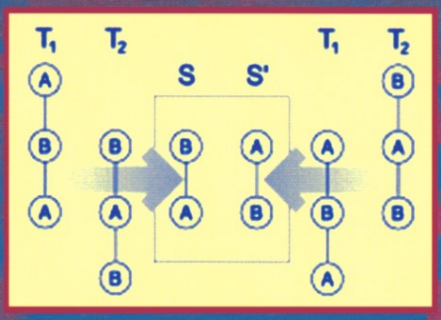
State-of-the-Art Survey

LNAI 3848

Jean-François Boulicaut
Luc De Raedt
Heikki Mannila (Eds.)

Constraint-Based Mining and Inductive Databases

European Workshop on Inductive Databases and Constraint Based Mining
Hinterzarten, Germany, March 2004, Revised Selected Papers



Springer

Lecture Notes in Artificial Intelligence 3848

Edited by J. G. Carbonell and J. Siekmann

Subseries of Lecture Notes in Computer Science

Jean-François Boulicaut Luc De Raedt
Heikki Mannila (Eds.)

Constraint-Based Mining and Inductive Databases

European Workshop on Inductive Databases and Constraint Based Mining
Hinterzarten, Germany, March 11-13, 2004
Revised Selected Papers

Springer

Series Editors

Jaime G. Carbonell, Carnegie Mellon University, Pittsburgh, PA, USA
Jörg Siekmann, University of Saarland, Saarbrücken, Germany

Volume Editors

Jean-François Boulicaut
INSA Lyon, LIRIS CNRS UMR 5205
69621 Villeurbanne, France
E-mail: Jean-Francois.Boulicaut@insa-lyon.fr

Luc De Raedt
Albert-Ludwigs-University, Institute for Computer Science
Georges-Köhler-Allee 79, 79110 Freiburg, Germany
E-mail: deraedt@informatik.uni-freiburg.de

Heikki Mannila
HIIT Basic Research Unit, University of Helsinki
and Helsinki University of Technology
P.O. Box 68, 00014 Helsinki, Finland
E-mail: Heikki.Mannila@cs.helsinki.fi

Library of Congress Control Number: 2005938512

CR Subject Classification (1998): I.2, H.2.8, H.2-3, D.3.3, F.1

ISSN 0302-9743
ISBN-10 3-540-31331-1 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-31331-1 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11615576 06/3142 5 4 3 2 1 0

Preface

The interconnected ideas of inductive databases and constraint-based mining are appealing and have the potential to radically change the theory and practice of data mining and knowledge discovery. Today, knowledge discovery is a very time-consuming and ad-hoc process, in which the analyst has to craft together a solution in a rather procedural manner. The ultimate goal of the inductive database framework is to develop an inductive query language, which would support the overall knowledge discovery process. Inductive queries specify constraints over patterns and models in a declarative way. Within this framework, the user then poses queries, which an inductive database management system has to answer, and knowledge discovery becomes an interactive querying process.

This book reports on the results of the European IST project cInQ (consortium on knowledge discovery by Inductive Queries) and its final workshop entitled "Inductive Databases and Constraint-Based Mining" organized in the Black Forest (Hinterzarten, Germany, March 11-14, 2004). The cInQ consortium consisted of INSA Lyon (France, coordinator: Jean-François Boulicaut), Università degli Studi di Torino (Italy, Rosa Meo and Marco Botta), the Politecnico di Milano (Italy, Pier-Luca Lanzi and Stefano Ceri), the Albert-Ludwigs-Universitaet Freiburg (Germany, Luc De Raedt), the Nokia Research Center in Helsinki (Finland, Mika Klemettinen and Heikki Mannila), and the Jozef Stefan Institute (Slovenia, Sašo Džeroski).

The workshop was attended by about 50 researchers, who presented their latest results in inductive querying and constraint-based data mining and also identified future directions. These results are presented in this book and provide a state-of-the-art overview of this newly emerging field lying at the intersection of data mining and database research. Even though we are still far away from inductive database management systems, a lot of progress has been made over the past few years, especially in constraint-based mining for local patterns (e.g., sets, sequential patterns, trees, graphs and rules), and in identifying some new primitives for data mining. Nevertheless, various important questions still remain open, such as the integration of query languages with databases and the fundamentals for inductive querying on global patterns.

The papers in this book can be categorized as follows (they are ordered in the book according to the name of the first author):

Keynote speakers: The chapter by Roberto J. Bayardo is an interesting position paper on various issues for constraint-based pattern mining. Johannes Gehrke and his co-authors provide a nice theoretical framework for optimizing constraint-based mining in difficult cases, typically when monotonicity properties are missing. Finally, Mohammed J. Zaki and his co-authors give a pragmatic view on the future of data mining software.

- *The Hows, Whys, and Whens of Constraints in Itemset and Rule Discovery* by Roberto J. Bayardo

- *How to Quickly Find a Witness* by Daniel Kifer, Johannes Gehrke, Cristian Bucila, and Walker White
- *Generic Pattern Mining via Data Mining Template Library* by Mohammed J. Zaki, Nilanjana De, Feng Gao, Paolo Palmerini, Nagender Parimi, Jeevan Pathuri, Benjarath Phoophakdee, and Joe Urban

Foundations: Several chapters address conceptual issues related to the inductive database framework, e.g., querying primitives, condensed representations, multiple uses of frequent sets, and the optimization of sequences of inductive queries:

- *A Relational Query Primitive for Constraint-Based Pattern Mining* by Francesco Bonchi, Fosca Giannotti and Dino Pedreschi.
- *A Survey on Condensed Representations for Frequent Sets* by Toon Calders, Christophe Rigotti and Jean-François Boulicaut
- *Boolean Formulas and Frequent Sets* by Jouni K. Seppänen and Heikki Mannila
- *Computation of Mining Queries: An Algebraic Approach* by Cheikh Talibouya Diop, Arnaud Giacometti, Dominique Laurent, and Nicolas Spyratos

Optimizing inductive queries on local patterns: Several chapters concern local pattern discovery by means of constraint-based mining techniques. A variety of pattern domains are considered such as trees, graphs, subgroups, inclusion dependencies, and association rules:

- *To See the Wood for the Trees: Mining Frequent Tree Patterns* by Björn Bringmann
- *Mining Constrained Graphs: The Case of Workflow Systems* by Gianluigi Greco, Antonella Guzzo, Giuseppe Manco, Luigi Pontieri, and Domenico Saccà
- *Relevancy in Constraint-Based Subgroup Discovery* by Nada Lavrač, and Dragan Gamberger
- *Adaptive Strategies for Mining the Positive Border of Interesting Patterns: Application to Inclusion Dependencies in Databases* by Fabien De Marchi, Frédéric Flouvat, and Jean-Marc Petit
- *A Novel Incremental Approach to Association Rules Mining in Inductive Databases* by Rosa Meo, Marco Botta, Roberto Esposito, and Arianna Gallo

Optimizing inductive queries on global patterns: Less research has been devoted to constraint-based mining of global patterns or models like clusters or classifiers. Important results in this direction are presented:

- *Inductive Queries on Polynomial Equations* by Sašo Džeroski, Ljupčo Todorovski, and Peter Ljubič
- *CrossMine: Efficient Classification Across Multiple Database Relations* by Xiaoxin Yin, Jiawei Han, Jiong Yang, and Philip S. Yu
- *Inductive Querying for Discovering Subgroups and Clusters* by Albrecht Zimmermann and Luc De Raedt

Applications: It is of course important to look at concrete applications of inductive querying techniques. Three chapters report on this:

- *Remarks on the Industrial Application of Inductive Database Technologies* by Kimmo Hätönen, Mika Klemettinen, and Markus Miettinen
- *Employing Inductive Databases in Concrete Applications* by Rosa Meo, Pier Luca Lanzi, Maristella Matera, Danilo Careggio, and Roberto Esposito
- *Contribution to Gene Expression Data Analysis by Means of Set Pattern Mining* by Ruggero G. Pensa, Jérémy Besson, Céline Robardet, and Jean-François Boulicaut

The editors would like to thank the EU (FET arm of the IST programme) for supporting the cInQ project as well as the Hinterzarten workshop, the partners in the cInQ consortium, and the participants in the workshop, especially our keynote speakers: Roberto J. Bayardo, Johannes Gehrke, and Mohammed J. Zaki. We hope that the readers will enjoy reading this book as much as we enjoyed the process of producing it.

September 2005

Jean-François Boulicaut
Luc De Raedt
Heikki Mannila

Table of Contents

The Hows, Whys, and Whens of Constraints in Itemset and Rule Discovery

Roberto J. Bayardo

IBM Almaden Research Center
bayardo@alum.mit.edu
http://www.almaden.ibm.com/cs/people/bayardo/

Abstract. Many researchers in our community (this author included) regularly emphasize the role constraints play in improving performance of data-mining algorithms. This emphasis has led to remarkable progress – current algorithms allow an incredibly rich and varied set of hidden patterns to be efficiently elicited from massive datasets, even under the burden of NP-hard problem definitions and disk-resident or distributed data. But this progress has come at a cost. In our single-minded drive towards maximum performance, we have often neglected and in fact hindered the important role of discovery in the knowledge discovery and data-mining (KDD) process. In this paper, I propose various strategies for applying constraints within algorithms for itemset and rule mining in order to escape this pitfall[1].

1 Introduction

Constraint-based pattern mining is the process of identifying all patterns in a given dataset that satisfy the specified constraints. There are many types of patterns we may wish to explore, depending on the data or its expected use. To name only a few, we have itemsets, sequences, episodes, substrings, rules, trees, cliques, and so on. The important aspect of constraint-based mining is not so much the specific patterns being identified, but the fact that we would like to identify *all* of them subject to the given constraints. This task of *constraint-based mining* is in contrast to *heuristic* pattern mining which attempts only to identify patterns which are likely (but not guaranteed) to be good according to certain criteria. A third task which I will touch upon only briefly, *optimization-based* pattern mining, attempts to identify only those patterns that are guaranteed to be (among the k-) best according to certain metrics.

While many may assign constraint-based mining a high face value solely from plethora of research on the topic, it is illustrative to take a step back and contemplate ***why*** it is a task worthy of our interest. Indeed, long before the "association rule" was a household name, heuristic pattern miners were proving extremely

[1] My use of the informal "I" rather than the typical "we" is to emphasize this paper is a personal position statement, along with a view of existing research in light of my position.

J.-F. Boulicaut et al. (Eds.): Constraint-Based Mining, LNAI 3848, pp. 1–13, 2005.

useful in machine learning circles. In fact, heuristic rule miners, which include decision tree ("divide and conquer") and covering ("separate and conquer") algorithms, remain essential components in the analyst's toolbox. I witnessed a growing interest in constraint-based mining once heuristic machine learning approaches gained reasonably widespread use in practice. The white-box nature of decision tree and other rule-based models were being used directly for end-user understanding of the data, even though they were not specifically intended for that purpose[2]. Use of these rule-based models for understanding led to questions such as the following:

- Do these rules capture and convey the "essence" of the relationship(s) in my data?
- Are there better rules (and who gets to define better)?

Note that each of these questions is open to some amount of subjective interpretation. But this is the point: the analyst is typically involved in *knowledge discovery* in which subjective and difficult to formalize notions of "goodness" are guiding the process, not simply data mining in which an algorithm follows a deterministic procedure to extract patterns that may or (more often) may not be of interest. Provided that constraints are used sensibly (and what "sensibly" means is the subject of this paper), constraint-based mining fosters discovery by providing the analyst with a broad result set capable of concretely answering a far wider set of questions than one that is heuristically determined.

A theme of this paper is that there are different phases of the knowledge discovery process in which we can exploit constraints, and the specific use of constraints should be dependent on ***when*** (in what phase) we are using them. During the mining phase, I argue that constraints should be *discovery preserving*. That is, they should filter out only those results that are *highly unlikely* to ever be of interest to the analyst. This admittedly informal notion of preserving discovery is in stark contrast to other proposals that envision query languages for constraint-based mining in which every imaginable constraint is enforced directly by the mining phase. The problem with this alternate view is simply that the analyst rarely knows the specific results of interest a priori (no pun intended). Constraints should therefore be used during the mining phase primarily for performance tractability. Discovering the precise results of interest is best left for post-processing of the mining results through interactive interfaces involving visualization, browsing, and ranking.

Recall that optimization-based pattern discovery forms an interesting middle-ground between the heuristic and constraint-based approaches: unlike heuristic approaches, it provides guarantees on result quality. Unlike constraint-based approaches, it provides these guarantees without requiring the extraction of all patterns matching the constraints, the number of which can be enormous. While these are desirable attributes, once again we are confronted with the question of what makes one rule better than the other. Optimization-based approaches allow

[2] It is therefore ironic that association rule miners are now commonly used in building general classification models, even though originally this was not their intended use!

no ambiguity on the part of the analyst since the ranking function is part of the input, if not hard-coded into the algorithm itself. Should an optimization-based approach be required (for example it is possible the pattern space is simply too large for constraints alone), I argue it is desirable for the approach to provide some ability to select and adjust the ranking criteria post-mining [6]. It is tempting to view an optimization criteria as itself just another constraint to be enforced by a constraint-based miner. Viewed as such, an optimization criteria is actually a constraint on the set of patterns rather than a constraint on the properties of the individual patterns. I believe this distinction is important enough to justify treating optimization-based approaches as separate from constraint-based ones.

As researchers, once we are convinced why something is useful, we become obsessed with ***how*** we can achieve it. And with constraint-based mining, the how part is particularly interesting due to huge computational challenges. Many constraint-based mining tasks can be proven NP-hard through reductions from problems such as constraint satisfaction, hitting set, prime implicant, and so on. Worse, the datasets involved often attain volumes beyond which standard data management strategies can efficiently cope. Then there is the issue of ensuring the results of our algorithms have statistical merit. This combination of search, data management, and statistical issues has provided ample research fodder for our community.

I cannot hope to even begin to address all interesting aspects of the hows in constraint-based mining in this short paper, but I will discuss some (often neglected) issues that I feel fit with in the context of discovery preservation. While much of what remains to be discussed applies to pattern mining in general, for concreteness sake, I focus in particular on itemsets and association rules. An itemset is simply a set of values appearing in a given dataset. An association rule is itself an itemset along with additional information specifying the division of items into antecedent and consequent subsets. The seminal work on association rule mining produced algorithms employing two distinct phases: (1) mine the frequent itemsets from the data, (2) output the rules of interest from them. While this two-phase approach was for the most part an operational detail of the mining algorithm, researchers (again, this author included) have been eager to build on only the first phase as if itemsets themselves are the output desired by the end user. I am quick to agree that itemsets are indeed *sometimes* the artifact of interest in data-mining. But that said, I believe, by and large, that the desired outcome of mining is more often *rules* since they express easy to interpret relationships between dataset elements that itemsets alone do not.

Luckily, many itemset constraints are themselves useful rule constraints, thus work in constraint-based itemset mining often has direct applications in constraint-based rule mining. There are, however, many constraints that are specific to rules such as bounds on confidence, lift, and other measures of predictive accuracy, and they have gone virtually ignored outside of result post-processing. To be fair, another reason rule-specific constraints have been ignored is that they do not fall into any of the convenient constraint classes that have been found to be easily enforceable during mining. But the fact is that many of these rule

constraints can be broken down into constituents that do fall into these classes. I will overview previous work in which properties of these constituents have been exploited for effective enforcement during mining *given appropriate structuring of the search*. That said, coming back to my original thesis, we typically would not want to enforce arbitrary rule constraints during mining to avoid hindering discovery. I therefore provide examples of rule constraints that can be regarded as discovery preserving, along with a framework for their enforcement during mining.

2 Constraints in the Discovery Process

It is well-known that knowledge discovery is a multi-phase and iterative process [11]. The data preparation and data-mining stages are often the most costly in terms of compute overhead. Thus, if possible, iteration should be restricted to subsequent phases (such as post-processing) in which it can be performed quickly. In the context of pattern mining, the role of the data-mining algorithm should be to transform the (preprocessed) dataset into a representation that allows for interactive browsing, ranking, and querying. "Interactive" means that the effects of changing a parameter, for example via a graphical control, are almost instantaneous. The following figure depicts this view.

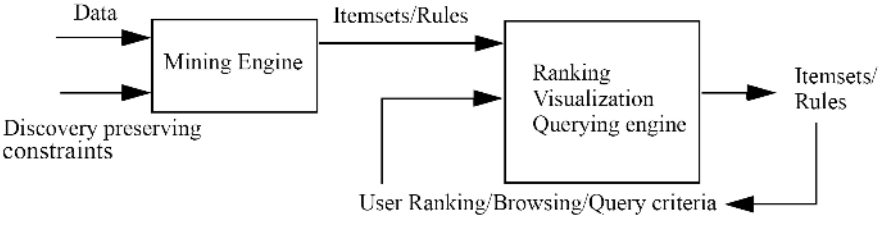


Fig. 1. Idealized View of the Mining Process

In some cases the input dataset may be sufficiently compact and the mining sufficiently trivial to allow the data-mining algorithm to be reapplied in real time to support interactivity. Mining caches can be used to further improve response [15,17], though I have doubts that cache hit rates will be significant enough for this to be of much use in practice.

More often, an intermediate representation is required to satisfy interactive response requirements. In the case of constraint-based rule and itemset mining, this intermediate representation is typically some collection of itemsets with their associated support values. For some datasets it might be possible to precompute the support of all possible itemsets and store them in an indexed database. However, most non-trivial datasets have enough items to make this impractical, as the number of itemsets increases exponentially with the number of items. A solution is to apply constraints to reduce the size of the mining result and the time required to obtain it, preferably without excluding patterns that are

of interest to the analyst. I argue that a good mining engine constraint has the following properties:

1. It is **tweakable**: post-mining, if the constraint is parameterized, the parameter should be adjustable without requiring expensive processing such as scanning or re-mining the original dataset.
2. It **provides efficiency**: applying the constraint should make the algorithm run significantly more efficiently. At this phase we are more concerned with using constraints for achieving tractability, and not necessarily in speeding up mining by a small constant.
3. It **preserves discovery**: the constraint, if it limits the sets of questions the analyst may efficiently pose during post-processing, should eliminate only those questions that are unlikely to be of value.

Properties 1 and 2 allow for the system itself to specify constraints automatically to ensure tractability of the mining run. The user is then able to efficiently adjust the constraints after the fact if necessary.

Property 3 implies that the system has a low probability of excluding patterns that may have otherwise been found interesting by the user. Property 3 is clearly the most subjective. Indeed, any pattern elimination can probably be rationalized as eliminating something useful for *some* purpose. However there are some constraints that do satisfy these properties in most settings. One example is a very low setting of minimum support. (1) Minimum support can be easily adjusted upwards post-mining without going back to the original dataset. One only needs to filter (or ignore) those itemsets whose supports falls below the modified limit. (2) Minimum support has been proven to provide significant boosts in efficiency during mining, even at relatively low settings. (3) Minimum support, provided it can be set low enough, preserves discovery since results with extremely low support are unlikely to be statistically valid.

Is minimum support enough? I feel it is safe to say that for "market-basket" and other sparse datasets, the answer is wholeheartedly yes. In fact, minimum support as exploited by the earliest of association rule miners (such as Apriori) is often entirely sufficient. In figure 2, I reprint with permission two graphs from a recent workshop on frequent itemset mining implementations (FIMI-03 [12]) in which participants submitted implementations for apples-to-apples comparison on a variety of datasets. For the sparse datasets, Borgelt's Apriori implementation outperformed most of the newer algorithms. Only for the very lowest support settings on the bmspos dataset was it outperformed by any significant amount. The point is that for any significantly complex mining task, the transformation and mining phases will be applied offline. Whether an algorithm requires one versus two hours to complete is not a major concern if iteration is relegated to post-processing.

Dense datasets tell a different story. Most tabular datasets with more than a handful of columns are sufficiently dense to render minimum support pruning woefully inadequate. In the FIMI-03 experiments, minimum support was the only constraint considered, and the minimum support levels attainable by any

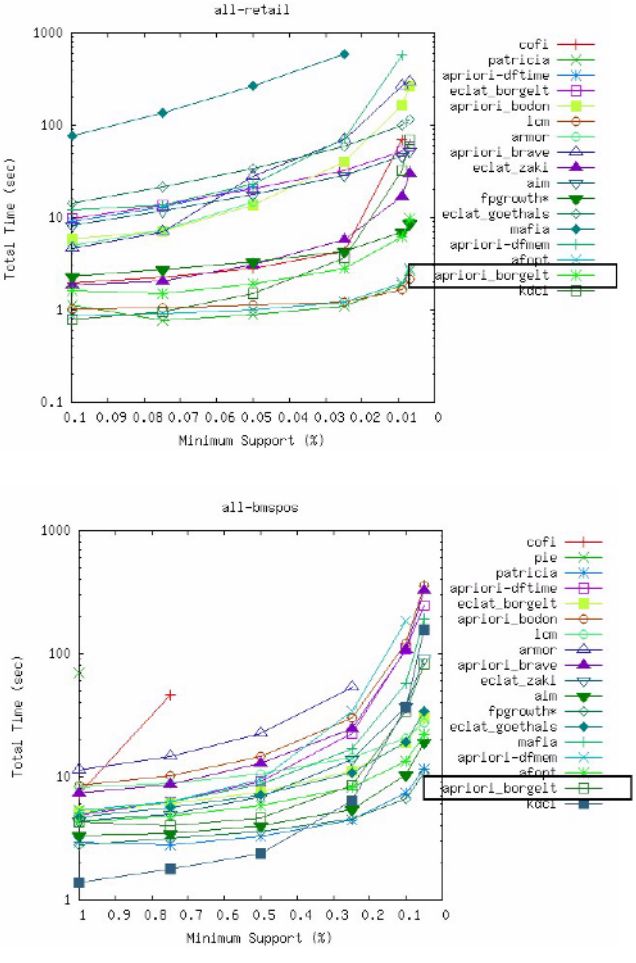


Fig. 2. Performance of the FIMI-03 implementations on sparse datasets

algorithm on the densest datasets were nowhere near what would be necessary to find any reasonably predictive rules [6]. We must therefore ask, what other constraints might we employ? Another good constraint is that the mining artifacts, whether itemsets or rules, be in a sense non-redundant. In the rule mining context, I noted in [8] that when an itemset has support equivalent to that of one of its subsets, it is redundant in the sense that it leads only to rules that are equivalent to existing rules in predictive accuracy and the population covered. It is a simple matter to prune such itemsets in order to avoid excessive counting due to equivalent supports. This idea is the basis of what is now commonly known as freeness and closure [13,19,25] in the context of itemset mining, and also what I called "antecedent maximality" in the context of rule mining [6]. Closure, while

not a parameterized property, is in a sense tweakable since we can easily regenerate non-closed itemsets (non-antecedent-maximal rules) from those in the set without dataset access. While it doesn't always help significantly (e.g. for sparse datasets), it certainly doesn't hurt performance. And finally, it doesn't hinder discovery whatsoever since it removes only redundant information which can be efficiently derived if necessary.

Unfortunately, I feel the benefits of closure are often overstated. It's no surprise that the Irvine "chess" and "connect-4" datasets are the most common benchmarks for demonstrating efficiency of closure-exploiting techniques. These datasets are relatively small and completely noise free. In the real world, data has noise. Noise quickly removes equivalence relations between itemsets, rendering closure-based pruning ineffective. Again, I refer to the FIMI-03 evaluation, where the algorithms employing closure remained ineffective on noisy dense datasets such as pumsb, except under an unreasonably high minimum support constraint.

As I further noted in [8], it is therefore worth exploiting "near equivalence", which is when an itemset has a support value that is within a very small amount of one of its subsets. This idea has since been better formalized as the principle of δ-free sets [13]. Pruning nearly equivalent itemsets restricts the questions we can ask from the result, but only slightly so. Further, it allows deriving of reasonably tight bounds on the support of any omitted itemset. While this makes the method very powerful in the itemset mining domain, these bounds cannot be straightforwardly used to intuitively quantify the effects on what rules are removed. But it is possible to perform some reasoning about rules using delta freeness, as demonstrated by Cremilleux and Boulicaut [14] who apply properties of delta free sets to characterizing classification rule conflicts. In the next section, I present what I believe to be a better constraint when rules instead of itemsets are the target pattern.

3 Discovery Preserving Rule-Specific Constraints

Rule-specific constraints are those that exploit properties specific to what distinguishes rules from itemsets; namely, the separation of the itemset into antecedent (A) and consequent (C) subsets (denoted $A \rightarrow C$). The most well known (and most often derided!) rule constraint is minimum confidence. Confidence, in the context of association rule mining, expresses the conditional probability with which the consequent holds given the antecedent:

$$conf(A \rightarrow C) = \frac{sup(A \cup C)}{sup(A)}$$

Confidence itself is not a bad metric. Along with knowledge of the background consequent probability (frequency), it conveys as much information as lift and related measures of predictive accuracy such as conviction [10]. It is my opinion that confidence, being a probability, is easier to interpret than these alternative measures. The problem with confidence stems from attempts at imposing a minimum bound in cases where the consequent of rules is allowed to vary,

as is the case in the traditional association rule mining problem [1]. A single fixed minimum on confidence will exclude highly predictive rules if their consequents have a very low frequency, and will allow completely non-predictive rules if their consequents have high a high frequency. But for rules which share the same consequent, the fact is that confidence, lift (also known as interest) and conviction rank rules identically [7]. A minimum confidence constraint in the case of *consequent-constrained* rule mining is actually a very useful constraint, as it can be used to concisely exclude only non-predictive rules[3]. I believe that excluding non-predictive rules is discovery preserving in most contexts, provided "non-predictive" is quantified in a sufficiently tight manner.

A constraint that excludes non-predictive results is a good start, but the fact is we can do even better without unduly hindering discovery. Many rules are highly predictive, but when considered in the appropriate context, are actually of little interest if the goal is indeed to understand predictive relationships. For example, we might find a highly predictive rule $\{i_1, i_2\} \rightarrow \{i_c\}$. But what if the rule $\{i_1\} \rightarrow \{i_c\}$ is even more predictive? The rule $\{i_1, i_2\} \rightarrow \{i_c\}$ considered in isolation of such subrules might lead to highly suboptimal decisions. The point is that one cannot fully understand the predictive nature of a rule without also considering the predictive behavior of all its proper subrules. (Formally, a subrule of a rule $A \rightarrow C$ is any rule $A' \rightarrow C$ such that $A' \subseteq A$.) This idea extends the notion that interpreting a rule from its confidence without considering its consequent frequency is virtually meaningless.

So if we are to accept that the analyst is interested in discovering predictive relationships, another interesting and discovery preserving constraint would be to remove all rules containing subrules that are more (or equally) predictive. (Note that this notion encompasses pruning with support equivalence, since it's easy to show a rule that improves upon the predictive accuracy of all its subrules has no functional dependencies between disjoint subsets of its antecedent.) When applying this constraint in practice, the effects are indeed dramatic, but problems remain. While it is strictly more powerful than pruning with closure, we are still plagued by "near equivalence" relationships between an itemset and its subsets. These near equivalences result in numerous rule variations, each reflecting roughly the same relationships along with one or more "noise" items. For example, we may again have that i strongly predicts , say, 1 ic with a confidence value of 90%. But we may also find that there exist dozens of other rules of the form $\{i_1\} \cup I \rightarrow \{i_c\}$ with confidence greater than 90% but perhaps less than 90.1%. Are these findings truly useful? I would argue in almost all cases they are not. One way to exclude such effects of near-equivalence is through a minimum positive bound on the predictive improvement a rule offers over all its subrules. Formally, I define the improvement value of a rule as the minimum of the differences between a rule's confidence and its proper subrules:

[3] A minimum confidence constraint also excludes negatively predictive rules, which are often of substantial interest. This can be avoided by also allowing the mining of rules that predict the negation of the desired consequent.

$$improvement(A \rightarrow C) = \min_{\forall A' s.t. (A' \subset A)} \{conf(A \rightarrow C) - conf(A' \rightarrow C)\}$$

Note that alternate definitions of the improvement value are possible, for example we could have defined improvement using ratios between confidence or lift values instead of differences. I chose differences between confidences because I feel it is easier to interpret.

Instead of simply requiring that improvement be positive, we can instead require that the improvement value of any rule exceed a specified bound. For example, a minimum improvement bound of .1 would exclude the noise rules from the examples above. Experiments show that even such a very low minimum improvement setting dramatically reduces the size of the result set and the time required to compute it [7]. Much as delta-free sets allow tight bounds on the support of any omitted itemset, a minimum improvement filtered rule set allows for tight upper bounds on the predictive ability of any omitted rule.

4 Enforcing Rule Constraints During Mining

Rule constraints such as minimums on confidence, lift, conviction and improvement are not exploitable through generic constraint-based itemset mining frameworks [18] because they are not classifiable as monotone, anti-monotone, succinct, convertible, or by any other simple and easily exploitable property. How then can we hope to exploit them during mining? Let us first consider the confidence value which I now rewrite slightly:

$$conf(A \rightarrow C) = \frac{sup(A \cup C)}{sup(A \cup C) + sum(A \cup \neg c)}$$

In the above expression, the notation c reflects a conceptual item that is contained by any record that does not contain the consequent itemset. Such records are deemed "negative examples" in the machine-learning context. Rewritten as such, note that the expression is obviously monotone in $sup(A \cup C)$ and anti-monotone in $sup(A \cup \neg c)$. Given that confidence consists of monotone and anti-monotone constituents, is it possible to compute a reasonably tight bound on the confidence achievable during mining? The answer is yes. The key is to explicitly keep track of all items that can be appended to an itemset to form a descendent of the itemset in a tree-structured search space. In the description of the Max-Miner algorithm [4], I referred to such a structure as an itemset "group", though a more mathematically precise term is perhaps a "subalgebra" of the itemset lattice [9]. This concept has its roots in algorithms for circuit optimization [22], though Webb [24] was first to formalize the concept within a generic search framework, and also exploit it in rule mining tasks.

More formally, let's assume the consequent itemset is fixed to C and we are searching over all possible antecedent itemsets for rules meeting various constraints. A node in the search space is represented by a head itemset H representing the antecedent of the rule enumerated by the node, and another ordered itemset T representing all items that can be appended to H to form descendents

of H in the search space. At each node in the search space, before expanding the children of the node, we filter items from T that cannot possibly lead to rules satisfying the constraints. In many cases, especially near the tree root, we may not be able to filter out any items. After filtering T, we obtain a new set T' whose size dictates the number of children of the node. For each item i in T', the child expansion policy creates a new node with head set $H \cup \{i\}$ and tail set $\{j | j \in T' \wedge j \text{ follows } i \text{ in the item ordering}\}$.

Note then as we descend in the tree, the itemset H always grows by exactly one item with each level (hence its support is monotonically decreasing), and the itemset $H \cup T$ either shrinks or stays the same (hence its support is monotonically increasing).

A bound on the confidence of any rule derivable by the node and its descendents can therefore be computed as follows:

$$conf\_bound(H \rightarrow C, T) = \frac{sup(H \cup C)}{sup(H \cup C) + sum(H \cup T \cup \neg c)}$$

Correctness of the bound computation follows directly from the monotonicity and anti-monotonicity properties stated earlier. Conveniently, being able to bound confidence allows us to also bound improvement of a rule: we simply compute $conf\_bound(H \rightarrow C, T) - conf(H' \rightarrow C)$ for the proper subrule $H' \rightarrow C$ of $H \rightarrow C$ with the highest confidence.

This is a simple but illustrative example. In fact, we can break down the improvement value itself into monotone and anti-monotone constituents to more directly derive a complementary bound on the improvement attainable by any of a node's descendents. This example is considerably more involved, but the essential concepts are the same. For the details I direct the reader to [7]. While not immediately obvious, this particular bounding technique effectively exploits near equivalences between antecedent subsets.

Constraint enforcement is a search space size issue, but in the data mining literature, its presentation is often confusingly intertwined with the particular data management and traversal strategies employed. The constraint enforcement techniques such as those from above are generic, and can be applied irrespective of breadth versus depth-first search, and irrespective of whether we use database scans to compute supports compared to more esoteric dataset representations involving database projections or projected FP-tree structures. Contrary to what I sometimes find implied in the literature, using a depth-first search does not magically provide more pruning opportunities, though it may simplify and optimize the gathering of necessary support information to be able to apply them.

5 Conclusions

In summary, I have attempted to make several (hopefully controversial!) points regarding the hows, whys, and whens of constraints in itemset and rule discovery:

1. First and foremost, while the utility of constraints in knowledge discovery is undeniable, they should be applied judiciously. In particular, apply during mining only those constraints that are in a sense *discovery preserving*.
2. Apriori adequately solves the problem of mining itemsets and rules from market-basket and other sparse datasets.
3. Itemset freeness/closure/equivalence is a powerful concept, but its effectiveness in practice is limited. Consider exploiting constraints based on near-equivalence instead, such as δ-freeness for itemsets and improvement thresholds for rules.
4. A rule cannot be fully interpreted in isolation from its subrules. This generalizes the well-known fact that confidence in absence of the consequent frequency is meaningless.
5. Itemset search methods should explicitly maintain the set of items that can be appended to the enumerated itemset in order to form its descendents in the search tree. Such explicit maintenance of lattice subalgebras allows exploiting both monotone and anti-monotone function constituents for more general and more powerful constraint enforcement.
6. Separating out issues of search space size from specific tree-traversal and data-management strategies improves understanding of algorithm performance, and increases the generality of constraint enforcement proposals.
7. Itemsets are not the only mining artifact of interest. Don't ignore the implications of mining-enforced constraints on what questions can be asked about the rules.

Now for the disclaimers. While I have stated each point as if it were maxim, I do not deny there are certain situations where some may fail to apply. Also, I do not claim to be the first to make them. Many similar points have been made within the literature (see for example [16] and [21] which are related point 1), and some I have picked up through osmosis from various talks and discussions. It is refreshing that even while I am writing this draft, I continue to come across new relevant work [3]. I must therefore apologize in advance for any references I have inevitably missed. For each point, I hope to minimally have provided some new perspectives.

References

1. R. Agrawal, T. Imielinski, and A. Swami. Mining Association Rules between Sets of Items in Large Databases. In *Proc. of the 1993 ACM-SIGMOD Conf. on Management of Data*, 207–216, 1993.
2. 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*, AAAI Press, 307–328, 1996.
3. C. Antunes and A. L. Oliveira. Mining Patterns Using Relaxations of User Defined Constraints. In *Proc. of the Workshop on Knowledge Discovery in Inductive Databases*, 2004.
4. R. J. Bayardo. Efficiently Mining Long Patterns from Databases. In *Proc. of the 1998 ACM-SIGMOD Int'l Conf. on Management of Data*, 85–93, 1998.

5. R. J. Bayardo, The many roles of constraints in data mining. (Letter from the guest editor.) *ACM SIGKDD Explorations* 4(1), i–ii, June 2002.
6. R. J. Bayardo and R. Agrawal. Mining the most interesting rules. In *Proc. of the Fifth ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, 145–154, 1999.
7. R. J. Bayardo, R. Agrawal, and G. Gunopulos. Constraint-based rule mining in large, dense databases. In *Proc. of the 15th Int'l Conf. on Data Engineering*, 188–197, 1999.
8. R. J. Bayardo. Brute-force mining of high confidence classification rules. In *Proc. of the Third International Conference on Knowledge Discovery and Data Mining*, 123–126, 1997.
9. C. Bucila, J. Gehrke, D. Kifer, DualMiner: A Dual-Pruning Algorithm for Itemsets with Constraints. In *Proc. SIGKDD* 2002.
10. S. Brin, R. Motwani, J. Ullman, and S. Tsur. Dynamic Itemset Counting and Implication Rules for Market Basket Data. In *Proc. of the 1997 ACM-SIGMOD Conf. on Management of Data*, 255–264, 1997.
11. R. J. Brachman and T. Anand. The Process Of Knowledge Discovery In Databases: A Human-Centered Approach. In A*dvances In Knowledge Discovery And Data Mining*, eds. U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, AAAI Press/The MIT Press, Menlo Park, CA., 37–57, 1996.
12. B. Goethals and M. J. Zaki. Advances in Frequent Itemset Mining Implementations: Introduction to FIMI-03. In *Proc. of the ICDM 2003 Workshop on Frequent Itemset Mining Implementations*, 2003.
13. J.-F. Boulicaut, A. Bykowski, and C. Rigotti. Approximation of frequency queries by means of free-sets. In *Proc. PKDD Int. Conf. Principles of Data Mining and Knowledge Discovery*, pages 75–85, 2000.
14. B. Cremilleux, J-F. Boulicaut. Simplest rules characterizing classes generated by delta-free sets. In: *Proceedings of the 22nd BCS SGAI International Conference on Knowledge Based Systems and Applied Artificial Intelligence*, Cambridge (UK), 33–46, 2002.
15. B. Jeudy and J.-F. Boulicaut. Using Condensed Representations for Interactive Association Rule Mining. In *Proc. of Principles of Data Mining and Knowledge Discovery: 6th European Conference (PKDD 2002)*, 228–236, 2002.
16. J. Hipp and U. Gntzer. Is pushing constraints deeply into the mining algorithms really what we want?: an alternative approach for association rule mining. *ACM SIGKDD Explorations* 4(1), 50–55, June 2002.
17. B. Nag, P. M. Deshpande, and D. J. DeWitt. Using a knowledge cache for interactive discovery of association rules. In *Proc. SIGKDD-1999*, 244–253, 1999.
18. R. Ng, L. V. S. Lakshmanan, J. Han, and A. Pang. Exploratory mining and pruning optimizations of constrained associations rules. In *Proc. SIGMOD-1998*, 13–24, 1998.
19. 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.
20. R. Rymon. Search through systematic set enumeration. In *Proc. of the Third Int'l Conf. on Principles of Knowledge Representation and Reasoning*, 539–550, 1992.
21. S. Sahar: Interestingness via What is Not Interesting. In *Proc. of SIGKDD-1999*: 332–336
22. J. R. Slagel, C.-L. Chang, and R. C. T. Lee. A New Algorithm for Generating Prime Implicants. *IEEE Trans. on Computers*, C-19(4):304–310, 1970.

23. R. Srikant, Q. Vu, and R. Agrawal. Mining Association Rules with Item Constraints. In *Proc. of the Third Int'l Conf. on Knowledge Discovery in Databases and Data Mining*, 67–73, 1997.
24. G. I. Webb. Opus: an efficient admissible algorithm for unordered search. *Journal of Artificial Intelligence Research* 3, 431–465, 1995.
25. M. J. Zaki. Generating non-redundant association rules. In *Proc. SIGKDD-2000*, pages 34–43, 2000.

A Relational Query Primitive for Constraint-Based Pattern Mining

Francesco Bonchi[1], Fosca Giannotti[1], and Dino Pedreschi[2]

[1] ISTI - CNR, Area della Ricerca di Pisa, Via Giuseppe Moruzzi, 1 - 56124 Pisa, Italy
[2] Dipartimento di Informatica, Via F. Buonarroti 2, 56127 Pisa, Italy

Abstract. As a step towards the design of an Inductive Database System, in this paper we present a primitive for constraint-based frequent pattern mining, which represents a careful trade-off between expressiveness and efficiency. Such primitive is a simple mechanism which takes a relational table in input and extracts from it all frequent patterns which satisfy a given set of user-defined constraints. Despite its simplicity, the proposed primitive is expressive enough to deal with a broad range of interesting constraint-based frequent pattern queries,using a comprehensive repertoire of constraints defined over SQL aggregates. Thanks to its simplicity, the proposed primitive is amenable to be smoothly embedded in a variety of data mining query languages and be efficiently executed, by the state-of-the-art optimization techniques based on pushing the various form of constraints by means of data reduction.

1 Introduction

Typically, two different kinds of structures are sought in data mining: *models* and *patterns* [31]. Models are high level, global, descriptive summaries of data sets. Patterns, on the other hand, are local descriptive structures. Patterns may be regarded as *local models*, and may involve just a few points or variables; i.e., they are descriptions of small fragments of the data, instead of overall descriptions. Accordingly, *Pattern Discovery* has a distinguished role within data mining technology. In particular, since *frequency* provides support to any extracted knowledge, it is the most used measure of interest for the extracted patterns. Therefore during the last decade a lot of researchers have focussed their studies on the computational problem of *Frequent Pattern Discovery*, i.e., mining patterns which satisfy a user-defined minimum threshold of frequency [3, 30].

The simplest form of frequent pattern is the frequent itemset: given a database of transactions (a transaction is a set of items) we want to find those subsets of transactions (itemsets) which appear together frequently.

Definition 1 (Frequent Itemset Mining). Let $\mathcal{I} = \{x_1, ..., x_n\}$ be a set of distinct literals, usually called *items*, where an item is an object with some predefined attributes (e.g., price, type, etc.). An *itemset* X is a non-empty subset of $\mathcal{I}$. If $|X| = k$ then X is called a *k-itemset*. A *transaction database* $\mathcal{D}$ is a bag of itemsets $t \in 2^{\mathcal{I}}$, usually called *transactions*. The *support* of an itemset X in

J.-F. Boulicaut et al. (Eds.): Constraint-Based Mining, LNAI 3848, pp. 14–37, 2005.

date	cust	item
11-2-97	cust1	beer
11-2-97	cust1	chips
11-2-97	cust1	wine
11-2-97	cust2	wine
11-2-97	cust2	beer
11-2-97	cust2	pasta
11-2-97	cust2	chips
13-2-97	cust1	chips
13-2-97	cust1	beer
13-2-97	cust2	jackets
13-2-97	cust2	col_shirts
13-2-97	cust3	wine
13-2-97	cust3	beer
15-2-97	cust1	pasta
15-2-97	cust1	chips
16-2-97	cust1	jackets
16-2-97	cust2	wine
16-2-97	cust2	pasta
16-2-97	cust3	chips
16-2-97	cust3	col_shirts
16-2-97	cust3	brown_shirts
18-2-97	cust1	pasta
18-2-97	cust1	wine
18-2-97	cust1	chips
18-2-97	cust1	beer
18-2-97	cust2	beer
18-2-97	cust2	beer
18-2-97	cust2	chips
18-2-97	cust2	chips
18-2-97	cust3	pasta

(a)

date	cust	itemset
11-2-97	cust1	{beer,chips,wine}
11-2-97	cust2	{wine,beer,pasta,chips}
13-2-97	cust1	{chips,beer}
13-2-97	cust2	{jackets,col_shirts}
13-2-97	cust3	{wine,beer}
15-2-97	cust1	{pasta,chips}
16-2-97	cust1	{jackets}
16-2-97	cust2	{wine,pasta}
16-2-97	cust3	{chips,col_shirts,brown_shirts}
18-2-97	cust1	{pasta,wine,chips,beer}
18-2-97	cust2	{beer,chips}
18-2-97	cust3	{pasta}

(b)

name	price	type
beer	10	beverage
chips	3	snack
wine	20	beverage
pasta	2	food
jackets	100	clothes
col_shirt	30	clothes
brown_shirt	25	clothes

(c)

Fig. 1. (a) A sample `sales` table, (b) its transactional representation and (c) the `product` table

database $\mathcal{D}$, denoted $supp_{\mathcal{D}}(X)$, is the number of transactions in $\mathcal{D}$ which are superset of X. Given a user-defined *minimum support* σ, an itemset X is called *frequent* in $\mathcal{D}$ if $supp_{\mathcal{D}}(X) \geq \sigma$.

The problem of mining all frequent itemsets in a database is the basis of the well-known association rule mining task [1]. However, frequent itemsets are meaningful not only in the context of association rules: they can be used as a basic mechanism in many other kind of analysis, ranging from classification [39, 40] to clustering [50, 58].

Example 2 (Market Basket Analysis). The classical context for association rule mining, as well as the most natural way to think about a transaction database, is the market basket setting, where we have a *sales database* of a retail store recording the content of each basket appearing at the cash register. In this context a transaction represent the content of a basket. In Figure 1(a) we have a relational table where a transaction or basket identifier is not explicitly given; however one could reconstruct transactions, for instance, grouping items by the pair (`date`, `cust`) as represented in Figure 1(b).

In principle, it is possible to express a query to count frequent itemsets in conventional SQL. This approach is examined in [57, 32, 52, 2]. For example,

in Figure 2 it is shown how to compute all 2-itemsets which are frequent in a relational database. We join `sales` with itself, with the condition that `tID` (transaction identifier) is the same, and the names of the two items are lexicographically different. We group the joined relation by the pair of items involved and check in the having clause that the group has at least 2 transactions.

```
SELECT  i1.item, i2.item
FROM  sales AS i1, sales AS i2
WHERE i1.item < i2.item AND
      i1.tID = i2.tID
GROUP BY i1.item, i2.item
HAVING 2 <= COUNT(i1.tID)
```

Fig. 2. Frequent 2-itemsets computation in SQL

The problem with this approach is that the right optimizations and "tricks" are beyond the state-of-the-art of conventional optimizers in commercial DBMS. Therefore a DBMS-based approach can not compete with specific algorithms employing ad hoc data structures. This is one of the main reasons, together with the stronger requirements of expressiveness, that justifies the need for specialized primitives and query languages for knowledge discovery.

1.1 Constraint-Based Pattern Discovery

Recently the research community has turned its attention to more complex kinds of frequent patterns extracted from more structured data: sequences, trees, and graphs. All these different kinds of patterns have different peculiarities and application fields, but they all share the same computational aspects: a usually very large input, an exponential search space, and a too large solution set. This situation – too many data yielding too many patterns – is harmful for two reasons. First, performance degrades: mining generally becomes inefficient or, often, simply unfeasible. Second, the identification of the fragments of interesting knowledge, blurred within a huge quantity of mostly useless patterns, is difficult.

Therefore, the paradigm of *constraint-based mining* was introduced. Constraints provide focus on the interesting knowledge, thus reducing the number of patterns extracted to those of potential interest. Additionally, they can be pushed deep inside the pattern discovery algorithm in order to achieve better performance [7, 4, 6, 9, 36, 37, 19, 25, 29, 38, 48, 49, 56, 44].

Definition 3 (Constrained Frequent Itemset Mining). A constraint on itemsets is a function $\mathcal{C} : 2^{\mathcal{I}} \rightarrow \{true, false\}$. We say that an itemset I satisfies a constraint if and only if $\mathcal{C}(I) = true$. We define the *theory* of a constraint as the set of itemsets which satisfy the constraint: $Th(\mathcal{C}) = \{X \in 2^{\mathcal{I}} \mid \mathcal{C}(X)\}$. Thus with this notation, the *frequent itemsets mining problem* requires to compute the set of all frequent itemsets $Th(\mathcal{C}_{freq[\mathcal{D},\sigma]})$. In general, given a conjunction of constraints $\mathcal{C}$ the *constrained frequent itemsets mining problem* requires to compute $Th(\mathcal{C}_{freq}) \cap Th(\mathcal{C})$.

According to the constraint-based mining paradigm, the data analyst must have a high-level vision of the pattern discovery system, without worrying about the details of the computational engine, in the very same way a database designer has not to worry about query optimizations. The analyst must be provided with a set of primitives to be used to communicate with the pattern discovery system, using a *Pattern Discovery Query Language*. The analyst just needs to declaratively specify in the pattern discovery query how the desired patterns should look like and which conditions they should obey (a set of constraints). Such rigorous interaction between the analyst and the pattern discovery system, can be achieved by means of a set of *pattern discovery primitives*, that should include:

- the specification of the source data,
- the kind of pattern to be mined,
- background or domain knowledge,
- the representation of the extracted patterns,
- constraints that interesting patterns must satisfy,
- interestingness measures for patterns evaluation.

Providing a query language capable to incorporate all these features may result, like in the case of relational databases, in a high degree of expressiveness in the specification of pattern discovery tasks, a clear and well-defined separation of concerns between logical specification and physical implementation of such tasks, and easy integration with heterogeneous information sources.

Clearly, the implementation of this vision presents a great challenge. A path to this goal is indicated in [41] where Mannila introduces an elegant formalization for the notion of interactive mining process: the term *inductive database* refers to a relational database plus the set of all sentences from a specified class of sentences that are true w.r.t. the data. In other words, the inductive database is a database framework which integrates the raw data with the knowledge extracted from the data and materialized in the form of patterns. In this way, the knowledge discovery process consists essentially in an iterative querying process, enabled by a query language that can deal either with raw data or patterns.

Definition 4. Given an instance $\mathbf{r}$ of a relation $\mathbf{R}$, a class $\mathcal{L}$ of sentences (patterns), and a selection predicate q, a pattern discovery task is to find a theory

$$\mathcal{T}h(\mathcal{L}, \mathbf{r}, q) = \{s \in \mathcal{L} | q(\mathbf{r}, s) \text{ is true}\}$$

The selection predicate q indicates whether a pattern s is considered interesting. In the constraint-based paradigm, such selection predicate q is defined by a conjunction of constraints.

1.2 Paper Contribution and Organization

As a step towards the design of an Inductive Database System, in this paper we present a primitive for constraint-based frequent pattern mining in a relational context. Such primitive is a simple mechanism which takes a relational table

in input and extracts from it all frequent patterns which satisfy a given set of user-defined constraints. Despite its simplicity, the proposed primitive is expressive enough to deal with a broad range of interesting constraint-based frequent pattern queries,using a comprehensive repertoire of constraints defined over SQL aggregates.

In this paper we do not deal specifically with the Data Mining Query Language issue, as our primitive can be embedded in any kind of query language based on the relational paradigm (SQL-like databases, logic-based deductive databases); basically we can view our primitive as an operator of the relational algebra.

Therefore, after introducing the primitive for constraint-based pattern mining we informally assess its expressiveness by means of example queries in SQL-like and DATALOG-like syntax. Then we concentrate on optimization issues: we summarize and amalgamates all the algorithmic results in constraint-based frequent pattern discovery obtained in the last years (2003-05) at Pisa KDD Laboratory, thus achieving an optimization framework.

The paper is organized as follows. In the next Section we define our primitive going through a rigorous identification of its basic components and of the constraints handled. In Section 3, we advocate the expressiveness and versatility of our proposal, showing how it can be embedded in query languages of different flavour. In Section 4 we recall the state-of-the-art classification of constraints and their properties, which can be exploited in order to achieve an efficient computation. In Section 5 we compose the state-of-the-art of the constraint pushing techniques in a breadth-first Apriori-like computation, achieving a very efficient evaluation strategy for our primitive. In particular, we adopt the strategy of pushing constraints in the computation mainly by means of data-reduction techniques: this approach enables us to exploit the different properties of constraints all together, and the total benefit is always greater than the sum of the individual benefits. Finally in Section 6 we conclude by describing on-going work on constrained frequent pattern discovery at Pisa KDD Laboratory, and drawing some future research paths.

2 A Primitive for Constraint-Based Mining

In this Section, going through a rigorous identification of all its basic components, we provide a definition of a primitive for constraint-based frequent pattern mining task over a relational database DB.

The first needed component is the data source: which table must be mined for frequent patterns, and which attributed do identify transactions and items.

Definition 5 (Mining View). Given a database DB any relational expression $\mathcal{V}$ on *preds*(DB) can be selected as data source, and named *mining view*.

Definition 6 (Transaction id). Given a database DB and a relation $\mathcal{V}$ derived from *preds*(DB). Let $\mathcal{V}$ with attributes $sch(\mathcal{V})$ be our mining view. Any subset of attributes $\mathcal{T} \subset sch(\mathcal{V})$ can be selected as transaction identifier, and named *transaction id*.

Definition 7 (Item attribute). Given a database DB and a relation $\mathcal{V}$ derived from *preds*(DB). Let $\mathcal{V}$ with attributes $sch(\mathcal{V})$ be our mining view. Given a subset of attributes $\mathcal{T} \subset sch(\mathcal{V})$ as transaction id, let $Y = \{y | y \in sch(\mathcal{V}) \backslash \mathcal{T} \wedge \mathcal{T} \rightarrow y \textit{ does not hold}\}$; we define an attribute $\mathcal{I} \in Y$ an *item attribute* provided the functional dependency $\mathcal{T}\mathcal{I} \rightarrow Y \backslash \mathcal{I}$ holds in DB.

Proposition 8. *Given a relational database* DB*, a triple* $\langle \mathcal{V}, \mathcal{T}, \mathcal{I} \rangle$ *denoting the mining view* $\mathcal{V}$*, the transaction id* $\mathcal{T}$*, the item attribute* $\mathcal{I}$*, uniquely identifies a transactional database, as defined in Definition 1.*

We next distinguish between attributes which describe items (descriptive attributes), from attribute which describe transactions (circumstance attributes).

Definition 9 (Circumstance attribute). Given a database DB and a relation $\mathcal{V}$ derived from *preds*(DB). Let $\mathcal{V}$ with attributes $sch(\mathcal{V})$ be our mining view. Given a subset of attributes $\mathcal{T} \subset sch(\mathcal{V})$ as transaction id, we define any attribute $\mathcal{A} \in sch(R)$ where R is a relation in *preds*(DB) *circumstance attribute* provided that $\mathcal{A} \notin \mathcal{T}$ and the functional dependency $\mathcal{T} \rightarrow \mathcal{A}$ holds in DB.

Definition 10 (Descriptive attribute). Given a database DB and a relation $\mathcal{V}$ derived from *preds*(DB). Let $\mathcal{V}$ with attributes $sch(\mathcal{V})$ be our mining view. Given a subset of attributes $\mathcal{T} \subset sch(\mathcal{V})$ as transaction id, and given $\mathcal{I}$ as item attribute; we define *descriptive attribute* any attribute $\mathcal{A} \in sch(R)$ where R is a relation in *preds*(DB), provided the functional dependency $\mathcal{I} \rightarrow \mathcal{A}$ holds in DB.

Consider the mining view: `sales(tID, locationID, time, product, price)` where each attribute has the intended semantics of its name and with `tID` acting as the transaction id. Since the functional dependency {`tID`} → {`locationID`} holds, `locationID` is a circumstance attribute. The same is true for `time`. We also have {`tID, product`} → {`price`}, and {`product`} → {`price`}, thus `product` is an item attribute, while `price` is a descriptive attribute.

Note that, from the previous definitions, *transaction id* and the *item attribute* must be part of the mining view, while circumstance and descriptive attributes could be also in other relations.

Constraints, as introduced in the previous Section (see Definition 3), describes properties of itemsets, i.e., a constraint $\mathcal{C}$ is a boolean function over the domain of itemsets: $\mathcal{C} : 2^{\mathcal{I}} \rightarrow \{true, false\}$. According to this view, constraints are only those ones defined on item attributes (Definition 7) or descriptive attributes (Definition 10).

Constraints defined over the *transaction id* (Definition 6) or over circumstance attributes (Definition 9) are not constraints in the strict sense. Indeed, they can be seen as selection conditions on the transactions to be mined and thus they can be satisfied in the definition of the *mining view*. Consider the relation: `sales(tID, locationID, time, product, price)` where each attribute has the intended semantics of its name and with `tID` acting as the transaction id. Since the functional dependency {`tID`} → {`locationID`} holds, `locationID` is a circumstance attribute. The constraints `locationID` ∈ {`Florence, Milan,`

`Rome`} is not a real constraint of the frequent pattern extraction, indeed it is a condition in the mining view definition, i.e., it is satisfied by imposing such condition in the relational expression defining the mining view (a `select` statement if we embed our primitive in a SQL-like language).

The following Definition lists all kinds of constraints that we consider as possible input for our primitive.

Definition 11 (Constraints on itemsets). In Table 1 all constraints admitted in our primitive are listed. The following notation is adopted:

- s *is an itemset;*
- $a_1, \ldots, a_n$ *are items;*
- d *is a descriptive attribute;*
- $d_1, \ldots, d_n$ *are values of a descriptive attribute;*
- m *is a numeric constant;*
- $\theta \in \{\leq, \geq, =\}$
- $aggr \in \{min, max, sum, avg, count, range, avg, median, var, std, md\}$

Table 1. Constraints admitted in our primitive

Constraint	Description
$s \supseteq \{a_1, \ldots, a_n\}$	itemset contains
$s \subseteq \{a_1, \ldots, a_n\}$	itemset domain
$count(s)\,\theta\,m$	itemset cardinality
$s.d \supseteq \{d_1, \ldots, d_n\}$	descriptive attribute contains
$s.d \subseteq \{d_1, \ldots, d_n\}$	descriptive attribute domain
$aggr(s.d)\,\theta\,m$	aggregate on descriptive attribute

We have provided all the needed components of our primitive, thus we are now ready to introduce it.

Definition 12 (Primitive for constraint-based itemset mining). Given a database DB, let the quintuple $\langle \mathcal{V}, \mathcal{T}, \mathcal{I}, \sigma, \mathcal{C} \rangle$ denotes the mining view $\mathcal{V}$, the transaction id $\mathcal{T}$, the item attribute $\mathcal{I}$, the minimum support threshold σ, and a conjunction of constraints on itemsets $\mathcal{C}$.

The primitive for constraint-based itemset mining takes in input such quintuple and returns a binary relation recording the set of itemsets which satisfy $\mathcal{C}$ and are frequent (w.r.t. σ) in the transaction database $\langle \mathcal{V}, \mathcal{T}, \mathcal{I} \rangle$, and their supports:

$$freq(\mathcal{V}, \mathcal{T}, \mathcal{I}, \sigma, \mathcal{C}) = \{(I, S) \mid \mathcal{C}(I) \ \wedge \ supp_{\langle \mathcal{V}, \mathcal{T}, \mathcal{I} \rangle}(I) = S \ \wedge \ S \geq \sigma\}$$

Example 13. A frequent pattern query for the `sales` table in Figure 1 (a), and the `product` table in Figure 1 (c), querying itemsets having a support ≥ 3 (transactions are made grouping by customer and date), and having a total price ≥ 30, could be simply defined as:

$freq(sales, \{date, cust\}, item, 3, sum\{p \mid i \in I \wedge product(i,p,t)\} \geq 30) = \{(I,S) \mid$
$sum\{p \mid i \in I \wedge product(i,p,t)\} \geq 30 \wedge supp_{\langle sales, \{date,cust\}, item\rangle}(I) = S \wedge S \geq 3\}$
The result of such query is a relation `(I,S)` with the following two entries: `({beer, wine},4)` and `({beer, wine, chips},3)`.

3 Embedding the Primitive into a Query Language

The issue of designing a query language capable of dealing with all requirements of knowledge discovery process, including definition of interestingness measures for extracted patterns and ad hoc exploitation of the application specific background knowledge, is a prominent research goal in data mining. This issue has been tackled both by a database perspective and a machine learning perspective (see [24] for a thorough discussion).

The proposal by a database perspective [33] is to combine relational query languages with data mining primitives in an overall framework capable of specifying data mining problems as an iterative and interactive querying session, where queries can involve both data and extracted models or patterns, and the result of a query becomes available for further querying (*closure principle*). In such a knowledge discovery system, identified by the term *inductive database*, query optimization and execution techniques typically rely on advanced ad hoc data mining algorithms. Past efforts for developing such languages can be classified in two categories:

- Developing mining tools tightly integrated with SQL DBMSs, representing both the source data and the induced patterns in database relations. Mining queries are specified in an SQL-like language [42, 43, 34, 35, 16, 27, 26, 55, 28, 29, 44].
- Exploiting logic to encode ad hoc data mining tasks and to specify background knowledge using the same language: typically some DATALOG extension [57, 53, 54, 21, 22, 14, 23].

On the machine learning side the main effort has been devoted to upgrading existing "propositional" data mining techniques to first order logic. This approach is denoted Inductive Logic Programming (ILP). ILP systems construct logic programs from examples (both positive and negative) and background knowledge: the challenge is to find a hypothesis that is consistent (w.r.t. negative examples) and complete (w.r.t. positive examples). The background knowledge and the hypothesis are expressed in two (not necessarily distinct) languages that are fixed in advance. In recent years, ILP has broadened its scope to cover standard data mining tasks such as classification, regression, clustering and association rules. This research is triggered by the need to pass from single-relational to *multirelational data mining* [20, 18, 17], i.e., a learning setting where every example is a set of facts or, equivalently, a (small) relational database.

An interesting approach would be to integrate the two different perspective in an overall framework, where multirelational data mining approach can meet

the potentials offered by the integration of mining and querying which is typical of the inductive database approach. A nice tentative in this direction is the logic language RDM [51] which uses terms for conjunctive queries: both constants and variables can be conjunctive queries, thus RDM queries can be regarded as higher order queries that seeks for standard conjunctive queries which satisfy some given constraints.

Although we are aware of the importance of designing a powerful query language for knowledge discovery, in this paper we do not provide yet another tentative in this direction. Less ambitiously, we provide a simple primitive for constraint-based frequent pattern queries, which can be embedded in any kind of query language and system coherent with the relational paradigm.

The rationale for this aim, is in our intention to focalize on the crucial point of the *Data Mining Query Language* problem: the hot-spot where an adequate trade-off between efficiency and expressiveness of the query language has to be found. The potential efficient evaluation of the proposed primitive is discussed in Section 4 and 5; here we advocate its adequate expressiveness and versatility showing how it can be embedded in query languages of different flavour.

Example 14 (Embedding the primitive in a SQL-like language). Consider the constraint-based frequent pattern query of Example 13. The following is some SQL-like syntactic sugar to express such query.

```
MINE PATTERNS Freq_pat, Support
FROM sales
GROUPING item BY day,cust
MINIMUM SUPPORT: 3
CONSTRAINTS ON name
FROM product
HAVING SUM(price) >= 30,
```

In the first line we define the name of the two output attributes `Freq_pat` and `Support` corresponding to the variables I and S of the query in Example 13. In the `FROM` clause we indicate the data source, or, in other words, the relation which plays the role of the mining view ($\mathcal{V}$): note that according to Definition 5 the mining view can be defined by a relational expression; this means that in the `FROM` clause we can have a full SQL `SELECT` clause. In the third line we indicate that transactions are created grouping the attribute `item` by the two attributes `day,cust`. In other word, we indicate the attribute which plays the role of item ($\mathcal{I}$), and a list of attributes which play the role of transaction identifier ($\mathcal{T}$). Note that such grouping is not really performed by the underlying DBMS: this is just syntactic sugar to express the input parameters for our primitive. In the fourth line we define the minimum support threshold σ. The last three lines define the additional constraint on the sum of prices. Since the attribute `price` is recorded in a relation different from the mining view we must indicate such relation (`product`) and the name of the item attribute in such relation (`name`).

An alternative could be to join the two tables in the definition of the mining view, as follows:

```
MINE PATTERNS Freq_pat, Support
FROM (SELECT *
      FROM sales JOIN product ON item = name
      )
GROUPING item BY day,cust
MINIMUM SUPPORT: 3
HAVING SUM(price) >= 30,
```

However, this requires that the underlying DBMS joins two tables, a costly operation not really necessary since our primitive can handle constraints defined on different relations. In general we can have how many clauses of the form "`CONSTRAINTS ON` *attribute* `FROM` *table* `HAVING` *constraint*" as necessary.

The simple query language whose syntax has been exemplified above, and whose semantics is indirectly given by Definition 12, seems to be an adequate trade-off between efficiency and expressiveness. It inherits its efficiency by our primitive, which is implemented by an ad hoc optimized mining algorithm (see Section 4 and 5). It is expressive since it allows full SQL definition of the source data, it can exploit a wide variety of different constraints, and more importantly, it is geared on frequent itemsets which is a primitive task for many complex queries. The most famous SQL-like data mining query language, MINE RULE [42], has not the same flexibility, being geared on association rules, nor the same efficiency, since it does not exploit constrain-pushing optimizations as those described in the next Section. MINE RULE allows to express queries for the mining of association rules whose body and head satisfy some structural constraints. In the following Example we show how such template-based queries can be expressed in a DATALOG-like query language which embeds our primitive as optimized mining engine.

Example 15 (Embedding the primitive in a DATALOG-like language). Consider again the relation in Figure 1(a) and (c). Suppose we want to compute simple association rules having support greater than 5 and confidence greater than 0.4 with exactly two items in the head (one of type `beverage` and one of type `snack`) and at least 3 items in the body.

As usually we can divide the association rule mining problem in two parts: the mining of frequent itemsets and the subsequent generation of valid rules. For the first subproblem we can use an inductive query which exploits our primitive in order to have an efficient mining, while the second post-processing subproblem can be solved by a simple DATALOG query.

The first inductive rule requires to compute itemsets with the proper support and containing at least one item of type `beverage` and one of type `snack`.

$$\begin{aligned}\texttt{frequentPatterns}(\texttt{Itemset},\texttt{Support}) \leftarrow\ & \texttt{Support} = \texttt{freq}(\texttt{Itemset},\texttt{X}),\\ & \texttt{X} = \langle \texttt{I} | \{\texttt{D},\texttt{C}\}\rangle,\\ & \texttt{sales}(\texttt{D},\texttt{C},\texttt{I}),\\ & \texttt{Support} >= 5,\\ & \texttt{product}(\texttt{L},\_,\texttt{beverage}),\\ & \texttt{product}(\texttt{J},\_,\texttt{snacks}),\\ & \{\texttt{L},\texttt{J}\} \subset \texttt{Itemset}.\end{aligned}$$

In the head of the rule we have the two output attributes `Itemset` and `Support`. The first clause in the body of the rule states that the variable `Support` stores the support of `Itemset` in the transaction database `X`. Such database is obtained from relation `sales(D,C,I)` (third clause) grouping `I` by {`D`,`C`} (second clause). The last three clauses in the body of the rule define the required constraints.

The second rule is a deductive DATALOG rule which computes the required association rules from the frequent itemsets by finding frequent itemsets of cardinality at least 5, having a frequent subset composed by two items to use as head of the association rule.

$$\begin{aligned}\texttt{rules}(\texttt{L},\texttt{R},\texttt{S},\texttt{C}) \leftarrow\ & \texttt{frequentPatterns}(\texttt{I},\texttt{S}),\\ & \texttt{cardinality}(\texttt{I}) >= 5,\\ & \texttt{frequentPatterns}(\texttt{R},\texttt{S}_1),\\ & \texttt{cardinality}(\texttt{R}) = 2,\\ & \texttt{subset}(\texttt{R},\texttt{I}),\texttt{difference}(\texttt{I},\texttt{R},\texttt{L}),\\ & \texttt{C} = \texttt{S}/\texttt{S1},\texttt{C} >= 0.4.\end{aligned}$$

As pointed out by these examples, our primitive is a propositional mechanism which works on a single relation, the input mining view. However, it is also possible in principle to embed the primitive into a multirelational data mining query language, such as RDM [51], equipped with high-order mechanisms to generate all possible mining views coherent with the specified item and transaction IDs, thus obtaining repeated invocation to the primitive on the various admissible mining views.

4 Constraint Properties and How to Exploit Them

Constrained frequent pattern mining can be seen as a query optimization problem: given a mining query $\mathcal{Q}$ containing a set of constraints $\mathcal{C}$, provide an efficient evaluation strategy for $\mathcal{Q}$ which is sound and complete (i.e. it finds all and only itemsets in $Th(\mathcal{C}_{freq}) \cap Th(\mathcal{C})$). A naïve solution to such a problem is to first find all frequent patterns ($Th(\mathcal{C}_{freq})$) and then test them for constraints satisfaction. However more efficient solutions can be found by analyzing the property of constraints comprehensively, and exploiting such properties in order to push constraints in the frequent pattern computation. Following this methodology, some classes of constraints which exhibit nice properties have been individuated. In this Section, by reviewing all basic works on the constrained frequent itemsets mining problem, we recall a classification of constraints and their properties.

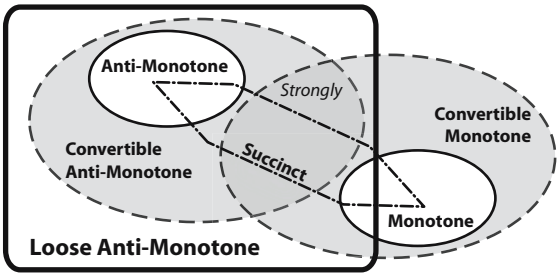


Fig. 3. Characterization of the classes of commonly used constraints

4.1 Anti-monotone and Succinct Constraints

A first work defining classes of constraints which exhibit nice properties is [44]. In that paper is introduced an Apriori-like algorithm, named CAP, which exploits two properties of constraints, namely *anti-monotonicity* and *succinctness*, in order to reduce the frequent itemsets computation. Four classes of constraints, each one with its own associated computational strategy, are defined:

1. Anti-monotone but not succinct constraints;
2. Anti-monotone and succinct constraints;
3. Succinct but not anti-monotone constraints;
4. Constraints that are neither.

Given an itemset X, a constraint $\mathcal{C}_{AM}$ is *anti-monotone* if $\forall Y \subseteq X : \mathcal{C}_{AM}(X) \Rightarrow \mathcal{C}_{AM}(Y)$. The frequency constraint is the most known example of a $\mathcal{C}_{AM}$ constraint. This property, *the anti-monotonicity of frequency*, is used by the Apriori [3] algorithm with the following heuristic: if an itemset X does not satisfy $\mathcal{C}_{freq}$, then no superset of X can satisfy $\mathcal{C}_{freq}$, and hence they can be pruned. This pruning can affect a large part of the search space, since itemsets form a lattice. Therefore the Apriori algorithm (see Algorithm1) operates in a level-wise fashion moving bottom-up on the itemset lattice, from small to large itemsets. At each iteration k Apriori counts the support of *candidate* itemsets (i.e. itemsets which have all subsets frequent) of size k, which are denoted by C_k.

Algorithm 1 Apriori

Input: $\mathcal{D}, \sigma$
Output: $Th(\mathcal{C}_{freq[\mathcal{D},\sigma]})$
1: $C_1 \leftarrow \{\{i\} \mid i \in \mathcal{I}\}$; $k \leftarrow 1$
2: **while** $C_k \neq \emptyset$ **do**
3: $L_k \leftarrow count(\mathcal{D}, C_k)$
4: $C_{k+1} \leftarrow generate\_apriori(L_k)$
5: $k++$
6: $Th(\mathcal{C}_{freq[\mathcal{D},\sigma]}) \leftarrow \bigcup_k L_k$

Those ones which have a support greater than the minimum support threshold σ are frequent itemsets. From the set of frequent itemsets of size k (denoted by L_k) the set of candidates for the next iteration C_{k+1} is generated by the *generate_apriori* procedure. Other $\mathcal{C}_{AM}$ constraints can easily be pushed deeply down into the frequent itemsets mining computation since they behave exactly as $\mathcal{C}_{freq}$: if they are not satisfiable at an early level (small itemsets), they have no hope of becoming satisfiable later (larger itemsets). Conjoining other $\mathcal{C}_{AM}$ constraints to $\mathcal{C}_{freq}$ we just obtain a more selective anti-monotone constraint.

A succinct constraint $\mathcal{C}_S$ is such that, whether an itemset X satisfies it or not, can be determined based on the singleton items which are in X. Informally, given A_1, the set of singleton items satisfying a succinct constraint $\mathcal{C}_S$, then any set X satisfying $\mathcal{C}_S$ is based on A_1 , i.e. X contains a subset belonging to A_1 (for the formal definition of succinct constraints see [44]). A $\mathcal{C}_S$ constraint is *pre-counting pushable*, i.e. it can be satisfied at candidate-generation time: these constraints are pushed in the level-wise computation by substituting the usual *generate_apriori* procedure, with the proper (w.r.t. $\mathcal{C}_S$) candidate generation procedure. For instance, consider the constraint $\mathcal{C}_S \equiv min(X.price) \leq v$, which is a succinct but not anti-monotone constraint. Given $A_1 = \{i \in \mathcal{I} \mid i.price \leq v\}$, we have that $Th(\mathcal{C}_S) = \{X \in 2^{\mathcal{I}} \mid \exists i \in X : i \in A_1\}$. Therefore this constraint can be satisfied at candidate-generation time. This can be done using a special candidate generation procedure, which takes care of the kind of the given constraint, and produces only candidates which satisfy it. Constraints that are both anti-monotone and succinct can be pushed completely in the level-wise computation before it starts (at pre-processing time). For instance, consider the constraint $min(X.price) \geq v$: if we start with the first set of candidates formed by all singleton items having price greater than v, during the computation we will generate only itemsets satisfying the given constraint. Constraints that are neither succinct nor anti-monotone are pushed in the CAP [44] computation by inducing weaker constraints which are either anti-monotone and/or succinct.

4.2 Monotone Constraints

Monotone constraints work the opposite way of anti-monotone constraints. Given an itemset X, a constraint $\mathcal{C}_M$ is monotone if: $\forall Y \supseteq X : \mathcal{C}_M(X) \Rightarrow \mathcal{C}_M(Y)$. Since the frequent itemset computation is geared on $\mathcal{C}_{freq}$, which is anti-monotone, $\mathcal{C}_M$ constraints have been considered more hard to be pushed in the computation and less effective in pruning the search space. In fact, many works [4, 19, 15, 13] have studied the computational problem $Th(\mathcal{C}_{freq}) \cap Th(\mathcal{C}_M)$, proposing some smart exploration of its search space, but all facing the inherent difficulty of the computational problem: the $\mathcal{C}_{AM}$-$\mathcal{C}_M$ *tradeoff*. Such tradeoff can be described as follows. Suppose that an itemset has been removed from the search space because it does not satisfy a monotone constraint. This pruning avoids checking support for this itemset, but on the other hand, if we check its support and find it smaller than the frequency threshold, we may prune away all the supersets of this itemset. In other words, by monotone pruning we risk to lose anti-monotone pruning opportunities given by the pruned itemset. The tradeoff is clear: pushing

monotone constraint can save frequency tests, however the results of these tests could have lead to more effective anti-monotone pruning.

In [7] a completely new approach to exploit monotone constraints by means of data-reduction is introduced. The *ExAnte Property* [7, 8] is obtained by shifting attention from the pattern search space to the input data. Indeed, the $\mathcal{C}_{AM}$-$\mathcal{C}_M$ *tradeoff* exists only if we focus exclusively on the search space of the problem, while if exploited properly, monotone constraints can reduce dramatically the data in input, in turn strengthening the anti-monotonicity pruning power. With data reduction techniques we exploit the effectiveness of a $\mathcal{C}_{AM}$-$\mathcal{C}_M$ *synergy*.

The ExAnte property states that a transaction which does not satisfy the given monotone constraint can be deleted from the input database since it will never contribute to the support of any itemset satisfying the constraint.

Proposition 16 (ExAnte property [7]). *Given a transaction database $\mathcal{D}$ and a conjunction of monotone constraints $\mathcal{C}_M$, we define the μ-reduction of $\mathcal{D}$ as the dataset resulting from pruning the transactions that do not satisfy $\mathcal{C}_M$: $\mu_{\mathcal{C}_M}(\mathcal{D}) = \{t \in \mathcal{D} \mid t \in Th(\mathcal{C}_M)\}$.*
It holds that this data reduction does not affect the support of solution itemsets:

$$\forall X \in Th(\mathcal{C}_M) : supp_{\mathcal{D}}(X) = supp_{\mu_{\mathcal{C}_M}(\mathcal{D})}(X).$$

A major consequence of reducing the input database in this way is that it implicitly reduces the support of a large amount of itemsets that do not satisfy $\mathcal{C}_M$ as well, resulting in a reduced number of candidate itemsets generated during the mining algorithm. Even a small reduction in the database can cause a huge cut in the search space, because all supersets of infrequent itemsets are pruned from the search space as well. In other words, monotonicity-based data-reduction of transactions strengthens the anti-monotonicity-based pruning of the search space. This is not the whole story, in fact, infrequent singleton items can not only be removed from the search space together with all their supersets, for the same anti-monotonicity property they also can be deleted from all transactions in the input database (this anti-monotonicity-based data-reduction is named α-*reduction*). Removing items from transactions offers another positive effect: reducing the size of a transaction which satisfies $\mathcal{C}_M$ can make the transaction violate it. Therefore a growing number of transactions which do not satisfy $\mathcal{C}_M$ can be found. Obviously, we are inside a loop where two different kinds of pruning (α and μ) cooperate to reduce the search space and the input dataset, strengthening each other step by step until no more pruning is possible (a fix-point has been reached).

The ExAMiner Algorithm. The recently introduced algorithm ExAMiner [6, 5], generalizes the ExAnte idea to reduce the problem dimensions at all levels of a level-wise Apriori-like computation. In this way, the $\mathcal{C}_{AM}$-$\mathcal{C}_M$ *synergy* is effectively exploited at each iteration of the mining algorithm, and not only at pre-processing as done by ExAnte, resulting in significant performance improvements. The idea is to generalize ExAnte's α-reduction from singletons level to the generic level k. This generalization results in the following set of data reduc-

tion techniques, which are based on the anti-monotonicity of $\mathcal{C}_{freq}$ (see [6] for the proof of correctness).

$\mathcal{G}_k(i)$**:** an item which is not subset of at least k frequent k-itemsets can be pruned away from all transactions in $\mathcal{D}$.

$\mathcal{T}_k(t)$**:** a transaction which is not superset of at least $k+1$ frequent k-itemsets can be removed from $\mathcal{D}$.

$\mathcal{L}_k(i)$**:** given an item i and a transaction t, if the number of frequent k-itemsets which are superset of i and subset of t is less than k, then i can be pruned away from transaction t.

Algorithm 2 *count&reduce*

Input: $\mathcal{D}_k, \sigma, \mathcal{C}_M, C_k, V_{k-1}$
Output: $\mathcal{D}_{k+1}, V_k, L_k$

```
1: forall i ∈ I do V_k[i] ← 0
2: forall tuples t in D_k do
3:    forall i ∈ t do if V_{k-1}[i] < k − 1
4:       then t ← t \ i
5:       else i.count ← 0
6:    if |t| ≥ k and C_M(t) then
7:       forall X ∈ C_k, X ⊆ t do
8:          X.count++; t.count++
9:          forall i ∈ X do i.count++
10:         if X.count = σ then
11:            L_k ← L_k ∪ {X}
12:            forall i ∈ X do V_k[i] + +
13:      if |t| ≥ k + 1 and t.count ≥ k + 1 then
14:         forall i ∈ t if i.count < k
15:            then t ← t \ i
16:            if |t| ≥ k + 1 and C_M(t) then
17:               write t in D_{k+1}
```

In ExAMiner [6] these data reductions are coupled with the μ-reduction for $\mathcal{C}_M$ constraints as described in Proposition 16. Essentially ExAMiner is an Apriori-like algorithm, which at each iteration $k-1$ produces a reduced dataset $\mathcal{D}_k$ to be used at the subsequent iteration k. Each transaction in $\mathcal{D}_k$, before participating to the support count of candidate itemsets, is reduced as much as possible by means of $\mathcal{C}_{freq}$-based data reduction, and only if it survives to this phase, it is effectively used in the counting phase. Each transaction which arrives to the counting phase, is then tested against the $\mathcal{C}_M$ (μ-reduction) , and reduced again as much as possible, and only if it survives to this second set of reductions, it is written to the transaction database for the next iteration $\mathcal{D}_{k+1}$. The procedure we have just described, is named *count&reduce* (see Algorithm 2), and substitutes the usual support counting procedure of Apriori (Algorithm 1).

In Algorithm 2 in order to implement the data-reduction $\mathcal{G}_k(i)$ we use an array of integers V_k (of the size of $Items$), which records for each item the number of frequent k-itemsets in which it appears. This information is then exploited during the subsequent iteration $k+1$ for the global pruning of items from all transaction in $\mathcal{D}_{k+1}$ (lines 3 and 4 of the pseudo-code). On the contrary, data reductions $\mathcal{T}_k(t)$ and $\mathcal{L}_k(i)$ are put into effect during the same iteration in which the information is collected. Unfortunately, they require information (the frequent itemsets of cardinality k) that is available only at the end of the actual counting (when all transactions have been used). However, since the set of frequent k-itemsets is a subset of the set of candidates C_k, we can use such data reductions in a relaxed version: we just check the number of candidate itemsets X which are subset of t ($t.count$ in the pseudo-code, lines 10 and 18) and which are superset of i ($i.count$ in the pseudo-code, lines 9 and 14).

4.3 Convertible Constraints

In [48, 49] the class of convertible constraints is introduced, and an FP-growth based methodology to push such constraints is proposed. A constraint $\mathcal{C}_{CAM}$ is convertible anti-monotone provided there is an order $\mathcal{R}$ on items such that whenever an itemset X satisfies $\mathcal{C}_{CAM}$, so does any prefix of X. A constraint $\mathcal{C}_{CM}$ is convertible monotone provided there is an order $\mathcal{R}$ on items such that whenever an itemset X violates $\mathcal{C}_{CM}$, so does any prefix of X. In [48, 49], two FP-growth based algorithms are introduced: $\mathcal{FIC}^{\mathcal{A}}$ to mine $Th(\mathcal{C}_{freq}) \cap Th(\mathcal{C}_{CAM})$, and $\mathcal{FIC}^{\mathcal{M}}$ to mine $Th(\mathcal{C}_{freq}) \cap Th(\mathcal{C}_{CM})$. A major limitation of any FP-growth based algorithm is that the initial database (internally compressed in the prefix-tree structure) and all intermediate projected databases must fit into main memory. If this requirement cannot be met, these approaches can simply not be applied anymore. This problem is even harder with $\mathcal{FIC}^{\mathcal{A}}$ and $\mathcal{FIC}^{\mathcal{M}}$: in fact, using an order on items different from the frequency-based one, makes the prefix-tree lose its compressing power. Thus we have to manage much greater data structures, requiring a lot more main memory which might not be available. Another important drawback of this approach is that it is not possible to take full advantage of a conjunction of different constraints, since each constraint in the conjunction could require a different ordering of items.

4.4 Loose Anti-monotone Constraints

In [12] a new class of tougher constraints, which is a proper superclass of convertible anti-monotone, is introduced together with an Apriori-like algorithm which exploit such constraints by means of data reduction.

Example 17 (var constraint is not convertible). Calculating the variance is an important task of many statistical analysis: it is a measure of how spread out a distribution is. The variance of a set of number X is defined as:

$$var(X) = \frac{\sum_{i \in X}(i - avg(X))^2}{|X|}$$

A constraint based on *var* is not convertible. Otherwise there is an order $\mathcal{R}$ of items such that $var(X)$ is a prefix increasing (or decreasing) function. Consider a small dataset with only four items $\mathcal{I} = \{A, B, C, D\}$ with associated prices $P = \{10, 11, 19, 20\}$. The lexicographic order $\mathcal{R}_1 = \{ABCD\}$ is such that $var(A) \leq var(AB) \leq var(ABC) \leq var(ABCD)$, and it is easy to see that we have only other three orders with the same property: $\mathcal{R}_2 = \{BACD\}, \mathcal{R}_3 = \{DCBA\}, \mathcal{R}_4 = \{CDBA\}$. But, for $\mathcal{R}_1$, we have that $var(BC) \not\leq var(BCD)$, which means that *var* is not a prefix increasing function w.r.t. $\mathcal{R}_1$. Moreover, since the same holds for $\mathcal{R}_2$, $\mathcal{R}_3$, $\mathcal{R}_4$, we can assert that there is no order $\mathcal{R}$ such that *var* is prefix increasing. An analogous reasoning can be used to show that it neither exists an order which makes *var* a prefix decreasing function.

Following a similar reasoning it can be shown that other interesting constraints, such as for instance those ones based on *standard deviation* (*std*) or *unbiased variance estimator* (var_{N-1}) or *mean deviation* (*md*), are not convertible as well. Luckily, all these constraints share a nice property that named *"Loose Anti-monotonicity"* [12].

While an anti-monotone constraint is such that, if satisfied by an itemset then it is satisfied by *all* its subsets, a loose anti-monotone constraint as such that, if it is satisfied by an itemset of cardinality k then it is satisfied by *at least one* of its subsets of cardinality $k-1$. Since some of these interesting constraints make sense only on sets of cardinality at least 2, in order to get rid of such details, we shift the definition of loose anti-monotone constraint to avoid considering singleton items.

Definition 18 (Loose Anti-monotone constraint). Given an itemset X with $|X| > 2$, a constraint is *loose anti-monotone* (denoted $\mathcal{C}_{LAM}$) if: $\mathcal{C}_{LAM}(X) \Rightarrow \exists i \in X \ : \ \mathcal{C}_{LAM}(X \setminus \{i\})$

The next proposition and the subsequent example state that the class of $\mathcal{C}_{LAM}$ constraints is a proper superclass of $\mathcal{C}_{CAM}$ (convertible anti-monotone constraints).

Proposition 19. *Any convertible anti-monotone constraint is trivially loose anti-monotone: if a k-itemset satisfies the constraint so does its* $(k-1)$*-prefix itemset.*

Example 20. We show that the constraint $var(X.A) \leq v$ is a $\mathcal{C}_{LAM}$ constraint. Given an itemset X, if it satisfies the constraint so trivially does $X \setminus \{i\}$, where i is the element of X which has associated a value of A which is the most far away from $avg(X.A)$. In fact, we have that $var(\{X \setminus \{i\}\}.A) \leq var(X.A) \leq v$, until $|X| > 2$.Taking the element of X which has associated a value of A which is the closest to $avg(X.A)$ we can show that also $var(X.A) \geq v$ is a $\mathcal{C}_{LAM}$ constraint. Since the standard deviation *std* is the square root of the variance, it is straightforward to see that $std(X.A) \leq v$ and $std(X.A) \geq v$ are $\mathcal{C}_{LAM}$. The mean deviation is defined as: $md(X) = (\sum_{i \in X} |i - avg(X)|) \ / \ |X|$. Once again, we have that $md(X.A) \leq v$ and $md(X.A) \geq v$ are loose anti-monotone. It is

easy to prove that also constraints defined on the unbiased variance estimator, $var_{N-1} = (\sum_{i \in X}(i - avg(X))^2) \ / \ (|X| - 1)$ are loose anti-monotone.

The next Proposition (see [12] for the proof) indicates how a $\mathcal{C}_{LAM}$ constraint can be exploited in a level-wise Apriori-like computation by means of data-reduction. It states that if at any iteration $k \geq 2$ a transaction is not superset of at least one frequent k-itemset which satisfy the $\mathcal{C}_{LAM}$ constraint (a solution), then the transaction can be deleted from the database.

Proposition 21. *Given a transaction database $\mathcal{D}$, a minimum support threshold σ, and a $\mathcal{C}_{LAM}$ constraint, at the iteration $k \geq 2$ of the level-wise computation, a transaction $t \in \mathcal{D}$ such that: $\nexists X \subseteq t, |X| = k, X \in Th(\mathcal{C}_{freq[\mathcal{D},\sigma]}) \cap Th(\mathcal{C}_{LAM})$ can be pruned away from $\mathcal{D}$, since it will never be superset of any solution itemsets of cardinality $> k$.*

As in ExAMiner [6] the anti-monotonicity based data reductions are coupled with the μ-reduction for $\mathcal{C}_M$ constraints, similarly we can exploit the above Proposition for $\mathcal{C}_{LAM}$ constraints, by embedding such loose anti-monotonicity based data reduction with-in the *count&reduce* procedure (see [12]).

5 Constraint Pushing Optimization

In this section we define an ad hoc optimized algorithm for the evaluation of our primitive on the basis of the state-of-the-art of constraint pushing techniques described in the previous Section. The proposed algorithm, is a breadth-first Apriori-like computation based on data-reduction techniques.

Adopting this kind of algorithmic architecture, i.e., moving level-wise and reducing data as much as possible, we can exploit different properties of constraints all together, and the global reduction benefit is always greater than the sum of the individual benefits.

In Table 2 we report the properties that our algorithm exploits for each constraint admitted in our framework. In the Table we do not report convertibility property since we do not exploit it. Convertibility, in fact, is well suited for FP-tree based depth-first algorithms [30, 48, 49] while we adopt an Apriori-like breadth-first computation. In our framework, the constraint based on the avg aggregate, which is the prototypical convertible constraint, is pushed in the computation by means of loose anti-monotonicity data reduction, obtaining stronger benefits. For a deeper discussion on this issue and for empirical comparison of the two different strategies see [12].

Algorithm 3 implements our primitive. Given $freq(\mathcal{V}, \mathcal{T}, \mathcal{I}, \sigma, \mathcal{C})$ Algorithm 3 computes $Th(\mathcal{C}_{freq[\mathcal{D},\sigma]}) \cap Th(\mathcal{C})$ where $\mathcal{D}$ is given by the triple $\langle \mathcal{V}, \mathcal{T}, \mathcal{I} \rangle$ as described in Section 2. In the pseudo-code the constraints in the conjunction $\mathcal{C}$ are partitioned in groups w.r.t. their properties. In particular:

- $\mathcal{C}_{AM}$ is the conjunction of constraints in $\mathcal{C}$ which are anti-monotone but not succinct;

Table 2. Properties of the constraints admitted in our primitive

Constraint	Properties
$S \subseteq V$	$\mathcal{C}_{AM}, \mathcal{C}_S, \mathcal{C}_{LAM}$
$S \supseteq V$	$\mathcal{C}_M, \mathcal{C}_S, \mathcal{C}_{LAM}$
$S.A \subseteq V$	$\mathcal{C}_{AM}, \mathcal{C}_S, \mathcal{C}_{LAM}$
$S.A \supseteq V$	$\mathcal{C}_M, \mathcal{C}_S, \mathcal{C}_{LAM}$
$min(S.A) \geq v$	$\mathcal{C}_{AM}, \mathcal{C}_S, \mathcal{C}_{LAM}$
$min(S.A) \leq v$	$\mathcal{C}_M, \mathcal{C}_S, \mathcal{C}_{LAM}$
$max(S.A) \geq v$	$\mathcal{C}_M, \mathcal{C}_S, \mathcal{C}_{LAM}$
$max(S.A) \leq v$	$\mathcal{C}_{AM}, \mathcal{C}_S, \mathcal{C}_{LAM}$
$count(S) \leq v$	$\mathcal{C}_{AM}, \mathcal{C}_{LAM}$
$count(S) \geq v$	$\mathcal{C}_M$
$count(S.A) \leq v$	$\mathcal{C}_{AM}, \mathcal{C}_{LAM}$
$count(S.A) \geq v$	$\mathcal{C}_M$
$sum(S.A) \leq v\ (\forall i \in S, i.A \geq 0)$	$\mathcal{C}_{AM}, \mathcal{C}_{LAM}$
$sum(S.A) \geq v\ (\forall i \in S, i.A \geq 0)$	$\mathcal{C}_M$
$range(S.A) \leq v$	$\mathcal{C}_{AM}, \mathcal{C}_{LAM}$
$range(S.A) \geq v$	$\mathcal{C}_M, \mathcal{C}_{LAM}$
$avg(S.A)\theta v$	$\mathcal{C}_{LAM}$
$median(S.A)\theta v$	$\mathcal{C}_{LAM}$
$var(S.A)\theta v$	$\mathcal{C}_{LAM}$
$std(S.A)\theta v$	$\mathcal{C}_{LAM}$
$md(S.A)\theta v$	$\mathcal{C}_{LAM}$

- $\mathcal{C}_{AMS}$ is the conjunction of constraints in $\mathcal{C}$ which are both anti-monotone and succinct;
- $\mathcal{C}_M$ is the conjunction of constraints in $\mathcal{C}$ which are monotone but not succinct;
- $\mathcal{C}_{MS}$ is the conjunction of constraints in $\mathcal{C}$ which are both monotone and succinct;
- $\mathcal{C}_{LAM}$ is the conjunction of constraints in $\mathcal{C}$ which are loose anti-monotone.

Note that this groups of constraints are not necessarily disjoint.

Example 22. The constraint $range(S.A) \geq v \equiv max(S.A) - min(S.A) \geq v$, is both monotone and loose anti-monotone. Thus, when we mine frequent itemsets which satisfy such constraint we can exploit the benefit of having together, in the same *count&reduce* procedure, the $\mathcal{C}_{freq}$-based data reductions, the μ-reduction for monotone constraints, and the reduction based on $\mathcal{C}_{LAM}$.

The possibility of exploiting different properties of constraints all together, exists not only for $\mathcal{C}_M$ and $\mathcal{C}_{LAM}$ constraints (as seen in Example 22), but also for any other kind of constraints. In fact, all the properties that we exploit are orthogonal and thus can be combined.

Example 23. Consider now the constraint $max(S.A) \geq v$. This constraint is monotone, succinct and loose anti-monotone. This means that we can exploit all

these properties by using it as a succinct constraint at candidate generation time, and using it as a monotone constraint and as a loose anti-monotone constraint by means of data-reduction at counting time.

Algorithm 3 Constraint-based Frequent Pattern Mining

Input: $\mathcal{D}, \sigma, \mathcal{C}$
Output: $Th(\mathcal{C}_{freq[\mathcal{D},\sigma]}) \cap Th(\mathcal{C})$

1: $L_1 \leftarrow \mathcal{I}$
2: $C_1 \leftarrow \{\{i\} \mid i \in \mathcal{I} \wedge \mathcal{C}_{AMS}(\{i\}) \wedge \mathcal{C}_{AM}(\{i\})\}$
3: $\mathcal{D}_1 \leftarrow \pi_{C_1}(\mathcal{D})$
4: $L_1, \mathcal{D}_1 \leftarrow count\_first\_iteration(\mathcal{D}_1, \sigma, C_1, \mathcal{C}_M, \mathcal{C}_{MS})$
5: **while** $L_1 \neq C_1$ **do**
6: $\quad C_1 \leftarrow L_1$;
7: $\quad L_1, \mathcal{D}_1 \leftarrow count\_first\_iteration(\mathcal{D}_1, \sigma, C_1, \mathcal{C}_M, \mathcal{C}_{MS})$
8: $C_2 \leftarrow generate(L_1, \mathcal{C}_{AM}, \mathcal{C}_{MS})$
9: **forall** $i \in L_1$ **do** $V_1[i] \leftarrow 0$
10: $k \leftarrow 2$
11: **while** $C_k \neq \emptyset$ **do**
12: $\quad L_k, \mathcal{D}_{k+1}, V_k \leftarrow count\&reduce(\mathcal{D}_k, \sigma, \mathcal{C}_M, \mathcal{C}_{MS}, \mathcal{C}_{LAM}, C_k, V_{k-1})$
13: $\quad C_{k+1} \leftarrow generate(L_k, \mathcal{C}_{AM}, \mathcal{C}_{MS})$
14: $\quad k++$
15: **for**$(i = 0; i \leq k; i++)$ **do**
16: $\quad$ **forall** $X \in L_i$ **do**
17: $\quad\quad$ **if** $\mathcal{C}_M(X) \wedge \mathcal{C}_{MS}(X) \wedge \mathcal{C}_{LAM}(X)$ **then return** X

Let us briefly describe the pseudo-code in Algorithm 3. First of all, note that as stated in Section 3 constraints which are both anti-monotone and succinct are pushed once and for all, at preprocessing, simply by considering in the forthcoming computation singleton items which satisfy them (Line 2). Lines from 3 to 7 together with procedure *count_first_iteration* (Algorithm 4), implement the Ex-Ante pre-processing [7]. Lines from 11 to 14 implements the typical central loop of the Apriori algorithm, where the *generate* procedure exploits succinctness and anti-monotonicity to reduce the set of candidates, and the *count&reduce* procedure exploits monotonicity and loose anti-monotonicity. Finally, lines from 15 to 17 implement the post-processing, where possible solution itemsets are check for satisfaction of those kinds of constraints, for which satisfaction not already guaranteed.

Our algorithm, by means of data-reduction, exploits a real synergy of all constraints that the user defines for the pattern extraction: each constraint does not only play its part in reducing the data, but this reduction in turns strengthens the pruning power of the other constraints. Moreover data-reduction induces a pruning of the search space, and the pruning of the search space in turn strengthens future data reductions.

The orthogonality of the exploited constraint pushing techniques has a twofold benefit: on one hand all the techniques can be amalgamated together achieving a

Algorithm 4 *count_first_iteration*

Input: $\mathcal{D}, \sigma, C, \mathcal{C}_M, \mathcal{C}_{MS}$
Output: $\mathcal{D}_1, L_1$

1: $L_1 \leftarrow \emptyset; \mathcal{D}_1 \leftarrow \emptyset$
2: **forall** $t \in \mathcal{D}$ **do**
3: **if** $\mathcal{C}_M(t) \wedge \mathcal{C}_{MS}(t)$ **then**
4: **forall** $i \in t$ **do** $i.count++$; **if** $i.count++ = \sigma$ **then** $L_1 \leftarrow L_1 \cup \{i\}$
5: $\mathcal{D}_1 \leftarrow \mathcal{D}_1 \cup t$
6: $\mathcal{D}_1 \leftarrow \pi_{L_1}(\mathcal{D}_1)$

very efficient computation (for the empirical evaluation of each single technique we address the interested reader to the respective paper cited in Section 4); on the other hand the framework can be easily extended to handle to other constraints.

Another positive effect of adopting an Apriori-like algorithm, is that in the implementation we can exploit all coding tricks and smart data structure that have been developed in the last decade for the Apriori algorithm (see, for instance, [45, 46]).

At Pisa KDD Laboratory developed a prototype of the optimized computational framework in tight collaboration with the authors of [45, 46], within the P^3D project[1].

6 Conclusions and Future Work

In this paper, we introduced a primitive for constraint-based pattern discovery, which represents a trade-off between simplicity and generality, as well as between expressiveness and efficiency. The versatility of the primitive is witnessed by its easy adaption within query languages of different nature; its efficiency is witnessed by the availability of systematic optimization methods, based on the properties of the specified constraints. We believe that this trade-off ia a step forward in the road to a realistic inductive database system. We are also aware that many issues remain open, and deserve further research:

- how to support user-defined constraints;
- how to integrate condensed representations of patterns in the constraint-based mining framework [47, 10, 11];
- how to tightly integrate our primitive within a relational DBMS: his issue is strictly connected with many other open problems, for instance, how to store and index frequent pattern query results;
- defining constraint-based incremental mining techniques, i.e., how to exploit the results of previous queries in order to have a more efficient computation for the forthcoming queries;
- developing a constraint-based mining framework for more complex kinds of patterns such as sequences and graphs.

[1] `http://www-kdd.isti.cnr.it/p3d/index.html`

Our objective is to integrate the results of these investigations in a unified system for exploratory constraint-based pattern discovery.

References

1. R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In *Proceedings of ACM SIGMOD'93*.
2. R. Agrawal and K. Shim. Developing tightly-coupled data mining applications on a relational database system. In *Proceedings of KDD'96*.
3. R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules in Large Databases. In *Proceedings of VLDB'94*.
4. F. Bonchi, F. Giannotti, A. Mazzanti, and D. Pedreschi. Adaptive Constraint Pushing in frequent pattern mining. In *Proceedings of PKDD'03*.
5. F. Bonchi, F. Giannotti, A. Mazzanti, and D. Pedreschi. Efficient Breadth-first Mining of Frequent Pattern with Monotone Constraints. To appear in *Knowledge and Information Systems - An International Journal (KAIS)*. Springer, Berlin.
6. F. Bonchi, F. Giannotti, A. Mazzanti, and D. Pedreschi. ExAMiner: Optimized level-wise frequent pattern mining with monotone constraints. In *Proceedings of ICDM'03*.
7. F. Bonchi, F. Giannotti, A. Mazzanti, and D. Pedreschi. ExAnte: Anticipated data reduction in constrained pattern mining. In *Proceedings of PKDD'03*.
8. F. Bonchi, F. Giannotti, A. Mazzanti, and D. Pedreschi. Preprocessing for Frequent Pattern Mining through Data Reduction. To appear in *IEEE Intelligent Systems*.
9. F. Bonchi and B. Goethals. FP-Bonsai: the Art of Growing and Pruning Small FP-trees. In *Proceedings of PAKDD'04*, 2004.
10. F. Bonchi and C. Lucchese. On closed constrained frequent pattern mining. In *Proceedings of ICDM'04*.
11. F. Bonchi and C. Lucchese. On Condensed Representations of Constrained Frequent Patterns. To appear in *Knowledge and Information Systems - An International Journal (KAIS)*. Springer, Berlin.
12. F. Bonchi and C. Lucchese. Pushing tougher constraints in frequent pattern mining. In *Proceedings of PAKDD'05*, 2005.
13. J.F. Boulicaut and B. Jeudy. Using constraints during set mining: Should we prune or not? In *Actes des Seizime Journes Bases de Donnes Avances BDA'00*.
14. J.F. Boulicaut, P. Marcel, and C.Rigotti. Query driven knowledge discovery in multidimensional data. In *Proceedings of DOLAP'99*.
15. C. Bucila, J. Gehrke, D. Kifer, and W. White. DualMiner: A dual-pruning algorithm for itemsets with constraints. In *Proceedings of ACM SIGKDD'02*.
16. S. Choenni and A. Siebes. Query Optimization to Support Data Mining. In *Proc. of the Int'l. Workshop on Database and Expert Systems Application 1997*.
17. L. Dehaspe and L. De Raedt. Dlab: A declarative language bias formalism. In *Proceedings of ISMIS'96*.
18. L. Dehaspe and H. Toivonen. Discovery of Frequent Datalog Patterns. *Journal of Knowledge Discovery and Data Mining*, 3(1):7–36, 1999.
19. L. De Raedt and S. Kramer. The levelwise version space algorithm and its application to molecular fragment finding. In *Proceedings of IJCAI'01*.
20. S. Džeroski and N. Lavrač, editors. *Relational Data Mining*. Springer, Berlin, 2001.
21. F. Giannotti and G. Manco. Querying Inductive Databases via Logic-Based User-Defined Aggregates. In *Proceedings of PKDD'99*.

22. F. Giannotti and G. Manco. Making Knowledge Extraction and Reasoning Closer. In T. Terano, editor, *Proceedings of PAKDD'00*.
23. F. Giannotti, G. Manco and F. Turini. Specifying Mining Algorithms with Iterative User-Defined Aggregates. IEEE Trans. Knowl. Data Eng. 16(10): 1232-1246 (2004).
24. F. Giannotti, G. Manco and J. Wijsen. Logical Languages for Data Mining. In *Logics for emerging Applications of Databases*. Springer, Berlin, 2003.
25. G. Grahne, L. Lakshmanan, and X. Wang. Efficient mining of constrained correlated sets. In *Proceedings of ICDE'00)*.
26. J. Han. Towards On-Line Analytical Mining in Large Databases. *Sigmod Records*, 27(1):97–107, 1998.
27. J. Han, S. Chee, and J. Chiand. Issues for On-Line Analytical Mining of Data Warehouses. In *Proceedings of DMKD'98*.
28. J. Han, Y. Fu, K. Koperski, W. Wang, and O. Zaiane. DMQL: A Data Mining Query Language for Relational Databases. In *Proceedings of DMKD'96*.
29. J. Han, L. V. S. Lakshmanan, and R. T. Ng. Constraint-based, multidimensional data mining. *Computer*, 32(8):46–50, 1999.
30. J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proceedings of ACM SIGMOD'00*.
31. D. Hand, H. Mannila, and P. Smyh. *Principles of Data Mining*. The MIT Press, 2001.
32. M. Houtsma and A. Swami. Set-oriented mining for association rules in relational databases. In *Proceedings of ICDE'95*.
33. T. Imielinski and H. Mannila. A database perspective on knowledge discovery. *Comm. Of The Acm*, 39:58–64, 1996.
34. T. Imielinski and A. Virmani. MSQL: A Query Language for Database Mining. *Data Mining and Knowledge Discovery*, 3(4):373–408, 1999.
35. T. Imielinski, A. Virmani, and A. Abdulghani. DMajor - Application Programming Interface for Database Mining. *Data Mining and Knowledge Discovery*, 3(4):347–372, 1999.
36. B. Jeudy and J.F. Boulicaut. Optimization of association rule mining queries. *Intelligent Data Analysis Journal*, 6(4):341–357, 2002.
37. S. Kramer, L. De Raedt, and C. Helma. Molecular feature mining in hiv data. In *Proceedings of ACM SIGKDD'01*.
38. L. V. S. Lakshmanan, R. T. Ng, J. Han, and A. Pang. Optimization of constrained frequent set queries with 2-variable constraints. *SIGMOD Record*, 28(2), 1999.
39. W. Li, J. Han, and J. Pei. CMAR: Accurate and efficient classification based on multiple class-association rules. In *In Proceedings of ICDM'01*.
40. B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *Proceedings of KDD'98*.
41. H. Mannila and H. Toivonen. Levelwise Search and Border of Theories in Knowledge Discovery. *Data Mining and Knowledge Discovery*, 3:241–258, 1997.
42. R. Meo, G. Psaila, and S. Ceri. A new SQL-like operator for mining association rules. In *Proceedings of VLDB'96*.
43. R. Meo, G. Psaila, and S. Ceri. A Tightly-Coupled Architecture for Data Mining. In *Proceedings of ICDE'98*.
44. R. T. Ng, L. V. S. Lakshmanan, J. Han, and A. Pang. Exploratory mining and pruning optimizations of constrained associations rules. In *Proceedings of the ACM SIGMOD'98*.
45. S. Orlando, P. Palmerini, and R. Perego. Enhancing the Apriori Algorithm for Frequent Set Counting. In *Proceedings of DaWak'01*.

46. S. Orlando, P. Palmerini, R. Perego, and F. Silvestri. Adaptive and Resource-Aware Mining of Frequent Sets. In *Proceedings of ICDM'02*.
47. N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In *Proceedings of ICDT'99*.
48. J. Pei and J. Han. Can we push more constraints into frequent pattern mining? In *Proceedings of ACM SIGKDD'00*.
49. J. Pei, J. Han, and L. V. S. Lakshmanan. Mining frequent item sets with convertible constraints. In *(Proceedings of ICDE'01)*.
50. J. Pei, X. Zhang, M. Cho, H. Wang, and P. Yu. Maple: A fast algorithm for maximal pattern-based clustering. In *Proceedings of ICDM'03*.
51. L. De Raedt. A logical database mining query language. In *Proceedings of ILP'00*.
52. S. Sarawagi, S. Thomas, and R. Agrawal. Integrating association rule mining with relational database systems: Alternatives and implications. In *Proceedings of the ACM SIGMOD'98*.
53. W. Shen and B. Leng. A Metapattern-Based Discovery Loop for Integrated Data Mining - Unsupervised Learning of Relational Patterns. *IEEE Trans. on Knowledge and Data Engineering*, 8(6):898–910, 1996.
54. W. Shen, K. Ong, B. Mitbander, and C. Zaniolo. Metaqueries for Data Mining. In *Advances in Knowledge Discovery and Data Mining*, pages 375–398. AAAI Press/The MIT Press, 1996.
55. A. P. J. M. Siebes and M. L. Kersten. Keso: Minimizing Database Interaction. In *Proceedings of KDD'97*.
56. R. Srikant, Q. Vu, and R. Agrawal. Mining association rules with item constraints. In *Proceedings of KDD'97*.
57. D. Tsur, J.D. Ullman, S. Abiteboul, C. Clifton, R. Motwani, S. Nestorov, and A. Rosenthal. Query flocks: A generalization of association-rule mining. In *Proceedings of ACM SIGMOD'98*.
58. M. L. Yiu and N. Mamoulis. Frequent-pattern based iterative projected clustering. In *Proceedings of ICDM'03*.

To See the Wood for the Trees: Mining Frequent Tree Patterns

Björn Bringmann

Lab for Machine Learning, Institute of Computer Science,
Albert-Ludwigs-University Freiburg,
Georges-Köhler-Allee 079, 79100 Freiburg, Germany
bbringma@informatik.uni-freiburg.de

Abstract. Various definitions and frameworks for discovering frequent trees in forests have been developed recently. At the heart of these frameworks lies the notion of matching, which determines if a pattern tree matches a tree in a data set. We compare four notions of tree matching for use in frequent tree mining and show how they are related to each other. Furthermore, we show how Zaki's TreeMinerV algorithm can be adapted to employ three of the four notions of tree matching. Experiments on synthetic and real world data highlight the differences between the matchings.

1 Introduction

In recent years, interest has grown in extending the frequent itemset paradigm to more expressive pattern types such as graphs, trees and sequences. Special attention has been devoted to semi-structured [1,2,3,4] and more specifically to tree-structured data [5,6,7]. These approaches aim at finding all frequent trees in a forest of rooted trees. They differ not only in the algorithms and implementation details, but more importantly also in the underlying notion of tree matching. When does one tree match another one? Asai *et al.* [5], Zaki [6] and Termier *et al.* [7] provide different answers to this question. Asai's notion is more restrictive than Zaki's, which is in turn more restrictive than Termier's. Termier *et al.* also have shown that it can be beneficial to work with more permissive notions of matching. However, this typically comes at a computational costs. Indeed, due to the expressiveness of their framework, Termier *et al.* cannot guarantee completeness, whereas the approaches of Zaki and Asai *et al.* are complete.

There are several important real-world applications for tree mining. First of all, consider the web usage mining problem [8]. Thousands of visitors maneuver through the well known web-sites like Amazon, Yahoo! and CNN each and every day. Most of these sites basically follow a hierarchical structure, i.e. a tree structure. Data Mining techniques created to handle trees can be used to gather information from the behavior of the visitors. The toy-example of an online shop shown in Figure (1) compares *tree embedding* and *tree incorporation* which we

J.-F. Boulicaut et al. (Eds.): Constraint-Based Mining, LNAI 3848, pp. 38–63, 2005.

will discuss in paragraph 2.2. While the latter, more expressive definition yields only one maximally specific pattern, the notion of tree embedding yields two. According to tree embedding some visitors looked at the blouse and some at the Fulgoni purse. The single maximally specific pattern according to the notion of tree incorporation offers some more information: *The visitors looking at the blouse and the visitors looking at the purse are the same persons.* This knowledge might be helpful when restructuring an online-shop to improve accessibility or placement of advertisements.

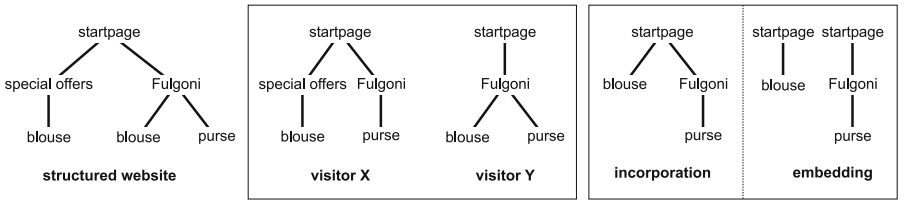


Fig. 1. Two *visitor* subtrees from a hierarchically structured web-site and the maximally specific patterns with regard to tree embedding and tree incorporation

XML has become a popular way of storing semi-structured data. As Goldfarb and Prescod write in their book [9]:

> *XML is a standardized notation for representing structured information. It is well-formed theoretically and is based on extensive industry experience. Although XML documents are simple, readily-transmitted character strings, the notation easily depicts a tree structure. A tree is a natural structure that is richer than a simple flat list, yet also respectful of cognitive and data processing requirements for economy and simplicity.*

XML data thus forms in general a 'source' for several important data mining domains. In bioinformatics tree structures arise as well. RNA structures essentially fold as trees. Newly sequenced RNA is compared with known RNA structures to draw conclusions about the functions of the RNA [10].

In this chapter, we compare the four notions *tree inclusion*, *tree embedding*, *tree incorporation*, and *tree subsumption* used in frequent tree mining and show how they are related to each other. Furthermore, we extend Zaki's TreeMinerV algorithm such that any of the first three named notions (i.e. all but *tree subsumption*) can be employed.

The chapter is organized as follows: Section 2 starts with definitions of trees and related objects. Using these, we formally define the four different notions of tree matching. In Theorem 1 we discuss the order among these notions. Section 2 ends with the definition of the tree-mining problem. In section 3, all concepts necessary for the tree-mining algorithm are described and the RETRO algorithm is explained. The following section discusses a novel pruning technique which reduces the memory consumption and time needed by the algorithm without

sacrificing any patterns. In section 5, we show two different ways to extract the maximally specific patterns from the set of frequent patterns found. The experiments in section 6 give insight into the performance of the algorithm employing different notions and pruning techniques as well as into the amount of patterns on real and artificial data. Finally, in sections 7 and 8 we touch upon related work and conclude.

2 Matching Trees

There exist several different matching notions for trees. All notions use a mapping function to match the nodes of one tree onto another tree adhering to several constraints. We will first define *trees* and several concepts regarding trees. Based on those concepts, we then define the four notions of matching we discuss in this chapter.

2.1 Trees

A *graph* $G = (V, E)$ is a set of *vertices* V (i.e. *nodes*), connected by *edges* $E \subseteq V \times V$ (i.e. links, arcs). The *order* of a graph is the number of its vertices $|V|$. If each edge is an ordered pair of vertices, the graph is a *directed graph*. A graph is *undirected* if each edge is an unordered pair of vertices. A graph can have labeled vertices, as well as labeled edges. We will denote a label on a vertex $v \in V$ or edge $e \in E$ with $\lambda(v)$ and $\lambda(e)$ respectively. A sequence of vertices such that each of its vertices has an edge to its successor vertex is called a *path*.

A *free tree* is a graph in which every pair of vertices is connected by *exactly* one path. In a *rooted tree*, the edges are directed (i.e. a rooted tree is a directed graph) and every node has exactly one incoming edge, except one designated node v_0 called *root*, which has no incoming edge. Nodes that have no outgoing edges are called *leaves*. Every node that is not a leaf is an *inner node*. In a rooted tree, a node c is called a *child* node of p if $(p, c) \in E$. Dually, p is called *parent* of c, denoted as $p = \pi(c)$. If there is a path from a node a to a node d, a is called an *ancestor* of d, and d is called a *descendant* of a. Hence, the root node of a tree is an ancestor of all other nodes in the tree. We use $\pi^*(d) =_{\text{def}} \{\pi(d)\} \cup \pi^*(\pi(d))$ to denote the set of all ancestors of a node d. For a tree with order k we write k-tree. In this work we concentrate onto *rooted trees*, *free trees* are not investigated. Hence, we simply use *tree* to denote a *rooted tree*. The child nodes of a node can be ordered. To denote the order from left to right, we use an operator $\prec$. If the child nodes of a node are ordered, the tree is called an *ordered rooted tree*. We can now define a formal language $\mathcal{L}$ composed of all possible labeled, ordered, rooted trees.

Furthermore, we need a notion of the *scope* of a node. This is a very useful notion for tree mining since the following proposition holds:

Proposition 1 (Scope). *Given the scope of a node a in a tree it can be checked in constant time if a second node b of the same tree is an ancestor, descendant, left or right sibling of a.*

The scope of a node n is an interval in $\mathbb{N}$. The nodes in a tree are indexed with their *preorder index*, i.e. they are enumerated in a depth first manner as depicted in Figure (2). Thus, the root node has the index 0 and its leftmost child is vertex 1. The rightmost descendant of the k-tree (i.e. the rightmost leaf) has the index $k-1$, where k is the number of nodes in the tree. Using a function $\gamma(x)$ that returns the index of node x, the scope $[x_l, x_r]$ of a node can be defined as:

$$x_l =_{\text{def}} \gamma(x) \quad \text{and} \quad x_r =_{\text{def}} \begin{cases} c_r & \text{if } c \text{ is the rightmost child of } x \\ \gamma(x) & \text{if } x \text{ is a leaf.} \end{cases} \tag{1}$$

An example for the depth-first enumeration and the scope-definition is shown in Figure (2).

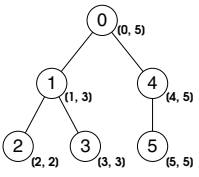


Fig. 2. A tree with its nodes labeled with their preorder index and each node annotated with its scope

2.2 Notions of Tree Matching

Previously [11], we presented three notions of tree matching and introduced also a novel one. In this section, we explain the differences and similarities between the four different notions in more detail.

First, all notions of *matching* map a tree $p = (V_p, E_p)$ onto another tree $t = (V_t, E_t)$, using a function $\varphi : V_p \to V_t$. In all four cases, the labels of the vertices are preserved: A vertex $v_p \in V_p$ can only be mapped to a vertex $v_t \in V_t$ if $\lambda(v_p) = \lambda(v_t)$, i.e. they both have the same labels.

The four notions have further similarities, but none is shared by all four of them. Figure (3) gives an example where the maximally specific patterns are

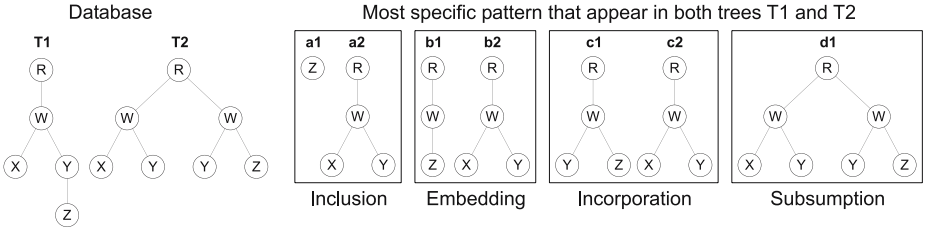


Fig. 3. Given a database consisting of the two trees $T1$ and $T2$ the four different notions yield four different sets of (maximally specific) patterns contained in both trees

different for all four notions. The most restrictive notion, called *tree inclusion*, states that a tree p is included in another tree t if there exists a subtree of t which is identical to p. It is defined as follows:

Definition 1. *Tree Inclusion*
A tree $p = (V_p, E_p)$ is included in a tree $t = (V_t, E_t)$, denoted as $match_{incl}(p, t)$, if there exists an injective mapping $\varphi : V_p \to V_t$ from the nodes of p to the nodes of t such that $\forall u, v \in V_p$

$$\begin{aligned}&\lambda(u) = \lambda(\varphi(u)) \wedge\\ &u \prec v \Leftrightarrow \varphi(u) \prec \varphi(v) \wedge\\ &\pi(u) = v \Leftrightarrow \pi(\varphi(u)) = \varphi(v).\end{aligned}$$

Tree inclusion has been extensively studied (in [12]) and can be decided in linear time. Asai *et al.* use this definition of matching in their algorithm FREQT. This notion might be to limited for several cases, but for other cases exactly this restrictiveness is required. Consider for example the representation of mathematical formulae as trees shown in Figure 4. In such a tree, each node corresponds to an operator or function and the leaves represent variables or numbers. Since the existence or absence of a single operation or function changes the whole meaning of the subtree, one would want patterns that preserve the parent-child relationship. Thus, more relaxed notions like *tree embedding* or *tree incorporation*, which allow to '*skip*' nodes in a pattern (and thus would extract A as a pattern for all three trees shown), are not useful here. Tree patterns in a set of formulae could be used to precalculate or optimize calculations that appear frequently.

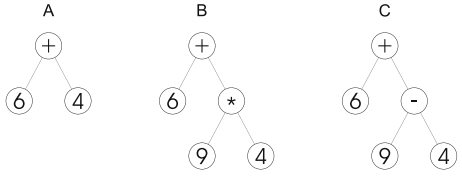


Fig. 4. In settings where a single node might influence the meaning of its whole subtree, tree inclusion will be the notion of choice

A more relaxed notion, called *tree embedding*, was first proposed in [13] and is based on an injective mapping preserving labels and ancestor-descendant relationships in the trees. In other words, we require that a parent-child relationship appears in the pattern p if and only if the two vertices are on the same path from the root to a leaf in the the tree t.

Definition 2. *Tree Embedding*
A tree $p = (V_p, E_p)$ is embedded in a tree $t = (V_t, E_t)$, denoted as $match_{emb}(p, t)$, if there exists an injective mapping $\varphi : V_p \to V_t$ from the nodes of p to the nodes of t such that $\forall u, v \in V_p$

$$\lambda(u) = \lambda(\varphi(u)) \wedge$$
$$u \prec v \Leftrightarrow \varphi(u) \prec \varphi(v) \wedge$$
$$v \in \pi^*(u) \Leftrightarrow \varphi(v) \in \pi^*(\varphi(u)).$$

Whereas the notion of tree inclusion is really useful when a node can change the meaning of its whole subtree, it will often be too restrictive if one deals with trees that contain some kind of additional or hierarchical information in the nodes which does not affect the meaning of nodes below. Consider for example a database of vehicles as in Figure 5, each represented as a hierarchical tree describing how the vehicle is composed of all its components. In such a case, trees may contain additional, more detailed information which is not available for all types of vehicles. Still, the make up of the components might be similar. Thus, a notion that allows to '*skip*' nodes can be very useful as shown in the example in Figure 5.

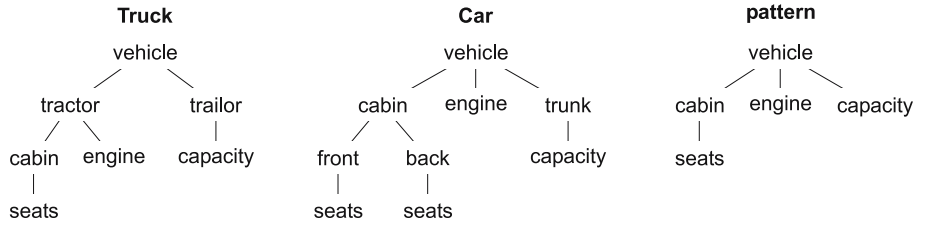


Fig. 5. Not enforcing the preservation of parent-child relationships allows to extract patterns that show more hidden similarities in the input data

Our definition, termed *tree incorporation*, is more relaxed than *tree embedding* since an ancestor-descendant relationship in the data does not have to hold in the pattern. Furthermore, it attempts to preserve the order among the children, but does not enforce it.

Definition 3. *Tree Incorporation*
A tree $p = (V_p, E_p)$ is incorporated in a tree $t = (V_t, E_t)$, denoted as $match_{icpr}(p, t)$, if there exists an injective mapping $\varphi : V_p \to V_t$ from the nodes of p to the the nodes of t such that $\forall u, v \in V_p$

$$\lambda(u) = \lambda(\varphi(u)) \wedge$$
$$u \prec v \Leftarrow \varphi(u) \prec \varphi(v) \wedge$$
$$v \in \pi^*(u) \Rightarrow \varphi(v) \in \pi^*(\varphi(u)).$$

The difference between *tree incorporation* and *tree embedding* can be described as follows. Let us consider two disjunct subsets X and Y of a set of trees $\mathcal{D}$ as shown in Figure (6), where all trees $x_i \in X$ and $y_j \in Y$ have two nodes labeled ψ and ϕ being descendants of a node labeled γ. Furthermore, in a tree $x_i \in X$, a node labelled ψ is an ancestor of the node labelled ϕ. For any tree of the other

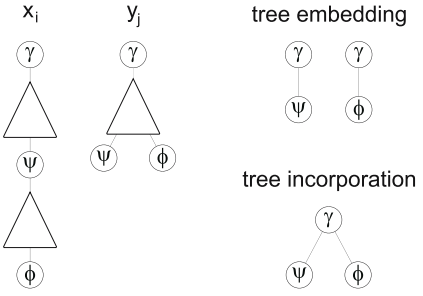


Fig. 6. The notion of *tree embedding* is more restrictive then *tree incorporation*, hence the latter one incorporates more information but less detail in the patterns found

subset Y, the nodes labelled ψ and ϕ are siblings, i.e. they have no ancestor-descendant relationship. Given an α such that $\max(|X|, |Y|) < \alpha \leq |X| + |Y|$, neither the pattern-tree where ψ is ancestor of ϕ nor the pattern-tree where ψ is a sibling of ϕ will be in at least α trees of the set $\mathcal{D}$. With regard to *tree embedding*, there are two tree-patterns with ψ being descendant of γ and ϕ being descendant of γ that match at least α trees of the set $\mathcal{D}$. In contrast, when using the notion of *tree incorporation*, there will be one tree-pattern where ψ and ϕ are siblings in at least α trees in $\mathcal{D}$. That way this result will show the information that the nodes labelled ψ and ϕ always appear in the same tree, while from the result using *tree embedding* this information cannot be obtained.

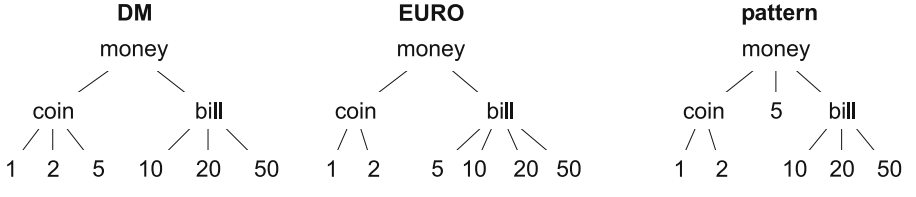


Fig. 7. The node labelled *5* denoting a bill and a coin respectively, appears as direct child of the root-node in the pattern

Furthermore, consider the example in Figure 7 where each tree represents the bills and coins of a currency. Both currencies have coins and bills with similar values. However, there is a 5 *Euro* bill but a 5 *DM* coin. Hence, the pattern that appears in both trees w.r.t. tree incorporation contains the node labelled *5* as child of the root node, since it can not be assigned to bill or coin in both cases. Using tree embedding, the pattern would or contain a *coin* and/or *bill* node or the node labelled *5* but never all three.

Finally, *tree subsumption* was introduced in [7]. It corresponds to representing the trees as relational formulae (cf. [14]), i.e. as a conjunction of all π^*, *edge* and *label* relations that hold in the tree. A tree then matches another tree if it θ-subsumes it. Theta-subsumption defined by Plotkin [15] is commonly employed

in the field of inductive logic programming (ILP, cf. [16]) and relational learning. Termier *et al.* [7] use this notion of tree subsumption in their TreeFinder algorithm.

Definition 4. *Tree Subsumption*
A tree $p = (V_p, E_p)$ is subsumed by a tree $t = (V_t, E_t)$, denoted as $match_{sub}(p, t)$, if there exists a mapping $\varphi : V_p \to V_t$ from the nodes of p to the nodes of t such that $\forall u, v \in V_p$

$$\lambda(u) = \lambda(\varphi(u)) \wedge$$
$$v \in \pi^*(u) \Rightarrow \varphi(v) \in \pi^*(\varphi(u)).$$

2.3 Order Among the Notions

The four notions given above are closely related. Indeed, the following theorem holds:

Theorem 1.

$$\begin{array}{l} \forall t, p \in \mathcal{L} \\ match_{incl}(p, t) \rightarrow match_{emb}(p, t) \rightarrow match_{icpr}(p, t) \rightarrow match_{sub}(p, t) \end{array} \quad (2)$$

Proof($match_{incl} \rightarrow match_{emb}$):
Since a parent node is also an ancestor node, i.e. $\pi(x) \in \pi^*(x)$, it follows that:

$$\begin{array}{l} \text{if } (v = \pi(u) \Leftrightarrow \varphi(v) = \pi(\varphi(u))) \\ \text{then } (v = \pi(u) \in \pi^*(u) \Leftrightarrow \varphi(v) = \pi(\varphi(u)) \in \pi^*(\varphi(u))). \end{array}$$

Hence, if a tree p is included in a tree t, it is also embedded in t. □

Proof ($match_{emb} \rightarrow match_{icpr}$):
Given $u \prec v \Leftrightarrow \varphi(u) \prec \varphi(v)$ and $v \in \pi^*(u) \Leftrightarrow \varphi(v) \in \pi^*(\varphi(u))$, it is obvious that also $u \prec v \Leftarrow \varphi(u) \prec \varphi(v)$ and $v \in \pi^*(u) \Rightarrow \varphi(v) \in \pi^*(\varphi(u))$ hold. Hence, a tree p embedded in a tree t is also incorporated in t. □

Proof ($match_{icpr} \rightarrow match_{sub}$):
Given an injective mapping φ from V_p to the nodes of V_t, such that $\forall u, v \in V_p$ $(\lambda(u) = \lambda(\varphi(u))) \wedge (u \prec v \Leftarrow \varphi(u) \prec \varphi(v)) \wedge (v \in \pi^*(u) \Rightarrow \varphi(v) \in \pi^*(\varphi(u)))$, it is obvious that only the part of the conditions required by *tree subsumption* will hold as well. Thus, if there exists a mapping φ such that a tree p is incorporated in a tree t it is also subsumed by t. □

2.4 Generality

On each of the matching notions, a more-general than (more-specific than) relation $\sqsubseteq$ ($\sqsupseteq$) can be defined. Given two trees $t, t' \in \mathcal{L}$ the relation can be defined as

$$t \sqsubseteq t' \Leftrightarrow \text{match}(t, t')$$

Hence, a tree t is called *more general* than a tree t', if and only if t is a pattern of t', according to the tree matching notion used. Corresponding to the four notions of matching there are four generality relations, i.e. for each $match_\chi$ there is a $\sqsubseteq_\chi$ with $\chi \in \{incl, emb, icpr, sub\}$. All four notions of generality induce a partial order on the pattern language $\mathcal{L}$. In contrast to the generality notion commonly used for frequent itemset mining[1], none of the four generality notions induces a lattice over $\mathcal{L}$. Figure (8) gives an example for this. In this example *two* elements S, S' are in the greatest lower bound of the trees T_1 and T_2 for any of the four notions. Since in a lattice the infimum (and supremum) are *unique*, this shows that no generality notion discussed here induces a lattice over $\mathcal{L}$.

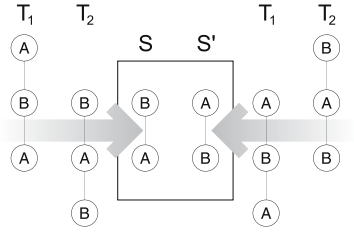


Fig. 8. An example that *none* of the four tree matching notions induces a lattice over $\mathcal{L}$. S and S' are maximally specific generalisations of the set $\{T_1, T_2\}$, i.e. there is no unique element that is the upper bound for this set.

2.5 Pattern Mining

After defining tree matching notions and therewith generality relations over $\mathcal{L}$, we can formalize the frequent tree mining problem. As stated before, let $\mathcal{L}$ be a formal language composed of all possible labeled, ordered, rooted trees and $\mathcal{D} \subseteq \mathcal{L}$ a database. To count the trees $t \in \mathcal{D}$ containing a pattern p with regard to a matching notion χ we define a function $d_{t,\chi} : \mathcal{L} \to \{0, 1\}$ as

$$d_{t,\chi}(p) =_{\text{def}} \begin{cases} 1 & \text{if } p \sqsubseteq_\chi t \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

that is 1 if the pattern p occurs at least once in the tree t, and 0 otherwise. The frequency of a pattern p in $\mathcal{D}$ can then be defined as

$$\sigma_{\mathcal{D},\chi}(p) =_{\text{def}} \Sigma_{t \in \mathcal{D}} d_{t,\chi}(p) \tag{4}$$

Using this definition, we now can define the task of *frequent tree mining*:

> Given a set of trees $\mathcal{D}$ and a minimum frequency α, the task of tree-mining is to find all patterns $p \in \mathcal{L}$ with regard to a matching notion χ such that $\sigma_{\mathcal{D},\chi}(p) \geq \alpha$ holds.

[1] In frequent itemset mining as introduced in [17], a pattern t is a set of items, and t is more general than t' if and only if $t \subseteq t'$.

All four matching notions can be used for searching patterns that occur with a minimum frequency.

Definition 5. *Antimonotonicity*
A constraint *or* selection predicate $q(p)$ *(such as minimum frequency* $\sigma_{\mathcal{D},\chi}(p) \geq \alpha$*) is* antimonotone *with regard to the specialization relation* $\sqsubseteq$ *if and only if*

$$\forall g \sqsubseteq s \Rightarrow (q(s) \Rightarrow q(g)). \tag{5}$$

The minimum-frequency predicate as defined above is antimonotone with regard to the specialization-relation $\sqsubseteq$, since for any pattern $g \sqsubseteq_\chi s$ it holds that $\sigma_{\mathcal{D},\chi}(g) \geq \sigma_{\mathcal{D},\chi}(s)$. Considering the framework introduced by Mannila and Toivonen [18], the theory of a database $\mathcal{D}$ with regard to a matching notion χ and a minimum frequency α can be defined as:

$$\mathcal{T}h(\mathcal{L}, \mathcal{D}, (\alpha, \chi)) = \{p \in \mathcal{L} \mid \sigma_{\mathcal{D},\chi}(p) \geq \alpha\}$$

As a consequence of Theorem 1, the set of frequent trees with regard to a data set $\mathcal{D}$ for *tree inclusion* is smaller than that for *tree embedding*, which is in turn smaller than that for *tree incorporation*. The set of frequent trees with regard to *tree incorporation* finally is smaller than that induced by *tree subsumption*. This motivates the use of the notion of matching as a parameter of frequent tree discovery tasks.

As pointed out by Mannila and Toivonen [18], a set $\mathcal{T}h \subseteq \mathcal{L}$ can be described by giving just the positive or the negative border. This helps to reduce the size for the representation of the solution. While the complete theory $\mathcal{T}h$ may contain thousands of patterns, the borders often contain only a small fraction of the patterns when compared to the whole version space. The *maximally specific set* $\mathbb{S}$, which is the same as the positive border ($\mathcal{B}d^+$), is defined as follows:

Definition 6. *Maximally Specific Set* - $\mathbb{S}$

$$\mathbb{S} =_{\text{def}} \{p \in \mathcal{T}h | \forall p' \in \mathcal{L} : p \sqsubset p' \Rightarrow p' \notin \mathcal{T}h\} \tag{6}$$

Minimum support is only one of several constraints that can be used to search for patterns occurring in a database. As long as the constraint is *antimonotone* with regard to the *more-general* relation, the resulting $\mathcal{T}h$ can be represented by its maximally specific set only. Hence, one could define constraints on tree-patterns like a maximum number of vertices, a maximum depth, or a maximum branching factor in a pattern. In addition to this extension, one can also use the generality relation $\sqsubseteq$ to define additional constraints. That will result in constraints like *maximally specific patterns*. I.e., given a set of maximally specific patterns $\mathbb{M}\text{ax} \subset \mathcal{L}$ there will be only patterns p in the result set such that $\forall m \in \mathbb{M}\text{ax} : p \sqsubseteq m$. Please note that the set $\mathbb{M}\text{ax}$ cannot always be reduced to just one tree, since the generality relation does *not* induce a lattice over $\mathcal{L}$.

Even further, *monotone constraints* could be used. I.e., a maximum support could be defined similar to the minimum support or a set of *maximally general patterns*. Similar to the *antimonotonicity* of constraints the *monotonicity* is defined.

Definition 7. *Monotonicity*
A constraint $q(p)$ *is* monotone *with regard to the specialization relation* $\sqsubseteq$ *if and only if*

$$\forall g \sqsubseteq s \Rightarrow (q(g) \Rightarrow q(s)). \tag{7}$$

If monotone constraints are used the corresponding border must be given to represent the resulting set $\mathcal{T}h$ in a compact way, i.e. only by its borders. This set $\mathbb{G}$ of *maximally general patterns* is defined analogue to the set $\mathbb{S}$ of *maximally specific patterns*.

Definition 8. *Maximally General Set* - $\mathbb{G}$

$$\mathbb{G} =_{\text{def}} \{p \in \mathcal{T}h | \forall p' \in \mathcal{L} : p' \sqsubset p \Rightarrow p' \notin \mathcal{T}h\} \tag{8}$$

Apart from the *maximum* and *minimum frequency* constraints, none of the constraints mentioned above requires to query the database. Following the notion of Ng *et al.* [19], constraints that do not require to query the database are called *domain constraints*. In the context of constraint-based mining one can see the notion of matching employed as providing a *constraint*. This type of constraint is comparable to the so-called *class constraints* mentioned by Ng *et al.*

Below we focus on the well known minimum frequency. Most of the other constraints can be treated in a similar way.

3 Mining Trees

Algorithms for mining frequent trees with regard to all four notions of tree matching exists. The algorithms for tree inclusion FreqT [5] and tree embedding TreeMinerV [6] work in a levelwise manner. Possible pattern trees are generated by extending known frequent trees with an additional node. After this extension, the supports for the new pattern trees are counted. The algorithm for mining incorporated trees RETRO [11] is an extension to the TreeMinerV algorithm for embedded trees. Thus, it works similarly, searching for frequent patterns in a levelwise manner.

However, the algorithm TreeFinder by Termier *et al.* [7] for mining frequent tree patterns is quite different. First, the trees are represented as relational formulae similar to representing trees within the Inductive Logic Programming (ILP) framework [16]. For every edge there is a binary predicate named after the labels of the nodes that are connected. The two arguments are unique identifiers of the connected nodes. After constructing all edge-predicates, their transitive closure is calculated. The algorithm then searches for all frequent ancestor-descendant relationships and clusters them. The resulting ancestor-descendant sets are re-transformed into edges from which a tree is constructed. For an in-depth explanation we refer the reader to [7].

The algorithm FreqT for searching included trees is not further described here, since the TreeMinerV algorithm will be extended in a way such that it can be used for mining frequent trees with regard to tree inclusion, tree embedding, and tree incorporation.

3.1 Systematically Enumerating Candidate Tree Patterns

As stated before, the algorithms for tree inclusion (FreqT), tree embedding (TreeMiner), and tree incorporation (RETRO) work in a levelwise manner. All three use a method called *rightmost expansion* to canonically enumerate all labeled, ordered, rooted trees. This technique was independently proposed by Zaki [6] and Asai *et al.* [5]. It works in a levelwise manner, adding a single node to a known frequent pattern in such a way that every possible candidate pattern will be generated *exactly* once. Thus the *rightmost expansion* is an optimal refinement operator, since every tree is enumerated but no tree is enumerated several times. Basically a k-tree is expanded to several $k+1$-trees P_i by adding new nodes only to its rightmost path as shown in Figure 9. The new node v_{k+1} in a pattern P_i is called *rightmost leaf* (RML) and the subtree without its RML is called *prefix* of the tree, denoted as $[P_i]$. The *rightmost path* of a tree is the path from the root node to the rightmost leaf. For efficient candidate generation the antimonotonicity of frequent patterns is used (i.e. a specialization s of a pattern p is not more frequent than p). Thus, we consider only frequent patterns for extension.

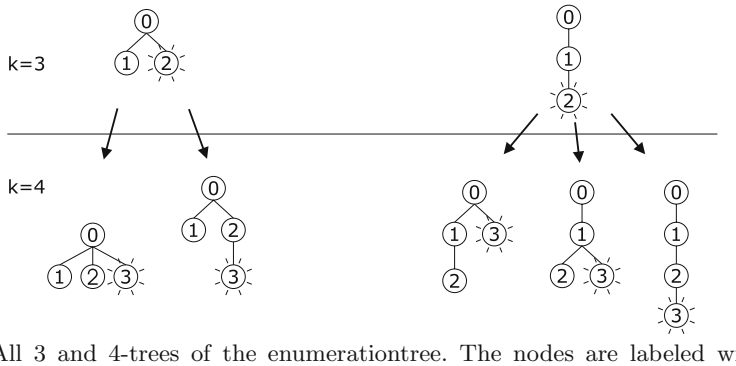


Fig. 9. All 3 and 4-trees of the enumerationtree. The nodes are labeled with their preorder index. New nodes attached at each level are marked.

3.2 Equivalence Classes and Instance Lists

With focus on the rightmost extension, patterns are organized in so called *equivalence classes* (EQ). An equivalence class contains all patterns that have the same prefix, i.e. differ in their rightmost leaves only. Each equivalence class contains the prefix $[P_i]$ only once and for each pattern P_i a tuple $\langle \lambda(\texttt{RML}_i), \gamma(\pi(\texttt{RML}_i)) \rangle$ that contains the label $\lambda(\texttt{RML}_i)$ of the new node and the index of the node it is attached to. If the RML is the root node of the pattern (i.e. the prefix is empty), this is denoted by a tuple $\langle \lambda(\texttt{RML}_i), -1 \rangle$. Since *prefix* and *equivalence class* denote essentially the same concept, $[P]$ is used to denote both, the prefix of a pattern P and the equivalence class that contains all patterns with the same prefix $[P]$. If there are no ambiguities, we use $\langle \lambda, \gamma \rangle$ to refer to a pattern P_i in $[P_i]$. With regard to the definition of *tree incorporation*, a pattern $\langle \lambda, j \rangle$ is a specialization of a pattern $\langle \lambda, i \rangle$ if $j > i$ and both patterns belong to the same equivalence

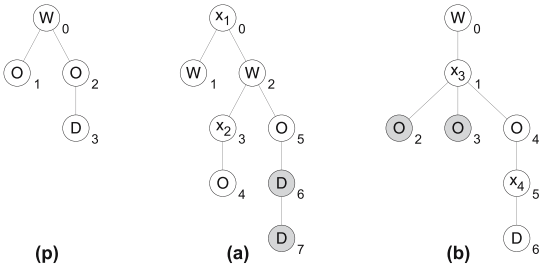


Fig. 10. A sample for four different instances of pattern p incorporated in a database consisting of two trees a and b. Both trees contain the pattern twice. In a the rightmost leaves of both instances are different, whereas in tree b their prefix is different. The grey marked nodes are the reason that there are two instances in each of the trees.

class. Hence, a function φ exists that maps the nodes of $\langle \lambda, i \rangle$ to the nodes of $\langle \lambda, j \rangle$ with respect to Definition 3. For example, for the two left-hand 4-trees in Figure (9), the right one is a specialization of the left one, but not vice versa.

To efficiently count the support of a pattern, the algorithm needs information about the instances in the data that support this pattern. Let pattern X be a k-subtree occurring in a tree T, φ the mapping from the pattern-nodes to the nodes of T, and x_k refer to the rightmost leaf of X. Following Zaki, we use $\mathcal{I}(X)$ to refer to the *instance-list* (also known as *scope-list*) of X. Each element of $\mathcal{I}(X)$ is a triple $\langle t, s, m \rangle$ identifying an instance of X where t is the identifier of the tree T the pattern X occurs in, $m =_{\text{def}} \{\gamma(\varphi(x_0)), \gamma(\varphi(x_1)), \ldots, \gamma(\varphi(x_{k-1}))\}$ is a list called *match label* of the prefix of X, and s is the scope of the node $\varphi(x_k)$, which the rightmost leaf of the pattern is mapped to. These instance lists contain all instances of a pattern with regard to *tree embedding*. An example is shown in Figure (10). Here the instance list for the pattern p is

$$\begin{aligned} \mathcal{I}(p) = \{\ & \langle a,\ [6,7], (2,4,5) \rangle, \\ & \langle a,\ [7,7], (2,4,5) \rangle, \\ & \langle b,\ [6,6], (0,2,4) \rangle, \\ & \langle b,\ [6,6], (0,3,4) \rangle\ \} \end{aligned}$$

For *tree incorporation*, we need the notion of *extended instance lists*. As stated before, a pattern $\langle \lambda, j \rangle$ is a specialization of a pattern $\langle \lambda, i \rangle$ if $j > i$ and both patterns belong to the same equivalence class. Hence, every instance that supports a pattern $\langle \lambda, j \rangle$ is also an instance that supports the pattern $\langle \lambda, i \rangle$. Using this information, the definition of an *extended instance list*, containing all instances that support a pattern X, is as follows:

$$\mathcal{I}^*(X) = \mathcal{I}^*(\langle \lambda, i \rangle) =_{\text{def}} \cup_{j \geq i} \mathcal{I}(\langle \lambda, j \rangle) \tag{9}$$

Given an equivalence class $[P]$, we use Zaki's *class extension* to obtain equivalence classes containing the successors of the patterns in $[P]$ with regard to the canonical enumeration scheme. The main idea is to consider each pair of patterns

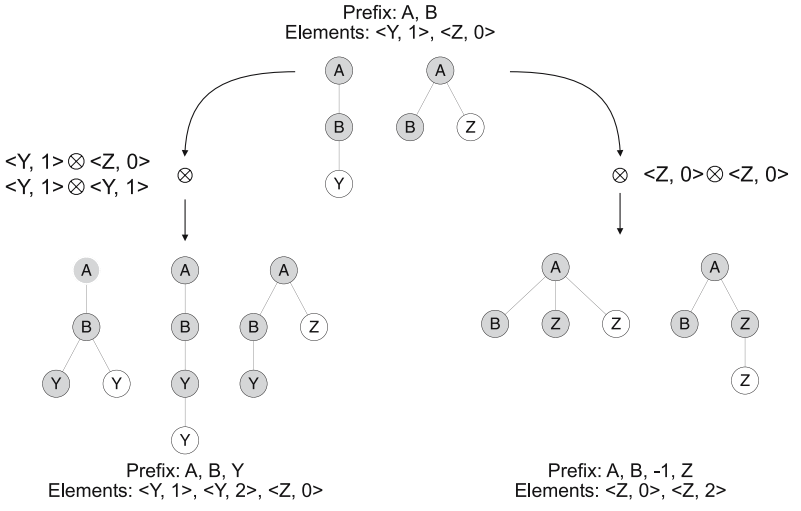


Fig. 11. An example for class extension. A class with two patterns is extended to obtain all its canonical successors. The grey nodes of each pattern represent the prefix common to all patterns in the equivalence class.

in the class for extension, including self extension. There can be up to two new candidates for each pair of patterns to be joined. Zaki's Theorem [6] formalizes this notion:

Theorem 2. *(Class Extension)*
Let $[P]$ *be an equivalence class and let* $\langle x, i\rangle$ *and* $\langle y, j\rangle$ *denote any two elements in the class. Let* $[P_x]$ *denote the class representing extensions of element* $\langle x, i\rangle$*. We define a join operator* $\otimes$ *on the two elements, denoted* $\langle x, i\rangle \otimes \langle y, j\rangle$*, as follows:*

$$\langle x, i\rangle \otimes \langle y, j\rangle =_{\text{def}} \begin{cases} \{\langle y, 0\rangle\} & \text{if} \quad i = j = -1 \\ \{\langle y, j\rangle, \langle y, \gamma(x)\rangle\} & \text{if} \quad i = j > -1 \\ \{\langle y, j\rangle\} & \text{if} \quad i > j \\ \{\} & \text{otherwise } (i < j) \end{cases} \tag{10}$$

Then all possible $(k+1)$*-subtrees in* $[P_x]$ *with the prefix* $[P]$ *will be enumerated by applying the join operator to each unordered pair of elements* $\langle x, i\rangle$ *and* $\langle y, j\rangle$*.*

As formalized in the join operation, there can be up to two outcomes of a join of two patterns. In Figure (11), showing the application of the join operator to an equivalence class consisting of two patterns, the self join of $\langle Z, 0\rangle$ results in two new patterns.

When two patterns A and B of the same equivalence class are joined, their instance lists are joined to obtain the instances that support the resulting patterns of the join. The corresponding operation is denoted $\cap_{\otimes}$. Let $R = \langle x, i\rangle \otimes \langle y, j\rangle$ denote the resulting set of patterns of the join. As stated before, there can be at most two elements (i.e. patterns) in R. In one of the elements, $\ddot{r}$, the node with

label y is a sibling of x and in the other element, $\dot{r}$, y is a child of x. Either one of the two elements can be in R. Similarly, the join of the instance lists results in at most two new instance lists, one for each element in R. When joining the instance lists $\mathcal{I}(X)$ and $\mathcal{I}(Y)$ of two patterns X and Y, all pairs $x = \langle t_x, s_x, m_x \rangle \in X$ and $y = \langle t_y, s_y, m_y \rangle \in Y$ of instances are considered. For two instances to be joined, i.e. recombined to a new instance, several conditions have to hold. First, both instances have to appear in the same tree. Second, both instances have to be extensions of the same prefix occurrence. Finally, the scopes $s_x = [l_x, u_x]$ and $s_y = [l_y, u_y]$ of the rightmost leaves of the instances have to be considered. If s_y is contained in s_x, the rightmost leaf of y is a descendant of the rightmost leaf of x. In this case we have a new instance for the pattern $\dot{r}$. If the rightmost leaf of y is a sibling (to the right) of the rightmost leaf of x we have a new instance for $\ddot{r}$. The conditions for a new instance for a pattern $\ddot{r}$ are called *out-scope* test, whereas the conditions for a pattern $\dot{r}$ are called *in-scope* test.

Formally speaking, the *out-scope* test and the *in-scope* test for two instances $x = \langle t_x, s_x, m_x \rangle$ and $y = \langle t_y, s_y, m_y \rangle$ (with $s_x = [l_x, u_x]$ and $s_y = [l_y, u_y]$) are defined as follows:

Definition 9. *In-Scope Test*
Given two instances $x = \langle t_x, s_x, m_x \rangle$ and $y = \langle t_y, s_y, m_y \rangle$ we say that y is in the scope of x if the following conditions hold:

1. $t_x = t_y$
2. $m_x = m_y$
3. $l_x < l_y \wedge u_x \geq u_y$

Definition 10. *Out-Scope Test*
Given two instances $x = \langle t_x, s_x, m_x \rangle$ and $y = \langle t_y, s_y, m_y \rangle$ we say that y is outscope of x if the following conditions hold:

1. $t_x = t_y$
2. $m_x = m_y$
3. $u_x < l_y$

If a new instance z is added to either $\ddot{r}$ or $\dot{r}$, it is composed by combining the instances x and y such that $z =_{\text{def}} \langle t_y, s_y, m_y \oplus \gamma(\texttt{RML}(x)) \rangle$, where the operator $\oplus$ adds a new element to the end of the list. Thus, the node the rightmost leaf of x refers to is now part of the match of the new instance.

This notion works for *embedded* trees. For *incorporated* as well as for *included* trees, some minor changes and extensions have to be made.

For *tree incorporation* we have already introduced *extended instance lists*. These lists contain every instance supporting a pattern with regard to *tree incorporation*. Furthermore the conditions of an *out-scope* test have to be modified such that condition 3 reads:

$$u_x < l_y \vee (l_y < l_x \wedge u_y \geq u_x).$$

That way, an *out-scope* test holds if the node the RML of y is mapped to is a right-sibling *or* an ancestor of the node the RML of x is mapped to.

For *tree inclusion* there are no ancestor-descendant relationships allowed, but only parent-child relationships. Fortunately, this additional constraint can be incorporated into the algorithm in the following way. The prefix of a tree, i.e. the part of the pattern that is common to all patterns in the same equivalence class is not changed when a pattern is extended. So we have to make sure that every pattern prefix is consistent with the parent-child constraint, such that only the rightmost leaves are allowed to be in an ancestor-descendant relationship[2]. To achieve the parent-child consistency in the prefix, an extension $\langle x, i\rangle \otimes \langle y, j\rangle$ of a pattern $\langle x, i\rangle$ is permitted only if the rightmost leaf of $\langle x, i\rangle$ is a proper child of its parent in the instance both nodes are mapped to. To do that, we keep track of the preorder indices of the nodes in the match-part of a pattern instance. To refer to a single node in the match $m = \{\gamma(\varphi(x_0)), \gamma(\varphi(x_1)), \ldots, \gamma(\varphi(x_n))\}$, we use $m_{[i]} = \gamma(\varphi(x_i))$. Please note that the elements $m_{[i]}$ in the match m of an instance are ordered with regard to the prefix order of the nodes x_i in the pattern tree X , i.e. $m_{[i]} = \gamma(\varphi(x_i))$. In addition, the notion of an instance is changed to a quadruple $\langle t, s, m, p\rangle$ where p is the preorder index of the parent node of the rightmost leaf in the instance. The other three parts remain as before, i.e. t is the identifier of the tree, s is the scope of the rightmost leaf of the instance, and m is the match of the instance. The *in-scope* and *out-scope* tests are modified such that there is an additional condition 4 for both cases which requires:

$$m_{[k]} = p$$

where k is the number of nodes in the prefix, i.e. the number of elements in the match m.

3.3 The RETRO Algorithm

As stated before, the RETRO (**Fre**quent **Tr**ee Disc**o**very) algorithm is a modification of Zaki's TreeMinerV algorithm. The main differences are the usage of the extended instance lists and the new condition for the *out-scope* test. The algorithm for computing frequent patterns with regard to tree incorporation is shown in Figure (12).

The first part of the algorithm computes the sets containing all frequent 1-trees (i.e. nodes) and 2-trees. Then the main loop starts by calling the function *Enumerate-Frequent-Subtrees* for every frequent 2-tree. The function *Enumerate-Frequent-Subtrees* generates all possible refinements of patterns in an EQ $[P]$. This is done by joining every pair $\langle x, i\rangle \otimes \langle y, j\rangle$ of patterns in $[P]$ including self-joins. Due to the rightmost expansion it is not allowed to join $\langle x, i\rangle \otimes \langle y, j\rangle$ with $i < j$ which would result in non-canonical expansions. A join results in one or two new patterns (R). Afterwards the respective instance lists are created by joining the instance lists of the patterns $\langle x, i\rangle$ and $\langle y, j\rangle$. Any new pattern that turns out to be frequent is added to the new equivalence class $[P_x]$. If all frequent

[2] This is imposed by the design of the algorithm.

FreQUENTTreeDiscoVERY($\mathcal{D}$, *minsup*):
  F_1 = { frequent 1-subtrees };
  F_2 = { classes $[P]_1$ of frequent 2-subtrees };
  for all $[P]_1 \in F_2$ **do**
    Enumerate-Frequent-Subtrees($[P]_1$);

Enumerate-Frequent-Subtrees($[P]$):
  for each element $(x, i) \in [P]$ **do**
    $[P_x] = \emptyset$
    for each element $(y, j) \in [P]$ with $i \geq j$ **do**
      $R = \{(x, i) \otimes (y, j)\}$;
      $\mathcal{I}(R) = \{\mathcal{I}^*(x, i) \cap_\otimes \mathcal{I}^*(y, j)\}$;
      if for any $p \in R$, p is frequent **then** $[P_x] = [P_x] \cup \{p\}$;
    Enumerate-Frequent-Subtrees($[P_x]$);

Fig. 12. TreeMining Algorithm

patterns of the new equivalence class $[P_x]$ are computed further refinements of these patterns are generated. Thus, the algorithm proceeds depth-first.

The Figure (13) shows one path of the enumeration tree when the algorithm is applied to the database consisting of the two trees $(x), (y)$. The extended instance lists are required explicitly in refinement step 3. Without the extended instance lists the algorithm would not refine further, hence it would not reach the incorporated pattern p.

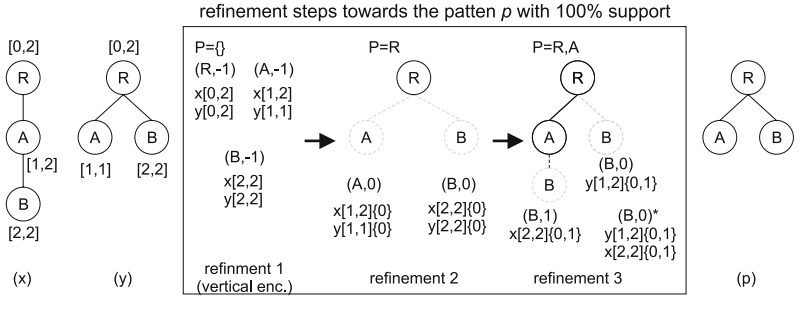


Fig. 13. An example for a tree pattern search with the notion of tree incorporation. Only the search path leading to the most specific pattern (p) in both trees $(x), (y)$ is shown. The shown pattern is *not* valid with regard to tree embedding where two different most specific patterns would be discovered.

4 Instance Pruning

Next to well-known pruning techniques like *node pruning* and *edge pruning* [5], we introduce a novel technique called *instance pruning* (*IP*) that reduces the

average computation time by 50%. It is not only applicable to the algorithm working on the novel pattern definition, but also to the TreeMinerV algorithm.

As stated before, $d_{t,\chi}(X)$ returns a 1 if there is at least one occurrence of pattern X in tree t, otherwise it returns 0. Hence, the frequency of a pattern depends only partly on the number of instances. The idea for *IP* is to keep only a subset of the instances necessary to discover all frequent patterns. If there are different instances of a pattern $\langle x, i\rangle$ in tree t, they are represented in the instance list $\mathcal{I}\langle x, i\rangle$ as $I_{1,0} = \langle t, a_0, s\rangle$ and $I_{2,0} = \langle t, b_0, r\rangle$. If the pattern

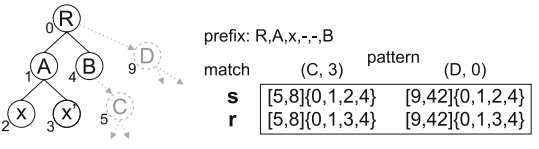


Fig. 14. One of the groups with the match label $s = \{0, 1, 2, 4\}$ or $r = \{0, 1, 3, 4\}$ can be removed, since both would yield the same result in future joins

$\langle x, i\rangle$ is joined with another pattern $\langle y, j\rangle$, all instances in I_1 and I_2 will be joined with the respective instances of (y, j). Consider two groups of instances $\langle t, a_0, s\rangle, \ldots, \langle t, a_m, s\rangle$ and $\langle t, b_0, r\rangle, \ldots, \langle t, b_n, r\rangle$ of the tree t with match labels s and r as shown in figure 14. If for every triple $\langle t, b_k, r\rangle$ there exists a triple $\langle t, a_l, s\rangle$ with $a_l = b_k$, all triples with the match label r can be removed from the `EQ`. This is possible, as for instances in the same tree with the same match label only the nodes a_l (or b_k) are of relevance for the extension of the instances. If a match label s includes all nodes b_k of a match label r, no instance can be created out of instances with match label r that cannot be created out of instances with match label s. This decrease in the number of instances can effectively reduce the memory consumption of the process. More importantly, it lowers computation time. Not only the removed instances themselves are not joined anymore, but also the ones that would have been created by joining them. For databases, with a low number of labels when compared to the number of nodes, *IP* can reduce the computation time by up to 80%.

5 Towards the $\mathbb{S}$-Set

As stated before, the pattern mining algorithm uses a canonical enumeration scheme for labeled, ordered, rooted trees. Due to this scheme, not every specialization s is created as a refinement of a more general pattern g. During the mining process this restriction is very useful, since it assures that no pattern is generated twice, i.e. it avoids redundancy. However, if we are only interested in the maximally specific patterns, the enumeration strategy gives rise to a problem:

> If a pattern cannot be refined further using the enumeration strategy, that does not imply that there is no further valid specialization.

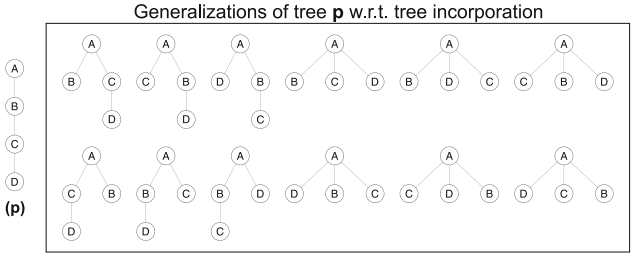


Fig. 15. Altogether there are 35 generalizations (including the empty tree) for the tree (p) on the left. The Figure shows only the 12 4-trees.

We focus on the most specific patterns only, i.e. the set of patterns $\mathbb{S}$ where every $s \in \mathbb{S}$ is frequent, but there is no specialization $s' \sqsupset s$ such that s' is frequent. A run of the algorithm, cf. Figure (12), yields all patterns that are found to be frequent during the search process. For the notions match_{incl} and match_{emb} these are *all* frequent patterns. Since the amount of frequent patterns with regard to the notion of tree incorporation is very high, (cf. Figure (15)) the algorithm focuses on the set $\mathbb{S}$ only. Hence, not all frequent patterns are generated during the search. Obviously, every pattern that can be refined further cannot be part of the set $\mathbb{S}$. But due to the canonical enumeration scheme, not every pattern that cannot be refined further is a maximally specific pattern. An example is shown in Figure (16). Hence, it is necessary to check if there exists a possible extension, whether it is canonical or not. Below we describe two ways to solve this problem.

First, the set $\mathbb{S}^+$ containing all patterns that could not be further refined by the algorithm, which is a superset of the desired set $\mathbb{S}$, can be filtered in a

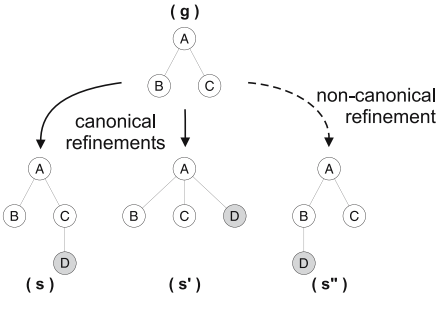


Fig. 16. The trees s, s', and s'' are specializations of the tree g. Whereas s and s' are canonical refinements which are generated by the rightmost expansion, s'' is not generated from g by rightmost expansion. Assume that g and s'' are frequent but neither s nor s' are frequent, there exists no frequent canonical refinement of g but there is a frequent specialization for g: s''. I.e., g would not be a maximally specific pattern.

post processing step. This can be done by applying the algorithm on each pair $s_1, s_2 \in \mathbb{S}^+, s_1 \neq s_2$ to search for patterns that appear in both trees, i.e. that have a frequency of 100%. If there is a pattern p that is equal to s_1 and appears in s_2 we know that $s_1 \sqsubseteq s_2$. Hence, s_2 is a specialization of s_1 and thus s_1 does not belong to the set $\mathbb{S}$. After checking every pair of pattern-trees[3] the set $\mathbb{S}$ is obtained. This method can be used for all three matchings tree inclusion, embedding, and incorporation. Unfortunately, the time-complexity is at $O(|\mathbb{S}|^2)$.

Fortunately, there is another way to discover if further specializations s' of a pattern s exist. Before inserting a frequent pattern s that is not further refinable into the solution set $\mathbb{S}$ it has to be checked if there exists no frequent specialization s' of s. For this, *non-canonical* expansions of the pattern s have to be considered as well.

During the search process, the algorithm traces instances of the patterns found. This trace-data can now be used to search for possible non-canonical expansions of patterns that cannot be generated with regard to the canonical enumeration scheme.

There are three possible specializations to consider: (1) **A new root node**, i.e. adding a parent node of the current root node to the pattern. (2) **A new child node**, i.e. adding a child node of the current root node to the pattern. (3) **Descent of a node**, i.e. specializing a pattern by moving a certain node downwards.

The first specialization, which can basically be described as adding a new node *above* the current pattern-root node is straight forward. For each tree t, we select the node r the pattern's root node is mapped to. We traverse the path starting from r to the root of t upwards and count the unique labels of the nodes seen. If any label's count reaches the minimum frequency, the search can be aborted: there exists a possible specialization of the considered pattern adding a new root node.

In the second case, where a new node is added *below* the pattern-root to generate a frequent non-canonical specialization, the process is more complicated. Since for every pattern p with a node n as some descendant of the root node there exists a more general pattern p with n as direct child of the root node, only expansions with new nodes as direct children of the root node have to be considered. Hence, there are as many possible positions (called *bins*) for a new node as the current pattern has direct children of the root node. expansion with a new node as the rightmost child of the root node, since this is already covered by the canonical expansion. Every node n that is not already part of the pattern could appear in a *bin* to the left as well as to the right of a direct child c of the root node, if n and c are on the same path to the root node. If a node n does not lie on a path with any direct child c of the root node, it could appear only in one *bin* between two existing children c_n, c_{n+1} of the root node (cf. node B in Figure 17). In general in a tree a node n lies on the same path as every ancestor of n and on the same path as every node in the subtree of n. Figure (17) shows a pattern and an instance of the pattern in a data tree. For non-canonical expansions there

[3] Of course, patterns that turn out to have a specialization are not reconsidered.

are two positions x_1, x_2 for a new node. The node D that is a descendant of A could appear at both, whereas the nodes B and E could only appear at position x_2. The second possible position for node E would be a canonical expansion and thus is not considered here. To find possible non-canonical expansions, the algorithm traverses the subtree of each instance root and counts the possible labels for the *bins*. If any label reaches the minimum support, we know that there is a frequent specialization of the pattern.

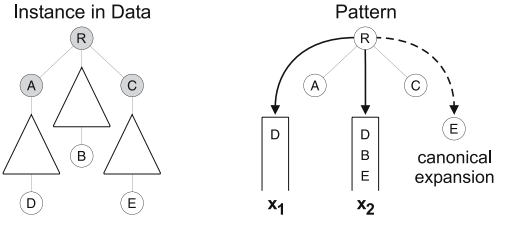


Fig. 17. Searching for non-canonical expansions the algorithm puts every node that is not part of the instance in up to two bins. Each bin is located on the left of each child of the root node. for both bins next to the corresponding between the 'root-childs' where they are located.

In the third and last case, no new node is added to the pattern. Instead, specializations of the pattern are considered where the nodes are moved further down in the tree. For this, every pair of nodes n_1, n_2 in the pattern is considered where n_1 is a sibling of n_2. The algorithm checks if there is an ancestor-descendant relationship in the data between the nodes n_1 and n_2 are mapped to. If the number of trees in which n_1 is an ancestor of n_2 is equal to or larger then the minimum frequency, the pattern considered can be specialized further. I.e., it is not maximally specific.

If none of the three specializations is possible, the pattern is maximally specific and can be added to the set $\mathbb{S}$.

5.1 Cardinality of Maximally Specific Sets

Due to Theorem 1, the resulting pattern space with regard to tree incorporation contains more patterns than of tree embedding, which in turn contains more patterns than that of tree inclusion. In contrast to the whole pattern space, there is no such relation among the $\mathbb{S}$ sets, i.e. the sets containing the most specific patterns only. Two examples are given in Figure (18) and Figure (19). The first shows that with the given database and a minimum frequency of two there are five maximally specific patterns for *tree inclusion* and six for *tree embedding*. With a minimum frequency of three, there are still five maximally specific patterns for *tree inclusion*, but only four for *tree embedding*. The example for *tree embedding* and *tree incorporation* in Figure (19) is similar. In the first case, there are four maximally specific patterns with regard to *tree embedding* and six for

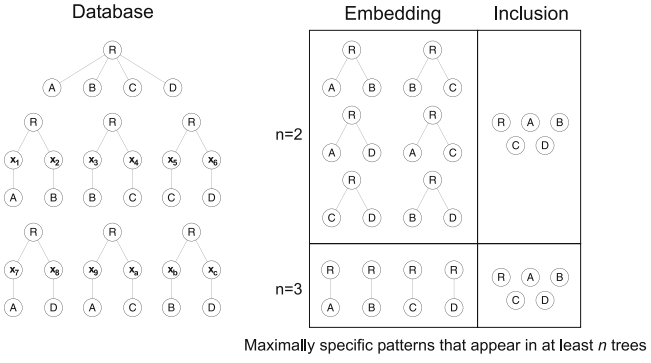


Fig. 18. A database and maximally specific patterns w.r.t. *tree inclusion* and *tree embedding* for a minimum support of n=2 and n=3 each. For n=2 there are more maximally specific patterns for embedding while for $n = 3$ there are more w.r.t. inclusion.

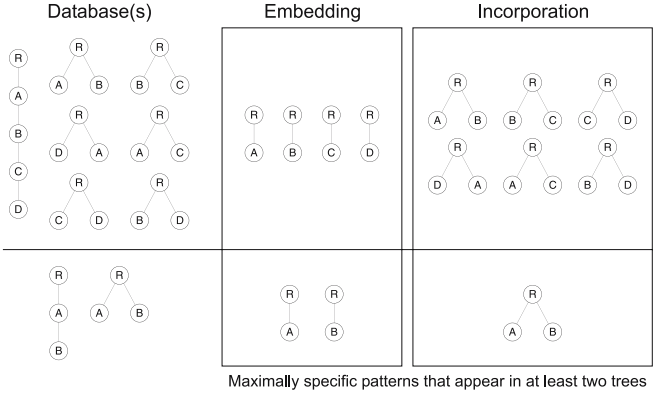


Fig. 19. Two databases and maximally specific patterns w.r.t. *tree embedding* and *tree incorporation* for a minimum support of $n = 2$ in both cases. In the upper case there are more maximally specific patterns for incorporation while for the database on the bottom there are more w.r.t. embedding.

tree incorporation. In the second case, there are more for *tree embedding* than for *tree incorporation*.

6 Experimental Results

A number of experiments were conducted on real-world and synthetic datasets. The real-world dataset (legcare [20]) consists of an online shop's web-log, containing 234942 visits. Each visit is regarded as a subtree of the hierarchically structured web-site. There where 694 unique labels for the database. For the synthetic dataset we implemented a data generator as described by Zaki [6]. All

the experiments were performed on a 3.2GHz Intel Pentium 4 with 2GB main memory, running SUSE 9.0. The algorithms were implemented in C++. For the tree embedding and tree incorporation, *instance pruning* is available. We compared the number of frequent patterns found by the algorithms and the size of the $\mathbb{S}$ set on both datasets with different minimum support. To calculate the $\mathbb{S}$ set an additional post-processing step was performed.

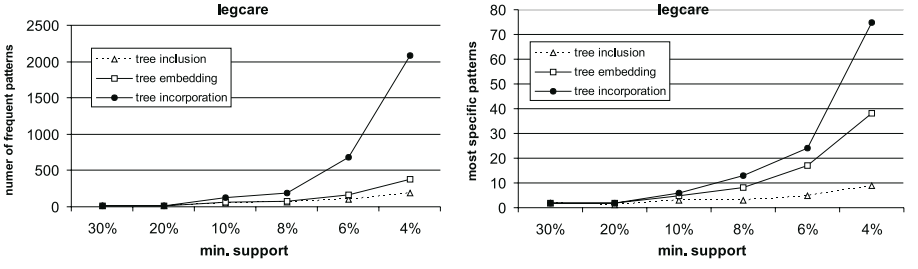


Fig. 20. Comparing the amount of the patterns generated during the search (*left*) and the size of the maximally specific border (*right*) of three tree matching notions for various minimum support levels

Figure (20) shows the number of patterns generated during the search and the number of patterns contributing to the $\mathbb{S}$ set for tree embedding and tree incorporation on the legcare dataset. As mentioned earlier, the notion of tree incorporation is more relaxed than tree embedding such that more patterns are generated and discovered during the search. In contrast to the figures shown here (Figure 20), there is no order between the notions with regard to the size of the according $\mathbb{S}$ set as explained in the Figures (18, 19). All three notions have in common that they exhibit exponential growth in the number of most specific patterns as well as in patterns considered during the search when reducing the minimum support. For the legcare-dataset there was no effect on computation time with and without *IP*. However, using *IP*, the memory consumption dropped dramatically for computing frequent pattern sets with minimum support below 10%. For the experiments with the synthetic data, a master-tree with 100 unique labels and 10000 nodes was generated with a maximum depth and fanout of 10. From this hypothetical web-site we generated 10000 visits, each a smaller subtree of the master-tree, as database. The graph in Figure (21) shows the results of experiments on this dataset regarding computation time and the effect of instance pruning. The plot clearly indicates an exponential growth in computation time when lowering the minimum support. The solid lines depict the required time with *IP*. Both, tree embedding and tree incorporation[4], show a significant speedup for low minimum support levels using *IP*. More experiments on synthetic data sets showed similar behaviour, i.e. the runtimes for the algorithms using *IP* are significantly lower, especially at low minimum support levels.

[4] Since *IP* was not implemented for tree inclusion there are no results here.

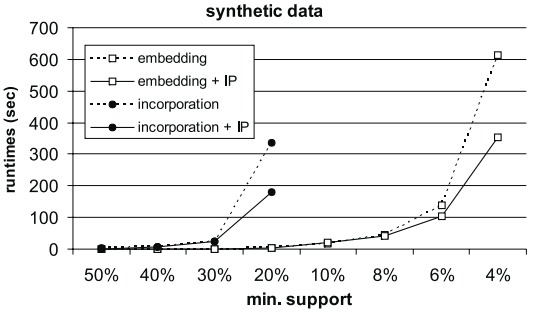


Fig. 21. Comparing the effect of instance pruning for embedding and incorporation on synthetic data

7 Related Work

The most directly related work to this paper is Zaki's TreeMinerV [6] as well as Termier's TreeFinder [7] and FreqT by Asai [5]. Zaki uses a smart, so-called vertical representation to facilitate the candidate count enabling a fast mining process that scales well even with large datasets. We adopted this idea for the presented treemining algorithm RETRO. Since the definition of tree incorporation is more general than tree embedding or tree inclusion, the algorithm yields more patterns even when focusing on the $\mathbb{S}$ set only. Therefore it is not surprising that the algorithm is slower than TreeMinerV. Using the RETRO algorithm to mine embedded trees there is a significant speedup compared to TreeMinerV due to the presented pruning technique. Compared to TreeFinder, it uses a less general pattern definitions, but all of them are complete with regard to the maximally specific patterns. For tree inclusion and tree embedding it is even complete with regard to the whole pattern space.

Other algorithms like FreeTreeMiner by Yun Chi *et al.* [21] and HybridTreeMiner [22] work on free trees. Furthermore there exist several algorithms like FreeTreeMiner by Rückert and Kramer [23] which searches for free-tree patterns in graphs or AGM [1], FSM [2], gSpan [3], and Gaston [24] that work completely on graphs rather than trees. They are restricted to subgraphs consisting of edges and if applied to trees would only discover frequent trees in the sense of subtree inclusion. The Gaston algorithm realizes the graph search stepwise. First frequent sequences are mined that are expanded to trees and later on to graphs.

8 Conclusions

The algorithm presented in this paper improves and expands the TreeMinerV algorithm. We added support for tree incorporation and and tree inclusion. The introduced *instance pruning* technique reduces the computation time as well as the memory usage of the algorithm in many cases. Since the amount of frequent

patterns in large databases grows fast when lowering the minimum support it seams to be useful to calculate the set $\mathbb{S}$ of all maximally specific patterns. An approach how to immediately calculate the set of all maximally specific patterns by considering non-canonical expansions was presented.

Furthermore, we have shown that there is an order among the four different matching notions. Each tree that matches another one with regard to tree inclusion also matches the same tree with regard to tree embedding. In the same way, each tree that is embedded in another tree is incorporated in the tree as well. Finally each incorporated tree is subsumed. Hence, the amount of patterns grows from tree inclusion, over embedding, incorporation to subsumption. In contrast this is not true for the set of maximally specific patterns.

With regard to the future, especially in a real-word environment, it would be nice to have more constraints, like maximally-general or maximally-specific pattern, to enable the user to focus the search as in MolFea [4]. Furthermore it would be interesting to extend the tree-mining process to first order logic which would give a much more expressive language for data and patterns. On the other hand the frequent patterns discovered could also be used as features for some classifier as in [25]. Considering the notion of tree incorporation, we still have to evaluate if the additional cost in time and memory is justified by more informative patterns. Finally, it depends on the data and on the requirements of the user which tree matching notion is the best.

Acknowledgments

We sincerely thank Luc De Raedt for motivating this work. We also thank Mohammed Zaki for providing the TreeMiner source code. Further many thanks to Ulrich Rückert, Albrecht Zimmermann, Kristian Kersting, and Andreas Karwath for constructive comments and helpful suggestions. The real world dataset was kindly provided by Blue Martini Software.

This work was partly supported by the EU IST project cInQ (Consortium on discovering knowledge with Inductive Queries). Contract no. IST-2000-26469.

References

1. Inokuchi, A., Washio, T., Motoda, H.: An apriori-based algorithm for mining frequent substructures from graph data. In: Proc. of PKDD. (2000) 13–23
2. Kuramochi, M., Karypis, G.: Frequent subgraph discovery. In: Proc. of ICDM. (2001) 439–442
3. Yan, X., Han, J.: gspan: Graph-based substructure pattern mining. In: Proc. of ICDM. (2002) 721–724
4. De Raedt, L., Kramer, S.: The levelwise version space algorithm and its application to molecular fragement finding. In: Proc. of IJCAI-01. (2001) 853–862
5. Asai, T., Abe, K., Kawasoe, S., Arimura, H., Sakamoto, H., Arikawa, S.: Efficient substructure discovery from large semi-structured data. In: Proc. of SIAM SDM. (2002) 158–174
6. Zaki, M.: Efficiently mining frequent trees in a forest. In: Proc. of KDD. (2002) 71–80

7. Termier, A., Rousset, M.C., Sebag, M.: Treefinder: a first step towards XML data mining. In: Proc. of ICDM. (2002) 450–457
8. Cooley, R., Mobasher, B., Srivastava, J.: Web mining: Information and pattern discovery on the world wide web. In: Proc. of ICTAI. (1997) 558–567
9. Goldfarb, C.F., Prescod, P.: The XML handbook. Prentice Hall (1998) ISBN 0-13-081152-1.
10. Shapiro, B.A., Zhang, K.: Comparing multiple RNA secondary structures using tree comparisons. Computer Applications in the Biosciences **6** (1990) 309–318
11. Bringmann, B.: Matching in frequent tree discovery. In: Proc. of ICDM. (2004) 335–338
12. Ramesh, R., Ramakrishnan, L.: Nonlinear pattern matching in trees. Journal of the ACM **39(2)** (1992) 295–316
13. Kilpeläinen, P.: Tree Matching Problems with Applications to Structured Text Databases. PhD thesis, University of Helsinki (1992)
14. Lloyd, J.W.: Foundations of logic programming; (2nd extended ed.). Springer-Verlag New York, Inc. (1987)
15. Plotkin, G.D.: A note on inductive generalization. In: Machine Intelligence. Volume 5. Edinburgh University Press (1970) 153–163
16. Muggleton, S., De Raedt, L.: Inductive logic programming: Theory and methods. Journal of Logic Programming **19/20** (1994) 629–679
17. Agrawal, R., Imielinski, T., Swami, A.N.: Mining association rules between sets of items in large databases. In: Proc. of ICMD, Washington, D.C. (1993) 207–216
18. Mannila, H., Toivonen, H.: Levelwise search and borders of theories in knowledge discovery. Data Mining and Knowledge Discovery **1** (1997) 241–258
19. Ng, R.T., Lakshmanan, L.V.S., Han, J., Pang, A.: Exploratory mining and pruning optimizations of constrained associations rules. In: SIGMOD '98: Proceedings of the 1998 ACM SIGMOD international conference on Management of data, ACM Press (1998) 13–24
20. Kohavi, R., Brodley, C., Frasca, B., Mason, L., Zheng, Z.: KDD-Cup 2000 organizers' report: Peeling the onion. SIGKDD Explorations **2** (2000) 85–98
21. Chi, Y., Yang, Y., Muntz, R.R.: Indexing and mining free trees. In: Proc. of ICDM. (2003) 509–512
22. Chi, Y., Yang, Y., Muntz, R.R.: Hybridtreeminer: An efficient algorithm for mining frequent rooted trees and free trees using canonical form. In: Proc. of SSDBM. (2004) 11–20
23. Rückert, U., Kramer, S.: Frequent free tree discovery in graph data. In: Proc. of ACM symposium on Applied computing, ACM Press (2004) 564–570
24. Nijssen, S., Kok, J.N.: A quickstart in frequent structure mining can make a difference. In: Proc. of KDD, ACM Press (2004) 647–652
25. Bringmann, B., Karwath, A.: Frequent SMILES. In: Proc. of LWA (FGML). (2004) 132–137

A Survey on Condensed Representations for Frequent Sets

Toon Calders[1], Christophe Rigotti[2], and Jean-François Boulicaut[2]

[1] University of Antwerp, Belgium
toon.calders@ua.ac.be
[2] INSA Lyon, LIRIS CNRS UMR 5205, France
{crigotti, jfboulicaut}@liris.cnrs.fr

Abstract. Solving inductive queries which have to return complete collections of patterns satisfying a given predicate has been studied extensively the last few years. The specific problem of frequent set mining from potentially huge boolean matrices has given rise to tens of efficient solvers. Frequent sets are indeed useful for many data mining tasks, including the popular association rule mining task but also feature construction, association-based classification, clustering, etc. The research in this area has been boosted by the fascinating concept of condensed representations w.r.t. frequency queries. Such representations can be used to support the discovery of every frequent set and its support without looking back at the data. Interestingly, the size of condensed representations can be several orders of magnitude smaller than the size of frequent set collections. Most of the proposals concern exact representations while it is also possible to consider approximated ones, i.e., to trade computational complexity with a bounded approximation on the computed support values. This paper surveys the core concepts used in the recent works on condensed representation for frequent sets.

1 Introduction

Knowledge Discovery in Databases (KDD) is a complex interactive and iterative process which involves many steps. Within the inductive database (IDB) framework, the needed data mining tasks are formalized as inductive queries which can be used to generate (mine), manipulate, and apply patterns [33,24]. The IDB framework is appealing because it employs *declarative* queries instead of ad-hoc *procedural constructs*. Since its introduction in [1], one of the most studied problems has been frequent itemset mining (FIM) and the popular post-processing of found itemsets into collections of association rules. Originally, this task has been dedicated to basket data analysis. Given a database of purchases or transactions, the association rule mining problem is to find associations between sets of products. In this context, the frequent itemsets correspond to sets of products that are often purchased together. Since then, the scope of association rule mining applications has been broadened towards many data analysis problems which are based on boolean or O/1 data (e.g., documents, WWW sessions, or microarray experiments can be considered as transactions whose items

J.-F. Boulicaut et al. (Eds.): Constraint-Based Mining, LNAI 3848, pp. 64–80, 2005.

are respectively descriptors, uploaded resources or gene expression properties). Finding the frequent itemsets given a user-defined support threshold is not only the computationally most intensive step of association rule mining, but also it can be used for many other mining tasks, e.g., feature construction for classification or clustering methods. As such, the efficient extraction of frequent itemsets directly leads to significant performance improvements for many interactive KDD processes. It is indeed widely recognized that mining frequent itemsets should be one of the main operations supported by an inductive database management system.

The FIM problem has been studied as an inductive querying problem (see, e.g., [10]) and it is a prototypical task for which the general idea of condensed representations introduced in [39] has been proved extremely useful. The simple model introduced in [40] enables to abstract the semantics of inductive queries. Given a language $\mathcal{L}$ of patterns, the *theory* of a database $\mathcal{D}$ w.r.t. $\mathcal{L}$ and a selection predicate $\mathcal{C}$ is the collection $Th(\mathcal{D}, \mathcal{L}, \mathcal{C}) = \{\phi \in \mathcal{L} \mid \mathcal{C}(\phi, \mathcal{D}) = true\}$. The predicate selection or *constraint* $\mathcal{C}$ indicates whether a pattern ϕ is interesting or not. We say that computing $Th(\mathcal{D}, \mathcal{L}, \mathcal{C})$ is the evaluation for the inductive query $\mathcal{C}$ where $\mathcal{C}$ is defined as a boolean expression over some primitive constraints. The FIM problem concerns inductive queries where the data is a set of transactions (i.e., a potentially huge boolean matrix), the patterns are itemsets (i.e., sets of columns of the boolean matrix), and the constraint $\mathcal{C}$ is reduced to a minimal support constraint. In the first years, most of the research on the FIM problem has concentrated on extracting *all* frequent sets as efficiently as possible. Levelwise and depth-first search methods based on the anti-monotonicity of minimal support, and efficient data structures have been studied. Since the first algorithm AIS [1], there have been important historical gains on performance such as: improving pruning (Apriori [2]) and counting (e.g., Partition [48], Sampling [49]), reducing the number of database scans (e.g., DIC [15]), and avoiding explicit candidate generation (e.g., FP-Growth [32]). This list is not exhaustive, and it should also be noticed that these approaches are often based on a mix of several improvements. Often, however, the number of frequent itemsets is so huge that their storage and support counting require unrealistic resources. This blow-up happens, for example, when we set the support threshold too low, or when the data is heavily correlated. Indeed, in the worst case, the number of frequent itemsets can be exponential in the number of items. Even though typical basket data is sparse and weakly correlated, many new applications of FIM have turned to be computationally too hard.

One solution to this problem relies on the *condensed representation* principle. The idea is to compute $\mathcal{CR} \subseteq \mathcal{L}$ which might be as concise as possible such that deriving $Th(\mathcal{D}, \mathcal{L}, \mathcal{C})$ from $\mathcal{CR}$ can be performed efficiently. In the context of huge database mining, efficiently means without any further access to $\mathcal{D}$. Using border sets [40], e.g., the maximal frequent itemsets for FIM [5], might be considered as a good solution: all the subsets of the maximal frequent itemsets are frequent itemsets (i.e., this condensed representation is a proper subset of the theory) and can be derived without looking at the data. In most of the applications of

FIM, however, the user wants not only the collection of the frequent patterns but also their supports (e.g., to compute association rule interestingness measures like the confidence values). Now, a condensed representation $\mathcal{CR}$ must enable to regenerate not only the patterns, but also the values of an evaluation function like the support without accessing the data. If the regenerated values are only approximated, the condensed representation is called *approximate*. Otherwise, it is called an *exact* condensed representation. For a condensed representation, different characteristics determine its usefulness, depending on the application area. It is clear that good characteristics are: the size of the representation (theoretically and in practice), the efficiency, and the completeness of the algorithms which compute these representations, the fast and complete generation of useful information from the representation (e.g., all the frequent itemsets and their supports, relevant association rules).

Starting from the formalization of ϵ-adequate representations [39] and its first concrete application to FIM in [11], many useful condensed representations have been designed over the last 5 years. The main objective of this survey is to present, in a synthetic way, the core concepts used in the recent works on condensed representation for frequent itemsets, including: *Closed Sets* [55,43,44,11] *δ-Free Sets* [12,13], *Disjunction-Free Sets* [17,18], *Generalized Disjunction-Free Sets* [37], *Non-Derivable Itemsets* [20], and the unified framework presented in [21].

The organization of the paper is as follows. In the next section, we recall some preliminary definitions. Then, we present several condensed representations in Sections 3 to 6. Section 7 concerns a recent framework which provides a unified view of most of these representations. Section 8 provides pointers to representative algorithms for computing condensed representations. Section 9 gives complementary bibliographic information concerning applications. Finally, Section 10 is a short conclusion.

2 Preliminary Definitions

The FIM problem is by now well known [1]. We are given a set of items $\mathcal{I}$ and a database $\mathcal{D}$ of subsets of $\mathcal{I}$ (to allow duplicates, $\mathcal{D}$ can be defined as a multi-set). The elements of $\mathcal{D}$ are called *transactions*. An *itemset* $I \subseteq \mathcal{I}$ is a set of items; its *support* in $\mathcal{D}$, denoted $supp(I, \mathcal{D})$, is defined as the number of transactions in $\mathcal{D}$ that contain all items of I. An itemset is called *σ-frequent* in $\mathcal{D}$ if its support in $\mathcal{D}$ exceeds σ. The goal is now, given a minimal support threshold and a database, to compute the collection $\mathcal{F}(\mathcal{D}, \sigma)$ of all frequent itemsets and their supports. We denote itemsets by strings, e.g., *abcd* denotes the set $\{a, b, c, d\}$.

The presentation of most of the condensed representations needs for the concept of *negative border* introduced in [40]. The negative border of a collection of itemsets $\mathcal{J}$, denoted $\mathcal{B}d^-(\mathcal{J})$ is the collection $\{X | X \subseteq \mathcal{I} \wedge X \notin \mathcal{J} \wedge (\forall Y \subset X, Y \in \mathcal{J})\}$. Intuitively, $\mathcal{B}d^-(\mathcal{J})$ contains the smallest itemsets not in $\mathcal{J}$. For instance, $\mathcal{B}d^-(\mathcal{F}(\mathcal{D}, \sigma))$ denotes the collection of the smallest (w.r.t. set inclusion) infrequent itemsets.

The last notion that we recall in this section, is *anti-monotonicity*. It is a commonly used property leading to safe pruning criteria and efficient pattern mining (e.g., [40]). A property ρ is anti-monotonic if and only if for all itemsets X and Y, $\rho(X)$ and $Y \subseteq X$ implies $\rho(Y)$. Clearly, the minimal support property is anti-monotonic.

3 Closed Sets

This representation is based on the notion of *closed set* used in *formal concept analysis* [51,28], a branch of lattice theory dedicated to the study of the lattice structure induced by a binary relation (structure called *Galois lattice* or *concept lattice*).

The application of this theory to frequent itemset mining has been proposed independently by Pasquier et al. in [43,44] and by Zaki and Ogihara in [55].

In this context, an itemset I is said to be *closed* in $\mathcal{D}$ if and only if no proper superset of I has the same support than I in $\mathcal{D}$. The *closure* of an itemset I in $\mathcal{D}$, denoted $cl(I)$, is the unique maximal superset of I having the same support than I and a closed itemset is equal to its closure. One elegant alternative definition is to consider the equivalence classes of the itemsets appearing in the same sets of transactions, i.e., the equivalence classes of the relation "has the same closure": closed itemsets are the unique maximal elements of each equivalence class [4].

For a given support threshold, it is thus sufficient to know the collection of all frequent closed itemsets (denoted *FreqClosed*) and their supports, to be able to generate all the frequent itemsets and their supports, i.e., $\mathcal{F}$. For example, consider an itemset X, if X has no superset in *FreqClosed*, this means that $cl(X)$ is not frequent, and thus X can not be frequent. If X has at least one superset in *FreqClosed*, then $supp(X) = supp(Y)$ where $Y = cl(X)$ is the smallest superset of X in *FreqClosed*.

Let us consider the database containing the following transactions: two transactions $\{a, b\}$, two transactions $\{a, b, c, d\}$ two transactions $\{a, b, c, d, e\}$ and one transaction $\{a, b, c, d, e, f\}$ (see Table 1).

In such a database, for example, the itemset abc is not closed, since it has the same support (i.e., 5 transactions) than $abcd$, one of its proper supersets.

Table 1. A toy database

	Items					
Trans.	a	b	c	d	e	f
t_1	1	1	0	0	0	0
t_2	1	1	0	0	0	0
t_3	1	1	1	1	0	0
t_4	1	1	1	1	0	0
t_5	1	1	1	1	1	0
t_6	1	1	1	1	1	0
t_7	1	1	1	1	1	1

The itemset *abcd* is the maximal superset of *abc* having the same support, and thus is the closure of *abc*. If we choose a support threshold of 2 transactions, then the frequent closed sets are *ab*, *abcd*, *abcde* and their respective supports are 7, 5 and 3. Having only at hand these frequent closed sets, to generate the support of *abc* we consider the smallest frequent closed set that is a superset of *abc*. This frequent closed set is *abcd* and its support (i.e., 5 transactions) gives us the support of *abc*.

4 Free Sets

The *free sets* (also termed δ-*free sets*) have been introduced in [12,13] and are based on the notion of δ-*strong rule*[1]. Informally, a δ-strong rule is an association rule of the form $X \Rightarrow^{\delta} a$, where $X \subseteq \mathcal{I}$, $a \in \mathcal{I} \setminus X$, and δ is a natural number. This rule is valid in a database if $supp(X) - supp(X \cup \{a\}) \leq \delta$, i.e., the rule is violated in no more than δ transactions. Since δ is supposed to be small w.r.t. $|\mathcal{D}|$, δ-strong rules have a high confidence (in particular confidence 1 when $\delta = 0$).

An itemset $Y \subseteq \mathcal{I}$ is a δ-free set if and only if there is no valid δ-strong rule $X \Rightarrow^{\delta} a$ such that $X \subset Y$, $a \in Y$ and where by definition $a \notin X$.

The set of all frequent δ-free sets, denoted $FreqFree^{\delta}$, and their supports enables to approximate the support of the frequent non-δ-free sets. Let us consider Y a frequent non-δ-free set. Then, there exists a valid δ-strong rule $X \Rightarrow^{\delta} a$ such that $X \subset Y$ and $a \in Y$. Moreover, $Y \setminus \{a\} \Rightarrow^{\delta} a$ is also valid. Thus the support of Y can be approximated by the support of the frequent set $Y \setminus \{a\}$ (more precisely, this support is an upper bound of $supp(Y)$). If $Y \setminus \{a\}$ is a free-set then we have its support, if not, it can be in turn approximated by the support of a smaller itemset. This recursive process gives an approximation of the support of Y. Using this principle, the best approximation is the lowest upper bound. Thus, in practice, the support of Y is approximated by the minimal support value of the frequent δ-free sets that are subsets of Y. The error made has been formalized using the framework of an ϵ-adequate representation [39], and is small on common real datasets [13].

When $\delta = 0$, the support of all frequent non-δ-free sets can be determined exactly. In fact, the 0-free sets corresponds to the *key patterns* (also called *generators*) developed independently in [4], and also used in other works, such as [36]. The following property mentioned by several authors (e.g., [4]) establishes a direct link between 0-free sets and closed sets: any frequent closed sets is the closure of at least one frequent 0-free sets. As a result, when considering each (frequent) 0-free set X, $cl(X)$ is a (frequent) closed set but also $X \Rightarrow cl(X) \setminus X$ is an association rule with confidence 1. In fact, 0-free sets are the minimal elements of the already mentioned equivalence classes. Since several minimal elements are possible, collections of 0-free sets are generally larger than collections of closed sets. In our toy example from Table 1, the 2-frequent 0-free sets are $\emptyset$, c, d and e.

Even though the frequent δ-free sets are sufficient to approximate the support of all frequent non-δ-free sets (or to determine this support exactly when $\delta = 0$),

[1] Stemming from the notion of *strong rule* of [46].

they are not sufficient to decide whether an itemset is frequent or not. For this purpose, the collection of frequent δ-free sets is completed by the collection of minimal infrequent δ-free itemsets, that can be defined as $\mathcal{B}d^-(\mathit{FreqFree}^\delta) \cap \mathit{Free}^\delta$, where Free^δ is the collection of δ-free sets. Now, given any itemset Y, if there exists $Z \subseteq Y$, such that Z is a minimal infrequent δ-free itemsets, then we know that Y is not frequent. In the other case, the support of Y can be approximated as described above.

5 Disjunction-Free Sets

5.1 Simple Disjunction-Free Sets

This representation has been proposed in [17,18] as a generalization of 0-free sets. It is based on *disjunctive rules* of the form $X \Rightarrow a \vee b$, where $X \subseteq \mathcal{I}$ and $a, b \in \mathcal{I} \setminus X$. Such a rule is said to be valid if any transaction containing X contains also a or b (maybe both).

Thus the support of X is equal to the sum of $\mathit{supp}(X \cup \{a\})$ and $\mathit{supp}(X \cup \{b\})$ minus $\mathit{supp}(X \cup \{a, b\})$ since the transactions containing $X \cup \{a, b\}$ have been counted both in $\mathit{supp}(X \cup \{a\})$ and $\mathit{supp}(X \cup \{b\})$. So, we have the relation $\mathit{supp}(X \cup \{a, b\}) = \mathit{supp}(X \cup \{a\}) + \mathit{supp}(X \cup \{b\}) - \mathit{supp}(X)$ and the satisfaction of this relation is equivalent to the validity of the rule $X \Rightarrow a \vee b$.

Similarly to δ-free sets, an itemset $Y \subseteq \mathcal{I}$ is a *disjunction-free set* if and only if there is no valid disjunctive rule $X \Rightarrow a \vee b$, such that $X \subset Y$, $a, b \in Y$ and where by definition $a \notin X$ and $b \notin X$. In the following, the collection of all frequent disjunction-free sets is denoted *FreqDFree*.

Knowing all elements in *FreqDFree* and their supports is not sufficient to determine the support of all frequent itemsets. For that purpose the representation can be completed in different ways. The representation based on disjunction-free sets proposed in [17] has been revisited in [36] and [18], leading to reduce the size of this border[2].

Intuitively, *FreqDFree* must be completed with the collection of all the valid rules of the form $X \Rightarrow a \vee b$, where $X \in \mathit{FreqDFree}$ and $X \cup \{a, b\}$ is frequent. This can be illustrated inductively as follows. Suppose that using *FreqDFree* (and the supports of its elements) and the collection of rules defined above, we are able to compute the support of any itemset having a size lesser or equal to k. Let us consider a frequent itemset Y such that $|Y| = k + 1$. If Y is disjunction-free then $Y \in \mathit{FreqDFree}$ and we know its support. If Y is not disjunction-free, then there exists a valid rule $X \Rightarrow a \vee b$ such that $X \subset Y$ and $a, b \in Y$. By definition of a valid rule, $Y \setminus \{a, b\} \Rightarrow a \vee b$ is also valid. Hence the relation $\mathit{supp}(Y) = \mathit{supp}(Y \setminus \{b\}) + \mathit{supp}(Y \setminus \{a\}) - \mathit{supp}(Y \setminus \{a, b\})$ holds. Since Y is frequent, the itemsets $Y \setminus \{b\}$, $Y \setminus \{a\}$ and $Y \setminus \{a, b\}$ are also frequent. Moreover, these three sets have a size strictly lesser than $k + 1$. Thus, by hypothesis, we can determine their supports, and then compute $\mathit{supp}(Y)$.

[2] The core part of the representation, i.e. the frequent disjunction-free sets (called frequent disjunction-free generators in [36]), remains the same.

5.2 Generalized Disjunction-Free Sets

The generalization of disjunction-free sets towards rules of the form $X \Rightarrow a_1 \vee \ldots \vee a_i \vee \ldots \vee a_n$, has been suggested in [17,18], and explored in [37]. In this context, an itemset X is a generalized disjunction-free set if and only if for any value of $n > 0$, there is no valid rule $X \setminus \{a_1, \ldots, a_i, \ldots, a_n\} \Rightarrow a_1 \vee \ldots \vee a_i \vee \ldots \vee a_n$, where $\{a_1, \ldots, a_i, \ldots, a_n\} \subseteq X$.

6 Non-derivable Itemsets

In [20], the *non-derivable itemsets* (NDIs) were introduced as a new condensed representation. The NDIs rely on a complete set of deduction rules that derive bounds on the support of an itemset. In this section, we first discuss the deduction rules, and then introduce the representation based on these rules.

6.1 Deduction Rules

In [20], formulas to bound the support of an itemset I, based on the supports of its subsets were introduced. For all $X \subseteq I$, the following rule holds:

$$supp(I) \leq \sum_{X \subseteq J \subset I} (-1)^{|I \setminus J|+1} supp(J) \qquad \text{if } |I \setminus X| \text{ odd}$$
$$supp(I) \geq \sum_{X \subseteq J \subset I} (-1)^{|I \setminus J|+1} supp(J) \qquad \text{if } |I \setminus X| \text{ even}$$

This rule is denoted $\mathcal{R}_X(I)$. The rules are based on the inclusion-exclusion principle [27]. For a proof of the rules, see [19]. Depending on the subset X of I, the bound is a lower or an upper bound. If $|I \setminus X|$ is odd, $\mathcal{R}_I(X)$ is an upper bound, otherwise it is a lower bound. Thus, given the supports of all subsets of an itemset I, we can derive lower and upper bounds on the support of I with the rules $\mathcal{R}_I(X)$ for all $X \subseteq I$.

Notice that these rules reflect the monotonicity principle. Let $i \in I$, then $\mathcal{R}_{I \setminus \{i\}}(I)$ is the following rule:

$$supp(I) \leq supp(I \setminus \{i\}) \ .$$

In Figure 1, all rules $\mathcal{R}_X(I)$, for $I = abcd$ and $X \subseteq I$ are given.

We denote the greatest lower bound on I by $LB(I)$ and the least upper bound by $UB(I)$. In practice it occurs often that $LB(I) = UB(I)$. Such a set I is called a *derivable itemset (DI)*, since we know without counting its support in the database, that $supp(I) = LB(I) = UB(I)$. In [20] it was shown that derivability is monotonic. Hence, if a set I is derivable, then are all its supersets.

Another interesting property proven in [20] is that for a non-derivable itemset, the interval width, that is, $w(I) := UB(I) - LB(I)$, decreases exponentially in $|I|$. Thus, $w(I \cup \{j\}) \leq w(I)/2$, for every itemset I and item j not in I. This property guarantees that non-derivable itemsets cannot be very large, because the intervals can only be halved a logarithmic number of times.

$$
\begin{array}{lll}
supp(abcd) \geq & supp(abc) + supp(abd) + supp(acd) + supp(bcd) & \\
& -supp(ab) - supp(ac) - supp(ad) - supp(bc) & \mathcal{R}_{\emptyset}(abcd) \\
& -supp(bd) - supp(cd) + supp(a) + supp(b) & \\
& +supp(c) + supp(d) - supp(\emptyset) & \\
supp(abcd) \leq & supp(a) - supp(ab) - supp(ac) - supp(ad) & \mathcal{R}_{a}(abcd) \\
& +supp(abc) + supp(abd) + supp(acd) & \\
supp(abcd) \leq & supp(b) - supp(ab) - supp(bc) - supp(bd) & \mathcal{R}_{b}(abcd) \\
& +supp(abc) + supp(abd) + supp(bcd) & \\
supp(abcd) \leq & supp(c) - supp(ac) - supp(bc) - supp(cd) & \mathcal{R}_{c}(abcd) \\
& +supp(abc) + supp(acd) + supp(bcd) & \\
supp(abcd) \leq & supp(d) - supp(ad) - supp(bd) - supp(cd) & \mathcal{R}_{d}(abcd) \\
& +supp(abd) + supp(acd) + supp(bcd) & \\
supp(abcd) \geq & supp(abc) + supp(abd) - supp(ab) & \mathcal{R}_{ab}(abcd) \\
supp(abcd) \geq & supp(abc) + supp(acd) - supp(ac) & \mathcal{R}_{ac}(abcd) \\
supp(abcd) \geq & supp(abd) + supp(acd) - supp(ad) & \mathcal{R}_{ad}(abcd) \\
supp(abcd) \geq & supp(abc) + supp(bcd) - supp(bc) & \mathcal{R}_{bc}(abcd) \\
supp(abcd) \geq & supp(abd) + supp(bcd) - supp(bd) & \mathcal{R}_{bd}(abcd) \\
supp(abcd) \geq & supp(acd) + supp(bcd) - supp(cd) & \mathcal{R}_{cd}(abcd) \\
supp(abcd) \leq & supp(abc) & \mathcal{R}_{abc}(abcd) \\
supp(abcd) \leq & supp(abd) & \mathcal{R}_{abd}(abcd) \\
supp(abcd) \leq & supp(acd) & \mathcal{R}_{acd}(abcd) \\
supp(abcd) \leq & supp(bcd) & \mathcal{R}_{bcd}(abcd) \\
supp(abcd) \geq & 0 & \mathcal{R}_{abcd}(abcd)
\end{array}
$$

Fig. 1. Tight bounds on $supp(abcd)$

The size of the rules $\mathcal{R}_I(X)$ increases exponentially with the cardinality of $I \setminus X$. The number $|I \setminus X|$ is called the *depth* of rule $\mathcal{R}_I(X)$. Since calculating all rules may require a lot of resources, in practise only rules of limited depth are used. The greatest lower and least upper bounds on the support of I resulting from evaluation of rules up to depth k are denoted $LB_k(I)$ and $UB_k(I)$. Hence, the interval $[LB_k(I), UB_k(I)]$ is formed by the bounds calculated by the rules $\{\mathcal{R}_X(I) \mid X \subseteq I, |I \setminus X| \leq k\}$.

Example 1. Consider the following database:

TID	Items
1	a
2	b
3	c
4	a, b
5	a, c
6	b, c
7	a, b, c

$$
\begin{array}{lll}
supp(abc) & \geq & 0 \\
& \leq & supp(ab) = 2 \\
& \leq & supp(ac) = 2 \\
& \leq & supp(bc) = 2 \\
& \geq & supp(ab) + supp(ac) - supp(a) = 0 \\
& \geq & supp(ab) + supp(bc) - supp(b) = 0 \\
& \geq & supp(ac) + supp(bc) - supp(c) = 0 \\
& \leq & supp(ab) + supp(ac) + supp(bc) \\
& & -supp(a) - supp(b) - supp(c) + supp(\emptyset) = 1
\end{array}
$$

These rules are $\mathcal{R}_{abc}(X)$ when X is respectively abc, ab, ac, bc, a, b, c, and $\emptyset$. The first rule has depth 0, the following three rules depth 1, the next three rules depth

2, and the last rule has depth 3. Hence, $LB_0(abc) = 0$, $LB_2(abc) = 0$, $UB_1(abc) = 2$, $UB_3(abc) = 1$. The interval width for abc is $UB(abc) - LB(abc) = 1$.

For ab, we have the following rules:

$$supp(ab) \geq 0 \qquad supp(ab) \leq supp(a) = 4$$
$$supp(ab) \geq supp(a) + supp(b) - supp(\emptyset) = 1 \qquad supp(ab) \leq supp(b) = 4$$

Therefore, $LB(ab) = 1$, and $UB(ab) = 4$. The interval width for ab is 3. Notice that the interval width for abc is indeed less than half of the interval width for ab.

6.2 Representation Based on Deduction Rules

In [20], the NDI representation was introduced, based on the deduction rules. The NDI-representation is defined as follows:

$$NDIRep(\mathcal{D}, \sigma) := \{(I, supp(I, \mathcal{D})) \mid supp(I, \mathcal{D}) \geq \sigma, LB(I) \neq UB(I)\}$$

From *NDIRep*, for every set I it can be decided whether or not it is frequent, and if it is frequent, its support can be derived. This can be seen as follows: every itemset I that is not in *NDIRep* is either infrequent, or derivable (or both). We calculate and compare the bounds $LB(I)$ and $UB(I)$. If they are not equal, I must be infrequent (otherwise I would have been in *NDIRep*). If they are equal, then we know $supp(I) = LB(I) = UB(I)$. In order to calculate the bounds on the support of I, however, we need to know the support of all subsets of I. This can be done in an iterative way; first we calculate the bounds on the subsets of I that are in the border of *NDIRep*. For these subsets, the bounds can be calculated. If one of them is infrequent, I must be infrequent as well. Otherwise, we know the supports of all subsets of I in the border of *NDIRep*. Subsequently, we can calculate bounds on the subsets of I that are just above the border, and so on, until either the supports of all subsets of I are known and we can calculate the bounds for I, or one of the subsets turned out to be infrequent.

7 Unified View

In [21], a unified view of 0-freeness, disjunction-freeness and non-derivability was given. In this framework, the notion of a k-free[3] set is central, as it captures different properties in several previously studied exact condensed representations. It was shown that the different representations can be described as a main component, that is based on frequent k-free, and a border. We now describe the main ideas of this unified view.

7.1 k-Free Sets

The k-free sets are a key tool in the unified framework.

[3] Notice that the k-free sets are different from the δ-free sets of Section 5.

Definition 1.
A set I is said to be k-free*, if $supp(I) \neq LB_k(I)$ and $supp(I) \neq UB_k(I)$.*
A set I is said to be ∞-free, if $supp(I) \neq LB(I)$, and $supp(I) \neq UB(I)$.
The set of all k-free (∞-free) sets is denoted $Free_k$ ($Free_\infty$).

As the next property states, these definitions cover freeness, disjunction-freeness, and generalized disjunction-freeness.

Property 1. [21] Let I be a frequent itemset.

- I is free (δ-free with $\delta = 0$) if and only if I is 1-free
- I is disjunction free if and only if I is 2-free.
- I is generalized disjunction-free if and only if I is ∞-free.

The next property forms the basis of the representations based on k-free sets.

Property 2. k-freeness is anti-monotonic; if a set I is k-free, then all its subsets are k-free as well. Moreover, if $supp(J) = LB_k(J)$ (resp. $supp(J) = UB_k(J)$), then also $supp(I) = LB_k(I)$ (resp. $supp(I) = UB_k(I)$), for all $J \subseteq I$.

The frequent k-free sets together with the border, that is, the collection

$$\{(I, supp(I)) \mid I \in FFree_k\} \cup \{(J, supp(J)) \mid \forall j \in J : J \setminus \{j\} \in FFree_k\} ,$$

forms a condensed representation. It can be shown by induction that for every itemset I, we can derive whether or not it is frequent, and if it is frequent, we can find its support. For the sets that are frequent and k-free or that are in the border, the support is known because they are in the representation. Next, let I be a set such that all its subsets are in the representation. Then the support of all subsets of I is known, as they are all in the representation. Also, I has at least one subset J in the border of the k-free sets (otherwise I would have been in the border itself, and thus in the representation). If J is infrequent, then I is as well. Otherwise, $supp(J)$ is either $LB_k(J)$ or $UB_k(J)$. Suppose that $supp(J) = LB_k(J)$. Then we know from Property 2 that also $supp(I) = LB_k(I)$. Since the support of all subsets of I are known, we can calculate $LB_k(I)$, and thus we can derive the support of I. Hence, for all itemsets that contain only one more item than the sets in the representation, we can find the support. We can now iteratively repeat this procedure to find the sets that contain two more items, three more items, and so on, until we have found all frequent itemsets.

7.2 Groups in the Border

Let us recall from Section 5 that frequent free sets alone do not form a condensed representation. In order to have a condensed representation, part of the border need to be stored as well. For disjunction-free and generalized disjunction-free sets, parts of the border are needed as well. The reason that some of the sets of the border are needed is because otherwise it is impossible to tell why the sets are not in the representation. For example, for the disjunction-free sets, were

they left out because the were infrequent, or because they were not disjunction-free? And if they are not disjunction-free, what rule should be used to derive the support? Because of the anti-monotonicity of both frequency and disjunction-freeness, it suffices to store only the sets on the border; if we know them, we know the rest as well; either the set on the border is infrequent, and then are all its supersets as well, or it is not disjunction-free with a certain rule, and in that case, its supersets are not disjunction-free as well, because of the same rule.

In general, as we illustrated in the previous subsection, this situation applies for k-free sets as well. Again, some elements of the border are needed to have a condensed representation.

In [21], a systematic study of which parts of the border are really needed was made. The border of the frequent k-free sets can be divided into different parts, based on the deduction rules. For example: the group of infrequent sets in the border, the group of sets I with $supp(I) \neq LB_1(I)$, or the group of frequent sets with $supp(I) = LB_\infty(I)$. In this way the existing representations could be improved by storing a smaller part of the border.

7.3 Relations Between the Different Representations

From the unified view of the different representations, many relations between the representations can be derived. In fact, the k-free based representations form an interesting hierarchy. The higher k is, the more complex the representation becomes, but at the same time, the more concise. For example, the disjunction-free sets are based on the 2-free sets, while the non-derivable itemsets are based on the ∞-free sets. Henceforth, on the one hand, the NDI-representation is more concise than the disjunction-free representation, but on the other hand, it can be far more costly to compute it and to derive the support of the sets which are not in the collection [21].

8 Algorithms

Many algorithms and variants have been proposed to extract condensed representations for frequent itemsets. The main principles are similar to the ones that have been proposed for the extraction of frequent itemsets. This includes two main aspects, firstly the strategy used to explore the pattern space and secondly the representation of the database used to count the support of the patterns.

Nearly all algorithms start the exploration from the empty itemset and go towards larger ones. This is performed either in a levelwise way (i.e., considering all patterns of size n and then all patterns of size $n + 1$) or using a depth-first approach. For the counting steps, three main representations have been adopted. The first one called *horizontal database layout* is a very natural one, in which the database is handle as a list of transactions. The second is based on a *vertical database layout* representation, so that for each pattern the algorithms store the identifiers of the transactions in which this pattern occur. Such a list, called *occurrence list* or *tid-list*, are used to count the support of the pattern and also to generate the occurrence lists of longer patterns. And finally, the third approach

that relies on *projected databases*, which contain in a compact way, subsets of the data needed to explore sub-spaces of the whole pattern space.

The main representative algorithms are a combination of these exploration strategies and database representations. The levelwise strategy is used together with an horizontal database layout to extract the closed sets by the algorithms Close [44] and Pascal [4], and also to mine the δ-free sets [12,13], the disjunction-free sets [17,18] (algorithm HLinHex) and the NDIs [20]. The depth-first strategy and a projected database approach are combined in the Closet [45] and VLinHex [17,18] algorithms to find respectively closed itemsets and disjunction-free sets. The vertical database layout has been used conjointly to a depth first exploration in the Charm [54] and the dfNDI [22] algorithms.

Beyond the usual pruning based on support, the various algorithms used pruning conditions stemming from properties of the different condensed representations (e.g., anti-monotonicity of freeness) to reduce the search space. It should be noticed that a major effort has been made to obtain efficient implementations (see, e.g., the first and second Workshop on Frequent Itemset Mining Implementations [31,6]).

9 Applications

Our goal is not to provide an exhaustive list of applications of condensed representations of frequent sets. Instead, we want to point out some typical examples of such works.

It is obvious that condensed representations of frequent sets can be used for any application of frequent sets: frequent sets and their supports are just computed faster from dense and/or correlated data. It is however important to notice that, when condensed representations enable a high condensation, the regeneration process might fail due to the size of the complete collection of the frequent sets. Therefore, it makes sense either to use condensed representations as cache mechanisms and/or to derive relevant patterns directly from the condensed representations. For instance, it is possible to provide summaries or even covers of large collections of association rules [53,3,30]. One typical application has been considered in [7] where 0-free sets and their closures are computed from a boolean gene expression data set. One can also point out the generation of a synthetic view of rule confidence variations from disjunction-free sets [16]. The recent Ph.D thesis [41] studies summarization techniques for large collections of patterns and thus many applications of condensed representations. Association-based classification (see, e.g., [38]) can also benefit from condensed representations. For instance, using δ-strong association rules built on δ-free itemsets and their closures has been proved useful in this context [23]. It is also possible to exploit condensed representations as patterns for themselves, e.g., closed sets in boolean gene expression data sets correspond to putative synexpression groups or transcription modules [8].

Condensed representations can be used for optimizing not only one inductive query on sets but also sequences of queries on set patterns [34,29]. One condensed

representation can also be used as an intermediate representation to mine efficiently another one (see, e.g., the generation of closed sets from disjunction-free sets [18]). Related to inductive querying on sets, one interesting issue concerns condensed representation mining when the minimal support constraint is not the only constraint. This has been considered, e.g., for free sets in [14] and for closed sets in [9].

Finally, we have removed maximal frequent itemsets from consideration while it can be useful for some applications where the support of every frequent itemset is not needed, e.g., feature construction. Indeed, border sets have many applications. For instance, border sets have been studied extensively in the context of conjunctions of minimal support and maximal support constraints (see, e.g., [25]).

10 Conclusion and Perspectives

This paper has surveyed the core concepts used in the recent works on condensed representations for frequent sets. These concepts have been proved extremely useful not only for an algorithmic breakthrough concerning the many applications of frequent set mining but also for deriving more useful patterns, e.g., covers of association rules. An important direction of work, is the detailed comparison of practical pros and cons of the different representations. This includes fair experiments on representative real data sets, to compare (1) the representation sizes (in number of patterns, and also their true sizes in bytes) and (2) their related time costs, (not only for their extractions, but also for the generation of patterns like frequent itemsets, rule covers, and for the derivation of other condensed representations). All the condensed representations mentioned in this paper are based on equality or inequality relations on itemset supports. Similar relation on support have been used by other authors in different contexts, e.g., for the approximation of the support of itemsets with negation in [39]. It might be interesting to consider whether the state-of-the-art in condensed representations enables or not to consider new data mining tasks based on, e.g., association rule with negated items.

The condensed representation principle can be applied for many other pattern domains and more sophisticated types of inductive queries. For instance, a similar concept of freeness has been studied for functional dependency discovery [42] and various condensed representations have been studied recently for frequent sequences, trees or graphs (see, e.g., [50,47,52]). It can be also studied w.r.t. quite general forms of inductive queries which are arbitrary boolean combinations of some primitive constraints. The results on using collections of version spaces as condensed representations for queries that involve arbitrary combinations of monotonic and anti-monotonic constraints provides an interesting starting point [26]. Also, the relationship between condensed representations and witnesses [35] might be explored.

As a conclusion, starting from efficient solutions to the Frequent Itemset Mining problem, the notion of condensed representation has been identified as a core

concept for inductive query optimization and its interest goes far beyond simple KDD processes based on itemsets, say standard association rule mining. We are pretty confident that this will become one major topic for research in the next few years, either for innovative applications of frequent pattern mining or for new pattern domains.

References

1. R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. ACM Int. Conf. on Management of Data SIGMOD'93*, pages 207–216, Washington, D.C., USA, May 1993. ACM Press.
2. R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proc. Int. Conf. on Very Large Data Bases VLDB'94*, pages 487–499, Santiago de Chile, Chile, Sept. 1994. Morgan Kaufmann.
3. Y. Bastide, N. Pasquier, R. Taouil, G. Stumme, and L. Lakhal. Mining minimal non-redundant association rules using frequent closed itemsets. In *Proc. Int. Conf. on Deductive and Object-Oriented Databases DOOD'00*, volume 1861 of *LNCS*, pages 972–986, London, UK, July 2000. Springer-Verlag.
4. Y. Bastide, R. Taouil, N. Pasquier, G. Stumme, and L. Lakhal. Mining frequent patterns with counting inference. *SIGKDD Explorations*, 2(2):66–75, Dec. 2000.
5. R. J. Bayardo. Efficiently mining long patterns from databases. In *Proc. ACM Int. Conf. on Management of Data SIGMOD'98*, pages 85–93, Seattle, USA, June 1998. ACM Press.
6. R. J. Bayardo, B. Goethals, and M. J. Zaki, editors. *Proc. Int. Workshop on Frequent Itemset Mining Implementations FIMI'04*, Brighton, UK, Nov. 2004.
7. C. Becquet, S. Blachon, B. Jeudy, J.-F. Boulicaut, and O. Gandrillon. Strong association rule mining for large gene expression data analysis: a case study on human SAGE data. *Genome Biology*, 12, 2002.
8. J. Besson, C. Robardet, J.-F. Boulicaut, and S. Rome. Constraint-based bi-set mining for biologically relevant pattern discovery in microarray data. *Intelligent Data Analysis*, 9(1):59–82, 2005.
9. F. Bonchi and C. Lucchese. On closed constrained frequent pattern mining. In *Proc. IEEE Int. Conf. on Data Mining ICDM'04*, pages 35–42, Brighton, UK, Nov. 2004. IEEE Computer Press.
10. 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.
11. J.-F. Boulicaut and A. Bykowski. Frequent closures as a concise representation for binary data mining. In *Proc. Pacific-Asia Conf. on Knowledge Discovery and Data Mining PAKDD'00*, volume 1805 of *LNAI*, pages 62–73, Kyoto, JP, Apr. 2000. Springer-Verlag.
12. J.-F. Boulicaut, A. Bykowski, and C. Rigotti. Approximation of frequency queries by mean of free-sets. In *Proc. Principles and Practice of Knowledge Discovery in Databases PKDD'00*, volume 1910 of *LNAI*, pages 75–85, Lyon, F, Sept. 2000. Springer-Verlag.
13. 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 journal*, 7(1):5–22, 2003.

14. J.-F. Boulicaut and B. Jeudy. Mining free itemsets under constraints. In *Proc. Int. Database Engineering and Application Symposium IDEAS'01*, pages 322–329, Grenoble, F, July 2001. IEEE Computer Press.
15. S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In *Proc. ACM Int. Conf. on Management of Data SIGMOD'97*, pages 255–264, Tucson, USA, May 1997. ACM Press.
16. A. Bykowski, T. Daurel, N. Méger, and C. Rigotti. Integrity constraints over association rules. In *Database Technologies for Data Mining - Discovering Knowledge with Inductive Queries*, volume 2682 of *LNCS*, pages 311–330. Springer-Verlag, 2004.
17. A. Bykowski and C. Rigotti. A condensed representation to find frequent patterns. In *Proc. ACM Symposium on Principles of Database Systems PODS'01*, pages 267–273, Santa Barbara, CA, USA, May 2001. ACM Press.
18. A. Bykowski and C. Rigotti. DBC: A condensed representation of frequent patterns for efficient mining. *Information Systems*, 28(8):949–977, 2003.
19. T. Calders. Deducing bounds on the support of itemsets. In *Database Technologies for Data Mining - Discovering Knowledge with Inductive Queries*, volume 2682 of *LNCS*, pages 214–233. Springer-Verlag, 2004.
20. T. Calders and B. Goethals. Mining all non derivable frequent itemsets. In *Proc. Principles and Practice of Knowledge Discovery in Databases PKDD'02*, volume 2431 of *LNAI*, pages 74–85, Helsinki, FIN, Aug. 2002. Springer-Verlag.
21. T. Calders and B. Goethals. Minimal k-free representations of frequent sets. In *Proc. Principles and Practice of Knowledge Discovery in Databases PKDD'03*, volume 2838 of *LNAI*, pages 71–82, Cavtat-Dubrovnik, HR, Sept. 2003. Springer-Verlag.
22. T. Calders and B. Goethals. Depth-first non derivable itemset mining. In *Proc. SIAM Int. Conf. on Data Mining SDM'05*, Newport Beach, USA, Apr. 2005.
23. B. Crémilleux and J.-F. Boulicaut. Simplest rules characterizing classes generated by delta-free sets. In *Proc. BCS Int. Conf. on Knowledge Based Systems and Applied Artificial Intelligence ES'02*, pages 33–46, Cambridge, UK, Dec. 2002. Springer-Verlag.
24. L. De Raedt. A perspective on inductive databases. *SIGKDD Explorations*, 4(2):69–77, 2003.
25. L. De Raedt. Towards query evaluation in inductive databases using version spaces. In *Database Technologies for Data Mining - Discovering Knowledge with Inductive Queries*, volume 2682 of *LNCS*, pages 117–134. Springer-Verlag, 2004.
26. L. De Raedt, M. Jaeger, S. D. Lee, and H. Mannila. A theory of inductive query answering. In *Proc. IEEE Int. Conf. on Data Mining ICDM'02*, pages 123–130, Maebashi City, JP, Dec. 2002. IEEE Computer Press.
27. J. Galambos and I. Simonelli. *Bonferroni-type Inequalities with Applications.* Springer, 1996.
28. B. Ganter and R. Wille. *Formal Concept Analysis, Mathematical Foundations.* Springer-Verlag, 1999.
29. A. Giacometti, D. Laurent, and C. T. Diop. Condensed representations for sets of mining queries. In *Database Technologies for Data Mining - Discovering Knowledge with Inductive Queries*, volume 2682 of *LNCS*, pages 250–269. Springer-Verlag, 2004.
30. B. Goethals, J. Muhonen, and H. Toivonen. Mining non derivable association rules. In *Proc. SIAM Int. Conf. on Data Mining SDM'05*, Newport Beach, USA, Apr. 2005.

31. B. Goethals and M. J. Zaki, editors. *Proc. Int. Workshop on Frequent Itemset Mining Implementations FIMI'03*, Melbourne, Florida, USA, Nov. 2003.
32. J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proc. ACM Int. Conf. on Management of Data SIGMOD'00*, pages 1 – 12, Dallas, Texas, USA, May 2000. ACM Press.
33. T. Imielinski and H. Mannila. A database perspective on knowledge discovery. *Communications of the ACM*, 39(11):58–64, 1996.
34. B. Jeudy and J.-F. Boulicaut. Using condensed representations for interactive association rule mining. In *Proc. Principles and Practice of Knowledge Discovery in Databases PKDD'02*, volume 2431 of *LNAI*, pages 225–236, Helsinki, FIN, Aug. 2002. Springer-Verlag.
35. D. Kifer, J. Gehrke, C. Bucila, and W. M. White. How to quickly find a witness. In *Proc. ACM Symposium on Principles of Database Systems PODS'03*, pages 272–283, San Diego, USA, June 2003. ACM Press.
36. M. Kryszkiewicz. Concise representation of frequent patterns based on disjunction-free generators. In *Proc. IEEE Int. Conf. on Data Mining ICDM'01*, pages 305–312, San Jose, USA, Nov. 2001. IEEE Computer Press.
37. M. Kryszkiewicz and M. Gajek. Concise representation of frequent patterns based on generalized disjunction-free generators. In *Proc. Pacific-Asia Conf. on Knowledge Discovery and Data Mining PAKDD'02*, volume 2336 of *LNCS*, pages 159–171, Taipei, Taiwan, 2002. Springer-Verlag.
38. B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rules mining. In *Proc. Int. Conf. on Knowledge Discovery and Data Mining KDD'98*, pages 80–86, New York, USA, 1998. AAAI Press.
39. H. Mannila and H. Toivonen. Multiple uses of frequent sets and condensed representations. In *Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining KDD'96*, pages 189–194, Portland, USA, 1996. AAAI Press.
40. H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1(3):241–258, 1997.
41. T. Mielikäinen. *Summarization Techniques for Pattern Collections in Data Mining.* PhD thesis, University of Helsinki, Department of Computer Science, Ph.D. thesis Report A-2005-1, 2005.
42. N. Novelli and R. Cicchetti. Mining functional and embedded dependencies using free sets. In *Actes Bases de Données Avancées BDA'00*, pages 201–220, 2000.
43. N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Pruning closed itemset lattices for association rules. In *Actes Bases de Données Avancées BDA'98*, Hammamet, Tunisie, Oct. 1998.
44. N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Efficient mining of association rules using closed itemset lattices. *Information Systems*, 24(1):25–46, Jan. 1999.
45. J. Pei, J. Han, and R. Mao. CLOSET an efficient algorithm for mining frequent closed itemsets. In *Proc. SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery DMKD'00*, Dallas, USA, May 2000.
46. G. Piatetsky-Shapiro. Discovery, analysis, and presentation of strong rules. In *Knowledge Discovery in Databases*, pages 229–248. AAAI Press, 1991.
47. U. Rückert and S. Kramer. Generalized version space trees. In *Proc. Int. Workshop on Inductive Databases KDID'03*, pages 119–129, Cavtat-Dubrovnik, HR, 2003. Rudjer Boskovic Institute, Zagreb, HR.
48. A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databases. In *Proc. Int. Conf. on Very Large Data Bases VLDB'95*, pages 432 – 444, Zürich, CH, Sept. 1995. Morgan Kaufmann.

49. H. Toivonen. Sampling large databases for association rules. In *Proc. Int. Conf. on Very Large Data Bases VLDB'96*, pages 134–145, Mumbay, India, Sept. 1996. Morgan Kaufmann.
50. J. Wang and J. Han. BIDE: Efficient mining of frequent closed sequences. In *Proc. IEEE Int. Conf. on Data Engineering ICDE'04*, pages 79–90, Boston, USA, Apr. 2004. IEEE Computer Press.
51. R. Wille. Restructuring lattice theory: An approach based on hierarchies of concepts. In I. Rival, editor, *Ordered Sets*, pages 445–470. Reidel, Dordrecht-Boston, 1982.
52. A. Xu and H. Lei. LCGMiner: Levelwise closed graph pattern mining from large databases. In *Proc. Int. Conf. on Scientific and Statistical Database Management SSDBM'04*, pages 421–422, Santorini Island, EL, June 2004. IEEE Computer Press.
53. M. J. Zaki. Generating non-redundant association rules. In *Proc. ACM Int. Conf. on Knowledge Discovery and Data Mining SIGKDD'00*, pages 34–43, Boston, USA, Aug. 2000. ACM Press.
54. M. J. Zaki and C.-J. Hsiao. CHARM: An efficient algorithm for closed itemset mining. In *Proc. SIAM Int. Conf. on Data Mining SDM'02*, Arlington, USA, Apr. 2002.
55. M. J. Zaki and M. Ogihara. Theoretical foundations of association rules. In *Proc. SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery DMKD'98*, pages 1–8, June 1998.

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.

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 φ is more general (resp. more specific) than θ, 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 θ and φ 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 ϵ 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. φ 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 δ, a database $\mathbf{d}$ and a predicate Q, an error measure ψ 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 φ.
- 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 φ.

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* $\mathbf{d} = \{r_1, r_2, r_3\}$ *over the schema* $\mathbf{R} = \{R_1, R_2, R_3\}$, *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* $\mathbf{d}$ *be a database over a schema* $\mathbf{R}$ *and* I *the set of satisfied INDs in* $\mathbf{d}$. *Let* $I_{R \to S}$ *be INDs from* R *to* S. *Then*

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

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* $\mathbf{R}$ *be a database schema and* (R, S) *a couple of relation schema of* $\mathbf{R}$. *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.

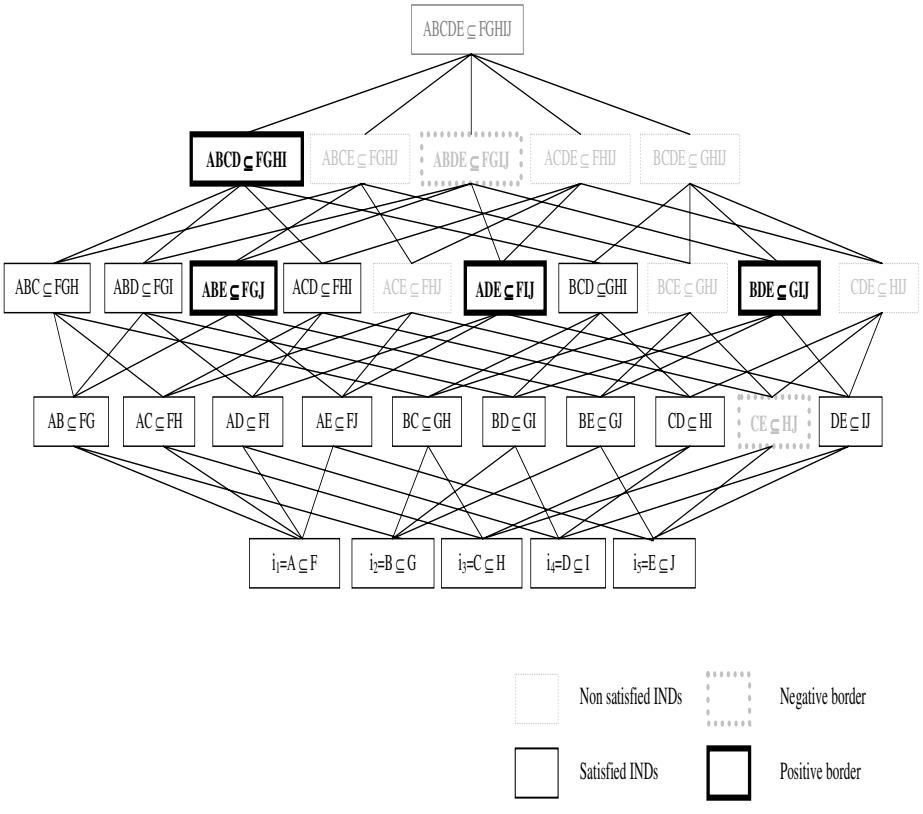


Fig. 1. A subset of the search space for INDs

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.

Computation of Mining Queries: An Algebraic Approach

Cheikh Talibouya Diop[1,4], Arnaud Giacometti[1], Dominique Laurent[2], and Nicolas Spyratos[3]

[1] LI, Université de Tours, 41000 Blois, France
giaco@univ-tours.fr
[2] LICP, Université de Cergy-Pontoise, 95 302 Cergy-Pontoise Cedex, France
dominique.laurent@dept-info.u-cergy.fr
[3] LRI, Université Paris 11, 91405 Orsay Cedex, France
spyratos@lri.fr
[4] Université Gaston Berger, Saint-Louis, Senegal
cdiop@ugb.sn

Abstract. Mining frequent queries often requires the repeated execution of some extraction algorithm for different values of the support, as well as for different source datasets. This is an expensive process, even if we use the best existing algorithms. Hence the need for *iterative* mining, whereby mining results already obtained are re-used to accelerate subsequent steps in the mining process.

In this paper, we present an approach for the iterative mining of frequent queries. Our approach is based on the notion of *mining context,* where a mining context is a set of queries over the same schema. We define operations on mining contexts, based on the standard relational algebra, and we also introduce new operators, one of which for computing frequent queries.

We first study the properties of the operators, then we consider particular mining contexts using biases for which frequent queries can be computed using any level-wise algorithm. Iterative mining is obtained by combining these particular contexts using our set of operations. We have implemented our approach and conducted experiments that show its efficiency in mining frequent queries.

1 Introduction

Association rule mining often requires the repeated execution of some extraction algorithm for different values of the support and confidence thresholds, as well as for different source datasets. This is an expensive process, even if we use the best existing algorithms. Hence the need for *iterative* mining, whereby mining results already obtained are re-used to accelerate subsequent steps in the mining process.

In this paper, we present an approach for the iterative mining of frequent queries that allows to generate *multi-dimensional* association rules, *i.e.,* association rules that involve one or more tables from a given relational database [11].

J.-F. Boulicaut et al. (Eds.): Constraint-Based Mining, LNAI 3848, pp. 102–126, 2005.

However, we do not consider the phase of generating the rules from the frequent queries.

In our formalism, frequent query mining takes place in a general framework in which the notion of *mining context* allows to specify the queries among which frequent queries are to be mined. Iterative mining is obtained by combining contexts using known relational algebra operations as well as new operators to be defined shortly. We call such combinations *mining queries,* in much the same way as in relational databases, queries are defined as combinations of relations using the operations of the relational algebra.

The contribution of the paper is twofold: First, we define and study operations on contexts that allow to consider complex mining queries and possible optimization techniques. Then, we focus on particular mining contexts from which frequent queries can be computed using standard level-wise algorithms, such as Apriori ([1]). For these mining contexts we study possible optimizations based on the properties of the operators introduced in the paper. Next, we present the main concepts of our approach.

1.1 Contexts and Biases

Whereas most formalisms to represent multi-dimensional association rules are based on Dalalog and first-order logic ([4,11,19]), we shall use the relational algebra to combine mining contexts.

In our approach, a context specifies a set of queries having the same schema. We extend the operators of the relational algebra to contexts and we introduce new operators that can be used for many purposes, for instance for computing frequent queries or for selecting queries according to criteria involving other queries. In this paper, we focus on the problem of mining frequent queries. Roughly speaking, given a query q, the support of q in an instance I of the underlying database is the ratio of the cardinality of the answer to q in I with respect to the cardinality of the answer in I to another query, that we call a *reference query.*

We focus on a specific set of mining contexts, whose definition is based on the notion of *mining bias.* A mining bias specifies the table in which frequent queries are to be mined, as well as selection conditions that appear in these queries. Let us explain these concepts through an example that will serve as a running example throughout the paper.

Example 1. Consider a database $DBSales$ containing the following tables:

- $Cust(Cid, Cjob, Caddr)$, where Cid, $Cjob$ and $Caddr$ are the ids, jobs and home addresses of customers, respectively,
- $Prod(Pid, Ptype)$, where Pid and $Ptype$ are the ids and the types of products, respectively,
- $Store(Sid, Sname, Saddr)$, where Sid, $Sname$ and $Saddr$ are the ids, names and addresses of stores, respectively,

– $Sales(Cid, Pid, Sid, Date)$, where a tuple $\langle c, p, s, d\rangle$ in the $Sales$ table means that customer c issued a transaction concerning product p in the store s on date d.

Assume that a user is interested in mining queries involving the jobs and addresses of customers and the types of products they buy. We can specify this by the relational expression $b_1 = Cust \bowtie Sales \bowtie Prod$.

Now, assume that, more precisely, this user is interested only in queries dealing with customers whose job is professor or lawyer, and who buy tea or milk. Then we can consider the set of selection conditions Σ_1 defined by: $\Sigma_1 = \{Cjob = Professor, Cjob = Lawyer, Ptype = Tea, Ptype = Milk\}$.

The pair $\mathcal{B}_1 = \langle b_1, \Sigma_1\rangle$ is an example of a *mining bias*, or simply a *bias*. The query b_1 and the set of selection conditions Σ_1 are called the *basis* and the *domain* of $\mathcal{B}_1$, respectively. The bias $\mathcal{B}_1$ specifies a context, denoted by $\mathcal{C}(\mathcal{B}_1)$, that contains all queries of the form $\sigma_S(b_1)$ where S is a conjunction of some conditions in Σ_1. The queries $\sigma_{Cjob=Lawyer}(b_1)$ and $\sigma_{Cjob=Professor \wedge Ptype=Tea}(b_1)$ are examples of queries in $\mathcal{C}(\mathcal{B}_1)$.

Let us now assume that supports of queries in $\mathcal{C}(\mathcal{B}_1)$ are to be computed with respect to all customers. In this case, we consider the query $r = \pi_{Cid}(Cust)$ as the *reference query*, and for every instance I of $DBSales$ and every query q in $\mathcal{C}(\mathcal{B}_1)$, we define the *support* of q in I as the ratio $|(\pi_{Cid}(r \bowtie q))(I)|/|r(I)|$. If this ratio is greater than a given support threshold then we say that q is *frequent* in I. We note that the set of all frequent queries is a subset of $\mathcal{C}(\mathcal{B}_1)$, and thus is itself a context.

Following this observation, we define an operator, denoted by $freq_I$, that takes as input a context $\mathcal{C}$, a support threshold α and a reference query r, and outputs a set denoted by $freq_I(\mathcal{C}, \alpha, r)$ which contains queries of $\mathcal{C}$ that are frequent in I with respect to α and r. □

1.2 Composition of Mining Contexts

One of the main contributions of this paper is to show the following: Suppose that a number of contexts of the form $freq_I(\mathcal{C}_i, \alpha_i, r)$ $(i = 1, \ldots, k)$ have already been computed and stored. Then any new context of the form $freq_I(\mathcal{C}, \alpha, r)$ where $\mathcal{C}$ is defined as a composition of the contexts $\mathcal{C}_i$ can be performed more efficiently than if it were performed directly, *i.e.,* without reference to the contexts $\mathcal{C}_i$. To this end, we define operations over contexts similar to those of the relational algebra, and we study their properties. The following example shows how contexts can be composed to create new contexts.

Example 2. In the database $DBSales$ of Example 1, assume that we want to discover relationships between the jobs of customers living in $Paris$ and the types of products they buy. Denoting by $\mathcal{C}_1$ the context $\mathcal{C}(\mathcal{B}_1)$ given previously, a new context $\mathcal{C}'_1 = \sigma_{Caddr=Paris}(\mathcal{C}_1)$ can be defined by applying the selection condition $(Caddr = Paris)$ on the queries of $\mathcal{C}_1$.

Given the new context $\mathcal{C}'_1$, let $q'_1 = \sigma_S(\sigma_{Caddr=Paris}(b_1))$ be a query of $\mathcal{C}'_1$. We can see easily that the answer to q'_1 is always included in the answer to the

query $q_1 = \sigma_S(b_1)$ of $\mathcal{C}_1$. More generally, it can be seen that every query q'_1 of $\mathcal{C}'_1$ is contained in a query q_1 of $\mathcal{C}_1$ (in the sense of query containment [20]).

Therefore, the support of every query q'_1 of $\mathcal{C}'_1$ is less than or equal to the support of a query q_1 of $\mathcal{C}_1$. It follows that if a query q'_1 of $\mathcal{C}'_1$ is frequent, then it is contained in a frequent query q_1 of $\mathcal{C}_1$. Conversely, if a query q'_1 of $\mathcal{C}'_1$ is not contained in a frequent query q_1 of $\mathcal{C}_1$, then it can not be frequent.

As a consequence, if we assume that the frequent queries of $\mathcal{C}_1$ are stored, this property can be used to prune some of the candidate queries when computing the frequent queries of $\mathcal{C}'_1$. Moreover, this new pruning step can be done without any access to the database, implying that the computation of the frequent queries of $\mathcal{C}'_1$ can be accelerated. □

In the remaining of the paper, we show how the above example can be generalized and how the computation of frequent queries in composed contexts can be optimized.

1.3 Related Work

So far, several *SQL* based query languages have been proposed in order to facilitate the specification of mining queries [2,9,10,14,19]. To our knowledge, the only language that allows to combine mining queries is Mine Rule [2]. However, in [2], the authors consider a single table and concentrate on selections for optimizing mining queries based on previously computed mining queries. Therefore, our approach is more general than that of [2], and is based on properties of the relational algebra, instead of being based on a specific language such as Mine Rule.

We also note that our notion of context is close to the notion of query flock introduced in [19]. However, the objective of [19] is not to store and use the results of mining queries in order to optimize new mining queries, but rather to optimize the computation of a single mining query using query containment.

In [12], the authors propose an algebra for the optimization of inductive queries. However, they only consider set-theoretic operations, and focus on monotonic queries, in order to design a strategy for the optimization of a single query. It turns out that our framework is more general than that of [12], and our optimization techniques are different than that of [12].

Storing and using the results of mining queries in order to optimize new mining queries has also been studied in [15,16,17]. However, these approaches compare only mining queries defined over a single reference and a single basis (using our terminology), and they do not propose any composition operations as we do.

In [15,16], the authors consider the traditional case where the source database contains only one table (the basis in our terminology), whose rows contain sets of items. In [15], mining queries are compared with respect to three criteria: the support thresholds, the selection conditions on the source dataset and the constraints on the patterns. Moreover, the authors use materialized views to store the results of mining queries already computed.

In [16], mining queries are mainly compared with respect to their support thresholds, and the authors propose different caching strategies aiming at storing

only the most useful frequent itemsets that can be used to answer new mining queries. In [17], the same authors present an extension of their approach, whereby the source database is not the traditional table, but a data cube.

In this paper, we focus on the case where the source database contains *multiple tables*. It follows that we can specify and compare mining queries with *different bases* or with *different references*.

The rest of the paper is organized as follows: In Section 2, we formally define the notions of mining context and of mining bias, and in Section 3 we extend the operations of the relational algebra to mining contexts and we study their properties. The computation of frequent queries is the subject of Section 4, where we define and study the appropriate operator, which we call the $freq$ operator. Section 5 deals with mining contexts that are defined using mining biases: we first show that in this particular case, frequent queries are computed using standard level-wise algorithms, and then we show how to improve the efficiency of frequent query discovery. In Section 6, we outline the implementation of our approach and report experiments. Section 7 concludes the paper and outlines further research directions based on this work.

2 Mining Context and Mining Bias

2.1 Background

The formalism used in this paper is based on the relational model of databases ([20]). We recall that a relational database schema is a set of relation names, each of which is associated with a set of attributes. We call schema of a relation name R the set of attributes associated with R and we denote it by $sch(R)$. For each attribute A, the possible values for A belong to a specific set of values, called the domain of A, denoted by $dom(A)$.

In the remainder of this paper, we assume a fixed database schema, $DB = \{R_1, R_2, \ldots, R_n\}$, and we call instance of DB any set of relations $I = \{r_1, r_2, \ldots, r_n\}$, where r_i is a relation over $sch(R_i)$, for every $i = 1, 2, \ldots, n$. Given a relational expression (or query) q, over DB, we call schema of q, denoted by $sch(q)$, the set of attributes over which q is defined. Moreover, we denote by $q(I)$ the answer to q in I.

As in [3], if q_1 and q_2 are two queries such that $sch(q_1) = sch(q_2)$, we say that q_1 is *contained* in q_2, or that q_1 is *more specific* than q_2, denoted by $q_1 \sqsubseteq q_2$, if for every instance I we have $q_1(I) \subseteq q_2(I)$. Queries q_1 and q_2 are said *equivalent,* denoted by $q_1 \equiv q_2$, if both $q_1 \sqsubseteq q_2$ and $q_2 \sqsubseteq q_1$ hold.

All results reported in this paper are up to query-equivalence, *i.e.,* each query q is considered to be a *representative* of its equivalence class.

2.2 Basic Definitions

In this section, we define formally the notions of *mining context* and *mining bias.*

Definition 1 - Mining Context. *Let X be a relation schema. A* mining context *$\mathcal{C}$, or simply a* context, over *X is a finite set of queries, all having X as their schema. The schema X is called* the schema of *$\mathcal{C}$, and is denoted by $sch(\mathcal{C})$.*

Contexts over the same schema can be compared according to two orderings: (1) set inclusion, and (2) the following ordering based on query-containment.

Definition 2 - Comparison of Contexts. *Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two contexts of same schema. $\mathcal{C}_1$ is* weakly-included *in $\mathcal{C}_2$, denoted by $\mathcal{C}_1 \sqsubseteq \mathcal{C}_2$, if for every q_1 in $\mathcal{C}_1$, there exists q_2 in $\mathcal{C}_2$ such that $q_1 \sqsubseteq q_2$.*

Clearly, if $\mathcal{C}_1 \subseteq \mathcal{C}_2$, then $\mathcal{C}_1 \sqsubseteq \mathcal{C}_2$, whereas the converse does not hold.

Definition 3 - Mining Bias. *A* mining bias *$\mathcal{B}$, or simply a* bias, *is a pair $\mathcal{B} = \langle b, \Sigma \rangle$ where:*

- *b is a relational expression called the* basis *of $\mathcal{B}$.*
- *Σ is a set of atomic selection conditions of the form $A = a$ where A is an attribute in $sch(b)$ and a is a value in $dom(A)$. Σ is called the* domain *of $\mathcal{C}$.*

Denoting by $\bot$ and $\top$ the selection conditions that are always false and true, respectively, let Σ^* be the following set of conjunctive selection conditions:
$\Sigma^* = \{\bot, \top\} \cup \{(A_1 = a_1) \wedge \ldots \wedge (A_k = a_k) \mid (\forall i = 1, \ldots, k)(A_i = a_i) \in \Sigma$ *and* $(\forall i, j \in \{1, \ldots, k\})(i \neq j \Rightarrow A_i \neq A_j)\}$.
The context of *$\mathcal{B}$, denoted by $\mathcal{C}(\mathcal{B})$, is defined by: $\mathcal{C}(\mathcal{B}) = \{\sigma_S(b) \mid S \in \Sigma^*\}$.*

According to Definition 3 above, we have $\sigma_\top(b) = b$ and $\sigma_\bot(b) = \emptyset$ (where $\emptyset$ refers to any query over $sch(b)$ whose answer is empty in any instance of DB). The roles of $\bot$ and $\top$ are explained later on in the paper (see Section 5).

We refer to Example 1 and Example 2 for examples of mining contexts and mining biases. It is easy to see that, in these examples, we have $\mathcal{C}_1' \sqsubseteq \mathcal{C}_1$. Indeed, every query q' of $\mathcal{C}_1'$ is of the form $\sigma_{Caddr=Paris}(q)$ where q is in $\mathcal{C}_1$, which entails that there exists a query q in $\mathcal{C}_1$ such that $q' \sqsubseteq q$. Note however that we do not have $\mathcal{C}_1' \subseteq \mathcal{C}_1$.

3 Operations on Contexts

In this section, we first define operations on contexts by extending the operators of the relational algebra, and we introduce new operators, called *τ-reduction* and *γ-reduction*. We then study the properties of these operators. Clearly, the operations defined in this section combine contexts in much the same way as the operations of the relational algebra combine relations. Therefore, these operations on contexts can be though of as means to specify what we call *mining queries*. In other words, in our approach, a mining query is an algebraic expression on contexts.

3.1 Unary Operations on Contexts

Definition 4 *Let $\mathcal{C}$ be a context.*

1. *Given a set of attributes X such that $X \subseteq sch(\mathcal{C})$, the* projection *of $\mathcal{C}$ over X, denoted by $\pi_X(\mathcal{C})$, is a context over X, defined by: $\pi_X(\mathcal{C}) = \{\pi_X(q) \mid q \in \mathcal{C}\}$.*

2. *Given a selection condition S involving only attributes in $sch(\mathcal{C})$, the* selection *of $\mathcal{C}$ with respect to S, denoted by $\sigma_S(\mathcal{C})$, is a context over $sch(\mathcal{C})$, defined by: $\sigma_S(\mathcal{C}) = \{\sigma_S(q) \mid q \in \mathcal{C}\}$.*
3. *Given two attributes $A \in sch(\mathcal{C})$ and $B \notin sch(\mathcal{C})$, the* renaming *of attribute A by B in $\mathcal{C}$, denoted by $\rho_{B \leftarrow A}(\mathcal{C})$, is a context over $(sch(\mathcal{C}) \setminus \{A\}) \cup \{B\}$, defined by: $\rho_{B \leftarrow A}(\mathcal{C}) = \{\rho_{B \leftarrow A}(q) \mid q \in \mathcal{C}\}$.*
4. *Given a selection condition S such that $att(S) \subseteq sch(\mathcal{C})$, the* τ-reduction *of $\mathcal{C}$ with respect to S, denoted by $\tau_S(\mathcal{C})$, is a context over $sch(\mathcal{C})$, defined by: $\tau_S(\mathcal{C}) = \{q \in \mathcal{C} \mid \sigma_S(q) = q\}$.*
5. *Given an operator $R \in \{\sqsubseteq, \sqsupseteq, =\}$ and a query q such that $sch(q) = sch(\mathcal{C})$, the* γ-reduction *of $\mathcal{C}$ with respect to R and q, denoted by $\gamma_{Rq}(\mathcal{C})$, is a context over $sch(\mathcal{C})$, defined by: $\gamma_{Rq}(\mathcal{C}) = \{q' \in \mathcal{C} \mid q' \ R \ q\}$.*

We illustrate these operations on contexts in the following example.

Example 3. First, if we consider the contexts $\mathcal{C}_1$ of Example 1 and $\mathcal{C}'_1$ of Example 2, then we have $\mathcal{C}'_1 = \sigma_{Caddr=Paris}(\mathcal{C}_1)$. Moreover, recalling that $\mathcal{C}_1$ is meant to contain queries dealing with customers whose job is either lawyer or professor and the types of items they buy, it can be seen that attributes such as *Date* or *Sid* have not to be considered. This can be achieved by considering the projection $\pi_X(\mathcal{C}_1)$ of $\mathcal{C}_1$ where $X = \{Cid, Cjob, Caddr, Pid, Ptype\}$. As will be seen in the paper, considering $\pi_X(\mathcal{C}_1)$ instead of $\mathcal{C}_1$ has no impact on the frequent queries computed with respect to the set of all customers.

On the other hand, considering the selection $Ptype = Tea$, the context $\tau_{Ptype=Tea}(\mathcal{C}_1)$ contains all queries q of $\mathcal{C}_1$ that are equal to $\sigma_{Ptype=Tea}(q)$. More precisely, as in Example 1, consider the queries $q_1 = \sigma_{Cjob=Lawyer}(b_1)$ and $q_2 = \sigma_{Cjob=Professor \wedge Ptype=Tea}(b_1)$. Then, clearly, $q_1 \neq \sigma_{Ptype=Tea}(q_1)$ whereas $q_2 = \sigma_{Ptype=Tea}(q_2)$, thus q_2 is a query of $\tau_{Ptype=Tea}(\mathcal{C}_1)$ and q_1 is not. We note that in this case, $\sigma_{Ptype=Tea}(\mathcal{C}_1)$ contains both queries $\sigma_{Ptype=Tea}(q_1)$ and $\sigma_{Ptype=Tea}(q_2)$ (which is equal to q_2), showing that $\sigma_S(\mathcal{C})$ and $\tau_S(\mathcal{C})$ are in general different contexts.

Let us now consider the query $q = \sigma_{Cjob=Professor \vee Ptype=Tea}(b_1)$. Then $\gamma_{\sqsubseteq q}(\mathcal{C}_1)$ contains all queries in $\mathcal{C}_1$ that deal either with tea or with professors. For instance the query q_2 above belongs to $\gamma_{\sqsubseteq q}(\mathcal{C}_1)$, whereas q_1 does not. □

We note from the previous example that the operations of τ- and γ-reduction can be seen as selection operations on contexts, whereas the operations of projection, selection and renaming define contexts containing "new" queries obtained through the corresponding operations of the relational algebra. Therefore, τ- and γ-reduction are useful for post-processing the results of previously computed mining queries. The following proposition states properties of the operations just defined.

Proposition 1. *Let $\mathcal{C}$ be a context.*

1. *For every selection condition S, $\tau_S(\mathcal{C}) \subseteq \mathcal{C}$, $\tau_S(\tau_S(\mathcal{C})) = \tau_S(\mathcal{C})$, $\tau_S(\sigma_S(\mathcal{C})) = \sigma_S(\mathcal{C})$ and $\sigma_S(\tau_S(\mathcal{C})) = \tau_S(\mathcal{C})$.*

2. *For every query q such that $sch(q) = sch(\mathcal{C})$, and every operator $R \in \{\sqsubseteq, \sqsupseteq, =\}$, $\gamma_{Rq}(\mathcal{C}) \subseteq \mathcal{C}$.*
3. *For all queries q_1 and q_2 of schema $sch(\mathcal{C})$, if $q_1 \sqsubseteq q_2$, then $\gamma_{\sqsubseteq q_1}(\mathcal{C}) \subseteq \gamma_{\sqsubseteq q_2}(\mathcal{C})$ and $\gamma_{\sqsupseteq q_2}(\mathcal{C}) \supseteq \gamma_{\sqsupseteq q_1}(\mathcal{C})$.*

PROOF: 1. Let S be a selection condition. By definition, we have $\tau_S(\mathcal{C}) \subseteq \mathcal{C}$. For every query q in $\tau_S(\mathcal{C})$, we have $q = \sigma_S(q)$. Thus, we have $\tau_S(\tau_S(\mathcal{C})) = \tau_S(\mathcal{C})$. On the other hand, for every query $q \in \sigma_S(\mathcal{C})$, there exists $q' \in \mathcal{C}$ such that $q = \sigma_S(q')$. It follows that $\sigma_S(q) = \sigma_S(\sigma_S(q')) = \sigma_S(q') = q$, which shows that $\sigma_S(q) = q$ and $\tau_S(\sigma_S(\mathcal{C})) = \sigma_S(\mathcal{C})$.
On the other hand, it is easy to see that the equality $\sigma_S(\tau_S(\mathcal{C})) = \tau_S(\mathcal{C})$ holds, based on Definition 4(4).

2. This point follows directly from the definition of γ-reduction.

3. Let q_1 and q_2 be two queries over $sch(\mathcal{C})$ such that $q_1 \sqsubseteq q_2$. Let $q \in \gamma_{\sqsubseteq q_1}(\mathcal{C})$. We have $q \sqsubseteq q_1$. Since $q_1 \sqsubseteq q_2$, we have $q \sqsubseteq q_2$, which shows that $q \in \gamma_{\sqsubseteq q_2}(\mathcal{C})$ and thus, $\gamma_{\sqsubseteq q_1}(\mathcal{C}) \subseteq \gamma_{\sqsubseteq q_2}(\mathcal{C})$. We can prove in the same way that $\gamma_{\sqsupseteq q_2}(\mathcal{C}) \subseteq \gamma_{\sqsupseteq q_1}(\mathcal{C})$. Thus, the proof is complete. ◇

3.2 Binary Operations on Contexts

A context being a set of queries, we consider the standard set-theoretic operators $\cup$, $\cap$ and $\setminus$ over contexts. Moreover, based on these set-theoretic operations, we define new operations on contexts in a relation-wise manner as follows.

Definition 5 *Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two contexts.*

1. *The join of $\mathcal{C}_1$ and $\mathcal{C}_2$, denoted by $\mathcal{C}_1 \bowtie \mathcal{C}_2$, is a context over $sch(\mathcal{C}_1) \cup sch(\mathcal{C}_2)$ defined by:*
   if $\mathcal{C}_1 \neq \emptyset$ and $\mathcal{C}_2 \neq \emptyset$ then $\mathcal{C}_1 \bowtie \mathcal{C}_2 = \{q_1 \bowtie q_2 \mid q_1 \in C_1$ and $q_2 \in C_2\}$
   otherwise, $\mathcal{C}_1 \bowtie \mathcal{C}_2 = \emptyset$.
2. *If $sch(\mathcal{C}_1) = sch(\mathcal{C}_2) = X$, the union of $\mathcal{C}_1$ and $\mathcal{C}_2$, denoted by $\mathcal{C}_1 \sqcup \mathcal{C}_2$, is a context over X defined by:*
   if $\mathcal{C}_1 \neq \emptyset$ and $\mathcal{C}_2 \neq \emptyset$ then $\mathcal{C}_1 \sqcup \mathcal{C}_2 = \{q_1 \cup q_2 \mid q_1 \in \mathcal{C}_1$ and $q_2 \in \mathcal{C}_2\}$
   otherwise, if $\mathcal{C}_1 = \emptyset$ then $\mathcal{C}_1 \sqcup \mathcal{C}_2 = \mathcal{C}_2$ else $\mathcal{C}_1 \sqcup \mathcal{C}_2 = \mathcal{C}_1$.
3. *If $sch(\mathcal{C}_1) = sch(\mathcal{C}_2) = X$, the intersection of $\mathcal{C}_1$ and $\mathcal{C}_2$, denoted by $\mathcal{C}_1 \sqcap \mathcal{C}_2$, is a context over X defined by:*
   if $\mathcal{C}_1 \neq \emptyset$ and $\mathcal{C}_2 \neq \emptyset$ then $\mathcal{C}_1 \sqcap \mathcal{C}_2 = \{q_1 \cap q_2 \mid q_1 \in \mathcal{C}_1$ and $q_2 \in \mathcal{C}_2\}$
   otherwise, $\mathcal{C}_1 \sqcap \mathcal{C}_2 = \emptyset$.
4. *If $sch(\mathcal{C}_1) = sch(\mathcal{C}_2) = X$, the difference of $\mathcal{C}_1$ and $\mathcal{C}_2$, denoted by $\mathcal{C}_1 \ominus \mathcal{C}_2$, is a context over X defined by:*
   if $\mathcal{C}_1 \neq \emptyset$ and $\mathcal{C}_2 \neq \emptyset$ then $\mathcal{C}_1 \ominus \mathcal{C}_2 = \{q_1 \setminus q_2 \mid q_1 \in \mathcal{C}_1$ and $q_2 \in \mathcal{C}_2\}$
   otherwise, $\mathcal{C}_1 \ominus \mathcal{C}_2 = \mathcal{C}_1$.

The following example illustrates the operations defined above.

Example 4. Considering again the database $DBSales$ of Example 1, we recall that the context $\mathcal{C}_1 = \mathcal{C}(\mathcal{B}_1)$ is defined from the bias $\mathcal{B}_1 = \langle b_1, \Sigma_1 \rangle$ where $b_1 =$

$Cust \bowtie Sales \bowtie Prod$ and $\Sigma_1 = \{Cjob = Professor, Cjob = Lawyer, Ptype = Tea, Ptype = Milk\}$.

The context $\sigma_{Ptype=Tea \wedge P=Milk}(\pi_X(\mathcal{C}_1) \bowtie \rho_{P \leftarrow Ptype}(\pi_X(\mathcal{C}_1)))$, where $X = sch(b_1) \setminus \{Pid\}$, contains queries that refer to those customers who are professors or lawyers and who buy tea and milk in the same store and on the same date.

On the other hand, the context $\sigma_{Ptype=Tea}(\mathcal{C}_1) \sqcap \sigma_{Caddr=Paris}(\mathcal{C}_1)$ contains queries that refer to those customers living in Paris, and who buy tea.

Note that this context should not be confused with the context $\sigma_{Ptype=Tea}(\mathcal{C}_1) \cap \sigma_{Caddr=Paris}(\mathcal{C}_1)$, which is empty. Indeed, if q is a query in $\sigma_{Ptype=Tea}(\mathcal{C}_1) \cap \sigma_{Caddr=Paris}(\mathcal{C}_1)$, then q is in $\sigma_{Ptype=Tea}(\mathcal{C}_1)$, meaning that q is of the form $\sigma_{Ptype=Tea \wedge S}(b_1)$ where $S \in \Sigma_1^*$. Since $Caddr = Paris$ is not in Σ_1, q cannot be in $\sigma_{Caddr=Paris}(\mathcal{C}_1)$. □

The following proposition shows that the operators just defined behave similarly as their standard conterpart over sets.

Proposition 2. *Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be contexts.*

1. $\pi_{X_i}(\mathcal{C}_1 \bowtie \mathcal{C}_2) \sqsubseteq \mathcal{C}_i$, *where* $X_i = sch(\mathcal{C}_i)$ *and* $i = 1, 2$.
2. $\mathcal{C}_1 \ominus \mathcal{C}_2 \sqsubseteq \mathcal{C}_1$.
3. *For* $i = 1, 2$, $\mathcal{C}_1 \sqcup \mathcal{C}_2 \sqsupseteq \mathcal{C}_i$ *and* $\mathcal{C}_1 \sqcap \mathcal{C}_2 \sqsubseteq \mathcal{C}_i$.

PROOF: Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be contexts. We first note that if $\mathcal{C}_1$ or $\mathcal{C}_2$ is empty, then the proof follows immediately from Definition 5 above. Let us assume that $\mathcal{C}_1$ and $\mathcal{C}_2$ are not empty.

1. Let $q \in \pi_{X_i}(\mathcal{C}_1 \bowtie \mathcal{C}_2)$ where $X_i = sch(\mathcal{C}_i)$ $(i = 1, 2)$. There exist $q_1 \in \mathcal{C}_1$ and $q_2 \in \mathcal{C}_2$ such that $q = \pi_{X_i}(q_1 \bowtie q_2)$. Moreover, we have $\pi_{X_i}(q_1 \bowtie q_2) \sqsubseteq q_i$. Thus, there exists a query $q_i \in \mathcal{C}_i$ such that $q \sqsubseteq q_i$, which shows that $\pi_{X_i}(\mathcal{C}_1 \bowtie \mathcal{C}_2) \sqsubseteq \mathcal{C}_i$.

2. Let $q \in \mathcal{C}_1 \ominus \mathcal{C}_2$. There exist $q_1 \in \mathcal{C}_1$ and $q_2 \in \mathcal{C}_2$ such that $q = q_1 \setminus q_2$. Moreover, we have $(q_1 \setminus q_2) \sqsubseteq q_1$. Thus, there exists a query $q_1 \in \mathcal{C}_1$ such that $q \sqsubseteq q_1$, which shows that $\mathcal{C}_1 \ominus \mathcal{C}_2 \sqsubseteq \mathcal{C}_1$.

3. Let $q \in \mathcal{C}_1$. Since $\mathcal{C}_2 \neq \emptyset$, let $q_2 \in \mathcal{C}_2$. Then, we have $q = q_1 \cup q_2 \in \mathcal{C}_1 \sqcup \mathcal{C}_2$. Moreover, $q_1 \sqsubseteq (q_1 \cup q_2)$. Thus, there exists a query $q \in \mathcal{C}_1 \cup \mathcal{C}_2$ such that $q_1 \sqsubseteq q$, which shows that $\mathcal{C}_1 \sqsubseteq \mathcal{C}_1 \sqcup \mathcal{C}_2$. The proof that $\mathcal{C}_1 \sqcap \mathcal{C}_2 \sqsubseteq \mathcal{C}_i$ is similar, and thus, is omitted. Therefore, the proof is complete. ◇

It is important to note that these properties can be used for query optimization in much the same way as in relational databases, where queries are optimized based on the properties of the relational algebra.

4 The Freq Operator

4.1 Support of a Query

We now define the support of a query. In our approach, the support of a query is defined with respect to a *reference* query that specifies the values subject to counting. It is important to note that, contrary to the operations defined previously, the $freq$ operator is defined for a particular instance I of DB.

Definition 6 - Support of a Query. *Let q and r be two queries. For every instance I, the* support of q relatively to r and I, *denoted by $Sup(q/r, I)$, is the following ratio:*

$$Sup(q/r, I) = \frac{|(\pi_K(r \bowtie q))(I)|}{|r(I)|}, \text{ where } K = sch(r).$$

The query r with respect to which the support of q is computed, is called the reference query.

The following proposition states the basic property of monotonicity of the support, which is similar to the one used for itemsets in [1].

Proposition 3. *Given a reference query r, let q_1 and q_2 be two queries such that $q_1 \sqsubseteq q_2$. Then for every instance I, we have: $Sup(q_1/r, I) \leq Sup(q_2/r, I)$.*

PROOF: Since $q_1 \sqsubseteq q_2$, then we have $\pi_K(r \bowtie q_1) \sqsubseteq \pi_K(r \bowtie q_2)$, which entails that $Sup(q_1/r, I) \leq Sup(q_2/r, I)$. ◇

Then, based on standard properties of the relational algebra, we have the following corollary.

Corollary 1. *Given a reference query r, let q_1 and q_2 be two queries. For every instance I, we have:*

1. *$Sup(\sigma_S(q_1)/r, I) \leq Sup(q_1/r, I)$ where S is a selection condition such that $att(S) \subseteq sch(q_1)$.*
2. *$Sup(\pi_X(q_1)/r, I) = Sup(q_1/r, I)$ where X is a set of attributes such that $sch(r) \subseteq X \subseteq sch(q_1)$.*
3. *$Sup(\rho_{B \leftarrow A}(q_1)/r, I) = Sup(q_1/r, I)$ where A and B are two attributes such that $A \notin sch(r)$ and $B \notin sch(r) \cup sch(q_1)$.*
4. *$Sup(q_1 \bowtie q_2/r, I) \leq Sup(q_i/r, I)$ for $i = 1, 2$.*

Moreover, if $sch(q_1) = sch(q_2)$ then we also have:

5. *$Sup(q_1 \cap q_2/r, I) \leq Sup(q_i/r, I)$ for $i = 1, 2$.*
6. *$Sup(q_2 \setminus q_1/r, I) \leq Sup(q_2/r, I)$.*
7. *$Sup(q_2 \cup q_1/r, I) \geq Sup(q_i/r, I)$ for $i = 1, 2$.*

PROOF: Using the notation and the hypotheses in the corollary, the proof is based on Definition 6 and on the following properties of the relational algebra, respectively: (1) $\pi_K(r \bowtie \sigma_S(q_1)) = \pi_K(\sigma_S(r \bowtie q_1))$, (2) $\pi_K(r \bowtie \pi_X(q_1)) = \pi_K(r \bowtie q_1)$, (3) $|(\rho_{B \leftarrow A}(q_1))(I)| = |q_1(I)|$, (4) $\pi_K(r \bowtie (q_1 \bowtie q_2)) \subseteq \pi_K(r \bowtie q_i)$ for $i = 1, 2$, (5) $q_1 \cap q_2 \sqsubseteq q_i$ for $i = 1, 2$, (6) $q_2 \setminus q_1 \sqsubseteq q_2$, and (7) $q_i \sqsubseteq q_1 \cup q_2$ for $i = 1, 2$. ◇

Definition 7 - Freq Operator. *Let $\mathcal{C}$ be a context, α a support threshold α, and r a reference query. For every instance I of DB, we define a context $freq_I(\mathcal{C}, \alpha, r)$ over $sch(\mathcal{C})$ by: $freq_I(\mathcal{C}, \alpha, r) = \{q \in \mathcal{C} \mid Sup(q/r, I) \geq \alpha\}$.*

Queries in $freq_I(\mathcal{C}, \alpha, r)$ are called α-frequent *in I, or simply* frequent *if α and I are understood.*

4.2 Properties of the Freq Operator

We first give basic properties of the $freq$ operator, and then we study its relationships to the unary and binary operations defined earlier.

Proposition 4. *Let $\mathcal{C}$, $\mathcal{C}_1$ and $\mathcal{C}_2$ be contexts, r a reference query and I an instance of DB. For all support thresholds α, α_1, α_2, the following holds.*

1. $freq_I(\mathcal{C}, \alpha, r) \subseteq \mathcal{C}$.
2. *If* $\alpha_2 \geq \alpha_1$, *then* $freq_I(\mathcal{C}, \alpha_2, r) \subseteq freq_I(\mathcal{C}, \alpha_1, r)$.
3. *If* $\mathcal{C}_1 \sqsubseteq \mathcal{C}_2$, *then* $freq_I(\mathcal{C}_1, \alpha, r) \sqsubseteq freq_I(\mathcal{C}_2, \alpha, r)$.
4. *If* $\mathcal{C}_1 \subseteq \mathcal{C}_2$, *then* $freq_I(\mathcal{C}_1, \alpha, r) \subseteq freq_I(\mathcal{C}_2, \alpha, r)$.

Proof: Let I be an instance of DB, $\mathcal{C}$ a context and r be a reference query.
1. This point follows directly from Definition 7 above.
2. Let α_1 and α_2 be two support thresholds such that $\alpha_2 \geq \alpha_1$, and $q \in freq_I(\mathcal{C}, \alpha, r)$. We have $Sup(q/r, I) \geq \alpha_2$, and thus, $Sup(q/r, I) \geq \alpha_2 \geq \alpha_1$, which shows that $q \in freq_I(\mathcal{C}, \alpha_1, r)$. Therefore, if $\alpha_2 \geq \alpha_1$, then $freq_I(\mathcal{C}, \alpha_2, r) \subseteq freq_I(\mathcal{C}, \alpha_1, r)$.

3. Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two contexts such that $\mathcal{C}_1 \sqsubseteq \mathcal{C}_2$. Given a support threshold α, let $q_1 \in freq_I(\mathcal{C}_1, \alpha, r)$. We have $Sup(q_1/r, I) \geq \alpha$. On the other hand, since $\mathcal{C}_1 \sqsubseteq \mathcal{C}_2$, there exists a query $q_2 \in \mathcal{C}_2$ such that $q_1 \sqsubseteq q_2$. It follows that $Sup(q_2/r, I) \geq Sup(q_1/r, I) \geq \alpha$. Thus, $q_2 \in freq_I(\mathcal{C}_2, \alpha, r)$, which shows that if $\mathcal{C}_1 \sqsubseteq \mathcal{C}_2$, then $freq_I(\mathcal{C}_1, \alpha, r) \sqsubseteq freq_I(\mathcal{C}_2, \alpha, r)$.

4. This case is obvious, and thus, the proof is complete. ◇

Considering the unary operators, we have the following two propositions.

Proposition 5. *Let I be an instance of DB, α a support threshold, and r a reference query. Let $\mathcal{C}$ be a context.*

1. *For every set of attributes X and every reference query such that $sch(r) \subseteq X \subseteq sch(\mathcal{C})$, $freq_I(\pi_X(\mathcal{C}), \alpha, r) = \pi_X(freq_I(\mathcal{C}, \alpha, r))$.*
2. *For every selection condition S, $freq_I(\sigma_S(\mathcal{C}), \alpha, r) \sqsubseteq freq_I(\mathcal{C}, \alpha, r)$.*
3. *For all attributes A and B, $freq_I(\rho_{B \leftarrow A}(\mathcal{C}), \alpha, r) = \rho_{B \leftarrow A}(freq_I(\mathcal{C}, \alpha, r))$.*
4. *For every selection condition S, $freq_I(\tau_S(\mathcal{C}), \alpha, r) = \tau_S(freq_I(\mathcal{C}, \alpha, r))$.*
5. *For every operator $R \in \{\sqsubseteq, \sqsupseteq, =\}$ and query q: $freq_I(\gamma_{Rq}(\mathcal{C}), \alpha, r) = \gamma_{Rq}(freq_I(\mathcal{C}, \alpha, r))$.*

Proof: Let I be an instance of DB, r a reference query, $\mathcal{C}$ a context, X a subset of $sch(\mathcal{C})$ such that $sch(r) \subseteq X$, and S a selection condition such that $att(S) \subseteq sch(\mathcal{C})$.
1. Let q be a query in $freq_I(\pi_X(\mathcal{C}), \alpha, r)$. There exists a query $q' \in \mathcal{C}$ such that $q = \pi_X(q')$, and $Sup(q/r, I) \geq \alpha$. Using Corollary 1(2), we have $Sup(q/r, I) = Sup(q'/r, I) \geq \alpha$. Thus, $q' \in freq_I(\mathcal{C}, \alpha, r)$ and $q \in \pi_X(freq_I(\mathcal{C}, \alpha, r))$, which shows that $freq_I(\pi_X(\mathcal{C}), \alpha, r) \subseteq \pi_X(freq_I(\mathcal{C}, \alpha, r))$.
Conversely, if $q \in \pi_X(freq_I(\mathcal{C}, \alpha, r))$, then there exists q' in $freq_I(\mathcal{C}, \alpha, r)$ such that $q = \pi_X(q')$. Thus, $q' \in \mathcal{C}$, and since $Sup(q/r, I) \geq \alpha$, we have $q \in$

$freq_I(\pi_X(\mathcal{C}), \alpha, r)$. Therefore, $\pi_X(freq_I(\mathcal{C}, \alpha, r)) \subseteq freq_I(\pi_X(\mathcal{C}), \alpha, r)$, which shows that $\pi_X(freq_I(\mathcal{C}, \alpha, r)) = freq_I(\pi_X(\mathcal{C}), \alpha, r)$.

2. Let $q \in freq_I(\sigma_S(\mathcal{C}), \alpha, r)$. We have $Sup(q/r, I) \geq \alpha$ and there exits a query $q' \in \mathcal{C}$ such that $q = \sigma_S(q')$. Using Proposition 3, since $q \sqsubseteq q'$, we have $Sup(q'/r, I) \geq Sup(q/r, I) \geq \alpha$. Thus, there exists a query $q' \in freq_I(\mathcal{C}, \alpha, r)$ such that $q \sqsubseteq q'$, which shows that $freq_I(\sigma_S(\mathcal{C}), \alpha, r) \sqsubseteq freq_I(\mathcal{C}, \alpha, r)$.

3. In the same way, we can easily prove this item using Corollary 1(3).

4. Let $q \in freq_I(\tau_S(\mathcal{C}), \alpha, I)$. By definition, $Sup(q/r, I) \geq \alpha$ and $q = \sigma_S(q)$. Thus, we have $q \in freq_I(\mathcal{C}, \alpha, I)$ and $q \in \tau_S(freq_I(\mathcal{C}, \alpha, r))$, which shows that $freq_I(\tau_S(\mathcal{C}), \alpha, I) \subseteq \tau_S(freq_I(\mathcal{C}, \alpha, r))$.
As it can be seen in a similar way that $\tau_S(freq_I(\mathcal{C}, \alpha, r)) \subseteq freq_I(\tau_S(\mathcal{C}), \alpha, I)$, we have $freq_I(\tau_S(\mathcal{C}), \alpha, r) = \tau_S(freq_I(\mathcal{C}, \alpha, r))$.

5. This property follows directly from Definition 4. Thus, the proof is complete. ◇

We point out that when considering contexts defined using biases, the result of Proposition 5(2) can be improved, as stated in the proposition below.

Proposition 6. *Let $\mathcal{B} = \langle b, \Sigma \rangle$ be a bias, r a reference query and S a selection condition. For every instance I of DB and every support threshold α, we have:*

1. $freq_I(\sigma_S(\mathcal{C}(\mathcal{B})), \alpha, r) \subseteq \sigma_S(freq_I(\mathcal{C}(\mathcal{B}), \alpha, r))$.
2. *If $S \in \Sigma^*$ then* $freq_I(\sigma_S(\mathcal{C}(\mathcal{B})), \alpha, r) = \tau_S(freq_I(\mathcal{C}(\mathcal{B}), \alpha, r))$.

Proof: In both cases, any q in $freq_I(\sigma_S(\mathcal{C}(\mathcal{B})), \alpha, r)$ is such that $Sup(q/r, I) \geq \alpha$ and $q = \sigma_S(\sigma_{S'}(b))$ where $S' \in \Sigma^*$.

1. Since $q \sqsubseteq \sigma_{S'}(b)$, we have $Sup(\sigma_{S'}(b)/r, I) \geq Sup(q/r, I) \geq \alpha$. Therefore, $\sigma_{S'}(b) \in freq_I(\mathcal{C}(\mathcal{B}), \alpha, r)$, and thus $q \in \sigma_S(freq_I(\mathcal{C}(\mathcal{B}), \alpha, r))$.

2. Since in this case S is in Σ^*, then so is $S \wedge S'$, and thus $q \in freq_I(\mathcal{C}(\mathcal{B}), \alpha, r)$. As $q = \sigma_S(q)$, we have that $q \in \tau_S(freq_I(\mathcal{C}(\mathcal{B}), \alpha, r))$. Thus, $freq_I(\sigma_S(\mathcal{C}(\mathcal{B})), \alpha, r) \subseteq \tau_S(freq_I(\mathcal{C}(\mathcal{B}), \alpha, r))$. As it can be shown in the same way that $freq_I(\sigma_S(\mathcal{C}(\mathcal{B})), \alpha, r) \supseteq \tau_S(freq_I(\mathcal{C}(\mathcal{B}), \alpha, r))$, the proof is complete. ◇

We note that the properties of the $freq$ operator given so far deal with the same reference query. However, in practice, changing the reference query can be useful for the user, as illustrated by the following example. In general, we have few interesting properties for the $freq$ operator in this case. Nevertheless, the following example shows a situation where some comparison is possible.

Example 5. Consider the database $DBSales$, the context $\mathcal{C}_1$ and the reference query r given in Example 1. We recall that queries in $\mathcal{C}_1$ are the queries of the form $\sigma_S(b_1)$ where $b_1 = Cust \bowtie Sales \bowtie Prod$ and S is a conjunctive selection condition built up from the conditions $Cjob = Professor$, $Cjob = Lawer$, $Ptype = Tea$ and $Ptype = Milk$. Moreover, we also recall that $r = \pi_{Cid}(Cust)$, meaning that the supports of queries in $\mathcal{C}_1$ are computed with respect to the set of all customers.

Assume now that, seeing that queries dealing with professors are frequent, the user wishes to compute supports with respect to *only* those customers who buy products and are professors. Then the corresponding reference query r' can be expressed by $r' = \pi_{Cid}(r \bowtie q)$, where $q = \sigma_{Cjob=Professor}(b_1)$. In such a case, the following proposition shows that $\gamma_{\sqsubseteq q}(freq_I(\mathcal{C}_1, \alpha, r)) \subseteq freq_I(\mathcal{C}_1, \alpha, r')$, meaning that the frequent queries with respect to r that deal with professors are among the frequent queries *with respect to* r' (which deal with professors). □

Proposition 7. *Let I be an instance of DB. Let $\mathcal{C}$ be a context. Given a query $q \in \mathcal{C}$ and a reference query r, let $r' = \pi_K(r \bowtie q)$ where $K = sch(r)$. For every support threshold α, we have: $\gamma_{\sqsubseteq q}(freq_I(\mathcal{C}, \alpha, r)) \subseteq freq_I(\mathcal{C}, \alpha, r')$.*

Proof: Let I be an instance of DB, r a reference query with $K = sch(r)$ and α a support threshold. Let $\mathcal{C}$ be a context.
Given a query $q \in \mathcal{C}$, let $q' \in \gamma_{\sqsubseteq q}(freq_I(\mathcal{C}, \alpha, r))$. By definition, we have $q' \sqsubseteq q$ and $Sup(q'/r, I) \geq \alpha$. Let $s_1 = \pi_K(r \bowtie q')$ and $s_2 = \pi_K(r' \bowtie q') = \pi_K(\pi_K(r \bowtie q) \bowtie q')$. Since $r' \sqsubseteq r$, we have $s_2 \sqsubseteq s_1$ and thus $s_2(I) \subseteq s_1(I)$. Now, we show that $s_1(I) \subseteq s_2(I)$. Let t be a tuple in $s_1(I)$. There exists a tuple $t' \in q'(I)$ such that for every attribute $A \in K \cap sch(q')$, $t'.A = t.A$. Since $q' \sqsubseteq q$, $t' \in q(I)$. Thus, $t \in \pi_K(r \bowtie q)(I)$. It follows that $t' \in (\pi_K(r \bowtie q) \bowtie q')(I)$ and $t \in s_2(I)$, which completes the proof that $s_1(I) \subseteq s_2(I)$. Since $s_2(I) \subseteq s_1(I)$ and $s_1(I) \subseteq s_2(I)$, we finally have $s_1(I) = s_2(I)$.

Now, we can compare the support of q' with respect to r and r'. Since $r' \sqsubseteq r$, we have $|r'(I)| \leq |r(I)|$. Therefore, $Sup(q'/r', I) = |s_2(I)|/|r'(I)| = |s_1(I)|/|r'(I)| \geq |s_1(I)|/|r(I)| = Sup(q'/r, I) \geq \alpha$. Thus, $q' \in freq_I(\mathcal{C}, \alpha, r)$, which completes the proof. ◇

The following proposition states the properties of $freq$ with respect to the binary operators.

Proposition 8. *Let I be an instance of DB, α a support threshold, and r a reference query. Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two contexts.*

1. *$\pi_{X_i}(freq_I(\mathcal{C}_1 \bowtie \mathcal{C}_2, \alpha, r)) \sqsubseteq freq_I(\mathcal{C}_i, \alpha, r)$ where $X_i = sch(\mathcal{C}_i)$ $(i = 1, 2)$, and $freq_I(\mathcal{C}_1 \bowtie \mathcal{C}_2, \alpha, r) \subseteq freq_I(\mathcal{C}_1, \alpha, r) \bowtie freq_I(\mathcal{C}_2, \alpha, r)$.*
2. *$freq_I(\mathcal{C}_i, \alpha, r) \sqsubseteq freq_I(\mathcal{C}_1 \sqcup \mathcal{C}_2, \alpha, r)$ for $i = 1, 2$.*
3. *$freq_I(\mathcal{C}_1 \sqcap \mathcal{C}_2, \alpha, r) \sqsubseteq freq_I(\mathcal{C}_i, \alpha, r)$ for $i = 1, 2$.*
4. *$freq_I(\mathcal{C}_1 \ominus \mathcal{C}_2, \alpha, r) \sqsubseteq freq_I(\mathcal{C}_1, \alpha, r)$.*

Proof: Let I be an instance of DB, α a support threshold and r a reference query with $K = sch(r)$. Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two contexts.
1. Using Proposition 5(1), we have $\pi_{X_i}(freq_I(\mathcal{C}_1 \bowtie \mathcal{C}_2, \alpha, r)) = freq_I(\pi_{X_i}(\mathcal{C}_1 \bowtie \mathcal{C}_2), \alpha, r)$ $(i = 1, 2)$. Moreover, using Proposition 2(1), we have $\pi_{X_i}(\mathcal{C}_1 \bowtie \mathcal{C}_2) \sqsubseteq \mathcal{C}_1$. Therefore, using Proposition 4(3), we have $freq_I(\pi_{X_i}(\mathcal{C}_1 \bowtie \mathcal{C}_2), \alpha, r) \sqsubseteq freq_I(\mathcal{C}_i, \alpha, r)$.
Now, let $q \in freq_I(\mathcal{C}_1 \bowtie \mathcal{C}_2, \alpha, r)$. There exist a query $q_1 \in \mathcal{C}_1$ and a query $q_2 \in \mathcal{C}_2$ such that $q = q_1 \bowtie q_2$. Moreover, we have $Sup(q/r, I) \geq \alpha$. Thus, since $\pi_K(r \bowtie q_1 \bowtie q_2) \sqsubseteq \pi_K(r \bowtie q_i)$ $(i = 1, 2)$, we have $Sup(q_i/r, I) \geq Sup(q/r, I) \geq$

α, which shows that for $i = 1, 2$, $q_i \in freq_I(\mathcal{C}_i, \alpha, r)$. Therefore, there exists $q_1 \in freq_I(\mathcal{C}_1, \alpha, r)$ and $q_2 \in freq_I(\mathcal{C}_2, \alpha, r)$ such that $q = q_1 \bowtie q_2$, and so $freq_I(\mathcal{C}_1 \bowtie \mathcal{C}_2, \alpha, r) \subseteq freq_I(\mathcal{C}_1, \alpha, r) \bowtie freq_I(\mathcal{C}_2, \alpha, r)$.

The proofs of items 2, 3 and 4 follow directly from Proposition 2(2, 3) and Proposition 4(3). ◇

The following proposition states the properties of $freq$ with respect to set-theoretic operations.

Proposition 9. *Let I be an instance of DB. Let α, α_1, α_2 be support thresholds, and r a reference query. Let $\mathcal{C}$, $\mathcal{C}_1$ and $\mathcal{C}_2$ be contexts.*

1. *For every operator $op \in \{\cap, \setminus, \cup\}$, we have:*
   $freq_I(\mathcal{C}_1 \ op \ \mathcal{C}_2, \alpha, r) = freq_I(\mathcal{C}_1, \alpha, r) \ op \ freq_I(\mathcal{C}_2, \alpha, r)$.
2. *If $\alpha_1 \leq \alpha_2$, then $freq_I(\mathcal{C}, \alpha_2, r) \setminus freq_I(\mathcal{C}, \alpha_1, r) = \emptyset$ and*
   $freq_I(\mathcal{C}, \alpha_1, r) \setminus freq_I(\mathcal{C}, \alpha_2, r) = freq_I((\mathcal{C} \setminus freq_I(\mathcal{C}, \alpha_2, r)), \alpha_1, r)$.
3. *If $\alpha_1 \leq \alpha_2$, then $freq_I(\mathcal{C}, \alpha_1, r) \cap freq_I(\mathcal{C}, \alpha_2, r) = freq_I(\mathcal{C}, \alpha_2, r)$ and*
   $freq_I(\mathcal{C}, \alpha_1, r) \cup freq_I(\mathcal{C}, \alpha_2, r) = freq_I(\mathcal{C}, \alpha_1, r)$.

Proof: 1. We prove this item in the case where *op* is the intersection operator, the other proofs being similar. A query q is in $freq_I(\mathcal{C}_1 \cap \mathcal{C}_2, \alpha, r)$ if and only if q is in $\mathcal{C}_1 \cap \mathcal{C}_2$ and $Sup(q/r, I) \geq \alpha$. Thus, q is in $freq_I(\mathcal{C}_1 \cap \mathcal{C}_2, \alpha, r)$ if and only if q is in $freq_I(\mathcal{C}_1, \alpha, r)$ and in $freq_I(\mathcal{C}_2, \alpha, r)$, which shows that $freq_I(\mathcal{C}_1 \cap \mathcal{C}_2, \alpha, r) = freq_I(\mathcal{C}_1, \alpha, r) \cap freq_I(\mathcal{C}_2, \alpha, r)$.

The proof of the last two items of the proposition is an immediate consequence of the fact that if $\alpha_1 \leq \alpha_2$, then $freq_I(\mathcal{C}, \alpha_2, r) \subseteq freq_I(\mathcal{C}, \alpha_2, r)$ (see Proposition 4(2)). Thus the proof is complete. ◇

In the following example, we outline situations in which properties of the $freq$ operator can be used for optimization.

Example 6. Consider again the database $DBSales$ of Example 1, the reference query $r = \pi_{Cid}(Cust)$ and the context $\mathcal{C}_1 = \mathcal{C}(\mathcal{B}_1)$ where $\mathcal{B}_1 = \langle b_1, \Sigma_1 \rangle$, $b_1 = Cust \bowtie Sales \bowtie Prod$ and $\Sigma_1 = \{Cjob = Professor, Cjob = Lawyer, Ptype = Tea, Ptype = Milk\}$. Assume in this example that the context $freq_I(\mathcal{C}_1, \alpha, r)$ has been computed and stored previously.

If we consider the context $\mathcal{C}_2 = \sigma_{Cjob=Lawyer}(\mathcal{C}_1)$, then, according to Proposition 6(2), for every support threshold α and every instance I of $DBSales$, we have: $freq_I(\mathcal{C}_2, \alpha, r) = \tau_{Cjob=Lawyer}(freq_I(\mathcal{C}_1, \alpha, r))$.

Thus, it turns out that the frequent queries of $\mathcal{C}_1$ that deal with lawyers are exactly the frequent queries of $\mathcal{C}_2$. As a consequence, $freq_I(\mathcal{C}_2, \alpha, r)$ can be computed without having to access the database.

Now, if we consider the context $\mathcal{C}'_1 = \sigma_{Caddr=Paris}(\mathcal{C}_1)$ given in Example 2, according to Proposition 6(1), for every instance I of $DBSales$, we have: $freq_I(\mathcal{C}'_1, \alpha, r) \subseteq \sigma_{Caddr=Paris}(freq_I(\mathcal{C}_1, \alpha, r))$. In this case, if a query q_1 of $\mathcal{C}_1$ is not frequent, the corresponding query $q'_1 = \sigma_{Caddr=Paris}(q_1)$ of $\mathcal{C}'_1$ is not frequent either, and this conclusion is reached without any access to the database. □

We conclude this section by the following important remark: although useful for optimization, the properties given so far say nothing about the actual computation of frequent queries in a given context. We address this problem in the following section, for contexts defined using mining biases.

5 Computation of Mining Contexts

In this section, we consider contexts that are expressed by a combination of contexts of the form $\mathcal{C}(\mathcal{B}_i)$ (where $\mathcal{B}_i$ are biases) using the operations given in this paper (except for $\sqcup$ and $\ominus$).

We first note in this respect that if the $freq$ operator does not occur in the defining expression, then the definitions given so far allow for the computation of the corresponding context. On the other hand, we show below that, under the restrictions stated just above, expressions of the form $freq_I(\mathcal{C}, \alpha, r)$ can be computed using any level-wise algorithm.

5.1 The Case of a Single Context

Simple Contexts. We show that given a bias $\mathcal{B} = \langle b, \Sigma \rangle$, a support threshold α, a reference query r and an instance I of DB, contexts of the form $freq_I(\mathcal{C}(\mathcal{B}), \alpha, r)$ can be computed using any level-wise algorithm as in [13].

To this end, we have to show that $\langle \mathcal{C}(\mathcal{B}), \sqsubseteq \rangle$ is a lattice. We recall from Definition 3 that $\mathcal{C}(\mathcal{B})$ is the set of all queries $\sigma_S(b)$ where S is either $\bot$, $\top$ or a conjunctive selection condition built up from atomic selection conditions of Σ. Let us define the following two operators $\overline{\wedge}$ and $\veebar$ on selection conditions of Σ^* : For all S_1 and S_2 in Σ^*, define:

- If $S_1 = \bot$ or $S_2 = \bot$ then $S_1 \,\overline{\wedge}\, S_2 = \bot$
  If $S_1 = \top$ (respectively $S_2 = \top$) then $S_1 \,\overline{\wedge}\, S_2 = S_2$ (respectively S_1)
  Otherwise, if $S_1 \wedge S_2 \in \Sigma^*$ then $S_1 \,\overline{\wedge}\, S_2 = S_1 \wedge S_2$ else $S_1 \,\overline{\wedge}\, S_2 = \bot$.
- If $S_1 = \bot$ (respectively $S_2 = \bot$) then $S_1 \veebar S_2 = S_2$ (respectively S_1)
  If $S_1 = \top$ or $S_2 = \top$ then $S_1 \veebar S_2 = \top$
  Otherwise, $S_1 \veebar S_2$ is the conjunction of all atomic selection conditions $(A_i = a_i)$ that occur in both S_1 and S_2.

It can be seen that for all queries $q_1 = \sigma_{S_1}(b)$ and $q_2 = \sigma_{S_2}(b)$ in $\mathcal{C}(\mathcal{B})$, we have:

1. The queries $\sigma_{S_1 \overline{\wedge} S_2}(b)$ and $\sigma_{S_1 \veebar S_2}(b)$ are in $\mathcal{C}(\mathcal{B})$,
2. $q_i \sqsubseteq \sigma_{S_1 \veebar S_2}(b)$ and $q_i \sqsupseteq \sigma_{S_1 \overline{\wedge} S_2}(b)$ for $i = 1, 2$,
3. $\sigma_{S_1 \veebar S_2}(b) = \min_{\sqsubseteq}\{q \in \mathcal{C}(\mathcal{B}) \mid q_1 \sqsubseteq q \text{ and } q_2 \sqsubseteq q\}$ and
   $\sigma_{S_1 \overline{\wedge} S_2}(b) = \max_{\sqsubseteq}\{q \in \mathcal{C}(\mathcal{B}) \mid q \sqsubseteq q_1 \text{ and } q \sqsubseteq q_2\}$.

As a consequence, $\langle \mathcal{C}(\mathcal{B}), \sqsubseteq \rangle$ is a lattice. Moreover, we recall from Proposition 3 that the support is monotonic with respect to the relation $\sqsubseteq$.

Therefore, in our approach, mining queries of the form $freq_I(\mathcal{C}(\mathcal{B}), \alpha, r)$ are computed using any level-wise algorithm ([13]). More precisely, the queries in $\mathcal{C}(\mathcal{B})$ are considered level by level, where level k consists of all queries of the

form $\sigma_S(b)$ such that S is the conjunction of k atomic selection conditions in Σ. At each level, the relation $(r \bowtie b)(I)$ is scanned once for computing the supports of the candidate queries. We note also that the computation starts at level 0 with the query b (or $\sigma_\top(b)$) and stops at the latest by considering the empty query (or $\sigma_\bot(b)$) which can not be frequent for any support threshold $\alpha > 0$.

Composed Contexts. We study the computation of contexts of one of the following forms

- $freq_I(op_1(\mathcal{C}(\mathcal{B})), \alpha, r)$ where op_1 is one of the unary operations given in the previous section, or
- $freq_I((\mathcal{C}(\mathcal{B}_1)\ op_2\ \mathcal{C}(\mathcal{B}_2)), \alpha, r)$ where op_2 is a binary operation in the set $\{\cap, \setminus, \cup, \sqcap, \bowtie\}$.

Given a bias $\mathcal{B} = \langle b, \Sigma \rangle$, the case of a unary operation can be summarized as follows:

- If op_1 is a projection over X such that $sch(r) \subseteq X \subseteq sch(b)$, or a renaming or a τ- or γ-reduction, then based on Proposition 5, we know that $freq_I(op_1(\mathcal{C}(\mathcal{B})), \alpha, r) = op_1(freq_I(\mathcal{C}(\mathcal{B}), \alpha, r))$. Therefore, $freq_I(\mathcal{C}(\mathcal{B}), \alpha, r)$ is computed first and then the operation op_1 is applied to the resulting context. We note that optimizations are possible in the case of γ- and τ-reductions. Indeed, at each level in the lattice $\mathcal{C}(\mathcal{B})$, it is not necessary to evaluate the supports of candidate queries not in $op_1(\mathcal{C}(\mathcal{B}))$.
- If op_1 is a selection, then it is easy to see that $\sigma_S(\mathcal{C}(\mathcal{B})) = \mathcal{C}(\mathcal{B}')$, where $\mathcal{B}' = \langle \sigma_S(b), \Sigma \rangle$. Therefore, in this case, the context $freq_I(\mathcal{C}(\mathcal{B}'), \alpha, r)$ is computed as explained previously in the case of simple contexts.

Now, given two biases $\mathcal{B}_1 = \langle b_1, \Sigma_1 \rangle$ and $\mathcal{B}_2 = \langle b_2, \Sigma_2 \rangle$, let us consider the case of a binary operation op_2 in the set $\{\cap, \setminus, \cup, \sqcap, \bowtie\}$.

If op_2 is one of the operations $\cap$, $\setminus$ or $\cup$, then based on Proposition 9(1), we have $freq_I((\mathcal{C}(\mathcal{B}_1)\ op_2\ \mathcal{C}(\mathcal{B}_2)), \alpha, r) = freq_I(\mathcal{C}(\mathcal{B}_1), \alpha, r)\ op_2\ freq_I(\mathcal{C}(\mathcal{B}_2), \alpha, r)$. Therefore, in order to compute $freq_I((\mathcal{C}(\mathcal{B}_1)\ op_2\ \mathcal{C}(\mathcal{B}_2)), \alpha, r))$, we first compute $freq_I(\mathcal{C}(\mathcal{B}_1), \alpha, r)$ and $freq_I(\mathcal{C}(\mathcal{B}_2), \alpha, r)$ and then we apply the operation op_2 to the resulting contexts. As in the case of unary operators, optimizations are possible in this computation. For example, if op_2 is the operator $\cap$, when computing $freq_I(\mathcal{C}(\mathcal{B}_1), \alpha, r)$, it is not necessary to evaluate the supports of the candidate queries not in $\mathcal{C}(\mathcal{B}_2)$.

If op_2 is either $\sqcap$ or $\bowtie$, then we have the following:

- $\mathcal{C}(\mathcal{B}_1) \sqcap \mathcal{C}(\mathcal{B}_2) = \mathcal{C}(\mathcal{B}')$, where $\mathcal{B}' = \langle b_1 \cap b_2, \Sigma_1 \cup \Sigma_2 \rangle$, and
- $\mathcal{C}(\mathcal{B}_1) \bowtie \mathcal{C}(\mathcal{B}_2) = \mathcal{C}(\mathcal{B}')$, where $\mathcal{B}' = \langle b_1 \bowtie b_2, \Sigma_1 \cup \Sigma_2 \rangle$.

Therefore, in these two cases, the computation of $freq_I((\mathcal{C}(\mathcal{B}_1)\ op_2\ \mathcal{C}(\mathcal{B}_2)), \alpha, r)$ is achieved through the computation of $freq_I(\mathcal{C}(\mathcal{B}'), \alpha, r)$ as defined above.

We note that the case of the operations $\sqcup$ and $\ominus$ cannot be treated as above because, if op_2 is one of these operations, then:

1. $freq_I((\mathcal{C}(\mathcal{B}_1)\ op_2\ \mathcal{C}(\mathcal{B}_2)), \alpha, r) \neq freq_I(\mathcal{C}(\mathcal{B}_1), \alpha, r)\ op_2\ freq_I(\mathcal{C}(\mathcal{B}_2), \alpha, r)$, and
2. there is no bias $\mathcal{B}'$ such that $\mathcal{C}(\mathcal{B}_1)\ op_2\ \mathcal{C}(\mathcal{B}_2) = \mathcal{C}(\mathcal{B}')$.

Summarizing this subsection, we have shown that any context of the form $freq_I(\mathcal{C}, \alpha, r)$, where $\mathcal{C}$ is a combination of contexts of the form $\mathcal{C}(\mathcal{B}_i)$ using any operation, except $\sqcup$ and $\ominus$, can be computed using any level-wise algorithm as in [13].

5.2 Iterative Computation

In this section, we study how to optimize the computation of mining queries of the form $freq_I(\mathcal{C}, \alpha, r)$ under the following assumptions:

1. The context $\mathcal{C}$ is an expression using operators introduced in this paper, except $\sqcup$ and $\ominus$, and involving contexts $\mathcal{C}(\mathcal{B}_1), \ldots, \mathcal{C}(\mathcal{B}_k)$, for some $k \geq 1$.
2. The contexts $freq_I(\mathcal{C}(\mathcal{B}_1), \alpha_1, r), \ldots, freq_I(\mathcal{C}(\mathcal{B}_k), \alpha_k, r)$ have already been computed and their positive boundaries are stored in the database.

We first study the case where $\alpha = \alpha_1 = \ldots = \alpha_k$, and outline the general case at the end of this section. Before doing so, we briefly review the basic concepts related to positive boundaries ([13]).

As in [13], we define the positive boundary of the set of frequent queries of a context $\mathcal{C}$ as being the set of all most specific queries in $\mathcal{C}$ that are frequent. More precisely, given a context $\mathcal{C}$, the positive boundary of $freq_I(\mathcal{C}, \alpha, r)$, denoted by $Bd_I^+(\mathcal{C}, \alpha, r)$, is defined by:

$$Bd_I^+(\mathcal{C}, \alpha, r) = \min_{\sqsubseteq}(freq_I(\mathcal{C}, \alpha, r))$$

Moreover, it is easy to see that $freq_I(\mathcal{C}, \alpha, r)$ can be computed from its positive boundary according to the following equality:

$$freq_I(\mathcal{C}, \alpha, r) = \{q \in \mathcal{C} \mid (\exists q^+ \in Bd_I^+(\mathcal{C}, \alpha, r))(q^+ \sqsubseteq q)\} \qquad (1)$$

We note that this computation can be achieved without accessing the database and that the supports of the queries in $freq_I(\mathcal{C}, \alpha, r)$ can be obtained through one pass over the database, only.

Now, let us consider the case where, in our previous assumptions, we have $\alpha = \alpha_1 = \ldots = \alpha_k$. Based on the properties of our operations on contexts, the following two cases occur:

1. *Equality:* In this case, $freq_I(\mathcal{C}, \alpha, r)$ is equal to an expression involving the contexts $freq_I(\mathcal{C}(\mathcal{B}_i), \alpha, r)$ $(i = 1 \ldots k)$. This case occurs for the operators projection, renaming, τ- and γ-reduction (see Proposition 5(1, 4, 5)), selection (see Proposition 6(2)), or for set-theoretic operators (see Proposition 9(1)).

2. *Inclusion:* In this case, $freq_I(\mathcal{C}, \alpha, r)$ is included (weakly or not) in an expression involving the contexts $freq_I(\mathcal{C}(\mathcal{B}_i), \alpha, r)$ $(i = 1 \ldots k)$. This case occurs for the operators selection (see Proposition 6(1)), join, or $\sqcap$ (see Proposition 8(1, 3)).

In each of these cases, we outline below optimizations for the computation of $freq_I(\mathcal{C}, \alpha, r)$.

(1) Equality. In this case, no access to data is needed to obtain $freq_I(\mathcal{C}, \alpha, r)$. In order to get the context $freq_I(\mathcal{C}, \alpha, r)$, one has just to compute the contexts $freq_I(\mathcal{C}(\mathcal{B}_i), \alpha, r)$ $(i = 1, \ldots, k)$ based on equality (1) above, and then to apply the corresponding operations to these contexts. Moreover, we recall that the supports of the queries in $freq_I(\mathcal{C}, \alpha, r)$ can be computed through only one pass over the data.

(2) Inclusion. In this case, the positive boundaries of the contexts already computed allow for additional pruning (with respect to standard level-wise algorithms) when generating the candidate queries of level $l+1$ from the frequent queries of level l.

To see this, let q be a candidate query in $\mathcal{C}$. We know that $freq_I(\mathcal{C}, \alpha, r)$ is included (weakly or not) in an expression involving the contexts $freq_I(\mathcal{C}(\mathcal{B}_i), \alpha, r)$. According to the properties given in the previous section, it follows that there exist queries $q_i \in freq_I(\mathcal{C}(\mathcal{B}_i), \alpha, r)$ such that $\pi_{X_i}(q) \sqsubseteq q_i$, where $X_i = sch(\mathcal{C}(\mathcal{B}_i))$. Thus, for every $i = 1, \ldots, k$, since q_i is frequent, there exists $q_i^+ \in Bd_I^+(\mathcal{C}(\mathcal{B}_i), \alpha, r)$ such that $q_i^+ \sqsubseteq q_i$. Therefore, conversely, if for some i, there is no query q_i^+ as above, then q is not frequent and thus, q can be removed from the set of candidate queries of $\mathcal{C}$. The problem of efficiently testing the existence of q_i^+ has been addressed in [6].

Example 7. Given three biases $\mathcal{B}_i = \langle b_i, \Sigma_i \rangle$ $(i = 1, 2, 3)$ such that $sch(b_1) \subseteq sch(b_2)$ and a minimum support threshold α, assume that the contexts $freq_I(\mathcal{C}_i, \alpha, r)$ where $\mathcal{C}_i = \mathcal{C}(\mathcal{B}_i)$ $(i = 1, 2, 3)$ have already been computed. Consider the context $\mathcal{C} = (\sigma_\varphi(\mathcal{C}_1) \cup \pi_{X_1}(\mathcal{C}_2)) \bowtie \mathcal{C}_3$ where φ is a selection condition and $X_1 = sch(\mathcal{C}_1)$. Using Propositions 5, 8 and 9, we have:

$$freq_I(\mathcal{C}, \alpha, r) \sqsubseteq (freq_I(\mathcal{C}_1, \alpha, r) \bowtie freq_I(\mathcal{C}_3, \alpha, r)) \cup (\pi_{X_1}(freq_I(\mathcal{C}_2, \alpha, r)) \bowtie freq_I(\mathcal{C}_3, \alpha, r)).$$

Moreover, let q be a candidate query in $\mathcal{C}$. q is either in $\sigma_\varphi(\mathcal{C}_1) \bowtie \mathcal{C}_3$ or in $\pi_{X_1}(\mathcal{C}_2) \bowtie \mathcal{C}_3$. In the first case, q is a query of the form $q = \sigma_{S_1 \wedge S_3}(\sigma_\varphi(b_1) \bowtie b_3)$ where $S_1 \in \Sigma_1^*$ and $S_3 \in \Sigma_3^*$. Thus, we have $\pi_{X_1}(q) \sqsubseteq \sigma_{S_1}(b_1)$ and $\pi_{X_1}(q) \sqsubseteq \sigma_{S_3}(b_3)$. Therefore, if one of the queries $\sigma_{S_1}(b_1)$ or $\sigma_{S_3}(b_3)$ is not in $freq_I(\mathcal{C}_1, \alpha, r)$ or $freq_I(\mathcal{C}_3, \alpha, r)$, then q is not frequent and thus, can be removed from the set of candidate queries in $\mathcal{C}$.

In the second case, q is a query of the form $q = \sigma_{S_2 \wedge S_3}(\pi_{X_1}(b_2) \bowtie b_3)$ where $S_2 \in \Sigma_2^*$ and $S_3 \in \Sigma_3^*$. Thus, we have $\pi_{X_1}(q) \sqsubseteq \pi_{X_1}(\sigma_{S_2}(b_2))$ and $\pi_{X_1}(q) \sqsubseteq \sigma_{S_3}(b_3)$. Therefore, if one of the queries $\sigma_{S_2}(b_2)$ or $\sigma_{S_3}(b_3)$ is not in $freq_I(\mathcal{C}_2, \alpha, r)$ or $freq_I(\mathcal{C}_3, \alpha, r)$, then q is not frequent and thus, can be removed from the set of candidate queries in $\mathcal{C}$. $\square$

We terminate this section by considering the general case whereby the contexts $freq_I(\mathcal{C}(\mathcal{B}_i), \alpha_i, r)$ have been computed with respect to different minimum support thresholds α_i $(i = 1, \ldots, k)$.

Let $\alpha' = \max(\{\alpha\} \cup \{\alpha_i \mid i = 1, \ldots, k\})$. For every $i = 1, \ldots, k$, we have $freq_I(\mathcal{C}(\mathcal{B}_i), \alpha', r) = \{q \in freq_I(\mathcal{C}(\mathcal{B}_i), \alpha_i, r) \mid Sup(q/r, I) \geq \alpha'\}$. Therefore, using $Bd_I^+(\mathcal{C}(\mathcal{B}_i), \alpha_i, r)$, the set $freq_I(\mathcal{C}(\mathcal{B}_i), \alpha', r)$ can be computed through one pass over the data, selecting the queries q in $freq_I(\mathcal{C}(\mathcal{B}_i), \alpha_i, r)$ such that $Sup(q/r, I) \geq \alpha'$. After the computation of all sets $freq_I(\mathcal{C}(\mathcal{B}_i), \alpha', r)$ $(i = 1, \ldots, k)$, it is then possible to compute the set $\mathcal{F} = freq_I(\mathcal{C}, \alpha', r)$, using the iterative approach presented just above.

If $\alpha' = \alpha$, then the computation of $freq_I(\mathcal{C}, \alpha, r)$ is complete. On the other hand, if $\alpha' > \alpha$, we still have to compute $\mathcal{F}' = freq_I(\mathcal{C}, \alpha, r) \setminus \mathcal{F}$. Using Proposition 9(2), we know that $\mathcal{F}' = freq_I(\mathcal{C} \setminus \mathcal{F}, \alpha, r)$. However, no further optimization is possible for queries in $\mathcal{C} \setminus \mathcal{F}$; one will have to use the standard approach [1].

6 Experimental Results

In order to evaluate the benefits of our iterative approach over non-iterative computations, we performed several experiments. We first describe the synthetic data sets used in the evaluation. Then, in the case of a selection, we study in detail the parameters that influence the gain in computation time. We also give some summary results in the case of a join.

6.1 Synthetic Data Sets

The data sets used in this paper have been generated in a similar way as in [1]. However, the generator used in [1] has been modified so as to comply with our environment that uses more than one table. We first summarize our method for generating data sets and then we give characteristics of the three groups of data sets that we considered in our experiments.

Given a bias $\mathcal{B} = \langle b, \Sigma \rangle$ and a support threshold α, we randomly generate queries of $\mathcal{C}(\mathcal{B})$ that are meant to be frequent in the data set to be generated. To this end, since a selection condition can be seen as a set of atomic selection conditions, we first choose randomly a positive number N for the average number of atomic selection conditions in the frequent queries. Then, N attributes are randomly chosen and for each of them, a value in the corresponding domain is randomly selected. The resulting selection conditions from Σ^* allow to build tuples that are inserted in the tables of the database under construction. We note in this respect that, for a given selection condition S in Σ^*, the number of inserted tuples is set so that the query $\sigma_S(b)$ be α-frequent.

In our experiments, we consider a database containing the three relations *Cust*, *Prod* and *Sales* from our running example. However, in order to increase the number of attributes, the relations *Cust* and *Prod* are defined over 10 attributes each, denoted by $X_1, \ldots, X_{10}$ for *Cust*, and by $Y_1, \ldots, Y_{10}$ for *Prod*. Different instances of this database have been generated, using different parameters settings. These instances are classified in the following three groups:

Table 1. First and second groups of data sets, with different sizes of domains ($D10T4S5$, $D40T4S5$ and $D80T4S5$) and different average sizes of queries ($D40T3S5$ et $D40T5S5$)

Data Sets	$D10T4S5$	$D40T4S5$	$D80T4S5$	$D40T3S5$	$D40T5S5$
Average Size of Patterns	4	4	4	3	5
Size of Domains	10	40	80	40	40
Number of Maximal Patterns	100	100	100	100	100
Cardinality of *Cust* Table	2,000	2,000	2,000	2,000	2,000
Cardinality of *Prod* Table	6,000	6,000	6,000	6,000	6,000
Cardinality of *Sales* Table	54,000	54,000	54,000	54,000	54,000
Minimal Support Threshold	5%	5%	5%	5%	5%

Table 2. Gain of computation time for the first and second groups of data sets

Without Optimization (sec.)	$D10T4S5$	$D40T4S5$	$D80T4S5$	$D40T3S5$	$D40T5S5$
Evaluation Phase	249	182	577	191	420
Total	282	189	599	197	436
Iterative Approach (sec.)					
N-Pass Evaluation	124	55	58	52	127
1-Pass Evaluation	86	32	41	22	100
New Pruning Phase	4	1	7	1	1
Total for N-Pass	224	75	92	59	141
Total for 1-Pass	193	53	81	30	113
Gain for N-Pass	45.0%	69.3%	88.0%	72.6%	69.5%
Gain for 1-Pass	56.0%	81.0%	89.8%	87.3%	75.9%

Group 1 - Instances D10T4S5, D40T4S5 and D80T4S5. In this first group, the cardinalities of the domains of attributes are 10, 40 and 80, respectively. On the other hand, the minimal support threshold is set to 5%, whereas the average size of the frequent queries is equal to 4 [1]. The cardinalities of the tables *Cust*, *Sales* and *Prod* are 2000, 6000 and 54000, respectively.

Group 2 - Instances D40T3S5 and D40T5S5. In this second group, the average size of the frequent queries is 3 (instance $D40T3S5$) or 5 (instance $D40T5S5$). In both cases, the cardinality of the domains of attributes is set to 40, and the minimal support threshold is equal to 5%.

Group 3 - Instances D10T4S3 and D10T4S5. In this third group, the instance $D10T4S3$ has the same characteristics as the instance $D10T4S5$, except that the minimal support threshold is set to 3%. These two instances have been considered to study how the value of the minimal support threshold can influence the gain in computation time.

The characteristics of the data sets in the first and second group are given in Table 1. For these groups, Table 2 shows the gains in computation time that we

[1] Given a bias $\mathcal{B} = \langle b, \Sigma \rangle$, the size of a query $q = \sigma_S(b) \in \mathcal{C}(\mathcal{B})$ is the number of all atomic selection conditions of Σ that occur in S.

Table 3. Candidate pruning for the first and second groups of data sets

Candidate Pruning	$D10T4S5$	$D40T4S5$	$D80T4S5$	$D40T3S5$	$D40T4S5$
Pruning Phase (sec.)	4	1	7	1	1
Pruning Time by Candidate (μs)	32.29	13.33	24.98	13.72	11.79
Nb of Candidates before Pruning	123,874	75,022	280,244	72,878	84,851
Nb of Candidates after Pruning	16,425	4,015	4,567	1,517	6,933
Percentage of Pruned Candidates	86.7%	94.6%	98.4%	97.9%	91.8%
Size of Positive boundary	7,568	662	559	690	589

Table 4. Third group of data sets, with different support thresholds

Data Sets	$D10T4S5$	$D10T4S3$
Average Size of Patterns	4	4
Cardinality of Domains	10	10
Number of Maximal Patterns	100	100
Cardinality of $Cust$ Table	2,000	2,000
Cardinality of $Prod$ Table	6,000	6,000
Cardinality of $Sales$ Table	54,000	54,000
Minimal Support Threshold	5%	3%

Table 5. Gain of computation time for the third group of data sets

Without Optimization (sec.)	$D10T4S5$	$D10T4S3$
Evaluation Phase	249	338
Total	282	609
Iterative Approach (sec.)		
N-Pass Evaluation Phase	124	195
1-Pass Evaluation	86	180
Pruning Phase	4	17
Total for N-Pass	224	616
Total for 1-Pass	193	601
Gain for N-Pass	45.0%	17.7%
Gain for 1-Pass	56.0%	20.2%

Table 6. Candidate pruning for the third group of data sets

Candidate Pruning	$D10T4S5$	$D10T4S3$
Pruning Phase (sec.)	4	17
Pruning Time by Candidate (μs)	32.29	68.7
Nb. of Candidates before Pruning	123,874	249,009
Nb. of Candidate after Pruning	16,425	55,451
Percentage of Pruned Candidates	86.7%	77.7%
Size of Positive Boundary	7,568	22,230

obtain using our iterative approach. To explain these gains, we show in Table 3 some important parameters of the pruning phase used to reduce the execution times of the mining queries.

In the same way, Table 4 summarizes the characteristics of the data sets in the third group. The gains in computation time are given in Table 5, whereas the parameters of the pruning phase are presented in Table 6.

It is important to note that in all these experiments, the positive boundaries of the mining queries already computed are stored in a cache in main memory.

6.2 Experiments with Synthetic Data

We first discus the experimental results obtained in the case of a selection, and then we give some summary results in the case of a join. More details on these experiments can be found in [5].

We recall that in the case of a selection (see Proposition 6), we have to distinguish two cases. Let $\mathcal{B} = \langle b, \Sigma \rangle$ be a mining bias and S be a selection condition. If S belongs to Σ^*, then we know that $freq_I(\sigma_S(\mathcal{C}(\mathcal{B})), \alpha, r)$ is a subset of $freq_I(\mathcal{C}(\mathcal{B}), \alpha, r)$, since by Proposition 6(2), we have $freq_I(\sigma_S(\mathcal{C}(\mathcal{B})), \alpha, r) = \tau_S(freq_I(\mathcal{C}(\mathcal{B}), \alpha, r))$.

Thus, if the positive boundary of $freq_I(\mathcal{C}(\mathcal{B}), \alpha, r)$ is stored, then the generation of the frequent queries in $freq_I(\sigma_S(\mathcal{C}(\mathcal{B})), \alpha, r)$ can be done without accessing the database. Moreover, the computation of their supports requires one pass through the data base. We call this evaluation phase, 1-*Pass Evaluation*.

On the other hand, if S does not belong to Σ^*, we know that the queries in $freq_I(\sigma_S(\mathcal{C}(\mathcal{B})), \alpha, r)$ are selections of queries in $freq_I(\mathcal{C}(\mathcal{B}), \alpha, r)$, since by Proposition 6(1), we have $freq_I(\sigma_S(\mathcal{C}(\mathcal{B})), \alpha, r) \subseteq \sigma_S(freq_I(\mathcal{C}(\mathcal{B}), \alpha, r))$. However, we do not know if queries in $\sigma_S(freq_I(\mathcal{C}(\mathcal{B}), \alpha, r))$ are frequent or not. Therefore, we have to compute their supports level-by-level, and we call this evaluation phase, N-*Pass Evaluation*.

In order to evaluate the gain in computation time in the case of a selection, we consider the reference query $r = \pi_{X_1}(Sales)$ and the bias $\mathcal{B} = \langle Cust \bowtie Sales \bowtie Prod, \Sigma \rangle$ where $\Sigma = \{X_i = x_i \mid i = 2, \ldots, 6 \ \wedge \ x_i \in dom(X_i)\} \cup \{Y_j = y_j \mid j = 1, \ldots, 5 \ \wedge \ y_j \in dom(Y_j)\}$. First, we compute the set of frequent queries $freq_I(\mathcal{C}(\mathcal{B}), \alpha, r)$ and store its positive boundary in a cache. Then, we consider the new context $\sigma_S(\mathcal{C}(\mathcal{B}))$ where $S = \top$, and we compare:

- the time for the computation of the frequent queries of $\sigma_S(\mathcal{C}(\mathcal{B})) = \mathcal{C}(\mathcal{B})$, without any optimization, and
- the time for the iterative computation of the frequent queries of $\sigma_S(\mathcal{C}(\mathcal{B}))$ using the positive boundary of $freq_I(\mathcal{C}(\mathcal{B}), \alpha, r)$.

Since $S = \top$ belongs to Σ^*, we recall that the evaluation phase for the computation of the frequent queries in $\sigma_S(\mathcal{C}(\mathcal{B}))$ can be done in one pass through the data set. However, we have also measured the gain in computation time using an N-Pass Evaluation. Such an experiment allows to evaluate the gain in computation when S belongs or not to Σ^*.

Let us consider the first group of data sets ($D10T4S5$, $D40T4S5$ and $D80T4S5$). We observe important gains in computation time in both cases, *i.e.,* 1-Pass or N-Pass Evaluation (see Table 2). This is due to the fact that about 90% of the candidates are pruned using the positive boundary. We also note that the gain in computation time increases with respect to the cardinality of the domains of attributes. More precisely, if the cardinality of the domains increases, then the number of candidates increases. Therefore, the number of candidates that are pruned before evaluation becomes more important and thus, the gain in computation time is higher.

The same reasoning holds for the second group of data sets ($D40T3S5$ and $D40T5S5$). The important gains in computation time are due to the high percentages of pruned candidates (97.9% and 91.8%). In that case, we note that the gains in computation time decrease with respect to the average size of the frequent queries. Indeed, the bigger the size of the frequent queries, the smaller the number of candidates to be pruned (the number of candidates being the same). This is why the gain in computation time decreases when the average size of the frequent queries varies from 3 to 5.

Finally, considering the instances $D10T4S5$ and $D10T4S3$, the gain in computation time decreases with respect to the minimal support threshold. This is due to the fact that if the minimal support threshold decreases, then the number of frequent queries increases. Therefore, the number of candidates to be pruned decreases.

To evaluate the gain in computation time in the case of a join, we have considered two mining biases $\mathcal{B}_1 = \langle b_1, \Sigma_1 \rangle$ and $\mathcal{B}_2 = \langle b_2, \Sigma_2 \rangle$ where $b_1 = Cust \bowtie Sales$ and $b_2 = Sales \bowtie Prod$. Using the same data sets as before, we compare the computation time without optimization to the computation time using our iterative approach. The gains in computation are given in Table 7.

Table 7. Gain of computation time in case of join of two contexts

Computation Time (sec.)	$D10$ $T4S5$	$D40$ $T4S5$	$D80$ $T4S5$	$D40$ $T3S5$	$D10$ $T4S5$	$D10$ $T4S3$
Without Optimization	282	189	599	197	436	609
Iterative Approach	259	133	400	138	298	598
Gain	8.2%	28.6%	33.2%	29.9%	31.7%	1.8%

7 Conclusion and Perspectives

We have introduced a new formalism for the iterative computation of frequent queries through composition of mining contexts. In our approach, mining contexts are sets of queries, and a mining query is a combination of contexts using operations on contexts. We recall that some of these operators are borrowed from the standard relational algebra, whereas new operators have been introduced. Among these new operators, we have specifically studied the $freq$ operator, which allows to compute the frequent queries in a given context.

We have given basic properties of our operators on contexts with the goal of mining query optimization, in much the same way as query optimization in relational databases. Then, based on these properties, we have shown that if we consider particular contexts, called biases, frequent queries can be efficiently computed. Moreover, we have reported on experiments for mining queries expressed by biases. In particular, we have shown that our approach results in a significant reduction in computation time for the computation of frequent queries of composed contexts.

It is important to notice that our formalism can be seen as an attempt to define an algebra for inductive databases. Indeed, the relational algebra together with the operations introduced in this paper constitute the basis for a language allowing to extract patterns and to query these patterns along with the data present in the database. Therefore, our approach can be seen as providing some of the basic ingredients required for the emergence of inductive databases as specified in [18]. However, in this setting, many important questions remain open, among which we mention the following:

1. The extension of our work to patterns other than frequent queries. Indeed, although our framework is very general, we did not consider, for instance, the case of association rules.
2. The study of the expressive power of the operations. Indeed, it is well known that the relational algebra has strong connections with first-order logic. Therefore a similar question seems relevant in our framework. However, this basic question is far from easy because patterns are formulas, and thus the comparison requires more than first-order logic.
3. The optimization of a single query has not been addressed in this paper. Although we mentioned that standard query optimization techniques can be used in our approach, we did not provide an actual policy. Preliminary results have been obtained in the case of conjunctive queries [8], but actual techniques and algorithms for the optimization of our algebra remain to be studied.
4. As our optimization technique heavily relies on the fact that previously computed contexts are stored, the influence of redundancies in this storage is crucial in our model. We are currently investigating this research direction, based on our preliminary work ([7]), and also on the approach of [16,17].

References

1. 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 309–328. AAAI-MIT Press, 1996.
2. M. Botta, R. Meo, and M. L. Sapino. Incremental execution of the mine rule operator. Technical Report RT66-2002, University of Turin, Turin, May 2002.
3. A.K. Chandra and P.M. Merlin. Optimal implementation of conjunctive queries in relational databases. In *Ninth ACM Symposium on Theory of Computing*, pages 77–90, 1977.

4. L. Dehaspe and H. Toivonen. Discovery of frequent datalog patterns. In *Data Mining and Knowledge Discovery*, volume 3, pages 7–36. Kluwer Academic Publishers, 1999.
5. C.T. Diop. *Etude et mise en oeuvre des aspects itératifs de l'extraction de règles d'association dans une base de données*. PhD thesis, Université de Tours, France, 2003.
6. C.T. Diop, A. Giacometti, D. Laurent, and N. Spyratos. Composition of mining contexts for efficient extraction of association rules. In *EDBT'02*, volume LNCS 2287, pages 106–123. Springer Verlag, 2002.
7. A. Giacometti, D. Laurent, and C.T. Diop. Condensed representations of sets of mining queries. In *Database Support for Data Mining Support*, volume LNCS 2682, pages 250–269. Springer Verlag, 2004.
8. A. Giacometti, D. Laurent, C.T. Diop, and N. Spyratos. Mining from views : An incremental approach. *International Journal Information Theories & Applications*, 9 (Techn. Report LI, Université de Tours, France), 2002.
9. J. Han, Y. Fu, W. Wang, K. Koperski, and O. Zaiane. Dmql : A data mining query language for relational databases. In *SIGMOD Workshop DMKD'96*, pages 27–34, 1996.
10. T. Imielinski and H. Mannila. A database perspective on knowledge discovery. In *Communications of the ACM*, volume 39, pages 58–64, 1996.
11. M. Kamber, J. Han, and J. Chiang. Metarule-guided mining of multi-dimensional association rules using data cubes. In *International Conference on Data Mining and Knowledge Discovery (KDD'97)*, pages 207–210, Newport Beach, USA, 1997.
12. S.D. Lee and L. De Raedt. An algebra for inductive query evaluation. In *IEEE ICDM*, pages 147–154, 2003.
13. H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. In *Data Mining and Knowledge Discovery*, volume 1(3), pages 241–258, 1997.
14. R. Meo, G. Psaila, and S. Ceri. A new sql-like operator for mining association rules. In *22nd VLDB Conf.*, pages 122–133, 1996.
15. T. Morzy, M. Wojciechowski, and M. Zakrzewicz. Materialized data mining views. In *PKDD'00*, volume LNAI 1910, pages 65–74. Springer Verlag, 2000.
16. B. Nag, P. Deshpande, and D.J. De Witt. Using a knowledge cache for interactive discovery of association rules. In *5th ACM SIGKDD Conference*, pages 244–253, 1999.
17. B. Nag, P. Deshpande, and D.J. De Witt. Caching for multi-dimensional data mining queries. In *Systemics, Cybernetics and Informatics (SCI)*, 2001.
18. L. De Raedt. A perspective on inductive databases. In *SIGKDD Explor. Newsletter*, volume 4(2), pages 69–77, 2002.
19. S. Tsur, J.D. Ullman, S. Abiteboul, C. Clifton, R. Motwani, S. Nestorov, and A. Rosenthal. Query flocks : A generalization of association-rule mining. In *ACM SIGMOD Conference*, pages 1–12, 1998.
20. J.D. Ullman. *Principles of Databases and Knowledge-Base Systems*, volume 1. Computer Science Press, 1988.

Inductive Queries on Polynomial Equations

Sašo Džeroski, Ljupčo Todorovski, and Peter Ljubič

Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia

Abstract. Inductive databases (IDBs) contain both data and patterns. Inductive Queries (IQs) are used to access, generate and manipulate the patterns in the IDB. IQs are conjunctions of primitive constraints that have to be satisfied by target patterns: they can be different for different types of patterns. Constraint-based data mining algorithms are used to answer IQs.

So far, mostly the problem of mining frequent patterns has been considered in the framework of IDBs: the types of patterns considered include frequent itemsets, episodes, Datalog queries, sequences, and molecular fragments. Here we consider the problem of constraint-based mining for predictive models, where the data mining task is regression and the models are polynomial equations.

More specifically, we first define the pattern domain of polynomial equations. We then present a complete and a heuristic solver for this domain. We evaluate the use of the heuristic solver on standard regression problems and illustrate its use on a toy problem of reconstructing a biochemical reaction network. Finally, we consider the use of a combination of different pattern domains (molecular fragments and polynomial equations) for practical applications in modeling quantitative structure-activity relationships (QSARs).

1 Inductive Databases

Inductive databases [12] embody a database perspective on knowledge discovery, where knowledge discovery processes are considered as query processes. In addition to normal data, inductive databases contain patterns (either materialized or defined as views). Data mining operations looking for patterns are viewed as inductive queries posed to the inductive database. Besides patterns (which are of local nature), models (which are of global nature) can also be considered.

Given an inductive database that contains data and patterns, several different types of queries can be posed. Data retrieval queries use only the data and their results are also data: no pattern is involved in the query. Cross over queries cross over patterns and data in order to obtain new data. In processing patterns, the patterns are queried without access to the data: this is what is usually done in the post-processing stages of data mining. Data mining queries use the data and their results are patterns: new patterns are generated from the data and this corresponds to the traditional data mining step. When we talk about inductive queries, we most often mean data mining queries.

J.-F. Boulicaut et al. (Eds.): Constraint-Based Mining, LNAI 3848, pp. 127–154, 2005.

A general formulation of data mining [16] involves the specification of a language of patterns and a set of constraints that a pattern has to satisfy with respect to a given database. The constraints that a pattern has to satisfy can be divided in two parts: language constraints and evaluation constraints. The first part only concerns the pattern itself, while the second part concerns the validity of the pattern with respect to a database. Constraints thus play a central role in data mining and constraint-based data mining is now a recognized research topic [2]. The use of constraints enables more efficient induction as well as focusing the search for patterns on patterns likely to be of interest to the end user.

In the context of inductive databases, inductive queries consist of constraints and the primitives of an inductive query language include language constraints (e.g., find association rules with item A in the head) and evaluation primitives. Evaluation primitives are functions that express the validity of a pattern on a given dataset. We can use these to form evaluation constraints (e.g., find all item sets with support above a threshold) or optimization constraints (e.g., find the 10 association rules with highest confidence).

Different types of patterns have been considered in data mining, including frequent itemsets, episodes, Datalog queries, and graphs. Designing inductive databases for these types of patterns involves the design of inductive query languages and solvers for the queries in these languages. For each type of pattern, or pattern domain, a specific solver is designed, following the philosophy of constraint logic programming [4].

While many different types of patterns have been considered in data mining, constraints have been mostly considered in mining frequent patterns, typically frequent itemsets and association rules. Some related tasks, such as mining frequent episodes, Datalog queries, molecular fragments, etc., have also been considered. Here we consider IDBs that contain predictive models in the form of polynomial equations.

We first define constrains on polynomial equations, including language and evaluation constraints. We then propose two solvers, a complete and a heuristic one. We evaluate the use of the heuristic solver on standard regression problems. Its use is also illustrated on a toy problem of reconstructing a biochemical reaction network, i.e., re-discovering differential equations that describe its dynamics. Finally, we consider the use of a combination of different pattern domains (molecular fragments and polynomial equations) for practical applications in modeling quantitative structure-activity relationships (QSARs). We conclude with a discussion of the issues raised by considering the task of mining predictive models in the IDB setting.

2 The Pattern Domain of Polynomial Equations

Here we consider the pattern domain of polynomial equations (`EQN`). We first define the language of polynomial equations, then consider syntactic/subsumption constraints on these. We next define several evaluation primitives for equations and finally discuss inductive queries in this domain.

2.1 The Language of Polynomial Equations

In this paper, we will concentrate on polynomial equations. Given a set of variables V, a polynomial equation has the form $T = P$, where T is a term and P is a polynomial over V. A polynomial P has the form $\sum_{i=1}^{r} \text{const}_i \cdot T_i$, where T_i are multiplicative terms, and const_i are real-valued constants. Of special interest will be equations that can be used to predict the value of a dependent variable $v_d \in V$. These will have the form $v_d = P$, where P is a polynomial over $V \setminus \{v_d\}$.

Each term in a polynomial is a finite product of variables from V, i.e., $T_i = \prod_{v \in V} v^{d_{v,i}}$, where $d_{v,i}$ is (a non-negative integer) degree of the variable in the term. The degree of 0 denotes that the variable does not appear in the term. The sum of degrees of all variables in a term is called the degree of the term, i.e., $\deg(T_i) = \sum_{v \in V} d_{v,i}$.

The degree of a polynomial is the maximum degree of a term in that polynomial, i.e., $\deg(P) = \max_{i=1}^{r} \deg(T_i)$. The length of a polynomial is the sum of the degrees of all terms in that polynomial, i.e., $\text{len}(P) = \sum_{i=1}^{r} \deg(T_i)$.

For example, consider a set of variables $V = \{x, y, z\}$, where z is chosen to be a dependent variable. The term x (that is equivalent to x^1y^0) has degree 1, the term x^2y has degree 3, while x^2y^3 is a term of degree 5. An example polynomial of degree 4 is $1.2x^2y + 3.5xy^3$. An example polynomial equation is $z = 1.2x^2y + 3.5xy^3$. This equation has $r = 2$, $d = 4$, and $len(P) = 7$.

2.2 Syntactic Constraints

We will consider two types of syntactic constraints on equations: parametric constraints and subsumption constraints.

Parametric constraints on polynomial equations include setting upper limits for the degree of a term d (in both the left-hand-side/LHS and right-hand-side/RHS of the equation), as well as the number of terms in the RHS polynomial. For example, one might be interested in equations of degree at most 3 with at most 4 terms. Such parametric constraints are taken into account by the equation discovery system LAGRANGE [8].

Of more interest are subsumption constraints, which bear some resemblance to subsumption/generality constraints on terms/clauses in first-order logic. A term T_1 is a sub-term of term T_2 if the corresponding multi-set M_1 is subset of the corresponding multi-set M_2. For example, xy^2 is sub-term of x^2y^4Z since $\{x, y, y\} \subset \{x, x, y, y, y, y, z\}$.

The sub-polynomial constraint is defined in terms of the sub-term constraint. Polynomial p_1 is a sub-polynomial of polynomial p_2, if each term in p_1 is a sub-term of some term in p_2. There are two options here: one may, or may not, require that each term in p_1 is a sub-term of a different term in p_2.

In looking for interesting equations, one might consider constraints of the form: LHS is a sub-term of t, t is a sub-term of LHS, RHS is a sub-polynomial of p, and p is a sub-polynomial of RHS. Here t and p are a term and a polynomial, respectively. These set upper and lower bounds on the lattice of equation structures, induced by the relations sub-term and sub-polynomial.

Consider the following constraint: LHS is a sub-term of x^2y and both xy and z are sub-polynomials of RHS. The equation $xy = 2x^2y^2 + 4z$ satisfies this constraint, under both interpretations of the sub-polynomial constraint. The equation $x^2y = 5xyz$, however, only satisfies the constraint under the first interpretation (different terms in p_1 can be sub-terms of the same term in p_2).

Subsumption constraints are often used in constraint-based mining of frequent patterns. They allow the user (domain expert) to specify domain knowledge and thus prune away large parts of the space of patterns and focus the search for patterns on parts of the space that are of special interest. We argue that subsumption constraints can be used in the same manner for mining predictive models.

2.3 Evaluation Primitives

The evaluation primitives for equations calculate different measures of the degree of fit of an equation to a given dataset/table. Assume that i is an index that runs through records/rows of a database table. Denote with y_i the value of the LHS of a given equation on record i (actual value of the dependent variable v_d); with $\hat{y}_i$ the value of the RHS as calculated for the same record (predicted value of v_d); and with $\overline{y}$ the mean value of y_i over the table records.

Below we define several measures for the degree of fit for an equation to a dataset. Two measures commonly used for regression are the mean squared error ($MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$) and the multiple correlation coefficient R (defined as $R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i-\hat{y}_i)^2}{\sum_{i=1}^{N}(y_i-\overline{y})^2}$) Džeroski and Todorovski [8] use a normalized version of MSE, denoted by S ($S^2 = \frac{\frac{1}{N}\sum_{i=1}^{N}(y_i-\hat{y}_i)^2}{\overline{y}^2+e^{-\overline{y}^2}}$).

Various types of prediction error measures are often used. These include mean absolute error ($MeanAE = \frac{1}{N}\sum_{i=1}^{N}|\hat{y}_i - y_i|$), maximum absolute error ($MaxAE = \max_{i=1}^{N}|\hat{y}_i - y_i|$), mean square error ($MSE = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$), and root mean square error ($RMSE = \sqrt{MSE}$). Most of these are well known from statistics.

In the machine learning literature, the measure RE, defined as $RE^2 = 1 - R^2$ is often used to evaluate the performance of regression approaches. Note that $RE^2 = MSE/\sigma^2$, where $\sigma^2 = \sum_{i=1}^{N}(y_i - \overline{y})^2$ is the variance of the dependent variable v_d. This normalization allows for comparisons of performance across different data sets.

We will also use a MDL (minimal description length) based heuristic function for evaluating the quality of equations that combines the degree of fit with the complexity of the equation

$$\mathrm{MDL}(v_d = P) = len(P)\log N + N \log \mathrm{MSE}(v_d = P),$$

where $len(P)$ is the length of polynomial P (the sum of the degrees of all terms in that polynomial) and N the number of training examples (data points in D). This evaluation function is based on the Akaike and Bayesian information criteria for regression model selection [9]. While the second term of the MDL

heuristic function measures the degree of fit of a given equation, the first term introduces a penalty for the complexity of the equation. Through this penalty the MDL heuristic function introduces a preference toward simpler equations.

2.4 Inductive Queries in the Domain of Equations

Inductive queries are typically conjunctions of constraints. The primitive constraints typically are evaluation and language constraints. Evaluation constraints set thresholds on acceptable values of the evaluation primitives: $M(e, D) < t$; $M(e, D) > t$, where t is a positive constant/threshold and M is one of the measures defined above.

Instead of evaluation constraints one can consider optimization constraints. Here the task is to find (the n best) e so that $M(e, D)$ is maximal / minimal. Language constraints, as discussed above, can be parametric and/or subsumption constraints.

It is rarely the case that an inductive query consists of a single constraint. Most often, at least one syntactic and at least one evaluation or optimization constraint would be a part of the query. For example, we might look for the equations, where the LHS is sub-polynomial of x^2y^3 and $x+z$ is a sub-polynomial of the RHS, which have the highest multiple correlation coefficient.

3 LAGRANGE-C: A Complete Solver for the Domain of Polynomial Equations

Our initial solver for the pattern domain `EQN`, named LAGRANGE-C, is based on the equation discovery system LAGRANGE [8], which discovers polynomial equations. Below, we first briefly describe LAGRANGE and then proceed to describe LAGRANGE-C.

3.1 Discovering Polynomial Equations with LAGRANGE

LAGRANGE [8] is the first system in the area of computational scientific discovery [15] that can discover both algebraic and ordinary differential equations. In this way, it can find quantitative laws describing the behavior of dynamic systems that change their state over time. Given measurements, LAGRANGE looks for polynomial equations on the set of measured variables and their time derivatives (rates of change over time).

The input to LAGRANGE is a table of measured values of a set of numeric variables V. In addition, the values of the following parameters have to be specified: d is the maximum degree of terms, r is the maximum number of terms, while t_R and t_S are thresholds on the R and S measures (evaluation functions). The task addressed by LAGRANGE is to find all valid equations (for which $R \leq t_R$ and $S \leq t_S$) of the form $T = P$, where T is a term and P a polynomial of at most r terms, both over variables from V and of degree at most d.

LAGRANGE exhaustively searches the space of polynomial equations that can be constructed, given the set of variables and the user-specified limits on the number of terms that can appear in the equations and their degree. LAGRANGE reports the equations that meet the evaluation constraints, i.e., are above/below the user specified thresholds for R/S. The main stages of the LAGRANGE algorithm are given in Table 1.

Table 1. The main stages of the LAGRANGE algorithm

procedure LAGRANGE(V, d, r, t_R, t_S)

1. Introduce time derivatives D of the measured variables V (optional)
2. Introduce new terms N with multiplication (up to degree d) of variables from $V \cup D$, yielding $T = V \cup D \cup N$
3. Generate and test equations with linear regression
   - Generate equation structures (select subsets of T with up to $r + 1$ terms)
   - Select a term (dependent variable) for the RHS from each subset
   - Fit coefficients with linear regression, calculate R and S
4. Report the equations that fit the data well, i.e., have R/S values above/below the user specified threshold t_R/t_S

To illustrate the workings of LAGRANGE, consider a population dynamics example [8], where a predator feeds on a prey and measurements of the population sizes/densities N and P are given. LAGRANGE would first introduce the time derivatives $\dot{N}$ and $\dot{P}$. Given $d = 2$ and $r = 2$, it would first introduce the new terms N^2, NP, $N\dot{N}$, $N\dot{P}$, P^2, $P\dot{N}$, $P\dot{P}$, $\dot{N}^2$, $\dot{N}\dot{P}$, $\dot{P}^2$, and then generate subsets of terms (equation structures). The subsets generated include $\{N\}, \{P\}, \{N, P\}, \{N, \dot{N}\}, \{N, P, \dot{N}\}, \{N, P, \dot{P}\}, \{N, \dot{N}, N^2\}, \{N, \dot{N}, NP\}$, and $\{P, \dot{P}, NP\}$. With the thresholds set to $t_R = t_S = 0.01$, the equations $NP = 160 \cdot N - 100 \cdot \dot{N}$ and $NP = 20 \cdot P + 100 \cdot \dot{P}$ are reported.

3.2 LAGRANGE-C: Using Constraints in LAGRANGE

LAGRANGE-C is an extension of LAGRANGE that includes the ability to take into account constraints. On the evaluation primitives side, we have extended LAGRANGE to calculate the values of four measures of degree of fit, in addition to R and S. These are specified in Section 2 and are: mean absolute error ($MeanAE$), maximum absolute error ($MaxAE$), mean square error (MSE), and root mean square error ($RMSE$).

The evaluation constraints can now specify thresholds on each of the above error measures, just like for R and S. In addition, optimization constraints based on these can be specified. For example, we may specify that we want the b best equations according to $MeanAE$ (or any other of the above evaluation primitives).

On the syntactic constraints side, we have extended LAGRANGE to take into account sub- and super-polynomial constraints. We can now specify that

the left-hand side of an equation should be a sub-term or a super-term of a given term t. Also, we can specify that the right-hand side of an equation should be a sub-polynomial or a super-polynomial of a given polynomial p.

LAGRANGE-C exhaustively searches the space of equations satisfying the combination of syntactic constrains given. At present, a conjunction of evaluation constraints or a single optimization constraint can be solved in addition to syntactic constraints. Only equations satisfying the syntactic constraints are evaluated for their degree of fit, and those satisfying the evaluation/optimization constraints are returned.

To illustrate the use of constraints in LAGRANGE-C, let us revisit the population dynamics example from the subsection above. Given the language constraints $d = 2$ and $r = 2$, LAGRANGE would consider a total of 469 equation structures before reporting the two valid equations. Namely, there are 14 terms alltogether, and we can construct $\binom{14}{1} + \binom{14}{2} + \binom{14}{3} = 469$ equations with at most 3 terms in total.

Suppose we give LAGRANGE-C the constraints that NP should be a subterm of the LHS and that $\dot{N}$ should be a subpolynomial of the RHS, in addition to the parametric constraints already listed. Given $d = 2$, this fixes the LHS to NP. The RHS must have one term that contains $\dot{N}$ and can optionally have a second term, different from the first one. This gives us $5 \times 13 = 65$ possible equation structures that are considered by LAGRANGE-C before reporting the equation $NP = 160 \cdot N - 100 \cdot \dot{N}$ (with the thresholds set to $t_R = t_S = 0.01$).

4 CIPER: A Heuristic Solver for the Domain of Polynomial Equations

While our complete solver LAGRANGE-C is able to take into account constraints and thus consider a reduced search-space, it still performs exhaustive search over this space, just like LAGRANGE [8]. With a slightly larger number of variables and longer target equations, this becomes infeasible. Hence the need for a heuristic solver for the pattern domain of polynomial equations.

In this section, we present a heuristic search algorithm CIPER that searches through the space of polynomial equations and finds one (or several equations) that satisfies the given set of constraints and has an optimal value of the given heuristic function. Here CIPER stands for Constrained Induction of Polynomial Equations for Regression. First, we introduce a refinement operator that generates more complex equations from simpler ones and thus orders the space of polynomial equations. Then, we present the heuristic function used to measure the quality of equations and discuss how constraints are handled in CIPER. After presenting the search algorithm based on beam search strategy, we look into how parameter fitting is optimized by taking into account properties of the refinement operator. Finally, we discuss the relation of CIPER to stepwise regression methods.

LAGRANGE and LAGRANGE-C look for implicit polynomial equations of the form $T = P$, where T is a term and P a polynomial over variables from V.

CIPER, on the other hand, looks for explicit regression equations of the form $v_d = P$, where v_d is a dependent variable and P a polynomial over the remaining variables from V. The task that CIPER addresses is to find the polynomial P which satisfies the given language constraints and has the lowest value of the heuristic function.

4.1 The Refinement Operator

Machine learning and data mining methods typically perform heuristic search over the space of possible models and patterns. After an initial model/pattern in this space is chosen (which usually does not fit the data very well), it is then gradually refined to fit the data. Refinement steps are performed by a so-called refinement operator and typically involve making small changes to the model, e.g., adding a condition to the antecedent of a classification rule.

In order to apply heuristic search methods to the task of inducing polynomial equations, we first order the search space of candidate equations. We introduce a refinement operator that orders this space according to equation complexity. Starting with the simplest possible equation and iteratively applying the refinement operator, all candidate polynomial equations can be generated.

Table 2. The refinement operator for ordering the space of polynomial equations

Original (current) equation
$v_d = \sum_{i=1}^{r} \text{const}_i \cdot T_i$
Refined equations that increase r (one for each $v \in V \setminus v_d$)
$v_d = \sum_{i=1}^{r} \text{const}_i \cdot T_i + \text{const}_{r+1} * v$, where $\forall i : v \neq T_i$
Refined equations that increase d (one for each T_j and $v \in V \setminus v_d$)
$v_d = \sum_{i=1, i \neq j}^{r} \text{const}_i \cdot T_i + T_j * v$, where $\forall i \neq j : T_j * v \neq T_i$

Assume we measure the complexity of the polynomial equation $v_d = P$ as $\text{len}(P)$. The refinement operator, defined in Table 2, increases the complexity of the equation by 1, either by adding a new linear term or by adding a variable to an existing term. First, we can add an arbitrary linear (first degree) term (that is a single variable from $V \setminus \{v_d\}$) to the current equation as presented in the first (upper) part of Table 2. Special care is taken that the newly introduced term is different from all the terms in the current equation. Second, we can increase the complexity $\text{len}(P)$ by adding a variable to one of the terms T_j in the current equation. Again, care should be taken that the changed term is different from all the other terms in the current equation. Note that the refinements of a given polynomial P are super-polynomials of P. They are minimal refinements in the sense that they increase the complexity of P by one unit.

The branching factor of the presented refinement operator depends on the number of variables $|V|$ and number of terms in the current equation r. The

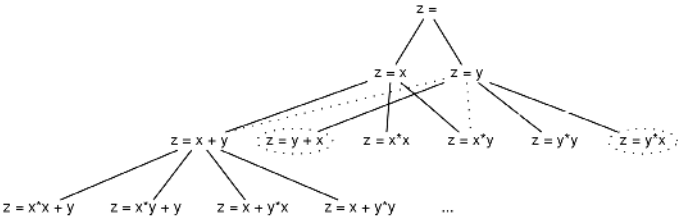


Fig. 1. The search space of polynomial equations over the set of variables $V = \{x, y, z\}$, where z is the dependent variable, as ordered by the refinement operator from Table 2. Note that for simplicity, real-valued constants are omitted from the equations.

upper bound of the branching factor is $\mathcal{O}((|V|-1)(r+1)) = \mathcal{O}(|V|r)$, since there are at most $|V| - 1$ possible refinements that increase r and at most $(|V| - 1)r$ possible refinements that increase d.

The ordering of the search space of polynomial equations, defined on the set of variables $V = \{x, y, z\}$, where z is the dependent variable, is presented in Figure 1. It shows that the defined refinement operator is not optimal, in the sense that each polynomial equation can be derived more than once. This is due to the commutativity of the addition and multiplication operators. An optimal refinement operator can be easily obtained by taking into account the lexical ordering of the variables in V. Then, only variables (and/or terms) with higher lexical rank should be added to the terms and/or equations. The dotted nodes in the graph in Figure 1 denote equations that would not be generated by the refinement operator that takes into account lexical order. The redundancy due to the sub-optimality of the refinement operator is taken care of during the beam search procedure by removing duplicates from the beam (see Section 4.4).

While an optimal refinement operator is desired for complete/exhaustive search, it may prevent the generation of good equations in greedy heuristic search. Suppose the polynomials x and z have low heuristic value, while y has a high heuristic value and $x+y$ is actually the best. Greedy search would choose y and the optimal refinement operator that takes into account lexicographic order would not generate $x + y$.

4.2 The Search Heuristic

Each polynomial equation structure considered during the search contains a number of generic constant parameters (denoted by $const_i$). In order to evaluate the quality of an equation, the values of these generic constants have to be fitted against training data consisting of the observed values of the variables in V. Since polynomial equations are linear in the constant parameters, the standard linear regression method can be used for this purpose.

The quality of the obtained equation is evaluated using a degree of fit measure that measures the discrepancy between the observed values of v_d and the values predicted using the equation. One such measure is mean squared error (MSE), defined in Section 2.3. CIPER uses an MDL (minimal description length) based heuristic

function for evaluating the quality of an equation $v_d = P$ that combines the degree of fit (MSE($v_d = P$)) with the complexity of the equation ($len(P)$), also defined in Section 2.3. The latter introduces a preference toward simpler equations.

4.3 Constraints in CIPER

The task of CIPER is to find polynomial equations that express the dependent variable v_d as a polynomial function of the dependent variables $V \setminus v_d$. The optimization constraints concern the heuristic function described in the previous subsection. CIPER takes as input a parameter b and uses heuristic search to find b polynomial equations that have the highest values of the heuristic.

CIPER also takes as input a set of language constraints, which can be parametric or subsumption constraints, as explained in Section 2. These constraints apply to the polynomial at the right-hand-side of the target equation. They place bounds on the lattice of equation structures introduced by the refinement operator defined earlier in this section.

If a constraint specifies that polynomial l should be a sub-polynomial of the RHS, then it bounds the lattice of equation structures from below. If a constraint specifies that polynomial u should be a super-polynomial of the RHS, then it bounds the lattice of equation structures from above. If a RHS structure violates such a super-polynomial constraint, then all of its refinements will also violate it (and can consequently be pruned from the search tree).

The parametric constraint on the number of terms limits the number of applications of the first rule of the refinement operator. The constraint on the degree of terms limits the number of applications of the second rule of the refinement operator to a single term. The length of a polynomial is the number of refinement steps needed to reach this polynomial from the simplest equation structure $v_d = const$. An interesting constraint, which we have not yet considered, would place a limit on the length of polynomials considered (and thus a limit on the depth of the refinement graph).

Table 3. A top-level outline of CIPER's beam search algorithm

```
   procedure CIPER(D, v_d, C, b)
 1 E_0 = simplest polynomial equation (v_d = const )
 2 E_0.MDL = FitParameters(E_0, D)
 3 Q = {E_0}
 4 repeat
 5    Q_r = {refinements of equation structures in Q
                 that satisfy the language constraints C}
 6    foreach equation structure E ∈ Q_r do
 7       E.MDL = FitParameters(E, D)
 8    endfor
 9    Q = {best b equations from Q ∪ Q_r according to MDL }
10 until Q unchanged during the last iteration
11 print Q
```

4.4 The Search Algorithm

CIPER employs beam search through the space of possible equations using the search algorithm presented in Table 3. The algorithm takes as input a training data set D containing the values of independent variables and the dependent variable v_d. CIPER also takes as input a set of language constraints C, which can be parametric or subsumption constraints. The output of CIPER consists of the b best polynomial equations, according to the MDL heuristic function defined in the previous section, that satisfy the language constraints in C.

Before the search procedure starts, the beam Q is initialized with the simplest possible polynomial equations of the form $v_d = \text{const}$. The value of the constant parameter const is fitted against the training data D using linear regression. In each search iteration, the refinements of the equations in the current beam are generated (using the refinement operator from Table 2). Those that satisfy the constraints in C are collected in Q_r (line 5).

In case redundant equations are generated due to the sub-optimality of the refinement operator, the duplicate equations are filtered out from the set Q_r. Linear regression is used to fit the constant parameters of the refinements against the training data D (lines 6-8). At the end of each search iteration, only the best b equations, according to the MDL heuristic function, are kept in the beam (line 10). The search stops when the performed iteration does not change the beam.

4.5 Optimization of Parameter Fitting

To evaluate each polynomial equation structure considered during the search, CIPER fits its constant parameters using least squares linear regression. Let X be the matrix of measurements of the vector of independent variables $V = (x_1, \ldots, x_n)$, augmented by a column of ones $X = [\underline{x}_1, \ldots \underline{x}_n, \underline{1}]$ and $\underline{y}$ the vector of measurements of the dependent variable v_d. In this case, we are looking for a vector of constant coefficients $\underline{c}$ that minimizes the expression $||\underline{c}X - \underline{y}||_2$. Such a vector of constant coefficients can be computed by using the formula $\underline{c} = (X^T X)^{-1}(X^T \underline{y})$.

As we can see, we have to compute the product of the two matrices X^T and X first. Let us take a look at some properties of $X^T X$. Since it is a matrix consisting of scalar products of vectors, the matrix $X^T X$ is symmetric. Therefore, we compute only the values above and on the diagonal.

Note that at each iteration of the beam search, we only consider equation structures that are refinements of the structures in the current beam. An important property of the refinement operator is used in CIPER to optimize the computation of $X^T X$. After we refine an equation, the corresponding X matrix only changes slightly: a new column is added to the matrix if a new term is added to the structure, or an existing column is replaced with a new column if an existing term is multiplied by a variable.

Given that the matrix X only changes slightly for a refined structure, we do not have to compute $X^T X$ anew (from scratch), but can reuse a large part of the $X^T X$ matrix for the original structure. When adding a new variable to

an equation, the dimension of $X^T X$ increases by one: we need to calculate are the scalar products of the newly added variable with all existing variables (plus the scalar product of newly added variable with itself). In case the refinement operator multiplies one term with a variable, the size of the $X^T X$ matrix remains the same. We discard the values of the multiplied term (one row and one column), and we compute the missing scalar products. For an illustration, we refer the reader to Todorovski et al. [20]. We can optimize the calculation of the product $X^T \underline{y}$ in a similar fashion: we need to calculate only the scalar product of the newly added variable and the dependent variable, while the rest of the product vector remains the same.

The complexity of computing $X^T X$ with v variables and N examples is $O(Nv^2)$. The complexity of our optimized approach is $O(Nv)$. For computing the product $X^T \underline{y}$, the optimization reduces the complexity from $O(Nv)$ to $O(N)$.

4.6 Stepwise Regression

The CIPER search algorithm is similar in spirit to the forward stepwise method for linear regression [9]. The stepwise regression method also starts with the simplest model $v_d = \text{const}$ and incrementally adds those independent variables to the model that most significantly improve its fit to the training data. To avoid overfitting, stepwise regression methods test the significance of the MSE improvement gained by refining the current equation and do not take into account those refinements that do not lead to significant improvements. The significance of the MSE improvement is based on the F statistic:

$$F = \frac{\text{MSE}(v_d = P) - \text{MSE}(v_d = P')}{MSE(v_d = P')} \cdot (m - r - 2),$$

where $v_d = P$ is the current equation, $v_d = P'$ is the candidate equation with the newly added term, r is the number of terms in the current equation, and m is the number of training examples. The improvement is significant, if the obtained F value is greater than the 95th percentile of the $F(1, m - r - 2)$ distribution [9]. The stepwise regression method proceeds with greedy search by choosing the best significant improvement and stops if no significant improvement is possible.

CIPER can be viewed as a stepwise method for polynomial regression with an MDL (instead of MSE) heuristic function. However, there are several other important differences between CIPER and the stepwise regression method. In particular, the refinement operator used in CIPER is better suited for polynomial regression. While the stepwise regression method can only refine the current equation by adding a new term to it, CIPER can also add a variable to an existing term in the current equation. Using this second kind of refinement, CIPER can generate polynomials of arbitrary degree.

To use the forward stepwise method for polynomial regression, terms of degree two and higher have to be precomputed and introduced as new independent variables. This, however, is a serious limitation of the stepwise method, since the

precomputation of higher degree terms requires the user to specify the maximal degree of the introduced terms upfront and introduces a potentially huge number of independent variables. The number of independent variables is of the order $\mathcal{O}(|V|^d)$, where d is the maximal degree of precomputed terms.

The huge number of precomputed higher degree terms means a high branching factor of the stepwise refinement operator. Since it adds a new term to the current equation, its branching factor equals the number of independent variables, i.e., $\mathcal{O}(|V|^d)$. Note that the branching factor of CIPER's refinement operator ($\mathcal{O}(|V|r)$) is linear in the number of independent variables. The lower branching factor of the refinement operator permits the use of higher beam widths in CIPER, which is in contrast with the beam width of one (greedy search) used for stepwise regression.

5 Evaluating CIPER on Standard Regression Datasets

Equation discovery approaches, such as LAGRANGE [8], have been typically evaluated in terms of the successful rediscovery of quantitative laws. In particular, data generated from known laws/models has been used. The emphasis has mainly been on the comprehensibility and general validity of the laws found, rather than their predictive power. One of the reasons for this has been the prohibitive computational complexity of applying exhaustive search to general regression problems involving many variables and potentially complex laws. CIPER, however, uses a heuristic approach for equation discovery, and its computational performance allows its use on standard regression datasets.

In this section, we perform a number of experiments to evaluate CIPER's predictive performance. We are especially interested in its performance in comparison with the standard regression methods for inducing linear and piecewise models, implemented in the data mining suite WEKA [25], as well as stepwise polynomial regression, as described in the previous section. Besides predictive performance, we also look at the size of the induced models.

The performance of the selected methods is evaluated on thirteen data sets, taken from the UCI Repository [3] and another publicly available collection of regression data sets [22]. These data sets have been widely used in other comparative studies. Table 4 presents the basic properties the data sets. In addition to the number of examples and variables, we also list the class variance.

5.1 Experimental Methodology and Settings

In all the experiments presented here, we estimate the predictive performance on unseen examples using 10-fold cross validation. The predictive performance of a regression model M is measured in terms of the relative mean squared error (RE) defined in Section 2.3. RE is a normalized performance measure that allows for comparisons of performance across different data sets.

We compare the performance of CIPER to the performance of three standard regression methods implemented in WEKA: linear regression, regression trees,

Table 4. Properties (number of variables n, number of examples m, and class variance $\mathrm{VAR}(v_d)$) of the thirteen regression data sets used in the experiments

Data set	n	m	$\mathrm{VAR}(v_d)$
auto_price	16	159	$3.433 \cdot 10^7$
baskball	5	96	0.01173
bodyfat	15	252	69.76
cal_housing	9	20640	$1.331 \cdot 10^{10}$
elusage	3	55	566.0
fried_delve	11	40768	24.97
house_8l	9	22784	$2.792 \cdot 10^9$
housing	14	506	84.4196
kin_8nm	9	8192	0.06948
mbagrade	3	61	0.1063
pw_linear	11	200	19.92
quake	4	2178	0.03587
vineyard	4	52	18.94

and model trees. The tree-based models are induced with the M5' algorithm [24]. All algorithms have been used with their default parameter settings. We also compare CIPER to stepwise polynomial regression (with maximal polynomial degree of 1, 2, and 3). The default beam width in CIPER is 16.

For pairwise comparison of methods, we calculate the relative error reduction achieved for a given data set by using method M_1 as compared to using method M_2: $1 - \mathrm{RE}(M_1)/\mathrm{RE}(M_2)$. The statistical significance of the difference is tested using the paired t-test (with the sample size equal to the number of folds; the same folds are used for all methods) with significance level of 99%. A $+/-$ sign to the right of a figure in Table 5 means that the difference is significant.

5.2 Experimental Results

We present the results of the experiments in Tables 5 and 6. The first table compares the regression methods in terms of their predictive error, while the second compares the complexity of the induced models.

Predictive error. CIPER clearly outperforms linear regression and stepwise linear regression ($d = 1$) on most of the experimental data sets (and significantly on five of them). The stepwise regression methods gain accuracy with increasing the maximal degree of precomputed terms d, but they induce much more complex models and tend to overfit the training data. Compare, for example, the results on the house_8L, pw_linear, and cal_housing data sets. Although insignificant, the differences in performance are still considerably large, especially for $d = 3$. In terms of significant differences, the performance of stepwise regression with $d = 2$ and $d = 3$ are comparable to CIPER.

The results of stepwise regression indicate that further improvement is possible if we increase d further: however this is intractable for large data sets. Also, stepwise regression tends to produce more and more complex models as

Table 5. Predictive performance of CIPER in terms of relative mean squared error (RE), as compared to stepwise regression (with maximal polynomial degree of 1, 2, and 3) and three regression approaches from WEKA: linear regression LR , model trees MT, and regression trees RT. A +/− sign to the right of a figure denotes that CIPER performed significantly better/worse.

		Stepwise regression			WEKA		
Data set	CIPER	$d = 1$	$d = 2$	$d = 3$	LR	MT	RT
auto_price	0.1610	0.2426	0.1985	0.3966	0.2168	0.1351	0.2896 +
baskball	0.6334	0.6334	0.6397	0.6218	0.6672	0.6334	0.8351 +
bodyfat	0.0282	0.0285	0.0324	0.0286	0.0295	0.0260	0.1025 +
cal_housing	0.2901	0.3639 +	0.3339	0.3510	0.3639 +	0.2376 −	0.2664 −
elusage	0.2720	0.3604	0.2720	0.2720	0.3731	0.3312	0.4827
fried_delve	0.1021	0.2773 +	0.1128	0.0436	0.2773 +	0.0765	0.1271
house_8L	0.3793	0.6193 +	0.4370 +	16.1411	0.6191 +	0.3545	0.3932
housing	0.1768	0.2866 +	0.1676	0.1680	0.2858 +	0.1745	0.2550 +
kin_8nm	0.3631	0.5871 +	0.4390 +	0.2684 −	0.5871 +	0.3673	0.4711 +
mbagrade	0.8403	0.8403	0.8450	0.8502	0.8403	0.8403	1.0209
pw_linear	0.3936	0.2504	0.1162	0.6122	0.2377	0.1047	0.3264
quake	1.0000	0.9964	0.9964	0.9997	0.9966	1.0035	1.0102
vineyard	0.5347	0.7400	0.6674	0.7679	0.7116	0.7404	0.7207
Average	0.3980	0.4789	0.4044	1.6555	0.4774	0.3865	0.4847

d increases. Finally, the performance of stepwise polynomial regression is very sensitive to the value of d. Selecting the optimal d value is a nontrivial problem, since it can differ from one data set to another: for practical reasons, the selection would be guided by computational complexity issues (the number of precomputed higher degree terms). The computational complexity of CIPER compares favorably to the stepwise regression method with $d = 3$: on average, CIPER considers 10 times fewer candidate equations than stepwise regression.

The overall accuracy of CIPER compares favorably to the accuracy of RTs: CIPER is significantly better on five data sets. The relative accuracy improvement is higher for smaller data sets [21], which provide insufficient statistical support for a number of partial models derived from parts of the data set (as in RTs and MTs), but a sufficient support for a single equation/model over the entire data set (as in CIPER). Finally, the overall accuracy of CIPER is comparable to the accuracy of MTs: MTs significantly outperform polynomial equations on only one data set. In addition, CIPER has much lower variance (in the bias-variance decomposition of the predictive error) than RTs and MTs [21].

Model Complexity. We assess the complexity of models based on polynomial equations in terms of the number of constant parameters #P, length LEN (as defined in Section 2.1), and polynomial degree DEG. The complexity of tree-based models is measured with the number of constant parameters in the leaf nodes #P, and the number of decision (internal) nodes #DN. Note that in regression trees, #P is the number of leaf nodes.

Table 6. The complexity of the models induced with CIPER as compared to stepwise regression, linear regression and tree-based models in terms of the number of constant parameters in the equation #P, polynomial length LEN and degree DEG, as well as the number of decision nodes #DN for tree-based piecewise models

	CIPER			Stepwise ($d = 2$)			Regr. Trees		Model Trees	
Data set	#P	LEN	DEG	#P	LEN	DEG	#P	#DN	#P	#DN
auto_price	5	5	2	7	10	2	8	7	19	6
baskball	3	2	1	3	3	2	2	1	3	0
bodyfat	8	11	3	4	5	2	16	15	12	5
cal_housing	24	81	17	40	71	2	499	498	898	268
elusage	3	3	2	3	3	2	3	2	6	1
fried_delve	7	7	2	66	120	2	1919	1918	2356	538
house_8l	35	163	13	34	60	2	266	265	434	127
housing	15	32	4	29	53	2	26	25	56	18
kin_8nm	13	16	2	33	56	2	264	263	409	105
mbagrade	3	2	1	3	2	1	1	0	3	0
pw_linear	10	12	2	13	18	2	14	13	12	1
quake	2	1	1	2	1	1	1	0	10	5
vineyard	4	4	2	3	3	2	4	3	6	1
Average	10.5	26.1	4.0	18.5	31.1	1.8	232.5	231.5	324.9	82.7

Table 6 presents the results of the model complexity comparison. CIPER produces less complex polynomials than stepwise regression with maximal degree of precomputed terms $d = 2$. Despite the (two times) higher average degree of the equations induced with CIPER, they are shorter and have two times fewer parameters than the equations induced with stepwise regression. The complexity of the polynomial models is much lower than the complexity of piecewise tree-based models. The factor of complexity reduction is of the order of a hundred for both regression and model trees.

6 Using Constraints in Modeling Chemical Reactions

CIPER can also discover differential equations. Like LAGRANGE [8], it can introduce the time derivatives of system variables by numerical derivation. It can then look for equations of the form $\dot{x_i} = P(x_1, x_2, ..., x_n)$, where $\dot{x_i}$ denotes the time derivative of x_i and $P(x_1, x_2, ..., x_n)$ denotes a polynomial of $x_1, x_2, ..., x_n$.

To illustrate the use of constraints in discovering differential equations, we address the task of reconstructing a partially specified (toy example) network of biochemical reactions [7]. Biochemical reactions take place continually in the metabolic processes of all living organisms. Biochemical kinetics studies the rates of biochemical reactions and the dynamic change of the concentration of various reactants (proteins and enzymes) involved in a particular metabolic process [23]. The latter is modeled using ordinary and partial differential equations.

The metabolic process is usually presented graphically as a network of chemical reactions (referred to as a metabolic pathway map). Table 7 gives an ex-

Table 7. A partially specified network (set) of chemical reactions, where the reactions are given in the notation $\{Inputs\} \rightarrow \{Outputs\}$

$\{x_5, \mathbf{x_7}\} \rightarrow \{x_1\}$
$\{x_1\} \rightarrow \{x_2, x_3\}$
$\{x_1, x_2, \mathbf{x_7}\} \rightarrow \{x_3\}$
$\{x_3\} \rightarrow \{x_4\}$
$\{x_4\} \rightarrow \{x_2, \mathbf{x_6}\}$
$\{\mathbf{x_4}, \mathbf{x_6}\} \rightarrow \{\mathbf{x_2}\}$

ample network of chemical reactions. The reactions are given in the notation $\{Inputs\} \rightarrow \{Outputs\}$. For example, the first reaction $\{x_5, x_7\} \rightarrow x_1$, takes the substrate substances x_5 and x_7 as inputs, and produces a single substance x_1.

There are several methods for transforming a metabolic pathway into ordinary differential equations for modeling the change of the concentrations of the substances involved. One of these models a network of chemical reactions with a set of polynomial differential equations (see, e.g., [13]). The transformation of a network to a set of differential equations is performed in the following manner. The reaction rate is proportional to the concentrations of inputs involved in the reaction. For example, consider the reaction $\{x_5, x_7\} \rightarrow x_1$, the first in Table 7. It takes x_5 and x_7 as inputs, therefore the corresponding term in the equations is $x_5 \cdot x_7$. The reaction rate influences the rate of change of all (input and output) compounds involved in the equations. Therefore, the term $x_5 \cdot x_7$, will appear in the equations for x_1, x_5, and x_7. In the equation for the output compound x_1 the term positively influences the change rate, while in the equations for the input compounds the term negatively influences the change rate.

Following the algorithm outlined above, the following set of differential equations can be composed and used for modeling the network from Table 7:

$$\begin{aligned}
\dot{x_1} &= 0.8 \cdot x_5 \cdot x_7 - 0.5 \cdot x_1 - 0.7 \cdot x_1 \cdot x_2 \cdot x_7 \\
\dot{x_2} &= 0.7 \cdot x_1 + 0.2 \cdot x_4 + 0.1 \cdot x_4 \cdot x_6 - 0.3 \cdot x_1 \cdot x_2 \cdot x_7 \\
\dot{x_3} &= 0.4 \cdot x_1 + 0.3 \cdot x_1 \cdot x_2 \cdot x_7 - 0.2 \cdot x_3 \\
\dot{x_4} &= 0.5 \cdot x_3 - 0.7 \cdot x_4 \cdot x_6 \\
\dot{x_5} &= -0.6 \cdot x_5 \cdot x_7 \\
\dot{x_6} &= 0.2 \cdot x_4 - 0.8 \cdot x_4 \cdot x_6 \\
\dot{x_7} &= -0.1 \cdot x_1 \cdot x_2 \cdot x_7 - 0.1 \cdot x_5 \cdot x_7
\end{aligned}$$

These equations were simulated for 1000 time steps of 0.01 from a randomly generated initial state (where the value of each variable was randomly chosen from the interval (0,1)). This provides a trace of the behavior of the 7 system variables over time, suitable for discovering differential equations with CIPER. Given such a behavior, the task is to reconstruct the metabolic pathway.

The domain of modeling networks of chemical reactions lends itself naturally to the use of constraints in polynomial equation discovery. On one hand, parametric constraints have a natural interpretation. A limit on r, the number of

terms in an equation, corresponds to a limit on the number of reactions a compound is involved in. A limit on d, the degree of terms, corresponds to a limit on the number of compounds that are input to a chemical reaction. On the other hand, subsumption constraints can also be used in a natural way. A partially specified reaction network gives rise to equations that involve subpolynomials of the polynomials modeling the entire network, as illustrated below.

Let us revisit the network in Table 7. The part of the network given in bold is assumed to be unknown (except for the fact that x_6 and x_7 are involved in the network), while the rest of the network is known. The task at hand is to reconstruct the complete structure of the network from a given behavior over time (mentioned above) and the partial structure (assumed to be known). This is a task of revising an equation-based model [19].

If only the bold part of the network is present, the following equations can be used to model its behavior.

$$\dot{x_1} = -const \cdot x_1 + const \cdot x_5 - const \cdot x_1 \cdot x_2$$
$$\dot{x_2} = const \cdot x_1 + const \cdot x_4 - const \cdot x_1 \cdot x_2$$
$$\dot{x_3} = const \cdot x_1 + const \cdot x_1 \cdot x_2 - const \cdot x_3$$
$$\dot{x_4} = const \cdot x_3 - const \cdot x_4$$
$$\dot{x_5} = -const \cdot x_5$$

The knowledge of the partial network can be used to constrain the search through the space of possible equations. The RHS polynomials in the equations for $\dot{x_1}$ ... $\dot{x_5}$ in the partial network should be subpolynomials of the RHS polynomials in the corresponding equations for the complete network. These subpolynomial constraints were given to CIPER together with the behavior trace for all 7 variables. No subsumption constraints were used for the equations defining $\dot{x_6}$ and $\dot{x_5}$. No parametric constraints were used for any of the equations. Beams of size 64 and 128 were used in the experiments.

CIPER then successfully reconstructs the equations for the entire network, i.e., for each of the 7 system variables, for each of the two beam sizes. Discovery without constraints, however, fails for two of the equations. If we provide CIPER only with the behavior trace and no constraints, it fails to reconstruct the equations for $\dot{x_1}$ (beam 128) and $\dot{x_2}$ (for both beams) correctly. In addition, unconstrained search inspects more equations than constrained search: for $\dot{x_2}$ and beam 128, unconstrained search considers 18643 equations, while constrained search considers 12901 equations. For comparison, exhaustive search through all equations with $d \leq 3$ and $r \leq 4$ would have to consider 637393 equations.

7 Combining Pattern Domains for Practical Applications: Towards Inductive Databases for QSAR

Here we first describe the pattern domain of molecular fragments. We then proceed with a proposal of how to integrate the pattern domains of equations and

molecular fragments in order to obtain an inductive database for QSAR (Quantitative Structure-Activity Relationships). Preliminary experiments in the domain of predicting biodegradability, illustrating how the two domains can be combined, are presented. Finally, two more complex scenarios for integrating the two pattern domains are considered and their use is illustrated on a problem from the area of predictive toxicology, namely predicting fish toxicity.

7.1 The Pattern Domain of Molecular Fragments

Here we briefly summarize the pattern domain of molecular fragments, introduced by Kramer and De Raedt [14], in terms of the syntactic constraints and evaluation primitives considered. The system MolFea, which implements a solver for this pattern domain, looks for interesting molecular fragments (features) in sets of molecules. Interestingness is defined as a conjunction of primitive constraint that the fragment has to satisfy.

A molecular fragment is defined as a sequence of linearly connected atoms. For instance, $o - c = o$ is a fragment meaning: "an oxygen atom with a single bond to a carbon atom with a double bond to an oxygen atom". In such expressions 'c', 'n', 'o', etc. denote elements, and '$-$' denotes a single bond, '$=$' a double bond, '$\#$' a triple bond, and '$\sim$' an aromatic bond. Only non-hydrogen atoms are considered. Fragments are partially ordered by the relation "is more general than"/"is a subsequence of": when fragment g is more general than fragment s, one writes $g \leq s$.

Kramer and De Raedt note that the representation of molecular fragments is relatively restricted compared to some other representations employed in data mining, such as first-order queries or subgraphs. Although fragments are a relatively restricted representation of chemical structure, it is easy for trained chemists to recognize the functional group(s) that a given fragment occurs in. Thus, the interpretation of a fragment reveals more than meets the eye.

The primitive language constraints that can be imposed on unknown target fragments f are of the form $f \leq p$, $p \leq f$, $\neg(f \leq p)$ and $\neg(p \leq f)$, where f is the unknown target fragment and p is a specific pattern. This type of primitive constraint denotes that f should (not) be more specific (general) than the specified fragment p. E.g., the constraint '$c = o$' $\leq f$ specifies that f should be more specific than '$c = o$', i.e., that f should contain '$c = o$' as a subsequence.

The evaluation primitive $freq(f, D)$ denotes the relative frequency of a fragment f on a set of molecules D. The relative frequency is defined as the percentage of molecules (from D) covered (by f). Evaluation constraints can be defined by specifying thresholds on the frequency or relative frequency of a fragment on a dataset: $freq(f, D_1) \leq t$ and $freq(f, D_1) \geq t$ denote that the relative frequency of f on D should be larger than (resp. smaller than) or equal to t. For example, the constraint $freq(f, Pos) \geq 0.95$ denotes that the target fragments f should have a minimum relative frequency of 95% on the set of molecules Pos.

The primitive constraints defined above can conjunctively be combined in order to declaratively specify the target fragments of interest. Note that the conjunction may specify constraints w.r.t. any number of datasets, e.g. imposing

a minimum frequency on a set of active molecules, and a maximum one on a set of inactive ones. For example, the constraint ($'c = o' \leq f) \wedge \neg(f \leq 'c-c-o-c = o')$ $\wedge freq(f, Deg) \geq 0.95 \wedge freq(f, NonDeg) \leq 0.05$) queries for all fragments that include the sequence $'c = o'$, are not a subsequence of $'c - c - o - c = o'$, have a frequency on Deg that is larger than 95 percent and a frequency on $NonDeg$ that is smaller than 5 percent.

7.2 Combining Molecular Fragments and Equations in an Inductive Database for QSAR

The basic idea of our proposal is to consider the pattern domains of equations and molecular fragments in a single inductive database. One could then easily use the results of one inductive query (e.g., the set of interesting features resulting from a MolFea query) as input to another inductive query (e.g. to find a QSAR equation for biodegradability). This is of interest as QSAR models most often take the form of equations and molecular fragments are often used as features.

This is similar to what Kramer and De Raedt [14] do. They look for interesting features with MolFea, then use these for predictive modeling (classification with a number of data mining approaches). However, no constraints are used in their predictive modeling phase. Also, the data mining approach that performs best in their setup is that of support vector machines, which does not yield interpretable models.

Each of the two pattern domains offers potentially useful functionality. Taken by themselves, equations are the approach of choice for QSAR modeling. While non-linear transformations of bulk features are sometimes used, most often linear equations are sought. Our constraint-based approach to equation discovery would allow the QSAR modeler to pose inductive queries that focus the search on interesting equations, such as: "find me the best equation that involves featureX", supposing featureX is of interest to the QSAR modeler.

Also, molecular fragments (or in general substructures) are often used as features in QSAR modeling. Using feature mining in the pattern domain of molecular fragments, the QSAR modeler can find patterns involving substructures of special interest. These can then be used in QSAR models.

The basic form of exploiting the connection between the two pattern domains is to use the molecular fragments found by MolFea as input for a data mining query looking for QSAR equations. Here one can exploit the subpolynomial constraints in the equations pattern domain to ask for equations that contain a specific molecular fragment: the fragment in question should be a subpolynomial of the RHS of the equation sought. However, additional constraints can be defined. For example, one may look for equations that involve a subfragment or superfragment of a given fragment, rather than just the fragment itself. We have implemented the latter in our system CIPER as an extension of the subpolynomial/superpolynomial constraint.

At present, we assume all frequent fragments are given and look for equations that involve the given equations and satisfy the constraints. As an illustration, in the biodegradability dataset, we may be interested in the two best equations

that contain only one feature that is a superfragment of '$c = o$'. The equations $logHLT = 6.9693 - 1.19013 * c = o$ and $logHLT = 6.91524 - 1.24286 * c - c = o$ are returned as a result of this query.

When we are looking for equations involving binary variables, such as the presence/absence of a molecular fragment in a molecule, we should take into account that higher degrees of such variables are identical to the variables themselves. Thus the higher degrees of molecular fragment should not appear in equations. This can drastically reduce the space of possible equations.

At present, the above is not taken into account explicitly. However, the search heuristic used in CIPER, which takes into account equation complexity (punishes complex equations) prevents equations with higher degrees of fragment features from being chosen. This because such equations have the same accuracy/degree of fit as the equation with a linear degree of the fragment and higher complexity.

Other constraints might arise from the properties of molecular fragments and generality/specificity relationships among them, which might be used to reduce/heuristically prune the space of possible QSAR equations. For example, when refining a fragment, i.e., making it more specific, the value of the corresponding feature can only go from one to zero for any molecule considered. This can be used to bind the possible error reduction and also the maximum heuristic value an equation can achieve.

Originally, our constraint-based approach to equation discovery was meant for regression problems (with continuous class variable). However, we can easily adapt it to classification problems where the features are numeric (including binary features). We can use CIPER to perform classification via regression (multi-response polynomial regression), an approach shown to perform well in many cases.

7.3 Experiments on the Biodegradability Dataset

The QSAR application that we consider here is the one of predicting biodegradability of compounds [6]. The database used was derived from the data in the handbook of degradation rates [11] and contains the chemical structure and measured (or estimated) degradation rates for 328 chemicals. In our study we focus on aqueous biodegradation half life time (HLT) in aerobic conditions. The target variable (denoted logHLT) is the natural logarithm of the arithmetic mean of the low and high estimate of the HLT for aqueous biodegradation in aerobic conditions, measured in hours. Two discretized versions (2-class and 4-class) of logHLT were also considered.

A global feature of each chemical is its molecular weight. This was included in the data. Another global feature is logP, the logarithm of the compound's octanol/water partition coefficient, used also in the mutagenicity application. This feature is a measure of hydrophobicity, and can be expected to be important since we are considering biodegradation in water. The two global features were used in addition to the discovered molecular fragments in the QSAR models developed.

Kramer and De Raedt [14] used the two-class version of the biodegradability dataset to mine for interesting features. They looked for features that are above a prespecified minimum frequency threshold and below a maximum threshold. No generality constraints were used. The resulting features (fragments) together with MolWeight and log P were then used for predictive modeling (classification with a number of data mining approaches, of which support vector machines performed best).

For our experiments, the features generated with thresholds of minimum 0.05 and maximum 0.95 were kindly provided by Stefan Kramer. A total of 124 fragments were found by MolFea. Of these, only 69 are distinct, in the sense that there is at least one compound where one fragment is present and the other is not. The major reason for getting duplicates is that MolFea reported all fragments that satisfied the frequency thresholds and not the minimal (maximal) ones: duplicates can be avoided by taking the borders of the versionspace produced by MolFea. Only 69 distinct fragments were taken as input for equation discovery.

Table 8. The equation generated by CIPER on the biodegradability dataset

```
logHLT = 0.0645738*logP - 0.825858*c=o + 0.374477*logP*cl + 0.487245*c-n
         + 2.43385*c~c~c~c~c~c~c~c~c~c~c~c~c~c - 0.529246*c~c~c~c~c
         + 0.757922*n=o + 0.570323*c-c-c~c~c~c~c~c
         - 0.632581*c-c-c-o + 0.817581*c-o-c - 0.621152*c-o
         + 0.00231708*MolWeight + 5.94176
```

Taking the 69 fragments, logP and MolWeight as the independent and logHLT (or the discretization thereof) as the dependent variable, we applied our system for equation discovery CIPER to the biodeg dataset. For the continuous version, the equation in Table 8 was generated by CIPER (with beam 4). For the two-class and four-class version, classification via regression was performed: as many equations were produced as the number of classes.

Table 9. The predictive performance of CIPER on the biodegradability dataset, estimated by 10-fold cross validation, as compared to several other learning algorithms

Data version / Algorithm	J48(Class via)	LinReg(Class via)	M5(Class via)	CIPER
2 class (acc.)	71.34	74.09	78.05	78.66
4 class (acc.)	56.40	52.24	54.57	56.40
continuous (RE)		0.627	0.560	0.576

The predictive performance of CIPER, estimated by 10-fold cross validation, was also measured (Table 9). It was compared to the performance of three other algorithms: J48, Linear Regression and M5. For the latter two, comparison was done for both the continuous (regression) and the discrete (classification via regression) versions of the dataset.

CIPER performs much better than J48 for the two-class version and the same for the four-class version. It performs better than linear regression (higher accuracy, lower error / RE). It performs better than M5 for classification via regression and slightly worse for regression. Overall, CIPER performs best. In addition, the equation produced can be easily interpreted (e.g., three fused aromatic rings greatly contribute to longer degradation time).

7.4 More Complex Scenarios for Combining Pattern Domains

In the previous subsection, we combined two pattern domains in a pipeline fashion: we took the frequent molecular fragments produced by MolFea as features and fed them into CIPER to find equations. Here we propose two iterative scenarios for combining the two pattern domains, where MolFea and CIPER are ran in a loop and outputs from one are fed as input into the other.

Table 10. An iterative scenario for QSAR with level-wise feature generation

- F_0 = Global features
- M' = Predictive model induced (by CIPER) on the features from F_0
- $L = 1$
- Repeat
  - $M = M'$
  - $F_L = F_{L-1} \cup$ frequent fragments of length L
  - M' = Predictive model induced (by CIPER) on the features from F_L
- Until M' is less accurate than M

In the first scenario, given in Table 9, frequent fragments of increasing length are sought by MolFea and incrementally made available to CIPER to construct equations: the process is repeated as long as the accuracy of the equations improves. The second scenario is similar, with the added constraint that the RHS of an equation found by CIPER at a given step has to be a super-polynomial of the RHS of the equation constructed at the previous step of the scenario. This is a theory revision scenario, with each model being a revision of the model constructed at the previous step.

To evaluate the two scenarios above, we consider a dataset from the area of predictive toxicology [10]. The fish-toxicity dataset comes from the Distributed Structure-Searchable Toxicity (DSSTox) Public Database Network [17] or more specifically the EPAFHM toxicity database. The latter contains data about toxicity response of the fathead minnow (freshwater fish *Pimephales Promelas*) for 617 industrial organic chemicals. For each considered chemical, the database provides several different measures of the minnow's toxicity response. We focus here on the median lethal concentration (LC50) in [mg/l], which measures the concentration of the chemical producing lethality in 50% of test animals after 96 hours of exposure. If more than one bioassay has been conducted for the same

Table 11. A theory revision scenario for QSAR

- F_0 = Global features
- M' = Equation induced by CIPER on the features from F_0
- $L = 1$
- Repeat
  - $M = M'$
  - $F_L = F_{L-1} \cup$ frequent fragments of length L
  - M' = Equation induced by CIPER on the features from F_L, where the RHS of M is a sub-polynomial of the RHS of M'
- Until M' is less accurate than M

chemical, geometric mean of LC50s is presented. If no mortality, or less than 50% mortality has been observed at 96 hours, LC50 value is not available. The actual target variable we want to predict is the negative value of the natural logarithm of LC50. The data set used in our study contains 574 chemical compounds, for which both the chemical structure and LC50 value of the activity are available.

Using the pipeline approach to combining the two pattern domains, we obtain 375 frequent molecular fragments and a correlation coefficient of 0.648. If we use the iterative scenario for level-wise feature generation, fragments up to length $L = 10$ are generated and a correlation coefficient of 0.6530 is achieved. Finally, applying the theory revision scenario generates fragments of length up to $L = 11$ and yields a correlation coefficient of 0.711.

8 Conclusion

Summary. Here we have considered the problem of predictive modeling via polynomial equations within the framework of inductive databases (IDBs). We have defined primitives for the pattern domain of polynomial equations, including language constraints (such as sub-polynomial), evaluation primitives, evaluation and optimization constraints. We have also shown how to combine such primitives to form inductive queries in this pattern domain.

We have then presented a complete and a heuristic solver for data mining queries in this pattern domain, focussing on the latter. The algorithm CIPER performs heuristic beam search through the space of polynomial equation that satisfy a set of given language constraints and reports the beam of equations that best satisfy the optimization constraints. It uses a refinement operator derived from the subsumption (sub-polynomial) relationship between polynomials and a heuristic function that takes into account both the accuracy and complexity of equations.

We have illustrated the use of the developed approach in two different areas. We have first applied CIPER to standard regression datasets, where it performs competitively to existing regression methods while producing more concise models. We have then shown that constraint-based discovery of polynomial equations is suitable for modeling the dynamics of chemical reaction networks, since the

language constraints in our pattern domain naturally correspond to complexity limits on reaction networks.

Finally, we have shown how the pattern domains of equations and molecular fragments can be combined into an IDB for QSAR. We first considered a simple scenario where the output of the discovery of molecular fragments is the input for equation discovery and illustrated its use on the problem of predicting biodegradability. We then considered two more complex (iterative) scenarios which interleave inductive queries from the two pattern domains and illustrated their use on a predictive toxicology problem.

Discussion. The present chapter emphasizes the use of equations as predictive models in data mining. Regression equations are commonly used for predictive modeling in statistics, but receive considerably less attention in data mining. While equation discovery [15] is a recognized research topic within machine learning, it has been previously mainly used to rediscover quantitative scientific laws, with an emphasis on the comprehensibility and general validity of the laws found, rather than their predictive power. Here we show that equation discovery can build models that have similar predictive power and lower complexity as compared to state of the art regression approaches.

In the framework of inductive databases (IDBs), different types of patterns have been considered within different so-called pattern domains [4]. Most considered pattern domains concern the discovery of frequent patterns, such as frequent itemsets, episodes, Datalog queries and sequences. One of the main contributions of this chapter is that global predictive models (equations) have been considered in the framework of IDBs, following the same general structure of a pattern domain as for the discovery of frequent patterns. Predictive modeling lends itself to being described in terms of the same types of primitives: language constraints, evaluation primitives, evaluation and optimization constraints, as shown by defining such primitives for the pattern domain of equations.

Considering predictive models in the same framework as frequent patterns brings out the contrast between the practice of mining patterns and models. Techniques for mining frequent patterns typically aim at finding all frequent patterns that satisfy a user provided set of constraints (such as minimal frequency). On the other hand, predictive modeling techniques heuristically search for a single model trying to maximize predictive accuracy (usually not taking into account any other constraints).

In the frequent pattern setting, frequency constraints enable pruning of the space of candidate patterns and using constraints can mean drastic improvements in terms of efficiency. The monotonicity/anti-monotonicity of such constraints, used in conjunction with a generality relation on the patterns, is crucial in this respect. In mining for models, one typically uses constraints (like accuracy) that are neither monotonic nor anti-monotonic. In such cases, the search space over the possible models can still be structured according to the constraints, but effective pruning is no longer possible. As a consequence, heuristic (rather than complete) solvers have to be designed, as was the case in our pattern domain of polynomial equations.

The use of constraints in predictive data mining is less likely to be driven primarily by efficiency considerations. This is because constraints there are less likely to be monotone/anti-monotone, i.e., to enable efficient pruning. In this context, it is important that the constraints are intuitively understandable for the domain experts and have natural interpretation in the domain of use. This has been illustrated by the use of constraints in modeling the dynamics of networks of chemical reactions.

So far, in the framework of IDBs, different types of patterns have been considered separately, within different pattern domains. We believe that both global models (like equations) and local patterns (like molecular fragments) will need to be considered in IDBs. Moreover, different types of patterns will need to be used in the same IDB for practical applications, such as QSAR (and other applications in bioinformatics) [5]. To our knowledge, global models in the form of equations have so far not been considered in IDBs (even though they are routinely used in practical applications, such as QSAR) and neither have combinations of local patterns and such global models.

The focus on individual pattern domains has had as a consequence the focus on inductive queries for individual data mining steps. However, IDBs in general are a promising framework for supporting the entire process of knowledge discovery. The more complex scenarios for combining molecular fragments and equations illustrate that IDBs can (at least in principle) support more complex operations of knowledge discovery that involve several data mining operations and where (in accordance to the closure principle) the results of one inductive query are input for another.

Further work. Concerning work on IDBs with predictive models, we have just began to scratch the surface. Even for the pattern domain of equations there are several directions for further work. One direction concerns the definition, implementation and use of new constraints, such as bounds on equation length or similarity constraints. The latter allow the formulation of queries such as: "find the equation most similar to e, which has mean absolute error smaller than x". Another direction concerns the extension of the pattern domain of equations towards general equations (non-polynomial ones). Here the connection to grammar-based equation discovery [18] might prove useful. Finally, extensions towards other types of regression models (e.g. model trees) as well as classification models would be in order. The latter would be especially relevant for predictive regression (modeling) applications, such as the ones considered in Section 4.

In the domain of modeling reaction networks, an immediate direction for further work concerns the further formulation and exploitation of constraints. For example, instead of a partially specified network with missing reactions, one might consider a given maximal network of reactions and look for a subnetwork. In this case, superpolynomial constraints might be used to focus/constrain the search for equations. Experiments on real data are needed to thoroughly evaluate the usefulness of our approach for modeling metabolic reaction networks.

The kinetics of biochemical metabolic pathways is an important potential application area of equation discovery and IDBs. The need for quantitative models of biological processes is growing rapidly, and we expect it to play a significant role in establishing the kind of mathematical understanding sought from enterprises like the Human Physiome Project [1]. We believe that equation discovery and IDBs of the form proposed here will greatly assist the analysis of the large quantities of data expected to be available as a result of the project.

Much work remains to be done to achieve IDBs where different pattern domains coexist and can be queried inductively. We have only considered frequent molecular fragments and predictive equation models, while it is obvious that many other forms of frequent patterns and predictive models deserve attention. Of special importance is the integration of such pattern domains and databases and the design of inductive query languages that would enable IDBs to support complex, non-trivial processes of knowledge discovery.

Acknowledgments

We acknowledge the support of the cInQ (Consortium on discovering knowledge with Inductive Queries) project, funded by the European Commission under contract IST-2000-26469. We would like to thank the cInQ partners for the cooperation during the cInQ project. Special thanks are due to Luc De Raedt and Jean-Francois Boulicaut for getting us involved in this research and their continued support thereof. Thanks to Stefan Kramer for interesting discussions on the topic of this paper and for providing the MolFea generated features for the biodegradability dataset. Thanks to Christoph Helma for providing the MolFea generated features for the fish toxicity dataset.

References

1. J. B. Bassingthwaighte, editor. *Web Page of the Physiome Project*, 2002 (Web page update). http://www.physiome.org/
2. R. Bayardo. Constraints in data mining. *SIGKDD Explorations*, 4(1), 2002.
3. C. L. Blake and C. J. Merz. UCI repository of machine learning databases, 1998. http://www.ics.uci.edu/ ∼ mlearn/MLRepository.html.
4. L. De Raedt. Data mining as constraint logic programming. In *Computational Logic: From Logic Programming into the Future (In honor of Bob Kowalski)*. Springer, Berlin, 2002.
5. L. De Raedt and S. Kramer. Inductive databases for bio and chemoinformatics. In P. Frasconi, R. Shamir (editors), *Artificial Intelligence and Heuristic Methods for Bioinformatics*. IOS Press, Amsterdam, 2003.
6. S. Džeroski, H. Blockeel, B. Kompare, S. Kramer, B. Pfahringer, and W. Van Laer. Experiments in predicting biodegradability. In *Proc. Ninth International Conference on Inductive Logic Programming*, pages 80–91. Springer, Berlin, 1999.
7. S. Džeroski, L. Todorovski, and P. Ljubič. Using constraints in discovering dynamics. In *Proc. Sixth International Conference on Discovery Science*, pages 297-305. Springer, Berlin, 2003.

8. S. Džeroski and L. Todorovski. Discovering dynamics: from inductive logic programming to machine discovery. *Journal of Intelligent Information Systems*, 4:89–108, 1995.
9. T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, Berlin, 2001.
10. C. Helma, editor. *Predictive Toxicology*. CRC Press, Boca Raton, FL, 2005.
11. Howard, P.H., Boethling, R.S., Jarvis, W.F., Meylan, W.M., and Michalenko, E.M. 1991. *Handbook of Environmental Degradation Rates*. Lewis Publishers.
12. T. Imielinski and H. Mannila. A database perspective on knowledge discovery. *Communications of the ACM*, 39(11):58–64, 1996.
13. J.R. Koza, W. Mydlowec, G. Lanza, J. Yu, and M.A. Keane. Reverse engineering of metabolic pathways from observed data using genetic programming. In *Proc. Sixth Pacific Symposium on Biocomputing*, pages 434-445. World Scientific, Singapore, 2001.
14. S. Kramer and L. De Raedt. Feature construction with version spaces for biochemical applications. In *Proc. Eighteenth International Conference on Machine Learning*, pages 258–265. Morgan Kaufmann, San Francisco, 2001.
15. P. Langley, H. A. Simon, G. L. Bradshaw, and J. M. Żytkow. *Scientific Discovery*. MIT Press, Cambridge, MA, 1987.
16. H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1(3):241–258, 1997.
17. A. Richard, editor. *Distributed Structure-Searchable Toxicity (DSSTox) Public Database Network*, 2004 (Web page update). `http://www.epa.gov/nheerl/dsstox/`.
18. L. Todorovski and S. Džeroski. Declarative bias in equation discovery. In *Proc. Fourteenth International Conference on Machine Learning*, pages 376–384. Morgan Kaufmann, San Francisco, CA, 1997.
19. L. Todorovski, and S. Džeroski. Theory revision in equation discovery. In *Proc. Fourth International Conference on Discovery Science*, pages 390-400. Springer, Berlin, 2001.
20. L. Todorovski, S. Džeroski, and P. Ljubic. Discovery of polynomial equations for regression. In *Proc. Sixth International Multi-Conference Information Society*, Volume A, pages 151-154. Jožef Stefan Institute, Ljubljana, 2003.
21. L. Todorovski, P. Ljubič, and S. Džeroski. Inducing polynomial equations for regression. In *Proc. Fifteenth International Conference on Machine Learning*, pages 441-452. Springer, Berlin, 2004.
22. L. Torgo. Regression data sets, 2001. http://www.liacc.up.pt/ ∼ ltorgo/Regression/DataSets.html.
23. E. O. Voit. *Computational Analysis of Biochemical Systems*. Cambridge University Press, Cambridge, UK, 2000.
24. Y. Wang and I. H. Witten. Induction of model trees for predicting continuous classes. In *The Proceedings of the Poster Papers of the Eighth European Conference on Machine Learning*, pages 128–137, University of Economics, Faculty of Informatics and Statistics, Prague, 1997.
25. I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, San Mateo, CA, 1999.

Mining Constrained Graphs: The Case of Workflow Systems

Gianluigi Greco[1], Antonella Guzzo[2], Giuseppe Manco[2], Luigi Pontieri[2], and Domenico Saccà[2]

[1] Department of Mathematics, University of Calabria, Italy
[2] ICAR, CNR, Italy
ggreco@mat.unical.it, {guzzo, manco, pontieri, sacca}@icar.cnr.it

Abstract. Constrained graphs are directed graphs describing the control flow of processes models. In such graphs, nodes represent activities involved in the process, and edges the precedence relationship among such activities. Typically, nodes and edges can specify some constraints, which control the interaction among the activities. Faced with the above features constrained graphs are widely used in the modelling and analysis of Workflow processes. In this paper we overview two mining problems related to the analysis of constrained graphs, namely the analysis of frequent patterns of execution, and the induction of a constrained graph from a set of execution traces. We discuss some complexity aspects related to the problem of reasoning and mining on constrained graphs, and overview two algorithms for the mentioned problems.

1 Introduction

Graph-based models have been widely used in several contexts as an intuitive and yet formal way of representing several kinds of data, like, e.g., web documents, chemical compounds, process models, behavioral patterns. Graph structures can be exploited both for representing a given application domain, and for modelling relationships between the involved objects, by means of constraints over the underlying graph structure. In this perspective, constrained graphs are a powerful means for representing many classes of applications requiring complex modelling structures, and can profitably support reasoning and mining tasks.

As an example, constrained graphs are used in the modelling of workflow processes. In Workflow Management Systems, the structure of a process is commonly represented by a *control graph*, where nodes correspond to the involved tasks while edges represent the potential flow of work, i.e., the precedence relationships defined among the tasks. An example constrained (control flow) graph is shown in Figure 1, for modelling a toy (*OrderManagement*) process for handling customers' orders. Several constraints can be specified over the control graph, expressing, e.g., conditions for the occurrence of some nodes in the graph in any execution. For example, some constraints in figure are the following: (i) exactly one of the outgoing edges of node b must appear in any (execution) instance including b, and (ii) if node l occurs in an instance, both its incoming edges must appear as well in the same instance.

J.-F. Boulicaut et al. (Eds.): Constraint-Based Mining, LNAI 3848, pp. 155–171, 2005.

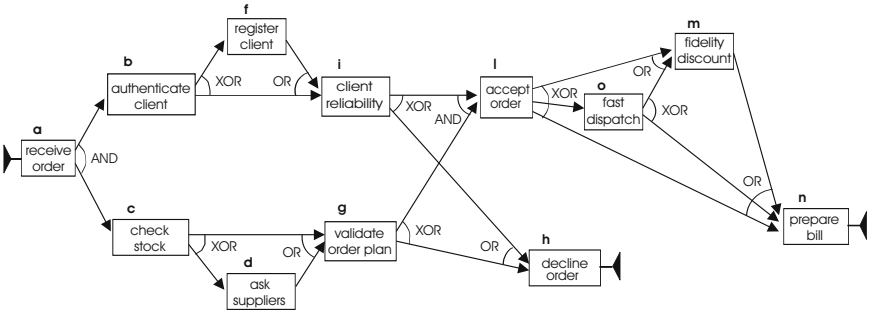
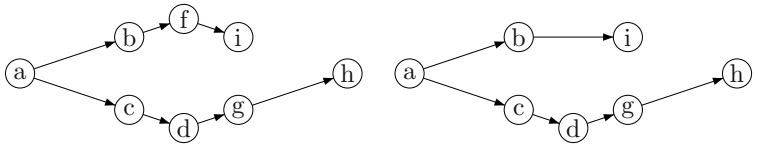


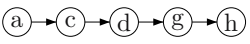
Fig. 1. Control flow graph for *OrderManagement* process

Thus, constraint graphs model the behavior of generic processes, whose executions can be traced and stored into database structures. In this context, a challenging research direction is the definition of both suitable pattern domains and mining techniques for the underlying data. Formally, the problem of mining constrained graphs can be formulated as follows. We are given a graph schema $\mathcal{WS}$ (i.e., a graph in which both nodes and edges must satisfy some specified constraints). An instance of a graph schema is any subgraph of $\mathcal{WS}$ which satisfies the constraints. An example is the subgraph containing nodes `a`, `c`, `b`, `i`, `g` and `h` in Figure 1. The subgraph describes the processing of an order which is declined due to a failure in the validation of the order plan. Hence, for a given pattern language $\mathcal{L}$, a set of instances $\mathcal{F}$ and a boolean inductive query Q, we aim at finding the inductive theory $\mathcal{T}h(\mathcal{F}, \mathcal{L}, Q) = \{p \in \mathcal{L} | Q(\mathcal{F}, p)\}$.

In this paper we describe two different pattern domains. A first case, formerly studied in [13], raises when patterns are subgraphs of the instances. Here, inductive queries can be used to formulate frequent pattern discovery problems. In particular, one can be interested in finding the discriminant factors which characterize a desired workflow configuration: essentially, this means finding frequent patterns containing some given portions of the workflow schema (i.e., finding all the patterns p satisfying $Q_f(\mathcal{F}, p) \wedge Q_1(\mathcal{F}, p) \wedge \ldots \wedge Q_n(\mathcal{F}, p)$, where $Q_f(\mathcal{F}, p)$ is true if p is frequent w.r.t. $\mathcal{F}$, and $Q_i(\mathcal{F}, p)$ is true if p contains a given subgraph g_i). For example, one could be interested in knowing which activities, among the ones described in Figure 1, are included within the frequent paths of execution which also include node `h` (representing the rejection of an order). Consider, e.g., the following toy instances:



Notice that both instances are subgraphs of the schema shown in Figure 1, and satisfy the constraints specified there. In addition, both the instances comply with the requirement of containing node `h`. Now, the subgraph



frequently occurs in both instances, and is characterized by the co-occurrence of activities `c`,`d`, `g`. Essentially, this means that in the modelled *OrderManagement* process, the rejection of an order is often characterized by the lack of availability of supplies. In a business intelligence perspective, this would require a better strategy for managing the store.

The challenge in the exemplified pattern domain is the generation of the desired patterns by a smart exploration of the search space, which benefits from the presence of schema constraints.

Another interesting situation occurs when the schema $\mathcal{WS}$ is not known a priori, although some instances are available and can be examined. In such a case, inductive queries can be used to formulate and solve the mining problem of inducing the schema. An example in this setting is the *Process mining problem* [12], where data collected during the enactment of a process is exploited to reconstruct the structure of the process. In more detail, we are given a set $\mathcal{F}$ of instances, and a language of patterns $\mathcal{L}$, modelling graph schemata. A boolean inductive query $Q_P(\mathcal{F}, p)$ is satisfied whenever p is a process model for $\mathcal{F}$, i.e., whenever each instance in $\mathcal{F}$ is also an instance of the graph schema represented by p. Again, many variants of the Q_P problem can be defined, by requiring, e.g., that p contains a given subgraph, or that satisfies a given constraint.

The main challenge here is devising efficient techniques to produce accurate and yet intuitive process models. As a matter of fact, since many models, in principle, could support a given set $\mathcal{F}$ of instances, a criterion could be devised to single the ones with satisfactory modelling features. For example, it is reasonable to require that, besides representing all the instances in $\mathcal{F}$, the discovered model has a limited description size and admits a minimal number of "spurious" execution paths, which do not have any correspondence in $\mathcal{F}$.

Objectives. In this paper, we elaborate the above described issues, by defining a formal model for mining constrained graphs, and by illustrating some efficient techniques for extracting patterns from graph-based data. In more detail, the contribution of the paper can be summarized as follows. In section 2, we introduce a formal framework for representing graph-based data, and classes of constraints over such data. We discuss some complexity results in reasoning on constrained graphs. Next, we state two mining problems, namely the mining of constrained subgraphs in section 3, and the induction of graph-based models in section 4. We discuss some approaches to the solution of the proposed mining problems, and show that they can be effectively exploited to support reasoning on constrained graphs. Throughout the paper, we exploit workflow management as a relevant application context for constrained graphs, and show that the proposed solutions suitably apply to the problem of workflow modelling.

Related Work. Mining workflow pattern is emerging as a novel field of research, promising interesting research issues and raising challenges in workflow applications. Since it is a pioneering study, techniques for workflow mining can be preliminary compared with several approaches proposed to mine patterns for structured or sequential data [2,14,3,25,24]. Moreover, they have also a strong relation with mining of graph structured data, occurring in several practical domains such as biology, chemistry and communication networking.

Many papers on graph mining have been proposed in the last years. A first category of studies applies greedy search to find subgraph patterns [4,33]. These approaches avoid the high complexity of the graph isomorphism problem, by mining an incomplete set of characteristic subgraphs. Conversely, a complete search for frequent subgraphs is guaranteed in WARMR, an ILP-based algorithm proposed by Dehaspe and Toivonen [9]. They formulated the problem of carcinogenesis prediction of chemical compounds with a set of grounded first order predicates representing graphs and they resolved this problem by combining ILP method with Apriori-like level wise search. Other approaches performing either level-wise search or projection methods to mine a complete set of subgraphs were proposed as well [17,19,31,32].

In principle, many of the above approaches could be used to mine constrained graphs. However, the adaptation of the above mentioned methods to workflow mining is a challenging task, and it results unpractical from both the expressiveness and the efficiency viewpoint. Indeed, generation of patterns with such traditional approaches does not benefit from the exploitation of the executions' constraints imposed by the workflow schema, such as precedences among activities, synchronization and parallel executions of activities (see, e.g, [18,29]).

In this setting, more sophisticated techniques have been successfully applied, in order to derive formal graph models from graph instances. The first example in the context of Workflow mining is in [1], where the main objective is the induction of a *directed graph model* exhibiting a limited number of control flow constructs.

Other more sophisticated approaches have been devised, relying, e.g., on the notion of grammar inference [23,5,6,7], or Petri Nets [29,27,28,30,8]. Starting from workflow logs, i.e., collections of linearized graph instances, the mentioned approaches propose algorithms for inducing complex control flow constraints. Further approaches have been devised in [15,16] and [26], where richer representation languages are adopted to discover more complex graph structures. In particular, the former approaches are devoted to the detection of redundancies in the workflow model, while the latter discover hierarchically structured workflow processes.

2 An Abstract Model for Constrained Graphs

A significant amount of research has been done in the specification of mechanisms for modelling processes; in particular, several formalisms have been proposed in the area of process modelling for software engineering (see, e.g. [10] for

an overview of different proposals). The most widely adopted formalism is the *control flow graph*, in which a process is represented by a labelled directed graph whose nodes correspond to the tasks to be performed, and whose arcs describe the precedences among them. More specifically, the control flow graph of a process P is a tuple $\mathcal{CF}(P) = \langle A, E, a_0, F \rangle$, where A is a finite set of *activities*, $E \subseteq (A - F) \times (A - \{a_0\})$ is a relation of precedences among activities, $a_0 \in A$ is the starting activity, $F \subseteq A$ is the set of final activities.

Any connected subgraph $I = \langle A_I, E_I \rangle$ of the control flow graph, such that $a_0 \in A_I$ and $A_I \cap F \neq \emptyset$ is a *potential instance* of P. In order to model restrictions on the possible instances, the description of a constrained graph is often enriched with *local* and *global* constraints, which express further relationships among the activities appearing in the control graph.

In particular, *local constraints* specify local properties of a given activity, with respect to its adjacents. For instance, possible local constraints are that an activity either can be executed only after all its predecessors are completed.

Most of the approaches proposed in the literature, even though with possibly different syntaxes, assume that the local constraints can be expressed in terms of three functions $\texttt{IN}, \texttt{OUT}_{min}$, and $\texttt{OUT}_{max}$ assigning to each node a natural number $(A \mapsto \mathbf{N})$ as follows:

- $\forall a \in A - \{a_0\}$, $0 < \texttt{IN}(a) \leq \mathit{InDegree}(a)$;
- $\forall a \in A - F$, $0 < \texttt{OUT}_{min}(a) \leq \texttt{OUT}_{max}(a) \leq \mathit{OutDegree}(a)$;
- $\texttt{IN}(a_0) = 0$, and $\forall a \in F$, $\texttt{OUT}_{min}(a) = \texttt{OUT}_{max}(a) = 0$.

where $\mathit{InDegree}(a) = |\{e = (b, a)\}|$,$\mathit{OutDegree}(a) = |\{e = (a, b)\}|$ and $e \in E$.

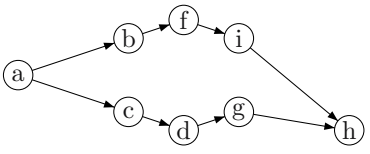
As for the semantics, an activity a can start as soon as at least $\texttt{IN}(a)$ of its predecessor activities have been completed. Two typical cases are: (i) if $\texttt{IN}(a) = \mathit{InDegree}(a)$ then a is an *and-join* activity, for it can be executed only after all of its predecessors are completed, and (ii) if $\texttt{IN}(a) = 1$ then a is an *or-join* activity, for it can be executed as soon as one of its predecessors is completed. Once finished, an activity a activates one non-empty subset of its outgoing arcs with cardinality between $\texttt{OUT}_{min}(a)$ and $\texttt{OUT}_{max}(a)$. If $\texttt{OUT}_{max}(a) = \mathit{OutDegree}(a)$ then a is a *full fork* and if also $\texttt{OUT}_{min}(a) = \texttt{OUT}_{max}(a)$ then a is a *deterministic fork* (also known as "and-split"), for it activates all of its successor activities. Finally, if $\texttt{OUT}_{max}(a) = 1$ then a is an *exclusive fork* (also called *xor-split* in the literature), for it activates exactly one of its outgoing arcs. Figure 1 shows an example schema containing the above mentioned constraints.

Global constraints specify relationships among not necessarily connected activities. Such constraints are richer in nature and their representation strongly depends on the particular application domain of the modelled process. Thus, they are often expressed using complex formalisms. Here, we assume that global constraints are propositional formulas expressing relationships among the nodes in A and edges in E. As an example, the constraint $f \rightarrow \neg m$ states that whenever activity f occurs, activity m cannot occur. This constraint, referred to Figure 1, has the intuitive meaning that fidelity discounts cannot be applied to new clients.

For a generic process P, a *workflow schema* for P, denoted by $\mathcal{WS}(P)$, is a tuple $\langle \mathcal{CF}(P), \mathcal{C}_L(P), \mathcal{C}_G(P) \rangle$, where $\mathcal{CF}(P)$ is the control flow graph of P, and $\mathcal{C}_L(P)$ and $\mathcal{C}_G(P)$ are sets of local and global constraints, respectively.

Given a subgraph I of $\mathcal{CF}(P)$ and a constraint c in $\mathcal{C}_L(P) \cup \mathcal{C}_G(P)$, we write $I \models c$ whenever I satisfies c in the associated semantics. Moreover, if $I \models c$ for all c in $\mathcal{C}_L(P) \cup \mathcal{C}_G(P)$ and contains both the starting activity a_0 and a final activity in F, then I is called an *instance* of $\mathcal{WS}(P)$, denoted by $I \models \mathcal{WS}(P)$. When the process P is clear from the context, a workflow schema will be simply denoted by $\mathcal{WS} = \langle \mathcal{CF}, \mathcal{C}_L, \mathcal{C}_G \rangle$.

Example 1. The following is an example instance of the workflow process $\mathcal{WS}$ shown in Figure 1.



Notice how each node appearing within the instance satisfies the constraints specified by $\mathcal{WS}$. ◁

Checking whether a workflow schema admits a successful execution is intractable.

Proposition 1 ([13]). *Let $\mathcal{WS} = \langle \mathcal{CF}, \mathcal{C}_L, \mathcal{C}_G \rangle$ be a workflow schema. Then, deciding whether there exists an instance I is* **NP***-complete, but the problem becomes* **P***-complete if all nodes are full-forks.* □

The above proposition has a strong negative impact: we cannot statically induce relevant properties of a workflow schema. This justifies the adoption of data mining techniques, which in principle allow to dynamically induce the desired properties from the instances resulting from past executions. Precisely, we assume that each instance is properly stored by the workflow management system in the log file, which can be seen as a set $\mathcal{F} = \{I_1, ..., I_n\}$ such that $\mathcal{WS} \models I_i$, for each $1 \leq i \leq n$. In the following, we denote by $\mathcal{I}(\mathcal{WS})$ the set of all the instances of a given workflow $\mathcal{WS}$.

Deciding whether a subgraph is an instance of $\mathcal{WS}$ is tractable although deciding the existence of an instance (i.e., whether $\mathcal{I}(\mathcal{WS}) \neq \emptyset$) is not because of Proposition 1.

Proposition 2 ([13]). *Let $\mathcal{WS} = \langle \mathcal{CF}, \mathcal{C}_L, \mathcal{C}_G \rangle$ be a workflow schema and I be a subgraph of $\mathcal{CF}$. Then, deciding whether I is an instance of $\mathcal{WS}$ can be done in polynomial time in the size of E.* □

Usually, logs are stored by means of traces. A *workflow trace s over A* is a string in A^*, representing an instance . Given a trace s, we denote by $s[i]$ the *i-th* task in the corresponding sequence, and by $lenght(s)$ the length of s. The set of

all the tasks in s is denoted by $tasks(s) = \bigcup_{1 \leq i \leq lenght(s)} s[i]$. Hence, a *workflow log for* $\mathcal{WS}(P)$, denoted by $\mathcal{L}_P$, is a multiset of traces: $\mathcal{L}_P = [\ s \mid s \in A^*\ ]$.

Let s be a trace in $\mathcal{L}_P$, $\mathcal{WS}$ be a workflow schema, and $I = \langle A_I, E_I \rangle$ be an instance of $\mathcal{WS}$. Then, s is *compliant with* $\mathcal{WS}$ *through* I, denoted by $\mathcal{WS} \models^I s$, if s is a topological sort of I, i.e., s is an ordering of the activities in A_I s.t. for each $(a, b) \in E_I$, $i < j$ where $s[i] = a$ and $s[j] = b$. Moreover, s is *compliant with* $\mathcal{WS}$, denoted by $\mathcal{WS} \models s$, if there exists I with $\mathcal{WS} \models^I s$. Finally, a weaker notion of compliance, which does not rely on the existence of an instance I, can be defined as $\mathcal{WS} \vdash s$. The latter holds whenever the order of appearance of the activities in s is compatible with the constraints specified in $\mathcal{WS}$.

Example 2. The following table reports example log traces for the process $\mathcal{WS}$ shown in Figure 1.

s_1 : acdbfgih	s_5 : abicglmn	s_9 : abficgln	s_{13} : abcidglmn
s_2 : abficdgh	s_6 : acbiglon	s_{10} : acgbfilon	s_{14} : acdbiglmn
s_3 : acgbfih	s_7 : acbgilomn	s_{11} : abcfdigln	s_{15} : abcdgilmn
s_4 : abcgiln	s_8 : abcfgilon	s_{12} : acdbfigln	s_{16} : acbidgln

By considering the instance I of example 1, we can observe that $\mathcal{WS} \models^I s_1$. ◁

Proposition 3. *Let* $\mathcal{WS} = \langle \mathcal{CF}, \mathcal{C}_L, \mathcal{C}_G \rangle$ *be a workflow schema and* s *be a trace of* $\mathcal{CF}$*. Then, deciding whether* $\mathcal{WS} \models s$ *is* **NP***-complete, but deciding whether* $\mathcal{WS} \vdash s$ *and, given an instance* I*, whether* $\mathcal{WS} \models^I s$ *can be done in polynomial time in the size of* E*.*

Proof. We first show that deciding whether $\mathcal{WS} \models s$ is **NP**-complete. Membership in **NP** is trivial. For the hardness, recall that, given a Boolean formula Φ over variables $X_1, ..., X_m$ the problem of deciding whether Φ is satisfiable is **NP**-complete. W.l.o.g. assume Φ to be in conjunctive normal form. Then, we define a workflow schema $\mathcal{WS}(\Phi) = \langle \mathcal{CF}, \mathcal{C}_L, \mathcal{C}_G \rangle$, where $\mathcal{CF} = \langle A, E, a_o, \{Sat\} \rangle$, such that A consists of an initial activity a_0, of the activities X_i, TX_i, FX_i, B_i for each $0 < i \leq m$, of the activities C_j for each distinct clause j of Φ, and the activity B, and of a final state Sat. The set of local constraints $\mathcal{C}_L$ and dependencies in E is defined as follows. Let $\texttt{IN}(Sat) = n$ (where n is the number of clauses contained in Φ), and $\texttt{IN}(a) = 1$ for any other activity $a \neq a_0$. Moreover:

- For each X_i, (X_i, TX_i), (X_i, FX_i), (B_i, TX_i), (B_i, FX_i), (TX_i, B), and (FX_i, B) are in E, with constraints $\texttt{OUT}_{min}(X_i) = \texttt{OUT}_{max}(X_i) = 1$ and $\texttt{OUT}_{min}(B_i) = \texttt{OUT}_{max}(B_i) = 1$. Thus, each time either the activity X_i or B_i is executed, it is required to make a choice between its possible successors; note that in our encoding, TX_i means that X_i is **true**, while FX_i means that X_i is **false**. Finally the arcs (a_0, X_i) and (a_o, B_i) are in E, and constraints $\texttt{OUT}_{min}(a) = \texttt{OUT}_{max}(a) = m + m$ are added.

– For each C_j, we have that (C_j, Sat) is in E, with constraints $\texttt{OUT}_{min}(Sat) = \texttt{OUT}_{max}(Sat) = 1$. Moreover, we have $(TX_i, C_j) \in E$ in the case X_j appears in the clause C_j, while we have $(FX_i, C_j) \in E$ in the case X_i appears negated in the clause C_j. Finally, for each node $a \in \{TX_i, FX_i\}$, $\texttt{OUT}_{min}(a) = 1$ and $\texttt{OUT}_{max}(a) = OutDegree(a) - 1$.

Global constraints in $\mathcal{C}_G$ are defined as follows. For each pair of activities of the form X_i and B_i, there is a constraint stating that the arc (B_i, TX_i) (resp. (B_i, FX_i)) cannot occur in the same execution with the arc (X_i, TX_i) (resp. (X_i, FX_i)). Moreover, for each activity of the form X_i, there is a constraint stating that arcs of the form (TX_i, C_j) (resp. (FX_i, C_j)) cannot occur in the same execution with arcs (TX_i, B) (resp. (FX_i, B)); finally, for each activity X_i, there is a constraint stating that an arc of the form (B_i, TX_i) (resp. (B_i, FX_i)) implies the activation of the arc (TX_i, B) (resp. (FX_i, B)).

Consider now a trace $s(\Phi) = a_0 B_1, ..., B_m X_1 ... X_m B C_1 ... C_m Sat$. Then, it is easy to see that $\mathcal{WS} \models s$ if and only if Φ is satisfiable.

To conclude the proof observe that (*1*) in order to decide whether $\mathcal{WS} \vdash s$ it is sufficient to tests the topological relationships locally induced by s, and that (*2*) in order to decide whether $\mathcal{WS} \models^I s$ it is sufficient to simulate the enactment of I. Both the above tasks are feasible in polynomial time. □

3 Mining Frequent Patterns

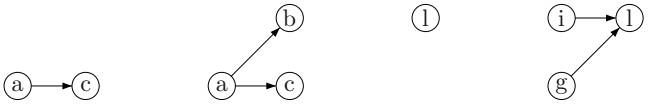
In this section we address the problem of mining connected frequent patterns (i.e., subgraphs) in workflow instances. Let us assume that a workflow schema $\mathcal{WS} = \langle \mathcal{CF}, \mathcal{C}_L, \mathcal{C}_G \rangle$ and a multiset of instances $\mathcal{F} = \{I_1, ..., I_n\}$ are given. A graph $p = \langle A_p, E_p \rangle \subseteq \mathcal{CF}$ is a *$\mathcal{F}$-pattern* (cf. $\mathcal{F} \models p$) if there exists $I = \langle A_I, E_I \rangle \in \mathcal{F}$ such that $A_p \subseteq A_I$ and p is the subgraph of I induced by the nodes in A_p. In the case $\mathcal{F} = \mathcal{I}(\mathcal{WS})$, the subgraph is simply said to be a *pattern*.

Let $supp(p) = |\{I \in \mathcal{F} | I \models p\}|/|\mathcal{F}|$, be the *support* of a $\mathcal{F}$-pattern p. Then, given a real number σ, we consider the following problem:

FCPD(σ)**:** *Frequent Connected Pattern Discovery*, i.e., finding all the connected patterns whose support is greater than σ.

A naive algorithm for mining frequent patterns can generate directly connected subgraphs, and then test in polynomial time whether it is indeed an instance of $\mathcal{WS}$. A different approach is based on the idea of reducing the number of patterns to generate. To achieve this aim, we can only consider connected subgraphs p which are "closed" w.r.t. local and global constraints, i.e., such that $p \models c$ for all $c \in \mathcal{C}_L \cup \mathcal{C}_G$. We shall denote such graphs *weak patterns*, or simply w-patterns.

Example 3. Let us consider the workflow graph of Figure 1, and the following subgraphs.



p_1 p_2 p_3 p_4

p_1 and p_3 are not w-pattern: indeed, a is a deterministic fork (thus triggering the occurrence of node b), whereas l is an and-join (thus triggering the occurrence of both i and g). Notice that both p_2 and p_4 are instead w-patterns, since each constraint involving the contained nodes is satisfied. ◁

The following proposition characterizes the complexity of recognition for the three notions of pattern; in particular, it states that testing whether a graph is a w-pattern can be done very efficiently in deterministic logarithmic space on the size of the graph $\mathcal{WS}$.

Proposition 4 ([13]). *Let $p \subseteq \mathcal{CF}$. Then*

1. *deciding whether p is a pattern is* **NP***-complete.*
2. *given a multiset $\mathcal{F}$ of instances, deciding whether p is an $\mathcal{F}$-pattern or whether p is a w-pattern is computable in polynomial time in the size of $\mathcal{F}$.* □

It turns out that the notion of weak pattern is the most appropriate from the computational point of view. Moreover, working with w-patterns rather than $\mathcal{F}$-patterns is not an actual limitation, since each frequent $\mathcal{F}$-pattern is bounded by w-patterns, as the following result states.

Lemma 1. Let p be a frequent $\mathcal{F}$-pattern. Then i) there exist a frequent w-pattern p' such $p \subseteq p'$, and ii) each weak pattern $p' \subseteq p$ is a frequent $\mathcal{F}$-pattern. □

We stress that a weak pattern is not necessarily an $\mathcal{F}$-pattern nor even a pattern. We shall use weak patterns to optimize the search space. The algorithm exploited uses a levelwise theory. Roughly speaking, we incrementally construct frequent weak patterns, by starting from frequent "elementary" weak patterns (defined below), and by extending each frequent weak pattern using two basic operations: adding a frequent arc and merging with another frequent elementary weak pattern. The correctness follows from the results of Proposition 1, and from the observation that the space of all connected weak patterns constitutes a lower semi-lattice, with a particular precedence relation $\prec$, defined next.

The elementary weak patterns, from which we start the construction of frequent patterns, are obtained as the deterministic closure of single nodes. A pattern is an elementary w-pattern (cf. *ew*-pattern) for a node a if it is the minimal (w.r.t. set inclusion) w-pattern containing a. The set of all *ew*-patterns is denoted by EW. Moreover, let p be a weak pattern, then EW_p denotes the set of the

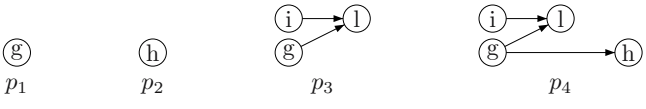
elementary weak patterns contained in p. Note that given an *ew*-pattern e, EW_e is not necessarily a singleton, for it may contain other *ew*-patterns. Moreover, given a set $E' \subseteq \mathrm{EW}$, $Compl(E') = \mathrm{EW} - \bigcup_{e \in E'} \mathrm{EW}_e$ contains all the elementary patterns which are neither in E' nor contained in some element of E'.

We now introduce a precedence relation $\prec$ among connected weak patterns. First of all, let us denote by $E^{\subseteq}$ the subset of arcs in $\mathcal{WS}$ whose source is not a deterministic fork, i.e., $E^{\subseteq} = \{(a,b) \in E \mid \mathtt{OUT}_{min}(a) < OutDegree(a)\}$. Given two connected w-patterns, say $p = \langle A_p, E_p \rangle$ and $p' = \langle A_{p'}, E_{p'} \rangle$, $p \prec p'$ if and only if:

a) $A_p = A_{p'}$ and $E_{p'} = E_p \cup \{(a,b)\}$, where $(a,b) \in E^{\subseteq} - E_p$ and $\mathtt{OUT}_{max}(a) > OutDegree_p(a)$ (i.e., p' can be obtained from p by adding an arc), or
b) there exists $p'' \in Compl(\mathrm{EW}_p)$ such that $p' = p \cup p'' \cup X$, where X is either empty if p and p'' are connected or contains exactly an edge in $E^{\subseteq}$ with endpoints in p and p'' (i.e., p' is obtained from p by adding an elementary weak pattern and possibly a connecting arc).

Note that $\perp \prec e$, for each $e \in \mathrm{EW}$.

Example 4. With reference to the workflow graph of Figure 1, let us consider the subgraphs shown below:



The subgraphs p_1, p_2 and p_3 are elementary patterns: indeed, p_1 is the deterministic closure of g and p_2 is the deterministic closure of h, whereas p_3 can be obtained from l. Also, notice that $p_1 \subset p_3$. p_4 is not an elementary pattern, as no node can generate it. Notice that $p_2 \prec p_4$ and $p_3 \prec p_4$, since $p_4 = p_2 \cup p_3 \cup \{(g,h)\}$. ◁

It can be shown that all the connected weak patterns of a given workflow schema can be constructed by means of a chain over the $\prec$ relation. As a consequence, it turns out that the space of all connected weak patterns is a lower semi-lattice w.r.t. the precedence relation $\prec$. The algorithm *w*-find, reported in Figure 2, exploits an apriori-like exploration of this lower semi-lattice.

At each stage, the computation of L_{k+1} (steps 5-14) is carried out by extending any pattern p generated at the previous stage ($p \in L_k$), in two ways: by adding frequent edges in $E^{\subseteq}$ (*addFrequentArc* function); or by adding an elementary weak patterns (*addEWFrequentPattern* function).

4 Mining Process Models

In this section we address the problem of inducing a model for a given process, based on data related to past executions. Let us assume that a workflow log

Input: A workflow Graph $\mathcal{WS}$, a set $\mathcal{F} = \{I_1, \ldots, I_N\}$ of instances of $\mathcal{WS}$.
Output: A set of frequent $\mathcal{F}$-patterns.
Method: Perform the following steps:

```
1   L_0 := {e | e ∈ EW, e is frequent w.r.t. F};
2   k := 0, R := L_0;
3   FrequentArcs := {(a,b) | (a,b) ∈ E^⊆, ⟨{a,b}, {(a,b)}⟩ is frequent w.r.t. F};
4   E_f^⊆ := E^⊆ ∩ FrequentArcs;
5   repeat
6     U := ∅;
7     forall p ∈ L_k do begin
8       U := U ∪ addFrequentArc(p);
9       forall e ∈ Compl(EW_p) ∩ L_0 do
10        U := U ∪ addFrequentEWPattern(p, e);
11    end
12    L_{k+1} := {p | p ∈ U, p is frequent w.r.t. F};
13    R := R ∪ L_{k+1};
14  until L_{k+1} = ∅;
15  return R;
```

Function *addFrequentEWPattern*$(p = \langle A_p, E_p\rangle, e = \langle A_e, E_e\rangle)$: **w-pattern**;
$p' := \langle A_p \cup A_e, E_p \cup E_e\rangle$;
if p' is *connected*, **then return** p' **else return** *addFrequentConnection*(p', p, e);

Function *addFrequentConnection*$(p' = \langle A_{p'}, E_{p'}\rangle, p = \langle A_p, E_p\rangle, e = \langle A_e, E_e\rangle)$: **w-pattern**;
$S := \emptyset$
forall frequent $(a, b) \in E_f^{\subseteq} - E_p$ s.t. $(a \in A_p,\ b \in A_e) \vee (a \in A_e,\ b \in A_p)$ **do begin**
  $q := \langle A_{p'}, E_{p'} \cup (a, b)\rangle$;
  if $\mathcal{WS} \models q$ **then** $S := S \cup \{q\}$;
end
return S

Function *addFrequentArc*$(p = \langle A_p, E_p\rangle)$: **pattern**;
$S := \emptyset$
forall frequent $(a, b) \in E_f^{\subseteq} - E_p$ s.t. $a \in A_p$, $b \in A_p$ **do begin**
  $p' := \langle A_p, E_p \cup (a, b)\rangle$
  if $\mathcal{WS} \models p'$ **then** $S := S \cup \{p'\}$;
end
return S

Fig. 2. Algorithm w-find($\mathcal{F}$,$\mathcal{WS}$)

$\mathcal{L}_P$ is given for a process P. In general, a process mining task consists in discovering a workflow schema $\mathcal{WS}(P)$, expressed through a suitable constrained graph, which describes the traces in $\mathcal{L}_P$. We are interested in devising a general approach which is independent of the particular syntax adopted for representing the global constraints. The solution we propose consists in discovering a set of alternative schemata having no global constraints, but collectively modelling the different behavioral patterns, instead of a single schema with global constraints explicitly expressed. To this purpose, we introduce the notion of *disjunctive workflow schema*.

A *disjunctive workflow schema* for a given process P, denoted by $\mathcal{WS}^{\vee}(P)$, is a set $\{\mathcal{WS}^1, ..., \mathcal{WS}^m\}$ of workflow schemata for P, with $\mathcal{WS}^j = \langle \mathcal{CF}^j, \mathcal{C}_L^j, \emptyset\rangle$, for $1 \leq j \leq m$. The *size* of $\mathcal{WS}^{\vee}(P)$, denoted by $|\mathcal{WS}^{\vee}(P)|$, is the number of workflow schemata it contains. An instance of any $\mathcal{WS}^j$ is also an instance of $\mathcal{WS}^{\vee}$, denoted by $\mathcal{WS}^{\vee} \models I$. Moreover, a trace s which is compliant with any $\mathcal{WS}^j$ is also compliant with $\mathcal{WS}^{\vee}$, denoted by $\mathcal{WS}^{\vee} \models s$.

Hence, given $\mathcal{L}_P$, we aim at discovering a disjunctive schema $\mathcal{WS}^{\vee}$ as "close" as possible to the actual unknown schema $\mathcal{WS}(P)$ that generated the log, according to the following *soundness* and *completeness* notions. We define *soundness of*

$\mathcal{WS}^\vee$ w.r.t. $\mathcal{L}_P$, the percentage of instances having corresponding traces in the log, i.e.,

$$soundness(\mathcal{WS}^\vee, \mathcal{L}_P) = \frac{|\{I \mid \mathcal{WS}^\vee \models I \wedge \exists s \in \mathcal{L}_P \text{ s.t. } \mathcal{WS}^\vee \models^I s\}|}{|\{I \mid \mathcal{WS}^\vee \models I\}|}$$

The *completeness of* $\mathcal{WS}^\vee$ w.r.t. $\mathcal{L}_P$, is instead the percentage of traces that are compliant with some trace in the log, ie.,

$$completeness(\mathcal{WS}^\vee, \mathcal{L}_P) = \frac{|\{s \mid s \in \mathcal{L}_P \wedge \mathcal{WS}^\vee \vdash s\}|}{|\{s \mid s \in \mathcal{L}_P\}|}$$

Thus, given two real numbers, namely α and σ, between 0 and 1, we say an induced schema $\mathcal{WS}^\vee$ is α-*sound* w.r.t. $\mathcal{L}_P$, if $soundness(\mathcal{WS}^\vee, \mathcal{L}_P) \leq \alpha$, whereas $\mathcal{WS}^\vee$ is σ-*complete* w.r.t. $\mathcal{L}_P$, if $completeness(\mathcal{WS}^\vee, \mathcal{L}_P) \geq \sigma$.

Notice that, for any value of α and σ, there always exists a trivial α-*sound* and σ-*complete* disjunctive schema $\mathcal{WS}^\vee$, consisting in the union of exactly one workflow (without global constraints) modelling each of the instances in $\mathcal{L}_P$. However, such model is not a syntectic view of the process P, for its size being $|\mathcal{WS}^\vee| = |\mathcal{L}_P|$. We thus introduce a bound on the size of $\mathcal{WS}^\vee$.

Then, given a workflow log $\mathcal{L}_P$ for the process P, a real number σ and a natural number m, we consider the following problem:

`MPD`(P, σ, m)**:** *Maximal Process Discovery*, i.e., find a σ-complete disjunctive workflow schema $\mathcal{WS}^\vee$, s.t. $|\mathcal{WS}^\vee| \leq m$ and $soundness(\mathcal{WS}^\vee, \mathcal{L}_P)$ is maximal.

Proposition 5 ([11]). `MPD`(P,σ,m) is an **NP**-complete optimization problem whose set of feasible solutions is not empty. □

Due to the above intractability result, the `MPD` problem is tackled with a greedy approach: in practice, we consider the variant `PD` problem, which consists in finding a σ-complete disjunctive schema with $|\mathcal{WS}^\vee| \leq m$, which is as sound as possible (i.e., a local optimum). In the rest of the section, we propose an efficient approach for solving the `PD` problem. The approach mainly relies on performing an iterative partitioning of the traces in the log, in order to find clusters of executions with a similar and unexpected (w.r.t. the local properties) behavior. Starting with a preliminary schema, which only accounts for the dependencies among the activities of P, the model is iteratively and incrementally refined by computing a specific workflow schema for each new cluster of traces. The schemata so obtained constitute a disjunctive workflow schema, which increases its soundness at each refinement step, still preserving its completeness. The algorithm exploits a "flat", relational representation of the traces obtained by projecting the instances on a suitable set of properly defined *features*.

The approach is encoded in the algorithm `ProcessDiscover`, shown in Figure 3, which computes a disjunctive schema $\mathcal{WS}^\vee$, taking as input a log $\mathcal{L}$ and three thresholds m, σ and $maxFeatures$ (which is an upper bound to the number of features that can be induced at each refinement step).

Input: A log $\mathcal{L}_P$, a real number σ, two natural numbers m and mF
Output: A disjunctive workflow schema $\mathcal{WS}^\vee$ (a solution of PD(P,σ,m)
Method: Perform the following steps:

1 $\mathcal{CF}_\sigma(\mathcal{WS}_0^1)$:=*minePrecedences*($\mathcal{L}_p$);
2 **let** $\mathcal{WS}_0^1$ be a schema, with $\mathcal{L}(\mathcal{WS}_0^1) = \mathcal{L}_P$;
3 *mineLocalConstraints*($\mathcal{WS}_0^1$);
3 $\mathcal{WS}^\vee := \mathcal{WS}_0^1$; *//Start clustering with the dependency graph only*
4 **while** $|\mathcal{WS}^\vee| < m$ **do**
5 $\mathcal{WS}_i^j$:=*leastSound*($\mathcal{WS}^\vee$);
6 $\mathcal{WS}^\vee := \mathcal{WS}^\vee - \{\mathcal{WS}_i^j\}$;
7 *refineWorkflow*(i,j);
8 **end while**
9 **return** $\mathcal{WS}^\vee$;

Procedure *refineWorkflow*(i: step, j: schema);

1 $\mathcal{F}$:=***identifyRelevantFeatures***($\mathcal{L}(\mathcal{WS}_i^j), \sigma, mF, \mathcal{CF}_\sigma$);
2 $\mathcal{R}(\mathcal{WS}_i^j)$:=***project***($\mathcal{L}(\mathcal{WS}_i^j), \mathcal{F}$);
3 $k := |\mathcal{F}|$;
4 **if** $k > 1$ **then**
5 $j := \max\{j \mid \mathcal{WS}_{i+1}^j \in \mathcal{WS}^\vee\}$;
6 $\langle \mathcal{WS}_{i+1}^{j+1}, ..., \mathcal{WS}_{i+1}^{j+k} \rangle := k\text{-}means(\mathcal{R}(\mathcal{WS}_i^j))$;
7 **for each** $\mathcal{WS}_{i+1}^h$ **do**
8 $\mathcal{WS}^\vee = \mathcal{WS}^\vee \cup \{\mathcal{WS}_{i+1}^h\}$;
9 $\mathcal{CF}_\sigma(\mathcal{WS}_{i+1}^h)$:=*minePrecedences*($\mathcal{L}(\mathcal{WS}_{i+1}^h)$);
10 *mineLocalConstraints*($\mathcal{WS}_{i+1}^h$);
11 **end for**
12 **else** *//Leaf of the tree*
13 $\mathcal{WS}^\vee = \mathcal{WS}^\vee \cup \{\mathcal{WS}_i^j\}$;
14 **end if**;

Function *identifyRelevantFeatures*(L: log, σ: threshold, mF: max nr. of features, $\mathcal{CF}_\sigma$: control flow graph): a set of minimal discriminant rules

1 $L_2 := \{[ab] \mid (a, b) \in E_\sigma\}$;
2 $k := 1$, $R := L_2$, $\mathcal{F} := \emptyset$;
3 **repeat**
4 $M := \emptyset$; $k := k + 1$;
5 **forall** $[a_i...a_j] \in L_k$ **do**
6 **forall** $[a_jb] \in L_2$ **do**
7 **if** $[a_{i+1}...a_j] \not\rightarrow_\sigma b$ is not in $\mathcal{F}$ **then**
8 $M := M \cup [a_i...a_jb]$;
9 **end for**
10 **forall** $p \in M$ of the form $[a_i...a_jb]$ **do**
11 **if** p is σ-frequent in $\mathcal{L}$ **then** $L_{k+1} := \{p\}$;
12 **else** $\mathcal{F} := \mathcal{F} \cup \{[a_i...a_j] \not\rightarrow_\sigma b\}$;
14 $\mathcal{F} := \{[a_i...a_j] \not\rightarrow_\sigma b\}$;
15 **end if**
13 **end for**
14 $R := R \cup L_{k+1}$;
15 **until** $L_{k+1} = \emptyset$;
16 **return** *mostDiscriminant*($\mathcal{F}$,mF);

Function *mostDiscriminantFeatures*($\mathcal{F}$: set of discriminant rules, mF: max nr. of features): set of discriminant rules;

1 $S' := \mathcal{L}$; $\mathcal{F}' := \emptyset$;
2 **do**
3 **let** $\phi = \texttt{argmax}_{\phi' \in \mathcal{F}} |w(\phi', S')|$;
4 $\mathcal{F}' := \mathcal{F}' \cup \{\phi\}$;
5 $S' := S' - w(\phi, S')$;
6 **while** ($|S'|/|\mathcal{L}_P| > \sigma$) and ($\mathcal{F}' < mF$);
7 **return** $\mathcal{F}'$;

Fig. 3. Algorithm `ProcessDiscover`

Notice that for the preliminary schema a control flow graph $\mathcal{CF}_\sigma$, expressing a minimal set of precedences with at least a given support σ, is computed through the procedure *minePrecedences* [1,28]. Each workflow schema $\mathcal{WS}_i^j$, eventually inserted in $\mathcal{WS}^\vee$, is identified by the number i of refinements needed, and an

index j distinguishing the schemata at the same refinement level. Moreover, we denote by $\mathcal{L}(\mathcal{WS}_i^j)$ the set of traces in the cluster defined by $\mathcal{WS}_i^j$. Notice that initially $\mathcal{WS}_0^1$, containing all the traces in $\mathcal{L}_P$, is put in $\mathcal{WS}^\vee$, and in Step 3 we refine the model by mining some local constraints, too.

At each step, function *refineWorkflow* is applied to a schema $\mathcal{WS}_i^j \in \mathcal{WS}^\vee$, chosen according a greedy heuristic: $\mathcal{WS}_i^j$ is the least sound schemata among the ones already discovered.

The function splits the traces of $\mathcal{WS}_i^j$ into k clusters, which are assigned to k distinct new schemata, $\mathcal{WS}_{i+1}^{j+1}, ..., \mathcal{WS}_{i+1}^{j+k}$ (where j is the maximum index of the schemata in $\mathcal{WS}^\vee$ with level $i+1$), which are put in $\mathcal{WS}^\vee$. For each schema a control flow graph and a set of local constraints are derived, which suitably model the associated traces.

The algorithm `ProcessDiscover` converges in at most m steps, and exhibits the following interesting property.

Lemma 2. *Given a disjunctive schema $\mathcal{WS}^\vee$, with $\mathcal{WS}_i^j \in \mathcal{WS}^\vee$, the disjunctive workflow schema $\mathcal{WS}_+^\vee$, obtained by refining $\mathcal{WS}_i^j$ through refineWorkflow(i,j), is such that soundness$(\mathcal{WS}_+^\vee) \geq$ soundness$(\mathcal{WS}^\vee)$.* □

The clustering of the log traces strongly relies on the procedures *identifyRelevantFeatures* and *project*. The former finds a set $\mathcal{F}$ of relevant features [21,20,22], whereas the latter projects the traces into a vectorial space whose components are, in fact, the mined features.

We formalize the key concept of *relevant feature* through the notion of *discriminant rule*. Let $\mathcal{L}$ be a set of traces, $\mathcal{CF}_\sigma$ be a mined control flow, for threshold σ, and E_σ be the edge set of $\mathcal{CF}_\sigma$. Then a sequence $[a_1...a_h]$ of tasks is σ-frequent in $\mathcal{L}$ if $|\{s \in \mathcal{L} \mid a_1 = s[i_1], ..., a_h = s[i_h] \wedge i_1 < ... < i_h\}|/|\mathcal{L}| \geq \sigma$. We say that $[a_1...a_h]$ *σ-precedes* a in $\mathcal{L}$, denoted by $[a_1...a_h] \rightarrow_\sigma a$, if both $[a_1...a_h]$ and $[a_1...a_h a]$ are σ-frequent in $\mathcal{L}$.

A *discriminant rule* (*feature*) ϕ is an expression of the form $[a_1...a_h] \not\rightarrow_\sigma a$, s.t. (i) $[a_1...a_h]$ is σ-frequent in $\mathcal{L}$, (ii) $(a_h, a) \in E_\sigma$, and (iii) $[a_1...a_h] \rightarrow_\sigma a$ does not hold. Moreover, ϕ is *minimal* if (iv) there is no b, s.t. $[a_1...a_h] \not\rightarrow_\sigma b$ and $[b] \rightarrow_\sigma a$, and (v) there is no j, s.t. $j > 1$ and $[a_j...a_h] \not\rightarrow_\sigma a$.

Example 5. In process *OrderManagament*, $[fil] \not\rightarrow_{.3} m$ is a minimal discriminant rule, prescribing that fidelity discounts are never applied for new clients. Notice that $[dgl] \not\rightarrow_{.3} o$ is a minimal discriminant rule as well. ◁

Again, the identification of the set $\mathcal{F}$ of discriminant rules can be carried out by a level-wise algorithm, as described in Figure 3.

The algorithm selects an optimal subset of features, with cardinality less or equal to *maxFeatures*, by exploiting the *mostDiscriminantFeatures* function, which works as follows. Let ϕ be a discriminant rule of the form $[a_i, ..., a_j] \not\rightarrow_\sigma b$, then *the witness of ϕ in $\mathcal{L}$*, denoted by $w(\phi, \mathcal{L})$, is the set of logs in which the pattern $[a_i, ..., a_j]$ occurs. Then, the set of the most discriminant feature

is computed through the heuristics of greedily selecting a feature ϕ covering the maximum number of traces, among the ones (S') not covered by previous selections.

5 Conclusions

In this paper we have introduced the problem of mining constrained graphs, with particular reference to the case of workflow systems. From an application viewpoint, the analysis of such models of execution can help in providing facilities for the human system administrator to monitor the actual behavior of many process models.

The paper proposes two distinct mining problems, and an overview of suitable solutions for such problems. In the context of inductive databases, the proposed problems raise interesting challenges, since the pattern languages introduced are worth even more complex mining tasks in which sophisticated constraints on the mining results can be specified. For example, one could be interested which discriminant factors characterize the failure or the success in the executions, or which is the choice that more frequently had led to a desired final configuration (e.g., to the acceptance of the order).

Interestingly, the techniques discussed in the previous sections are the adaptation of traditional learning techniques to a more structured domain in which background knowledge is available, and can be exploited for a smarter exploration of the search space. Indeed, frequent pattern discovery is essentially the adaptation of the *apriori* algorithm [2] to the case of workflow systems. Moreover, the Process Mining problem can be seen as a special case of inductive logic programming, in which the task is the mining of a set of consistent and complete clauses modelling the positive cases, and the latter correspond to log traces. Both the approaches presented in this paper have been extensively studied from an experimental point of view in [13,12], thus demonstrating their effectiveness w.r.t. traditional approaches which do not properly exploit the available domain knowledge.

In this context, a challenging research direction is to extend the proposed techniques in a full multirelational setting. Indeed, the proposed model is essentially a *propositional* model, for it assumes a simplification of the constrained graphs in which many real-life details are omitted. However, we believe that the model can be easily updated to cope with more complex constraints, such as time constraints, pre-conditions and post-conditions, and rules for exception handling.

References

1. R. Agrawal, D. Gunopulos, and F. Leymann. Mining process models from workflow logs. In *Proc. 6th Int. Conf. on Extending Database Technology (EDBT'98)*, pages 469–483, 1998.
2. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. of the 20th Int'l Conference on Very Large Databases*, pages 487–499, 1994.

3. R. Agrawal and R. Srikant. Mining sequential patterns. In *Proc. 11th Int. Conf. on Data Engineering (ICDE95)*, pages 3–14, 1995.
4. D. J. Cook and L. B. Holder. Substructure Discovery Using Minimum Description Length and Background Knowledge. *Journal of Artificial Intelligence Research*, 1(1):231–255, 1994.
5. J.E. Cook and A.L. Wolf. Automating process discovery through event-data analysis. In *Proc. 17th Int. Conf. on Software Engineering (ICSE'95)*, pages 73–82, 1995.
6. J.E. Cook and A.L. Wolf. Event-based detection of concurrency. In *Proc. 6th Int. Symposium on the Foundations of Software Engineering (FSE'98)*, pages 35–45, 1998.
7. J.E. Cook and A.L. Wolf. Software process validation: quantitatively measuring the correspondence of a process to a model. *ACM Trans. Softw. Eng. Methodol.*, 8(2):147–176, 1999.
8. A.K.A de Medeiros, B.F. van Dongen, W.M.P. van der Aalst, and A.J.M.M. Weijters. Process mining: Extending the a-algorithm to mine short loops. Technical report, University of Technology, Eindhoven, 2004. BETA Working Paper Series, WP 113.
9. L. Dehaspe and H. Toivonen. Discovery of Frequent DATALOG Patterns. *Data Mining and Knowledge Discovery*, 3(1):7–36, 1999.
10. D. Georgakopoulos, M. Hornick, and A. Sheth. An overview of workflow management: From process modeling to workflow automation infrastructure. *Distributed and Parallel Databases*, 3(2):119–153, 1995.
11. G.Greco, A.Guzzo, G.Manco, and D. Saccà. Mining frequent instances on workflows. In *Proc. 7th Pacific-Asia Conference (PAKDD'03)*, pages 209–221, 2003.
12. G.Greco, A.Guzzo, L.Pontieri, and D. Saccà. Mining expressive process models by clustering workflow traces. In *Proc. 8th Pacific-Asia Conference (PAKDD'04)*, pages 52–62, 2004.
13. G. Greco, A. Guzzo, G. Manco, and D. Saccà. Mining and reasoning on workflows. *IEEE Trans. on Data and Knowledge Eng.*, 17(4):519–534, 2005.
14. J. Han, J. Pei, and Y. Yi. Mining frequent patterns without candidate generation. In *Proc. Int. ACM Conf. on Management of Data (SIGMOD'00)*, pages 1–12, 2000.
15. J. Herbst. Dealing with concurrency in work?ow induction. In *Procs. European Concurrent Engineering Conference*, 2000.
16. J. Herbst and D. Karagiannis. Integrating machine learning and workflow management to support acquisition and adaptation of workflow models. *Journal of Intelligent Systems in Accounting, Finance and Management*, 9:67–92, 2000.
17. A. Inokuchi, T. Washi, and H. Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In *Proc. 4th European Conf. on Principles of Data Mining and Knowledge Discovery*, pages 13–23, 2000.
18. P. Koksal, S.N. Arpinar, and A. Dogac. Workflow history management. *SIGMOD Recod*, 27(1):67–75, 1998.
19. M. Kuramochi and G. Karypis. Frequent subgraph discovery. In *Proc. IEEE Int. Conf. on Data Mining (ICDM'01)*, pages 313–320, 2001.
20. H. Motoda and H. Liu. Data reduction: feature selection. *Handbook of data mining and knowledge discovery*, pages 208–213, 2002.
21. N.Lesh, M.J. Zaki, and M.Ogihara. Mining features for sequence classification. In *Proc. 6th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'00)*, pages 342–346, 1999.

22. B. Padmanabhan and A. Tuzhilin. Small is beautiful: discovering the minimal set of unexpected patterns. In *Proc. 6th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'00)*, pages 54–63, 2000.
23. R. Parekh and V. Honavar. Grammar Inference, Automata Induction and Language Acquisition. In *Handbook of Natural Language Processing*. Marcel Dekker, 2000.
24. J. Pei, J. Han, H. Lu, S. Nishio, S. Tang, and D. Yang. H-Mine: Hyper-structure mining of frequent patterns in large databases. In *Proc. IEEE Int. Conf. on Data Mining (ICDM'01)*, pages 441–448, 2001.
25. J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M. Hsu. Prefixspan: Mining sequential patterns by prefix-projected growth. In *Proc. IEEE Int. Conf. on Data Engineering (ICDE'2001)*, pages 215–224, 2001.
26. Guido Schimm. Mining most specific workflow models from event-based data. *Business Process Management*, pages 25–40, 2003.
27. W.M.P. van der Aalst and B.F. van Dongen. Discovering workflow performance models from timed logs. In *Proc. Int. Conf. on Engineering and Deployment of Cooperative Information Systems (EDCIS 2002)*, pages 45–63, 2002.
28. W.M.P. van der Aalst, B.F. van Dongen, J. Herbst, L. Maruster, G.Schimm, and A.J.M.M. Weijters. Workflow mining: A survey of issues and approaches. *Data and Knowledge Engineering*, 47(2):237–267, 2003.
29. W.M.P. van der Aalst and K.M. van Hee. *Workflow Management: Models, Methods, and Systems*. MIT Press, 2002.
30. W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster. Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*. To appear.
31. X. Yan and J. Han. gSpan: Graph-based substructure pattern pining. In *Proc. IEEE Int. Conf. on Data Mining (ICDM'02)*, 2001. An extended version appeared as UIUC-CS Tech. Report: R-2002-2296.
32. X. Yan and J. Han. CloseGraph: Mining closed frequent graph patterns. In *Proc. ACM Int. Conf. on Knowledge Discovery and Data Mining (KDD'03)*, pages 286–295, 2003.
33. K. Yoshida, H. Motoda, and N. Indurkhya. Graph- based induction as a unified learning framework. *Journal of Applied Intel.*, 4:297–328, 1994.

CrossMine: Efficient Classification Across Multiple Database Relations*

Xiaoxin Yin[1], Jiawei Han[1], Jiong Yang[1], and Philip S. Yu[2]

[1] University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA
{xyin1, hanj, jioyang}@uiuc.edu
[2] IBM T.J. Watson Research Center, Yorktown Heights, N.Y. 10598, USA
psyu@us.ibm.com

Abstract. Most of today's structured data is stored in relational databases. Such a database consists of multiple relations that are linked together conceptually via entity-relationship links in the design of relational database schemas. Multi-relational classification can be widely used in many disciplines including financial decision making and medical research. However, most classification approaches only work on single "flat" data relations. It is usually difficult to convert multiple relations into a single flat relation without either introducing huge "universal relation" or losing essential information. Previous works using Inductive Logic Programming approaches (recently also known as *Relational Mining*) have proven effective with high accuracy in multi-relational classification. Unfortunately, they fail to achieve high scalability w.r.t. the number of relations in databases because they repeatedly join different relations to search for good literals.

In this paper we propose CrossMine, an efficient and scalable approach for multi-relational classification. CrossMine employs *tuple ID propagation*, a novel method for virtually joining relations, which enables flexible and efficient search among multiple relations. CrossMine also uses aggregated information to provide essential statistics for classification. A selective sampling method is used to achieve high scalability w.r.t. the number of tuples in the databases. Our comprehensive experiments on both real and synthetic databases demonstrate the high scalability and accuracy of CrossMine.

1 Introduction

Relational databases are the most popular format for structured data, and is thus the richest source of knowledge in the world. There are many real world applications involving decision making process based on information stored in relational databases, such as credit card fraud detection and loan application.

* The work was supported in part by National Science Foundation under Grants IIS-02-09199/IIS-03-08215, and an IBM Faculty Award. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding agencies.

J.-F. Boulicaut et al. (Eds.): Constraint-Based Mining, LNAI 3848, pp. 172–195, 2005.

Approaches that can perform in-depth analysis on relational data is of crucial importance in such applications. Therefore, multi-relational data mining has become a field with strategic importance.

There have been many important approaches for classification, such as neural networks [11] and support vector machines [6]. They can only be applied to data represented in single, "flat" relations. Multiple relations in a database are usually connected via *semantic links* such as *entity-relationship links* of an ER model used in the database design [8]. Data stored in the same relation often have closer semantic relationship than those reachable via remote links. It is counter-productive to simply "convert" multi-relational data into a single flat data relation because such conversion may lead to the generation of a huge universal relation [8] but lose some essential semantic information carried by the semantic links in the database design.

Inductive Logic Programming (ILP) [12,10] is the most widely used category of approaches to multi-relational classification. There are many ILP approaches [1,3,4,5,13,14,15,18], which use very different philosophies in identifying hypotheses that fit the background knowledge. The ILP approaches achieve good classification accuracy. Unfortunately, most of them are not highly scalable w.r.t. the number of relations and the number of attributes in databases, thus are usually inefficient for databases with complex schemas.

In a database for multi-relational classification, there is one target relation R_t, whose tuples are called *target tuples*. Each target tuple is associated with a class label. To build a good multi-relational classifier, one needs to find good literals in each non-target relation R that help distinguish positive and negative target tuples. The target relation can usually join with every non-target relation via multiple join paths. Thus in a database with reasonably complex schema, there are a large number of join paths that need to be explored, each leading to dozens of literals in a certain relation. In order to identify the best literals and construct good clauses, many ILP approaches repeatedly join the relations along different join paths and evaluate literals based on the joined relation. This is very time consuming, especially when the joined relation contains much more tuples than the target one.

There are two major challenges in multi-relational classification: one is efficiency and scalability, and the other is the accuracy of classification. When building a classifier for a database with many relations, the search space is usually very large, and it is unaffordable to perform exhaustive search. On the other hand, the semantic linkages usually become very weak after passing through a long chain of links. Therefore, a multi-relational classifier needs to handle both efficiency and accuracy problems.

In this paper we propose CrossMine, a scalable and accurate approach for multi-relational classification. Its basic idea is to propagate the tuple IDs (together with their associated class labels) from the target relation to other relations. In the relation to which the IDs are propagated, each tuple t is associated with a set of IDs, which represent the target tuples that are joinable with t. Tuple ID propagation is a convenient and flexible method for virtually joining

different relations, with as low cost as possible. Tuple IDs can be easily propagated between any two relations, which enables CrossMine to search freely in multiple relations for good literals and clauses. CrossMine obtains high efficiency and scalability by tuple ID propagation.

CrossMine uses a sequential covering algorithm, which repeatedly constructs clauses and removes positive examples covered by each clause. To construct a clause, it repeatedly searches for the best literal and appends it to the current clause. During the searching process, CrossMine limits the search space to relations related to the target relation or related to relations used in the clause. In this way the strong semantic links can be identified and the search process is controlled in promising directions. On the other hand, the search space of CrossMine is larger than typical ILP approaches. By using tuple ID propagation and *look-one-ahead*, CrossMine considers literal sequences of length up to three at a time. It achieves both high efficiency and high accuracy by controlling the search space and identifying strong semantic links.

Unlike most previous approaches on multi-relational classification that only use simple literals, CrossMine uses both simple literals and literals involving aggregations on attribute values. For example, in the database of a CS department, a student's average grade or number of publications might be very important features for judging the academic performance of a student. The aggregations provide statistics about the target tuples, which often provide essential information for classification.

In many sequential covering algorithms, the negative examples are never removed in the clause building process, which makes the algorithm inefficient for databases with large numbers of tuples. It is common that before building a clause, there are much less positive examples than negative ones, which causes the algorithm to spend a large amount of time to build low-quality clauses. To address this issue, CrossMine employs a selective sampling method to reduce the number of negative tuples when the numbers of positive and negative tuples are unbalanced. This helps CrossMine achieve high scalability w.r.t. the number of tuples in databases. Our experiments show that the sampling method decreases the running time significantly but only slightly sacrifices the accuracy.

The remaining of the paper is organized as follows. In Section 2 we introduce the related work. The problem definition is presented in Section 3. Section 4 introduces the idea of tuple ID propagation and its theoretical background. We describe the algorithm and implementation issues in Section 5. Section 6 describes the negative tuple sampling technique. Experimental results are presented in Section 7. We made discussions in Section 8 and the study is concluded in Section 9.

2 Related Work

The most important category of approaches in multi-relational classification is ILP [12,10], which is defined as follows. Given background knowledge B, a set of positive examples P, and a set of negative examples N, find a hypothesis H, which is a set of Horn clauses such that:

– $\forall p \in P : H \cup B \models p$ (completeness)
– $\forall n \in N : H \cup B \not\models n$ (consistency)

The well known ILP systems include FOIL [18], Golem [14], and Progol [13]. FOIL is a top-down learner, which builds clauses that cover many positive examples and few negative ones. Golem is a bottom-up learner, which performs generalizations from the most specific clauses. Progol uses a combined search strategy. Some recent approaches TILDE [3], Mr-SMOTI [1], and RPTs [15] use the idea of C4.5 [17] and inductively construct decision trees from relational data. These approaches are usually more efficient than traditional ILP approaches due to the divide-and-conquer nature of decision tree algorithm.

Efficiency and scalability are two major issues in ILP. In [4] an approach was proposed to handle data stored on disks. In [5] the authors proposed an approach that can evaluate packs of queries which can be handled together. This approach is similar to CrossMine because both of them can utilize common prefix of different clauses. But CrossMine can propagate tuple IDs freely among different relations, which is more convenient in building clauses.

Besides ILP, probabilistic approaches [19,16] are also popular for multi-relational classification and modelling. Probabilistic relational models [19] is an extension of Bayesian networks for handling relational data, which can integrate the advantages of both logical and probabilistic approaches for knowledge representation and reasoning. In [16] an approach is proposed to integrate ILP and statistical modelling for document classification and retrieval.

We take FOIL as a typical example of ILP approaches and show its working procedure. FOIL is a sequential covering algorithm that builds clauses one by one. Each clause is built by repeatedly adding literal. At each step, every possible literal is evaluated and the best one is appended to the current clause. To evaluate a literal p, p needs to be appended to the current clause c to get a new clause c'. Then it constructs a new dataset which contains all target tuples satisfying c', evaluates p based on the number of positive and negative target tuples satisfying c'. For databases with complex schemas, the search space is huge and there are many possible literals at each step. Thus FOIL needs to repeatedly construct datasets by physical joins to find good literals, which is very time-consuming. This is also verified by our experiments.

3 Preliminaries

3.1 Basic Definitions

A database D consists of a set of relations, one of which is the target relation R_t, with class labels associated with its tuples. The other relations are non-target relations. Each relation may have one primary key and several foreign keys. The following types of joins are considered in CrossMine:

1. Join between a primary key k and some foreign key pointing to k.
2. Join between two foreign keys k_1 and k_2, which point to the same primary key k. (For example, the join between Loan.account-id and Order.account-id.)

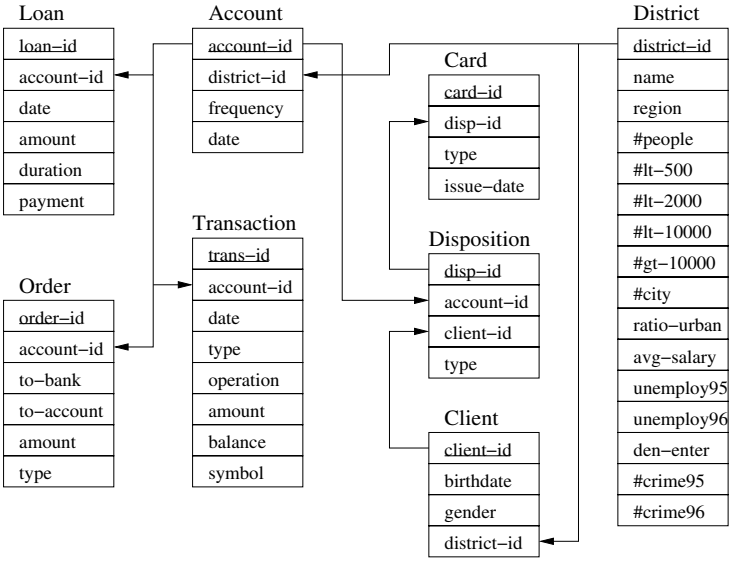


Fig. 1. The financial database from PKDD CUP 99

We ignore other possible joins because they do not represent strong semantic relationships between entities in the database. Figure 1 shows an example database. Arrows go from primary-keys to corresponding foreign-keys. The target relation is *Loan*. Each target tuple is either positive or negative, indicating whether the loan is paid on time.

CrossMine is a clause-based classifier on relational data. In general, each clause consists of a list of literals and the predicted class. Each literal is either a simple literal on the value of an attribute, or an aggregation literal on the aggregated value of an attribute.

3.2 Literals

In general, a literal is a constraint on a certain attribute in a certain relation. For example, literal "$l_1 = Loan(L, \_, \_, \_, >= 12, \_)$" means that the duration of loan L is no less than 12 months. In relational databases a literal is often defined based on a certain join path. For example, "$l_2 = Loan(L, A, \_, \_, \_, \_), Account(A, \_, monthly, \_)$" is defined on the join path $Loan \bowtie Account$, which means that the associated account of a loan has frequency "monthly".

There are two types of attributes: categorical attributes and numerical attributes. There are three types of literals:

1. **Categorical literal:** A categorical literal is defined on a categorical attribute. It is a constraint that this attribute must take a certain value, such as l_2 in the above example.

2. **Numerical literal:** A numerical literal is defined on a numerical attribute. It contains a certain value and a comparison operator, such as l_1, in the above example.
3. **Aggregation literal:** An aggregation literal is similar to a numerical literal, but is defined on the aggregated value of an attribute. It contains an aggregation operator, a certain value, and a comparison operator. For example, $l_3 = Loan(L, A, \_, \_, \_, \_), Order(\_, A, \_, \_, sum(amount) >= 1000, \_)$ is an aggregation literal, which requires the sum of amount of all orders related to a loan is no less than 1000. The following aggregation operators can be used: count, sum, avg.

3.3 Clauses

CrossMine is a clause-based classifier, which aims at finding clauses that distinguish positive examples from negative ones. Each clause contains a list of literals, associated with a class label. To integrate the join path into the clauses, CrossMine uses a form of clauses that is different from the traditional ILP approaches. Instead of using conventional literal, *complex literal* is used here as the element of clauses. A complex literal $\hat{l}$ contains two parts:

1. *prop-path*, i.e., propagation path, which indicates how to propagate IDs. For example, "$Loan.account\_id \rightarrow Account.account\_id$" indicates propagating IDs from the *Loan* relation to the *Account* relation using the join condition "$Loan.account\_id = Account.account\_id$". [1]
2. *constraint*: which indicates the constraint on the relation which the IDs are propagated to. For example, "$Account.frequency = monthly$" indicates that tuples in the *Account* relation should have value "monthly" on attribute *frequency. The constraint is actually a literal that is either categorical, numerical, or involves aggregation.*

A complex literal is usually equivalent to two conventional literals. For example, the clause "$Loan(L, +) :- Loan(L, A, \_, \_, \_, \_), Account(A, \_, monthly, \_)$" can be represented by "$Loan(+) :- [Loan.account\_id \rightarrow Account.account\_id, Account.frequency = monthly]$".

A clause contains a list of literals. A target tuple satisfies a clause if and only if it satisfies every literal of the clause. To judge whether a target tuple t satisfies a clause c, one needs to join t with tuples in other relations according to the join path of c. We will introduce how to efficiently find out all target tuples satisfying a clause later.

We use the database in Figure 2 as an illustrative example. Suppose clause $c = Loan(+) :- [Loan.account\_id \rightarrow Account.account\_id, Account.frequency = monthly]$. We say a tuple t in *Loan* satisfies c if and only if **any** tuple in *Account* that is joinable with t has value "monthly" in the attribute of *frequency*. In this example, there are two tuples (with account-id 124 and 45) in *Account* that satisfy the literal "$Account(A, \_, monthly, \_)$". So there are four tuples (with loan-id 1, 2, 4, and 5) in *Loan* that satisfy this clause.

[1] The prop-path of a complex literal may be empty if we already have the right tuple IDs on the relation to which the constraint is applied.

Loan					
loan-id	account-id	amount	duration	payment	*class*
1	124	1000	12	120	+
2	124	4000	12	350	+
3	108	10000	24	500	−
4	45	12000	36	400	−
5	45	2000	24	90	+

Account		
account-id	frequency	date
124	monthly	960227
108	weekly	950923
45	monthly	941209
67	weekly	950101

Fig. 2. A sample database (The last column of Loan relation contains class labels)

3.4 Evaluation of Literals and Clauses

To generate a clause, CrossMine starts at an empty clause, keeps selecting the best literal and add it to the current clause. At each step, we need to evaluate every literal and select the best one. *Foil gain* is used [18] to measure the goodness of a literal.

Definition 1 (Foil gain). *For a clause c, we use $P(c)$ and $N(c)$ to denote the number of positive and negative examples satisfying c. Suppose the current clause is c. We use $c + l$ to denote the clause constructed by appending literal l to c. The foil gain of literal l is defined as follows,*

$$I(c) = -\log \frac{P(c)}{P(c) + N(c)} \tag{1}$$

$$foil\_gain(l) = P(c + l) \cdot [I(c) - I(c + l)] \tag{2}$$

Intuitively $foil\_gain(l)$ represents the total number of bits saved in representing positive examples by appending l to the current clause. It indicates how much the predictive power of the clause can be increased by appending l to it.

After generating a clause c, we need to evaluate c by estimating its accuracy. Suppose there are N^+ positive and N^- negative tuples satisfying c in the training set. The accuracy of c can be estimated using the method in [7], which is shown in the following equation:

$$Accuracy(c) = (N^+ + 1)/(N^+ + N^- + C) \tag{3}$$

where C is the number of classes.

4 Tuple ID Propagation

In this section we present the idea of tuple ID propagation and method of finding good literals with that. In essence, tuple ID propagation is a method for virtually joining non-target relations with the target relation. It is a convenient method that enables flexible search in relational databases, and is much less costly than physical join in both time and space.

4.1 Search for Literals by Joins

Consider the sample database in Figure 2. Suppose we want to compute the foil gain of literals in a non-target relation, such as *Account*. We need to find out for all positive and negative target tuples satisfying each literal l in the *Account* relation.

One approach is to join the two relations together and compute the foil gain of all literals, as shown in Figure 3. With the joined relation, the foil gain of every literal in both relations can be computed. To compute the foil gain of all literals on a certain attribute, one only needs to scan the corresponding column in the joined relation once. It can also handle continuous attribute as in [17]. To find the best literal on attribute *Account.date*, one can first sort that column, then iterate from the smallest value to the largest value, and for each value d, compute the foil gain of two literals "date $\leq d$" and "date $\geq d$".

Loan $\bowtie$ Account							
l-id	a-id	amount	dur	pay	freq	date	*class*
1	124	1000	12	120	monthly	960227	+
2	124	4000	12	350	monthly	960227	+
3	108	10000	24	500	weekly	950923	−
4	45	12000	36	400	monthly	941209	−
5	45	2000	24	90	monthly	941209	+

Fig. 3. The join of *Loan* and *Account*

It is quite expensive to use physical joins to evaluate literals for the following two reasons. First, in a database with complex schema, there are usually a large number of join paths that need to be explored. For example, in the database shown in Figure 1, *Loan* can join with *Account*, *Order*, *Transaction* and *Disposition*. Each of the four relations can join with several other relations, such as *Disposition* that can join with *Card*, *Client*, or back to *Account* and *Order*. Therefore one needs to repeatedly perform physical joins and create many joined relations. Second, there may be much more tuples in a joined relation than in the target relation. For example, a loan may join with several orders or dozens of transactions. Thus the joined relation may contain a large number of tuples when the join path is long.

The above two challenges prevent most traditional ILP approaches from efficiently searching among different relations. In the next section we will introduce *tuple ID propagation*, a technique that enables free search in relational databases. When searching for good literals, one can propagate tuple IDs from any relation that IDs have been propagated to, which requires much less computation and data transfer. The tuple IDs can be easily propagated between any two relations, which makes it possible to "navigate freely" among different relations.

4.2 Tuple ID Propagation

Suppose the primary key of the target relation is an attribute of integers, which represents the ID of each target tuple. Consider the sample database shown in Figure 4, which has the same schema as in Figure 2. Instead of performing physical join, the IDs and class labels of target tuples can be propagated to the *Account* relation. The procedure is formally defined as follows.

Loan

loan-id	account-id	amount	duration	payment	*class*
1	124	1000	12	120	+
2	124	4000	12	350	+
3	108	10000	24	500	−
4	45	12000	36	400	−
5	45	2000	24	90	+

Account

account-id	frequency	date	IDs	*class labels*
124	monthly	960227	1, 2	2+, 0−
108	weekly	950923	3	0+, 1−
45	monthly	941209	4, 5	1+, 1−
67	weekly	950101	–	0+, 0−

Fig. 4. Example of tuple ID propagation

Definition 2 (Tuple ID propagation). *Suppose two relations R_1 and R_2 can be joined by attributes $R_1.A$ and $R_2.A$. Each tuple t in R_1 is associated with a set of IDs in the target relation, represented by $idset(t)$. For each tuple u in R_2, we set $idset(u) = \bigcup_{t \in R_1, t.A = u.A} idset(t)$.*

The following lemma and its corollary show the correctness of tuple ID propagation and how to compute foil gain from the propagated IDs.

Lemma 1. *Suppose two relations R_1 and R_2 can be joined by attribute $R_1.A$ and $R_2.A$, and R_1 is the target relation, with primary key $R_1.id$. All the tuples in R_1 satisfy the current clause (others have been eliminated). The current clause contains a literal "$R_1(R_1.id, R_1.A, \cdots)$", which enables the join of R_1 with R_2. With tuple ID propagation from R_1 to R_2, for each tuple u in R_2, $idset(u)$ represents all target tuples joinable with u, using the join path specified in the current clause.*

Proof. *From definition 2, we have $idset(u) = \bigcup_{t \in R_1, t.A = u.A} idset(t)$. That is, $idset(u)$ represents the target tuples joinable with u using the join path specified in the current clause.*

Corollary 1. *Suppose two relations R_1 and R_2 can be joined by attribute $R_1.A$ and $R_2.A$, R_1 is the target relation, and all the tuples in R_1 satisfy the current*

clause (others have been eliminated). If R_1's IDs are propagated to R_2, then the foil gain of every literal in R_2 can be computed using the propagated IDs on R_2.

Proof. *Given the current clause c, for a literal l in R_2, such as $R_2.B = b$, its foil gain can be computed based on $P(c)$, $N(c)$, $P(c + l)$ and $N(c + l)$. $P(c)$ and $N(c)$ should have been computed during the process of building the current clause. $P(c + l)$ and $N(c + l)$ can be computed in the following way: (1) find all tuples t in R_2 that $t.B = b$; (2) with the propagated IDs on R_2, find all target tuples that can be joined with any tuple found in (1) (using the join path specified in the current clause); and (3) count the number of positive and negative tuples found in (2).*

For example, suppose "$Loan(L, +) :- Loan(L, A, \_, \_, \_, \_)$" is the current clause. For literal "$Account(A, \_, monthly, \_)$", we can first find out tuples in the *Account* relation that satisfy this literal, which are {124, 45}. Then we can find out tuples in the *Loan* relation that can be joined with these two tuples, which are {1, 2, 4, 5}. We maintain a global table of the class label of each target tuple. From this table, we know that tuples {1, 2, 4, 5} contain three positive and one negative examples. With this information we can easily compute the foil gain of literal "$Account(A, \_, monthly, \_)$".

Besides propagating IDs from the target relation to relations directly joinable with it, one can also propagate IDs transitively by propagating the IDs from one non-target relation to another, according to the following lemma.

Lemma 2. *Suppose two non-target relations R_2 and R_3 can be joined by attribute $R_2.A$ and $R_3.A$, and all the tuples in R_2 satisfy the current clause (others have been eliminated). For each tuple v in R_2, $idset(v)$ represents the target tuples joinable with v (using the join path specified by the current clause). By propagating IDs from R_2 to R_3 through the join $R_2.A = R_3.A$, for each tuple u in R_3, $idset(u)$ represents target tuples that can be joined with u (using the join path in the current clause, plus the join $R_2.A = R_3.A$).*

Proof. *Suppose a tuple u in R_3 can be joined with $v_1, v_2, \cdots, v_m$ in R_2, using join $R_2.A = R_3.A$. Then $idset(u) = \bigcup_{i=1}^{m} idset(v_i)$. A target tuple t is joinable with any one of $v_1, v_2, \cdots, v_m$ if and only if $t.id \in \bigcup_{i=1}^{m} idset(v_i)$. Therefore, a target tuple t is joinable with u (using the join path in the current clause, plus the join $R_2.A = R_3.A$) if and only if $t.id \in idset(u)$.*

A corollary similar to corollary 1 can be proved for Lemma 2. That is, by tuple ID propagation between non-target relations, one can also compute the foil gain based on the propagated IDs.

4.3 Analysis and Constraints

The idea of *label propagation* was proposed in [2], which propagates class labels along join paths for evaluating literals. This approach is effective for n-to-1 relationships. But for join paths that involve 1-to-n or n-to-n relationships, it cannot find the numbers of positive and negative target tuples satisfying each

literal. For example, suppose there are 10 tuples in the *Loan* relation, 5 being positive and 5 being negative. 4 positive and 5 negative tuples are joinable with 1 account each, while the other positive tuple is joinable with 10 accounts. Suppose all above accounts satisfy a literal l. Then one can see that there are 5 positive and 5 negative target tuples satisfying l, indicating that l has low foil gain. However, if only class labels are propagated, we will not be able to distinguish class labels from different target tuples, and will say that there are 14 positive and 5 negatives tuples satisfying l, indicating that l has high foil gain. A real database usually contains many 1-to-n and n-to-n relationships, thus one needs to propagate IDs instead of labels when building classifiers.

Tuple ID propagation is a way to perform virtual join. Instead of physically joining relations, they are virtually joined by attaching the tuple IDs of the target relation to the tuples of a non-target relation, using a certain join path. In this way the literals can be evaluated as if physical join is performed. Tuple ID propagation is a flexible and efficient method. IDs (and their associated class labels) can be easily propagated from one relation to another. By dong so, literals in different relations can be evaluated with little redundant computation. The required space is also small because the IDs do not take much additional storage space. Moveover, a relation may be associated with multiple set of IDs corresponding to different join paths. This enables CrossMine to search for good literals freely across relations.

ID propagation, though valuable, should be enforced with certain constraints. There are two cases that such propagation could be counter-productive: (1) propagate via large fan-outs, and (2) propagate via long weak links.

The first case happens if the there are too many tuples that can be produced via propagation. Suppose after the IDs are propagated to a relation R, it is found that every tuple in R can be joined to many target tuples and every target tuple can be joined to many tuples in R. Then the semantic link between R and the target relation is usually very weak because the link is very unselective. For example, propagation among people via birth-country links may not be productive. Therefore, our system discourages propagation if the current link has very large fan-out.

The second case happens if the propagation goes through long weak links, e.g., linking a student with his car dealer's pet (via car, and then dealer) may not be productive either. From the consideration of both efficiency and accuracy, our system discourages propagation via such links.

5 Clause Generation

In this section we present CrossMine's algorithm for generating clauses by tuple ID propagation. A sequential covering algorithm is developed that repeatedly builds clauses and removes positive tuples satisfying the clause. To build a clause, it repeatedly searches for the best literal and adds it to the current clause. This algorithm is selected because it guarantees the quality of each clause by always keeping a large number of negative examples, and moreover, its greedy nature makes it efficient in large databases.

5.1 Finding Best Literal

Suppose CrossMine is searching for the best literal in a certain relation R, and tuple IDs have been propagated to R so that one will know the target tuples joinable with each tuple in R. To find the best literal in R, CrossMine evaluates the literals in each attribute of R. Different algorithms are used for categorical and numerical attributes.

Suppose the best literal on a categorical attribute A_c is to be found. Suppose A_c has l values $a_1, \ldots, a_l$. For each value a_i, a literal $l_i = [R.A_c = a_i]$ is built. Then CrossMine scans the values of each tuple on A_c to find out the numbers of all positive and negative target tuples satisfying each literal l_i. With this information, the foil gain of each l_i can be computed and the best literal can be found.

Suppose the best literal on a numerical attribute A_n is to be found, and a sorted index for values on A_n has been built beforehand. CrossMine iterates from the smallest value of A_n to the largest value. When iterating to each value v_i, all tuples having value v_i are found, and their associated IDs are added into a pool. This pool of IDs represent all target tuples satisfying literal $[A_n \leq v_i]$. In this way, one can compute the foil gain of every literal of the form $[A_n \leq v_i]$ for every value v_i of A_n. Then CrossMine iterates from the largest value to the smallest value to evaluate the literals of the form $[A_n \geq v_i]$. In this way the best numerical literal can be found for A_n.

To search for the best aggregation literal for A_n, CrossMine first finds some statistics for each target tuple. By scanning the tuple IDs associated with tuples in R, for each target tuple t^*, CrossMine can find the tuples in R joinable with t^*, and calculate the count, sum, and average of the values of those tuples on A_n. Then CrossMine computes the foil gain of all aggregation literals using an approach similar to the approach for finding best numerical literals. In this way the best aggregation literal can be found.

5.2 Clause Generation Algorithms

Given a relational database with one target relation, CrossMine builds a classifier containing a set of clauses, each of which contains a list of complex literals and a class label. The overall idea is to repeatedly build clauses. After each clause is built, remove all positive target tuples satisfying it. The algorithm is shown in Figure 5.

To build a clause, one repeatedly searches for the best complex literal and appends it to the current clause, until the stop criterion is met. A relation is *active* if it appears in the current clause, or it is the target relation. Every active relation is required to have the correct propagated IDs on every tuple before searching for the next best literal. The algorithm is shown in Figure 6.

The following procedure is used to find the best literal: (1) for every active relation $\hat{R}$, find the best complex literal whose constraint applies on $\hat{R}$ (no ID propagation involved), and (2) for every relation $\bar{R}$ that can be joined with some active relation $\hat{R}$, propagate IDs from $\hat{R}$ to $\bar{R}$, and find the best complex literal on

Algorithm 1. Find-Clauses
Input: a relational database D with a target relation R_t.
Output: a set of clauses for predicting class labels of target tuples.

Procedure
clause set $R \leftarrow emptyset$;
do
clause $c \leftarrow$ *Find-A-Clause()*;
add c to R;
remove all positive target tuples satisfying c;
while(there are more than 10% positive target tuples left);
return R;

Fig. 5. Algorithm *Find-Clauses*

Algorithm 2. Find-A-Clause
Input: a relational database D with a target relation R_t.
Output: a clause for predicting class labels of target tuples.

Procedure
clause $c \leftarrow$ *empty-clause*;
set R_t to active;
do
Complex literal $l \leftarrow$ *Find-Best-Literal()*;
if $foil\_gain(l) <$ MIN_FOIL_GAIN;
then break;
else
$c \leftarrow c + l$;
remove all target tuples not satisfying c;
update IDs on every active relation;
if $l.constraint$ is on an inactive relation
then set that relation active;
while($c.length <$ MAX_CLAUSE_LENGTH);
return c;

Fig. 6. Algorithm *Find-A-Clause*

$\bar{R}$. Consider the database in Figure 1. Originally only *Loan* is active. Suppose the first best complex literal is "[$Loan.account\_id \rightarrow Account.account\_id$, $Account.frequency = monthly$]". Now *Account* becomes active as well. And we will try to propagate the tuple IDs from *Loan* or *Account* in every possible way to find the next best literal.

The idea behind the algorithm of building a clause is as follows. Starting from the target relation R_t, find the best complex literal $\hat{l}$, which propagates IDs from R_t to another relation $\bar{R}$. Then start from either R_t or $\bar{R}$ to find the next complex literal. This algorithm is greedy in nature. It extends the clause using only those literals in either the active relations or the relations directly joinable with an active relation.

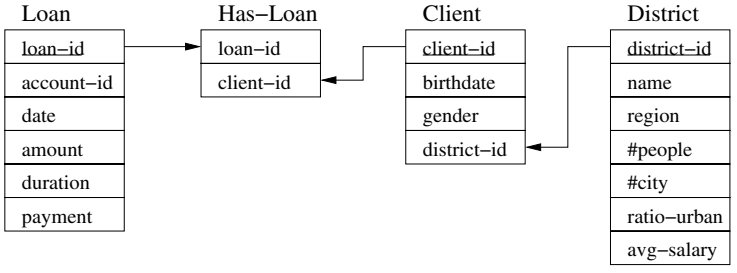


Fig. 7. Another sample database

The above algorithm may fail to find good literals in databases containing some relations that are used to join with other relations, such as the database shown in Figure 7. In this database there is no meaningful literal in the Has_Loan relation. Therefore, the clauses built will never involve any literals on the $Client$ relation and the $District$ relation.

This problem can be solved using the *look-one-ahead* method. When searching for the best literal, after IDs have been propagated to a relation $\bar{R}$, if $\bar{R}$ contains a foreign-key pointing to relation $\bar{R}'$, IDs are propagated from $\bar{R}$ to $\bar{R}'$, and used to search for good literals in $\bar{R}'$. By this method, in the example in Figure 7, one can find clauses such as "$Loan(+) :- [Loan.loan\_id \rightarrow Has\_Loan.loan\_id, Has\_Loan.client\_id \rightarrow Client.client\_id, Client.birthdate < 01/01/60]$".

With the correct IDs on a relation $\bar{R}$, one can scan $\bar{R}$ once to compute the number of positive and negative target tuples satisfying every literal in $\bar{R}$, using the approach in [9]. The algorithm for searching for the best complex literal is shown in Figure 8.

The above algorithms show the procedure of building clauses in CrossMine. The basic idea of building a clause is to start from the target relation, keep appending literals in active relations or relations related to some active relation, until the stopping criterion is met. The running time of CrossMine is not much affected by the number of relations in the database, because the size of the search space is mainly determined by the number of active relations and the number of joins on each active relation. This is also verified in our experiments on synthetic databases.

To achieve high accuracy in multi-relational classification, an algorithm should be able to find most of the useful literals in the database, and builds good clauses with them. In most commercial databases following E-R model design there are two types of relations: *entity* relation and *relationship* relation. Usually each entity relation is reachable from some other entity relations via join paths going through relationship relations. Suppose an entity relation R contains useful information for classification. There are usually many join paths between R and the target relation R_t, some representing important semantic links. It is likely that R can be reached from some other useful entity relations through

Algorithm 3. Find-Best-Literal
Input: a relational database D with a target relation R_t, and current clause c.
Output: the complex literal with most foil gain.

Procedure
Complex literal $l_{max} \leftarrow empty$;
for each active relation $\hat{R}$
Complex literal $l \leftarrow$ best complex literal in $\hat{R}$;
if $foil\_gain(l) > foil\_gain(l_{max})$
then $l_{max} \leftarrow l$;
for each relation $\bar{R}$
for each key/foreign-key k of $\bar{R}$
if $\bar{R}$ can be joined to some active relation $\hat{R}$ with $\bar{R}.k$
then
propagate IDs from $\hat{R}$ to $\bar{R}$;
$l \leftarrow$ best complex literal in $\bar{R}$;
if $foil\_gain(l) > foil\_gain(l_{max})$
then $l_{max} \leftarrow l$;
for each foreign-key $k' \neq k$ of $\bar{R}$
propagate IDs from $\bar{R}$ to relation $\bar{R}'$
that is pointed to by $\bar{R}.k$;
$l \leftarrow$ best complex literal in $\bar{R}'$;
if $foil\_gain(l) > foil\_gain(l_{max})$
then $l_{max} \leftarrow l$;
return l_{max};

Fig. 8. Algorithm *Find-Best-Literal*

relationship relations. Therefore, by using the method of look-one-ahead, it is highly probable that one can utilize the information in R.

Most ILP approaches also perform heuristical search when building clauses. However, the search spaces of those approaches are usually much smaller than that of CrossMine. By using complex literals, CrossMine considers two literals at a time (one for join and another for value constraint). By using look-one-ahead, it can consider up to three literals together in clause generation. This enables CrossMine to find good literals and build more accurate classifiers than traditional ILP approaches. On the other hand, CrossMine is rather different from joining a large number of relations indiscriminately, such as the "universal relation" approach. Instead, it limits the search process (i.e., tuple ID propagation) among only active relations with at most one look-ahead. Thus the search space is more confined, following more promising and active links than indiscriminate joins, and thus lead to both high efficiency and classification accuracy.

5.3 Predicting Class Labels with Clauses

After generating clauses, CrossMine needs to predict the class labels of unlabelled target tuples. CrossMine also needs to predict the class labels of the tuples in

the training set to estimate the accuracy of each clause. Therefore, an efficient algorithm is needed for finding out all target tuples satisfying each clause.

CrossMine uses an efficient algorithm based on tuple ID propagation to find out all target tuples satisfying a certain clause c. Suppose $c = R_t(+) :- l_1, l_2, \ldots, l_k$. ($l_i$ $(1 \leq i \leq k)$ is a complex literal.) The main idea of the algorithm is to propagate the IDs of all target tuples along the prop-path of each literal l_i, and prune all IDs of target tuples not satisfying the constraint of l_i.

To illustrate this procedure, let us examine an example. Suppose $c = Loan(+) :- [Loan.account\_id \rightarrow Account.account\_id, Account.frequency = monthly]$, $[Account.district\_id \rightarrow District.district\_id, avg\_salary > 80000]$. First, the IDs of all target tuples are propagated to the *Account* relation via the prop-path $Loan.account\_id \rightarrow Account.account\_id$. All target tuples whose associated account has value "monthly" on attribute $frequency$ are found, and the IDs of all the other tuples are pruned. Then the remaining IDs are propagated to the *District* relation via the prop-path $Account.district\_id \rightarrow District.district\_id$, and target tuples satisfying the second literal are found, which are all tuples satisfying this clause.

Given a set of target tuples whose class labels need to be predicted, CrossMine first finds out the tuples satisfying each clause. For each target tuple t, the most accurate clause that is satisfied by t is found, and the class label of that clause is used as the predicted class. If multiple classes are presented in the training set, then for each class C, CrossMine takes tuples of C as positive tuples and all the other tuples as negative ones to build clauses for class C. The same algorithm is used for predicting the class labels of unseen tuples.

6 Tuple Sampling

From Algorithm 1 we can see that during the procedure of building clauses, the number of positive tuples keeps decreasing and the number of negative tuples remains unchanged. Each clause covers a certain proportion of the remaining positive tuples (usually 5% to 20%), thus the first several clauses can often cover the majority of the positive tuples. However, even if most of positive tuples have been covered, it still takes a similar amount of time to build a clause because all the negative tuples remain there.

Let $c.sup^+$ and $c.sup^-$ be respectively the number of positive and negative tuples satisfying a clause c. Let $c.bg^+$ and $c.bg^-$ be respectively the number of positive and negative tuples satisfying c when c is built. The accuracy of c can be estimated using the method in [7], which is shown in the following equation:

$$Accuracy(c) = (c.sup^+ + 1)/(c.sup^+ + c.sup^- + C) \tag{4}$$

where C is the number of classes.

In the algorithm described above, $c.bg^-$ always equals to the number of negative tuples. When $c.bg^+$ is small, even if $c.bg^-$ is large, the quality of c cannot be guaranteed. That is, if $c.bg^+$ is small, one cannot be confident that $Accuracy(c)$ is a good estimate for the real world accuracy of c. Therefore, although much

time is spent in building these clauses, the quality of the clauses is usually much lower than that of the clauses with high bg^+ and bg^-.

Based on this observation, the following method is proposed to improve its effectiveness. Before a clause is built, we require that the number of negative tuples is no greater than NEG_POS_RATIO times the number of positive tuples. Sampling is performed on the negative tuples if this requirement is not satisfied. We also require that the number of negative tuples is smaller than MAX_NUM_NEGATIVE, which is a large constant.

Here we analyze the improvement on efficiency by sampling. Our experiments show that, when only a small portion of positive tuples remain, each clause generated usually covers an even smaller portion of the remaining positive tuples. The possible reason is that, there are usually many "special positive cases" that cannot be covered by any good clause. The consequence is that the number of generated clauses usually increases with the number of target tuples. When sampling is not used, the time for building each clause is proportional to the total number of target tuples. Thus the total runtime increases sharply as the number of tuples increases, because more clauses are needed and longer time is used for building each clause. When sampling is used, the time for building a clause is proportional to the number of remaining positive tuples. Because the first several clauses can often cover the majority of positive tuples, the total number of tuples decreases sharply after finding them, and the algorithm becomes highly scalable.

When sampling is used, the accuracy of clauses should be estimated in a different way. Suppose before building clause c, there are P positive and N negative tuples. N' negative tuples are randomly chosen by sampling ($N' < N$). After building clause c, suppose there are l positive and n' negative tuples satisfying c. We need to estimate n, the number of negative tuples satisfying c. The simplest estimation is $n \approx n' \frac{N}{N'}$. However, this is not a safe estimation because it is quite possible that c luckily excludes most of the N' negative examples but not the others. We want to find out a number n, so that the probability that $n' \leq n\frac{N'}{N}$ is 0.9. Or to say, it is unlikely that $\frac{n}{N} \leq \frac{n'}{N'}$.

As we know, N' out of N negative tuples are chosen by sampling. Assume we already know that n negative tuples satisfy c. Consider the event of a negative tuple satisfying c as a random event. Then n' is a random variable obeying binomial distribution, $n' \sim B(N', \frac{n}{N})$. n' can be considered as the sum of N' random variable of $B(1, \frac{n}{N})$. When N' is large, according to central limit theorem, we have $\frac{n'}{N'} \sim N(\frac{n}{N}, \frac{\frac{n}{N}(1-\frac{n}{N})}{N'})$. For a random variable $X \sim N(\mu, \sigma^2)$, $P(X \geq \mu - 1.28\sigma) \approx 0.9$. So we require

$$\frac{n'}{N'} = \frac{n}{N} - 1.28\sqrt{\frac{\frac{n}{N}(1-\frac{n}{N})}{N'}} \tag{5}$$

Let $x = \frac{n}{N}$ and $d = \frac{n'}{N'}$. Equation (5) is converted into

$$\left(1 + \frac{1.64}{N'}\right)x^2 - \left(2d + \frac{1.64}{N'}\right)x + d^2 = 0 \tag{6}$$

Equation (6) can be easily solved with two solutions x_1 and x_2, corresponding to the positive and negative squared root in equation (5). The greater solution x_2 should be chosen because it corresponds to the positive squared root. If there are x_2N negative tuples satisfying the clause before sampling, then it is unlikely that there are less than n' tuples satisfying the clause after sampling. Therefore, we use x_2N as the safe estimation of n. From the estimated n, we can estimate the accuracy of c based on equation (4).

7 Experimental Results

We have performed comprehensive experiments on both synthetic databases and real databases to show the accuracy and scalability of CrossMine. We compare CrossMine with FOIL [18] and TILDE [3] in every experiment, where the source code of FOIL and binary code of TILDE are from their authors. CrossMine and FOIL are run on a 1.7GHz Pentium 4 PC running on Windows 2000 Professional. TILDE is run on a Sun Blade 1000 workstation. Ten-fold experiments are used unless specified otherwise.

The following parameters are used in our experiments for testing CrossMine: MIN_FOIL_GAIN = 2.5, MAX_CLAUSE_LENGTH = 6, NEG_POS_RATIO = 1, and MAX_NUM_NEGATIVE = 600. Moreover, we have found that the accuracy and running time of CrossMine are not sensitive to these parameters.

7.1 Synthetic Databases

To evaluate the scalability of CrossMine, a set of synthetic relational databases are generated. These databases mimic the real world relational databases. Our data generator takes the parameters shown in Table 1 to generate a database. The three columns of Table 1 represent the parameter name, description, and default value.

To generate the database, we first generate a relational schema with $|R|$ relations, one being the target relation. The number of attributes of each relation obeys exponential distribution with expectation A and is at least A_{min}. One of the attributes is the primary-key. All attributes are categorical, and the number of values of each attribute (except the primary key) obeys exponential distribution with expectation V and is at least V_{min}. Besides these attributes, each relation has a few foreign-keys, pointing to the primary-keys of other relations. The number of foreign-keys of each relation obeys exponential distribution with expectation F and is at least F_{min}.

After the schema is generated, we generate clauses that are lists of complex literals. The number of complex literals in each clause obeys uniform distribution between L_{min} and L_{max}. Each complex literal has probability f_A to be on an active relation and probability $(1 - f_A)$ to be on an inactive relation (involving a propagation). Only categorical literals are used. The class label of each clause is randomly generated, but the number of positive clauses and that of negative clauses differ by at most 20%.

Table 1. Parameters of data generator

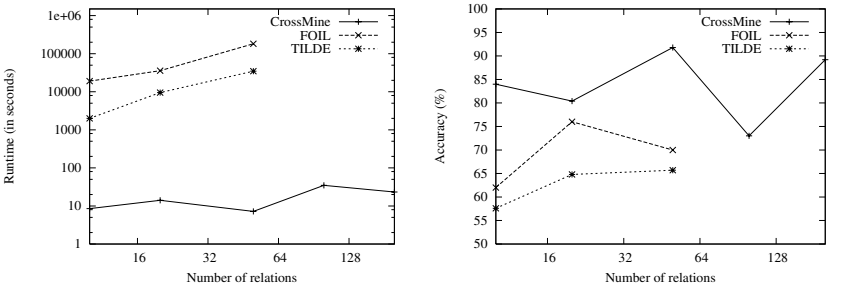
Name	Description	Def.
$\|R\|$	# relations	x
T_{min}	Min # tuples in each relation	50
T	Expected # tuples in each relation	y
A_{min}	Min # attributes in each relation	2
A	Expected # attributes in each relation	5
V_{min}	Min # values of each attribute	2
V	Expected # values of each attribute	10
F_{min}	Min # foreign-keys in each relation	2
F	Expected # foreign-keys in each relation	z
$\|c\|$	# clauses	10
L_{min}	Min # complex literals in each clause	2
L_{max}	Max # complex literals in each clause	6
f_A	Prob. of a literal on active relation	0.25

The generated tuples are added to the database. The target relation has exactly T tuples. Each target tuple is generated according to a randomly chosen clause. In this way we also need to add tuples to non-target relations to satisfy the clause. After all target tuples are generated, we add more tuples to non-target relations. For each non-target relation R, the number of tuples obeys exponential distribution with expectation T and is at least T_{min}. If R already has enough tuples, we leave it unchanged. Otherwise we randomly generate tuples and add them to R until it has enough tuples. We use "$Rx.Ty.Fz$" to represent a synthetic database with x relations, expected y tuples in each relation, and expected z foreign-keys in each relation.

For a multi-relational classification approach, we are most interested in its scalability w.r.t. the size of database schema, the number of tuples in each relation, and the number of joins involving each relation. Therefore, experiments are conducted on databases with different number of relations, different number of tuples in each relation, and different number of foreign-keys in each relation. In each experiment, the running time and accuracy of CrossMine, FOIL, and TILDE are compared.

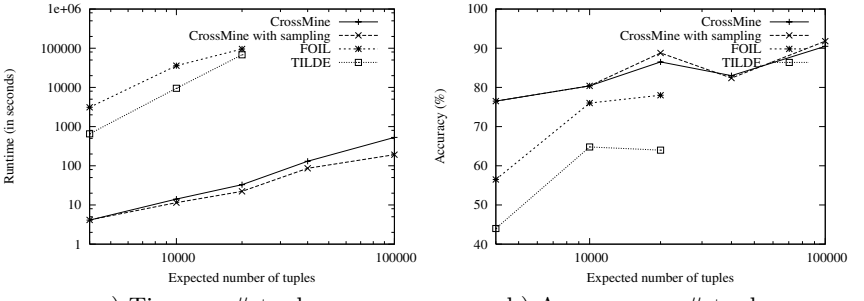
To test the scalability w.r.t. the number of relations, five databases are created with 10, 20, 50, 100, and 200 relations respectively. In each database, the expected number of tuples in each relation is 500 and the expected number of foreign-keys in each relation is 2.

Figure 9 (a) shows the running time of the three approaches. Ten-fold experiments are used in most tests, and the average running time of each fold is shown in the figure. If the running time of an algorithm is close to or greater than 10 hours, only the first fold is tested in our experiments. We stop an experiment if the running time is much greater than 10 hours. From the experimental results, one can see that CrossMine is thousands of times faster than FOIL and TILDE in most cases. Moreover, its running time is not affected much by the number of relations. FOIL and TILDE are not scalable with the number of relations. The



a) Time vs. #relation b) Accuracy vs. #relation

Fig. 9. Runtime and accuracy on R*.T500.F2



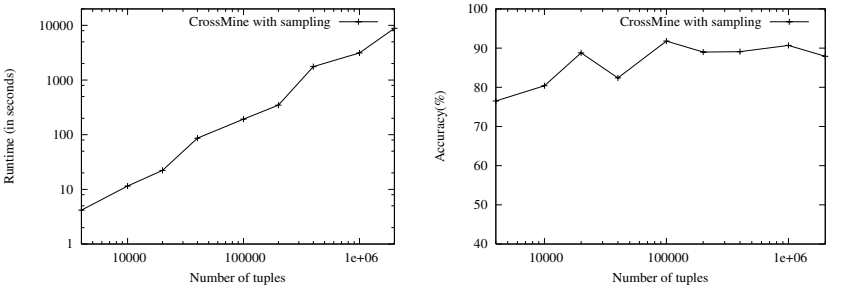
a) Time vs. # tuple b) Accuracy vs. # tuple

Fig. 10. Runtime and accuracy on R20.T*.F2

running time of FOIL increases 9.6 times when the number of relations increases from 10 to 50, whereas the running time of TILDE increases 17.3 times. The accuracy of the three approaches are shown in Figure 9 (b). One can see that CrossMine is more accurate than FOIL and TILDE.

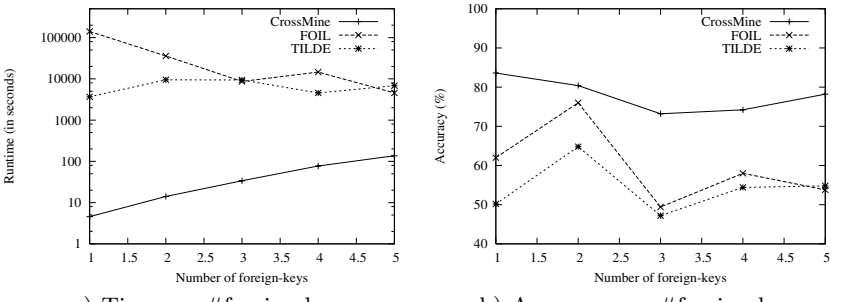
To test the scalability w.r.t. the number of tuples, five databases are created with the expected number of tuples in each relation being 200, 500, 1000, 2000, and 5000, respectively. There are twenty relations in each dataset, thus the expected number of tuples range from 4K to 100K. The expected number of foreign-keys in each relation is 2. In this experiment, the performance of CrossMine with sampling is also tested to show the effectiveness of sampling. Figure 10 (a) shows the running time of the four approaches.

One can see that CrossMine is more scalable than FOIL and TILDE. The running time of CrossMine increases 8 times when the number of tuples increases from 200 to 1000, while those of FOIL and TILDE increase 30.6 times and 104 times, respectively. With tuple sampling, CrossMine becomes more scalable (running time decreases to one third of non-sampling version when the number of tuples is 5000). The accuracy of the three approaches is shown in Figure 10 (b). CrossMine is more accurate than FOIL and TILDE, and the sampling method only slightly sacrifices the accuracy.



a) Time vs. # tuple b) Accuracy vs. # tuple

Fig. 11. Runtime and accuracy on large datasets



a) Time vs. #foreign-key b) Accuracy vs. #foreign-key

Fig. 12. Runtime and accuracy on R20.T500.F*

We also test CrossMine (with negative sampling) on large datasets to show its high scalability. We generate nine datasets with expected number of tuples in each relation from 200 to 100K. Since there are twenty relations in each dataset, the expected numbers of tuples range from 4K to 2M. The running time and accuracy of CrossMine are shown in Figure 11 (a) and (b). It can be seen that CrossMine is highly scalable for large datasets.

Finally, we test the scalability w.r.t. the number of foreign-keys. Again, five databases are created with the expected number of foreign-keys in each relation being 1 to 5. The number of relations is 20 and the expected number of tuples in each relation is 500. The running time of the three approaches are shown in Figure 12 (a) and the accuracy are shown in Figure 12 (b). One can see that CrossMine is not very scalable w.r.t. the number of foreign-keys, although it is still much more efficient than FOIL and TILDE. Fortunately, in most commercial databases the number of foreign-keys in each relation is quite limited. And CrossMine is very efficient when this number is not large.

7.2 Real Databases

Experiments are also conducted on two real databases to compare the efficiency and accuracy of CrossMine, FOIL and TILDE. The first database is the financial

Table 2. Performances on the financial database of PKDD CUP'99

Approach	Accuracy	Runtime
CrossMine w/o sampling	89.5%	20.8 sec
CrossMine with sampling	88.3%	16.8 sec
FOIL	74.0%	3338 sec
TILDE	81.3%	2429 sec

Table 3. Performances on the Mutagenesis database

Approach	Accuracy	Runtime
CrossMine	89.3%	2.57 sec
FOIL	79.7%	1.65 sec
TILDE	89.4%	25.6 sec

database used in PKDD CUP 1999. Its schema is shown in Figure 1. We modify the original database by shrinking the *Trans* relation which was extremely huge, and removing some positive tuples in the *Loan* relation to make the numbers of positive tuples and negative tuples more balanced. The final database contains eight relations and 75982 tuples in total. The Loan relation contains 324 positive tuples and 76 negative ones. The performances on this database is shown in Table 2. All three types of literals are considered in this experiment.

The second database is the Mutagenesis database, which is a frequently used ILP benchmark. It contains four relations and 15218 tuples. The target relation contains 188 tuples, in which 124 are positive and 64 are negative. The Mutagenesis database is pretty small and the sampling method has no influences to CrossMine. The performances is shown in Table 3.

From the experiments one can see that CrossMine achieves good accuracy and efficiency. It is much more efficient than traditional ILP approaches, especially on databases with complex schemas.

8 Discussions

In this paper it is assumed that the dataset can fit in main memory, so that random access can be performed on tuples in different relations. In some real applications the dataset cannot fit in main memory. Instead, the data are stored in a relational database in the secondary storage. However, this will not affect the scalability of CrossMine. In this section we show that all the operations of CrossMine can be performed efficiently on data stored on disks.

8.1 Tuple ID Propagation

Tuple ID propagation is the most basic operation of CrossMine. When data is in main memory, a set of tuple IDs associated with a relation R are stored in

a separate array. When data cannot fit in main memory, we can store a set of tuple IDs as an attribute of R. Since CrossMine limits the fan-out of tuple ID propagation (Section 4.3), the number of IDs associated with each tuple is limited, thus the IDs can be stored as a string of fixed or variable length.

In CrossMine, only joins between keys or foreign-keys are considered (Section 3.1). An index can be created for every key or foreign key. When propagating IDs from R_1 to R_2, only the tuple IDs and the two joined attributes are needed. If one of them can fit in main memory, this propagation can be done efficiently. Otherwise, a join operation can be performed between R_1 and R_2 to find joinable tuples and propagated IDs.

8.2 Evaluating Literals

Suppose tuple IDs have been propagated to a relation R, and the best literal on R need to be identified. If all attributes of R are categorical, then the numbers of positive and negative target tuples satisfying every literal can be calculated by one sequential scan on R. With this sequential scan, we can also generate simple statistics (sum, average, etc.) for every target tuple and every numerical attribute. The best aggregation literal can be found by these statistics. For a numerical attribute A, suppose a sorted index has been built on A. Then a sorted scan on A is needed to find the best literal on A. If this index and the tuple IDs can fit in main memory, this can be done efficiently.

9 Conclusions and Future Work

Multi-relational classification is an important issue in data mining and machine learning involving large, real databases. It can be widely used in many disciplines, such as financial decision making, medical research, and geographical applications. Many traditional ILP approaches are inefficient and unscalable for databases with complex schemas because they evaluate a huge number of clauses when selecting literals. In this paper we propose CrossMine, an efficient approach for multi-relational classification. It uses tuple ID propagation to reduce the computational cost dramatically, which makes CrossMine highly scalable w.r.t. the size of database schemas. In the process of building clauses, CrossMine performs search in wider space than traditional ILP approaches by considering up to three literals at a time. This enables CrossMine to identify better-quality literals and build more accurate clauses. Experiments show that CrossMine is highly efficient comparing with the traditional ILP approaches, and it achieves high accuracy. These features make it appropriate for multi-relational classification in real world databases.

There are several possible extensions to CrossMine. Although CrossMine searches a wider space to select better-quality clauses than most ILP approaches, it is still a greedy algorithm and searches only a small part of the whole search space. Moreover, it is interesting to study how to integration CrossMine methodology with other classification methods (such as SVM, Neural Networks, and k-nearest neighbors) in the multi-relational environment to achieve even better accuracy and/or scalability.

References

1. A. Appice, M. Ceci, and D. Malerba. Mining model trees: a multi-relational approach. In *Proc. 2003 Int. Conf. on Inductive Logic Programming*, Szeged, Hungary, Sept. 2003.
2. J. M. Aronis, F. J. Provost. Increasing the Efficiency of Data Mining Algorithms with Breadth-First Marker Propagation. In *Proc. 2003 Int. Conf. Knowledge Discovery and Data Mining*, Newport Beach, CA, 1997.
3. H. Blockeel, L. De Raedt, and J. Ramon. Top-down induction of logical decision trees. In *Proc. 1998 Int. Conf. Machine Learning*, Madison, WI, Aug. 1998.
4. H. Blockeel, L. De Raedt, N. Jacobs, and B. Demoen. Scaling up inductive logic programming by learning from interpretations. *Data Mining and Knowledge Discovery*, 3(1):59-93, 1999.
5. H. Blockeel, L. Dehaspe, B. Demoen, G. Janssens, J. Ramon, and H. Vandecasteele. Improving the efficiency of inductive logic programming through the use of query packs. *Journal of Artificial Intelligence Research*, 16:135-166, 2002.
6. C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–168, 1998.
7. P. Clark and R. Boswell. Rule induction with CN2: Some recent improvements. In *Proc. 1991 European Working Session on Learning*, pages 151–163, Porto, Portugal, Mar. 1991.
8. H. Garcia-Molina, J. D. Ullman, and J. Widom. *Database Systems: The Complete Book*. Prentice Hall, 2002.
9. J. Gehrke, R. Ramakrishnan, and V. Ganti. Rainforest: A framework for fast decision tree construction of large datasets. In *Proc. 1998 Int. Conf. Very Large Data Bases*, New York, NY, Aug. 1998.
10. N. Lavrac and S. Dzeroski. *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood, 1994.
11. T. M. Mitchell. *Machine Learning*. McGraw Hill, 1997.
12. S. Muggleton. *Inductive Logic Programming*. Academic Press, New York, NY, 1992.
13. S. Muggleton. Inverse entailment and progol. In *New Generation Computing, Special issue on Inductive Logic Programming*, 1995.
14. S. Muggleton and C. Feng. Efficient induction of logic programs. In *Proc. 1990 Conf. Algorithmic Learning Theory*, Tokyo, Japan, 1990.
15. J. Neville, D. Jensen, L. Friedland, and M. Hay. Learning Relational Probability Trees. *Proc. 2003 Int. Conf. Knowledge Discovery and Data Mining*, Washtington, DC, 2003.
16. A. Popescul, L. Ungar, S. Lawrence, and M. Pennock. Towards structural logistic regression: Combining relational and statistical learning. In *Proc. Multi-Relational Data Mining Workshop*, Alberta, Canada, 2002.
17. J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
18. J. R. Quinlan and R. M. Cameron-Jones. FOIL: A midterm report. In *Proc. 1993 European Conf. Machine Learning*, Vienna, Austria, 1993.
19. B. Taskar, E. Segal, and D. Koller. Probabilistic classification and clustering in relational data. In *Proc. 2001 Int. Joint Conf. Artificial Intelligence*, Seattle, WA, 2001.

Remarks on the Industrial Application of Inductive Database Technologies

Kimmo Hätönen, Mika Klemettinen, and Markus Miettinen

Nokia Research Center, P.O.Box 407, FIN-00045 Nokia Group, Finland
{kimmo.hatonen, mika.klemettinen, markus.miettinen}@nokia.com

Abstract. The research in the area of inductive databases has taken huge steps forward during recent years. Various results have been produced and published by many groups all around the world. The next big challenge for the research community together with industry is to integrate these results to the existing systems and to enhance current solutions to better answer to the real world challenges. In this article we give an industrial perspective for exploring, validating and exploiting new techniques like inductive databases. We discuss various requirements that industrial processes set for the methods and tools. Based on our own ten year experience in the field we also study reasons and background for why some systems are taken into use and some are not.

1 Introduction

The research in the area of inductive databases (IDB) has taken considerable steps forward during recent years. Various results have been produced and published by many groups all around the world. For example, in recently finished EU funded research project *cInQ* (consortium on discovering knowledge with Inductive Queries, IST-2000-26469) the project participants have published tens of publications on the application of IDB technologies on different domains like medical science, bioinformatics, telecommunications and web mining.

One of the challenges that follows successful basic research is how the research community together with industry could transfer the created compentence to be applied in the industry. Achieved results should be evaluated and integrated to existing systems to enhance current solutions to better answer to the real world challenges. This should be done while the basic research and method development are continuing.

There are a couple of basic questions that must be considered here. First of all does the research answer the needs of the community, industry and the markets? Are the studied problems interesting from applied research point of view and do they answer practical needs of industry? Another issue is whether the presented results create added value for the current solutions in such a way that it is worth for the industry to take the risk to implement them as a part of existing solutions. Will there be enough markets for the new solutions and will they enhance the existing solutions in such a way that it will bring leading edge benefits for the applier?

J.-F. Boulicaut et al. (Eds.): Constraint-Based Mining, LNAI 3848, pp. 196–215, 2005.

If and when the answers to those questions are in the affirmative, what should be done to convince the industry to apply presented results? What makes an IDB algorithm or a system succeed in practice?

Answering these questions starts from the needs of an applier. It is essential to understand what kind of issues affect the application, in what kind of environment the algorithm or system is used, who are using it, what it is used for and what kind of legacy solutions or other resources are available to the users. In this article we describe and study the conditions in which different types of telecommunication applications are used. Based on our ten year experience in the field we identify a set of requirements for data mining (DM) and IDB applications. They affect the usability and interestingness of IDB research results from the industrial perspective.

In section 2 we discuss requirements that are introduced by the telecommunication industry set up. In section 3 we present some scenarios about decision making situations in telecommunication network operation, and in section 4 we show how some research results have answered to specific tasks and their requirements. In section 5 we line up the results of this paper with earlier studies made with expert systems and show that actually in both application genres - when applied to the same industrial domain - similar types of hypotheses apply.

2 Framework for Exploitation

The perspective of an academic researcher differs somewhat from the view point of an industrial researcher. While the academic researcher is often mainly interested in doing basic research and proving that generally speaking something is possible and holds, the industrial researcher is usually tied to the real world and its particular problems and their peculiarities. Very seldom an industrial researcher can start to work on a new topic without any existing legacy systems that will continue to be used and to which the new methods must offer backward compatibility. The industrial world sets also many other requirements for exploiting new methods. For example, the new techniques have to be in line with the selected product strategy, there must be enough resources to implement the new methods, there must be customer demand – either explicit or anticipated – for the application of the methods, and so on, to name a few.

In order to illustrate these demands and where they originate, we present a model about a tool provider and a user of the provided tools in the telecommunciations domain. For modelling these two worlds we have chosen to use Leavitt's diamond model [12]. It describes organisations as four interrelated components: tasks, technology, persons and structure, where structure represents the organisation as well as external stakeholders such as competitors. The interdependence between the different components of the model is strong. When one component changes it also influences the others. For example, changes in technology affect the way in which individuals relate themselves to the tasks they are responsible for and to the organisational structure. The model has been used as an analysis framework for, e.g., information systems [11], information system personnel and

their roles [15], telecommuting [4], and telecommunication network planning tool implementation [16].

For our study we separate the model into two views [16]: a developer view and a user view. The developer view illustrates a method provider or developer organisation who selects whether DM or IDB techniques are going to be used in tools. In the user view, a user organisation can be a telecommunication network operator or an IT department of an enterprise. The views are presented in figures 1 and 2.

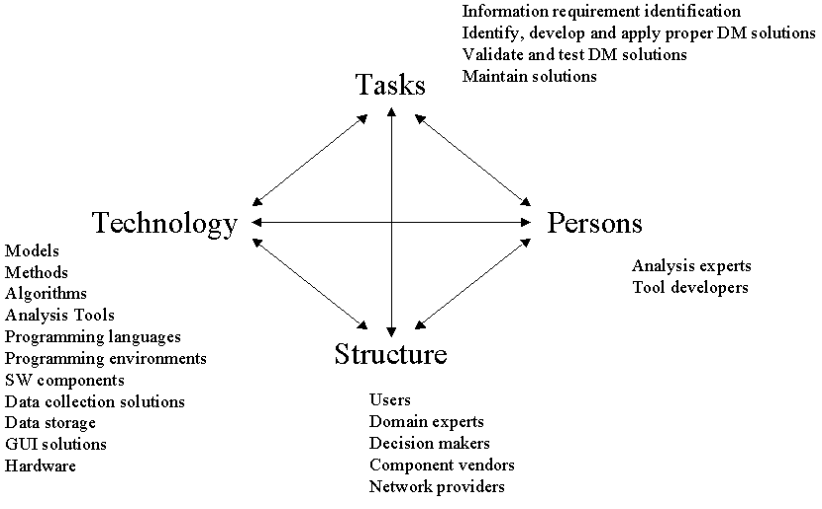


Fig. 1. Interactions between data mining application issues from the developer perspective

Figure 1 shows the application of Leavitt's diamond model to development of the DM and IDB domain from the developer perspective. The tasks consist of requirements management, test data acquisition, method development, method and tool verification and tool maintenance. The directly involved persons are analysis experts and tool developers. Technology consists of models, methods, algorithms, DM tools and environments, programming languages and environments, software components, data collection and storage solutions, legacy data management and reporting solutions, GUI solutions and, finally, of the analysed network and hardware. The structure contains tool users, domain experts, decision makers, and software tool, component and platform vendors.

Figure 2 shows the model from the data mining and IDB applications user perspective. The essential tasks – basically making decisions in different types of situations – are related to network operation and development. Such tasks include, for example, configuring a new network segment, optimising the services in some cells or fixing acute and critical faults. The technology component consists of numerous items of the application domain and the monitored network, its

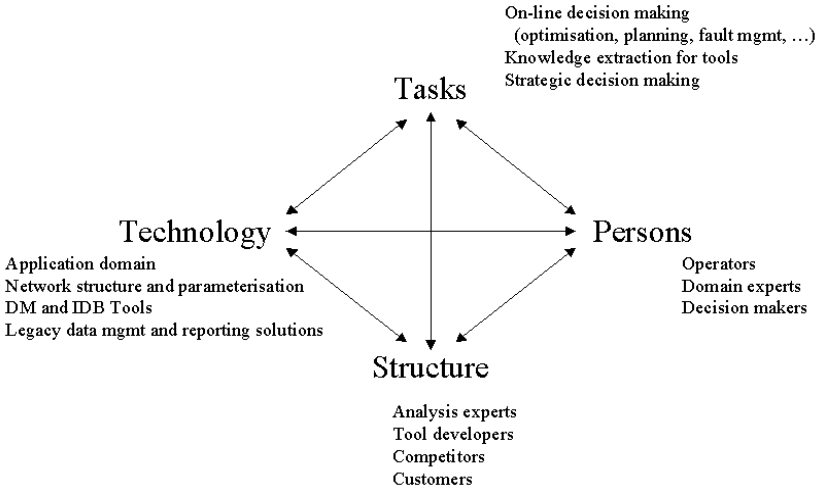


Fig. 2. Interactions between data mining application issues from the user perspective

structure and parameterisation. The topic of this article, DM and IDB methods, is seen as technology embedded in domain specific tools from the perspective of the engineers and other operator staff using them. From their perspective, these tools should be integrated to the legacy data management and reporting solutions that still offer the major functionality of the monitoring system. From the user perspective, the structure contains analysis experts, tool developers, customers and competitors.

These views are interdependent. For example, the technology component of the user view is linked with the task component of the developer view as the developed methods and tools are the key results that are used by the users. Also the analysis experts and tool developers of the persons component in the developer view can be modeled to be in the structure component of the user view, and vice versa.

The interdependence between the two views is a reason for conflicts since the needs of developers and users are contradicting [16]. For example, from the user point of view the tools should make the execution of simple network analysis tasks very fast whereas from the developer point of view the tools should be easy to implement and maintain.

The successful exploitation of DM and IDB tools requires understanding of the requirements set for the tools from the user point of view. If those requirements are not met, then the users very easily just do not use the new technology but stick with the existing solutions and their direct enhancements.

When users are selecting their tools, they set requirements for the possible candidates. For applications in industrial production these requirements are quite strict and the technological excellence is only one aspect in the selection process. Other requirements are set for understandability, integrability, effective-

ness, continuation of development, guaranteed support and maintenance, and so on. If these requirements are not met by the tool and its seller, the method or tool might be abandoned without a second look at its technological achievements.

Below we point out some connections in the user view that affect the acceptability of new technologies. These are, e.g., connections between

- technology and persons,
- technology and tasks, and
- technology and structure.

2.1 Technology and Persons

Persons who use DM and IDB in user organisations can be, e.g., technicians, top level domain experts or top managers with a lot of experience in the business. A common factor among all of them is that they are typically skilled in what they are doing, namely in running telecommunications networks. They probably don't know too much about statistics or data mining techniques.

This sets a requirement for any proposed tool or method: it must provide results using the terminology and semantics of the application domain. For example, pure statistical figures without a good explanation about what causes them and what they tell to an analyst are not necessarily understandable for a domain expert. In other words, the tool provider has to attach a semantic interpretation in application domain terms to each used statistical figure.

As observed, experts are willing to assist in the development and adopt a planning tool if it provides immediate and accurate results to them already during the development period [16]. This is most probably true also with any DM or IDB tool. This is essential, since without the domain knowledge that the experts provide, the developer is not able to do the needed semantic localisation of the tool to the application domain. If the method is easy to understand and provides accurate results, experts will use it to assist them in their daily tasks, to play around with it and provide the semantic connection by themselves. This will require also a user friendly and usable user interface for the method.

2.2 Technology and Tasks

DM and IDB tools. In network operation there are plenty of different tasks with different time constraints. The most urgent task is to fix critical faults that disturb communications of a large number of mobile phones. These faults are monitored and, if detected, analysed online 24 hours per day. Less critical faults are analysed in priority order based on daily fault and performance reports. Every now and then the operator personnel go through the whole network in order to detect cells that are not working optimally.

For all of the above mentioned analysis tasks the operator has plenty of monitoring and reporting tools that follow up different parts and aspects of the network. Any DM or IDB tool is an enhancement for the existing tools. They should assist the persons in their tasks, which they are used to perform based on the information provided by the existing tools. These tools are typically tightly

linked to different management applications, with which the operators tune and fix the network remotely. This setup requires proper input and output interfaces to the new enhancements, which have to be integrated to the existing infrastructure.

Network structure. The structure and parameterisation of the network evolves constantly. Quite a large number of cell configurations – e.g., one percent out of thousands of cells – are updated on a weekly basis. This sets a challenge for the personnel: the so called normal or optimal value ranges of several indicators derived from a cell group or a cell are constantly changing. These changes have to be identified from the series of measurement values and verified against general domain knowledge.

On-line exploration vs. offline DM. In telecommunications there are plenty of different decision making situations, which have different time frames and characteristics. The role of DM and IDB applications in the domain is to support these decision tasks.

The shortest decision making loops have been automated. There are closed control loops that monitor one or more indicator time series and adjust process parameters as a response to the incoming data. For these control functions the DM and IDB applications can provide information about the effects of different traffic and configuration combinations. This information can be extracted off-line either from a history data set or a simulated laboratory data set.

Another natural target for support are strategic decisions, which are based on data and information in various formats coming from several different sources. Analysis of this information closely resembles a classical data mining process, where also analysis experts are involved.

Probably the hardest target for decision support are the tactical and short term strategic decisions, where the time to do the decision is limited, the problem occurs either very seldom or is totally new and for which no analysis expert is available. In these tasks the DM and IDB tools have to be so easy to use that a domain expert is able to quickly extract needed information by himself. There is no room for iteration or full scale data exploration, but in spite of that the analysis has to be well focused and straightforward to use.

2.3 Technology and Structure

Analysis experts. One of the most critical differences between developer and user views is in the role of analysis experts. They are DM and IDB experts that develop used methods. In the developer view they are in the persons component. This means that they are available inside the organisation and actively taking part in different tasks.

In the user view, analysis experts are in the structure component. They are not part of the organisation using the tools and methods but rather externals, probably personnel of a tool provider or some consulting company. This makes them temporary options for any continuous analysis task. They might be used for

giving training in the roll-out phase of a tool, but later it is usually an expensive option to use constant consultations.

Competitors. A basis for all the user organisation acquisitions is the amount of expected utility. The utility can be in a form of more effective operations and cost savings, improved product quality, new and impressive services and so on. If it is possible to manage the business with the old existing infrastructure and the expected utility that could be gained with the new solutions are less than what is required to update the old system and maintain the new one, then the acquisition will not be made. For example, if updating the legacy solution would require re-programming some of the central building blocks of the existing system and thus re-testing and debugging of all the solutions depending on it, the expected utility gain has to be very large before the organisation is willing to consider taking the risk of updating the system.

One element in the structure component – competitors – are the source for the need to upgrade operation solutions. If competitors are able to achieve lower maintenance costs by using more efficient analysis tools, this probably drives the organisation towards considering to use them. Otherwise, if their running costs are higher than those of the competitors, it will mean losing profits in the longer run.

3 Decision Making Scenarios in Telecommunications

In telecommunication business there are plenty of different types of decision making situations. They vary from very fast optimisation decisions to large strategic decisions about the infrastructure and business opportunities. The common aspect for all of these is that there are more data available than can be analysed. Therefore, DM and IDB tools provide very promising alternatives for different decision making tasks. In this section we give examples of tasks that were included in the tasks component of the user view in section 2.

3.1 Knowledge Extraction

DM and IDB methods are used in the knowledge extraction task to discover knowledge that can be either encoded into the applications running the short term control loops or integrated to the knowledge base of expert systems. As an example of a knowledge extraction task we introduce a task of finding rules and patterns for an alarm correlation engine. These tasks are typically executed offline and the execution closely resembles the classical knowledge extraction process [5].

Rule and pattern extraction. In a network management system there are plenty of tools that use some sort of rule, pattern, or expression base to filter incoming data and/or to identify different conditions in the network. These include, e.g., spam filters, alarm correlators, expert systems identifying faults and

so on. Knowledge bases of these applications require constant maintenance and updating.

New rules can be found by analysing the collected data and identifying possible signs of searched conditions. When there appears to be a new kind of problem that can not be dealt with the knowledge already included in the system, then new rules have to be defined. Also, when such a system is for the first time installed into the network, a rulebase has to be created and localised for that particular network. This creation can be done by analysing the recent history data and by identifying possible extensions to existing knowledge.

The extracted rules can be either based on propositional logic (e.g., rules containing only event types) or predicate logic. The latter type of rules describe dependencies between parameterised event types, so that an operator can see not only event type sequences, but also what kind of parameter combinations occur in the event sequences. These sequential patterns and rules derived from them can be very helpful, since they represent real causalities in the network's behaviour.

3.2 On-Line Support for Tactical Decisions

Operators monitor the network state and events in the network elements to ensure integrity and security in a telecom network. This monitoring is based on data collected from the network elements. Suitable data for this purpose are, e.g., different security application logs, and operating system and application logs from the various nodes in the network.

Decisions made based on the monitoring are typically so called tactical decisions. Their objective is to optimise performance of the network by tuning the configuration or fixing different problems around the network. The decisions are made on-line. This is to say that there are quite strict time limits for the tasks. Typically analysed situations are searches for cells with lowered performance and when such cells are found, analysis of their performance and surroundings in order to identify the root cause for the trouble.

Performing routine searches for deviating behaviour. An operator analyses the event or alarm log contents with the goal of finding indications of system failures or security breaches. The analysis software presents him sufficient information from which he recognises deviating behaviour and identifies the relevant factors related to this deviation. Based on this information, more thorough analysis of the problem can be undertaken.

In practice, the operator browses the log data and views analysis reports produced by the analysis system. An analysis report shows frequently occurring log entry patterns, thus quickly giving an overview of the most common activities in the network element. Individual occurrences of such log rows that belong to some log entry pattern are removed from the analysis view. This reduces the amount of shown data and makes it thus easier for the network monitoring officer to find seldomly occurring, or unique, events in the network.

Investigating a discovered malfunction in the network. Every now and then there occurs a malfunction in the network. Usually signs of such malfunction are detected by, e.g., the alarm system that generates a set of alarms describing the situation. The task is to find out reasons for the alarms. The operator investigates information that is related to the alarms and alarming elements. He views the overall picture of the situation and focuses on interesting details in that. To do this, the operator looks at views of logs that are related to the alarming element. He analyses the alarm sequences and switches, if necessary, to a view, where the details of all log entries coming from the element are visible. The role of DM and IDB tools in this kind of task is to filter redundant event sequences and to show connections between related alarms.

3.3 Off-Line Support for Strategic Decisions

In strategic decision making, system performance data, expected development scenarios in customer behaviour and external factors are combined with the risk policy and company values. The management of the company selects a probable scenario of the future as a basis of decisions and defines required actions. The data used for the scenario creation – especially if coming from external sources – can be in various formats and requires interpretation for the DM methods. Strategic decision tasks typically require either the same kind of on-line support as tactical decision making described above, or they can be more like the knowledge extraction tasks presented in section 3.1.

Churn analysis. An expert analyses data about customers who have recently changed their operator away from the company. He tries to understand why they left and if there was anything the company could have done better to keep the customers. He pays attention especially to those customers and customer groups who were among the most profitable ones. An outcome of the effort can be a list of suggestions for improvements in marketing, pricing, and investment policies. The analysis can be based on the traffic profiles of the users as well as events preceding the change.

3.4 Data Management

IDB methods provide potential enhancements for different data management tasks, for example, for semantic compression methods.

Event logs produced by different network elements and functions have to be stored for a certain period of time. Security-related logs, for example, might be needed several months or even years after their creation in order to properly analyse security breaches that took place long time ago. Due to the huge volume of the data it has to be compressed for storage. Such a form of compression should be used that the data is all the time queryable without prior decompression.

This kind of semantic compression can be done by discovering patterns of frequently repeating items from the log data and using these patterns as a codebook for compressing the data. When log data arrive to be deposited in the log data

repository, they are mined for such frequent patterns. Some of these frequent patterns (i.e. such that have the most suitable statistical properties) are selected to be used in building a codebook that is used to encode the log data. When the most suitable attribute patterns have been found, the log data are compressed using the created codebook and stored in the repository in compressed format.

4 User Tasks and Solution Examples

The *cInQ* project has developed several techniques that can be used in assisting telecommunication network operation tasks. Most of them are modifications and enhancements for the calculation of frequent patterns and their derivatives. In this section we discuss how they can be used to handle log analysis problems.

The telecommunication networks produce large amounts of different types of events that are logged in central monitoring points. These events include log entries reflecting normal operation of the network as well as alarm information about faults and problems that occur.

The log files may be very large. During one day, millions of lines might be accumulated into a log file. A solution to browse the data is either to search for patterns that are known to be interesting with high probability or to filter out patterns that most probably are uninteresting. A system can assist in this but the evaluation of interestingness is left to an expert. To be able to make the evaluation an expert has to check the found log entries. Often he has to return to the original log file and iteratively check all probably interesting entries and their surroundings.

In the log analysis, there are at least two kinds of problems present. In the pool of discovered information - among the interesting pieces - there are facts, e.g., patterns or relations, that have been generated between items or patterns of items that can not have any interdependence with each other, and large amounts of (often very similar) proper facts that are of no or very low interest. These two problem areas have characterised the work in the telecommunications application area during the cInQ project.

We show how frequent patterns and their derivatives can be used to assist in different types of telecom operation tasks and what kind of issues have to be considered in applying these techniques. We start in section 4.1 with tasks concerning knowledge extraction either for different tools or off-line decision making. Both of these tasks can be assisted by data mining projects following the classical knowledge extraction process [5]. Then we discuss on-line support for tactical decision making and special system functionalities, which require good usability and understandability or even automation from the tools used.

4.1 Knowledge Extraction for Tools and Off-Line Tasks

Frequent patterns are value or event combinations that occur often together in the data. They provide information, which can be used to find rules or patterns of correlated or otherwise searched event combinations. Thus they are useful in

knowledge discovery tasks described in sections 3.1 and 3.3. A pattern is called frequent if the number of its occurrences in the data is larger than a given threshold. The patterns capture common value combinations that occur in the logs and often they cover also most of the volume of the data. A formal definition of frequent patterns can be found, e.g., in [6].

The execution of algorithms for finding frequent patterns easily becomes untractable. Algorithms try to overcome this by using effective methods to prune and limit the search space. Unfortunately, however, the log data contain a lot of redundant value combinations that make most of these algorithms reach their limits very soon. This happens especially when the interesting patterns are not those that occur most often in the data.

Frequent patterns provide a representation of the data for several methods that extract different types of rules from the data set. These rules can then be integrated to existing knowledge bases in industrial systems. Such formalisms are, e.g., association rules and structural rules.

Association rules [1,2] describe local correlations between values in the database. Their direct derivative, episode rules (see, e.g. [14]), find frequent event type combinations. The biggest problem with algorithms searching for rules is that they easily provide an overwhelming amout of them. Different types of methods, like statistical descriptors or interactive browsing environments have been suggested in order to simplify identification of interesting rules.

Another form of rules that can be found based on frequent patterns are so called structural rules. For finding structural rules in data streams, modified sequence mining methods can be employed. One approach is the algorithm *MineSeqLog* [13]. These algorithms are able to find not only propositions between event types like episode rules but also the parameter combinations attached to the event types. Also the usability of these rules suffers from the easily overwhelming size of the result.

When we consider to use DM and IDB tools in telecommunication network operation, we have to take into account what is available for the end user and what is not. If we look at figure 2 in section 2, we can see that the technology as well as persons components consist of strong competence on the domain. There are plenty of knowledge and data about the network and its structure as well as data collected from the network. How would it be possible to take advantage of them? One possibility is to use the available structural information to steer the analysis and to filter the results. These results, achieved by the DM and IDB methods, can then be verified with available legacy network inspection and analysis solutions.

Domain structures in filtering irrelevant frequent patterns. In the network there are all the time plenty of independent processes going on. These processes emit alarms, when they get disturbed by faults. It often happens that many independent processes get simultaneously affected by a fault and they all start to alarm, not necessarily about the fault itself, but about its secondary reflections. Thus, generated alarms and log entries actually carry second-hand

information about the incident. They do not necessarily identify the primary fault at all.

Alarms that network processes emit are collected to centralised monitoring points. This makes the analysis even more difficult, because at each monitoring point, the symptoms and reflections of separated problems are merged into one information flow. The combined flow also contains entries caused by normal maintenance operations or by natural phenomena like thunderstorms.

A starting point for a network analyst in a fault condition is always localisation and isolation of the fault, i.e., finding the area where the problem is located and identifying all network elements that are affected by the fault. Localisation and isolation is based on the assumption that it is probable that the fault itself is local although its reflections are widespread. In this situation alarms coming from the same network element or its direct neighbours are related to one reflection of the fault. After the localisation has been done it is easier to do the actual identification of the fault.

Episode and association rule based techniques [2,14,9,10] have been used in semi-automatic knowledge acquisition from alarm data in order to collect the required knowledge for knowledge based systems like alarm correlators. Given such rules holding in an alarm database, a fault management expert is able to verify whether the rules are useful or not. Some of the rules may reflect known causal connections, some may be irrelevant, while some rules give new insight to the behaviour of the network elements. Selected rules can be used as a basis for correlation patterns for alarm correlation systems. However, if no constraints are applied, the discovered result set of, say, episode rules might become huge and contain mostly trivial, uninteresting or even impossible rules.

The basic methods for finding episodes use event distances in time but ignore all the other knowledge about the domain. There is, however, plenty of other useful domain knowledge, e.g., topological information, available from telecommunications networks [8]. Also different taxonomies and definitions of control hierarchy and data flows between objects are usually available. This background knowledge can be used as a basis for domain specific *distance measures*. These distance measures can then be used to prune out accidental event occurences that are caused by simultaneous but independent phenomena in the network.

Distance measures. In a general case a data set can be seen as a collection of events so that each event has attached properties, which include event type and its occurrence time. Traditionally event time and type have been seen as more important than other properties such as cancellation time and severity of the event. However, this is not necessarily always the case. Any one of the properties can be used as a target for pattern mining. Depending on the application and the data set, some other properties than traditional time and type might be even more informative.

In earlier analyses the time has been used as the only property that separates possible event patterns from each other. In our research, also domain structures have been introduced for the same purpose [8]. In general, there might be several different types of constraining distance measures that could be used. The most

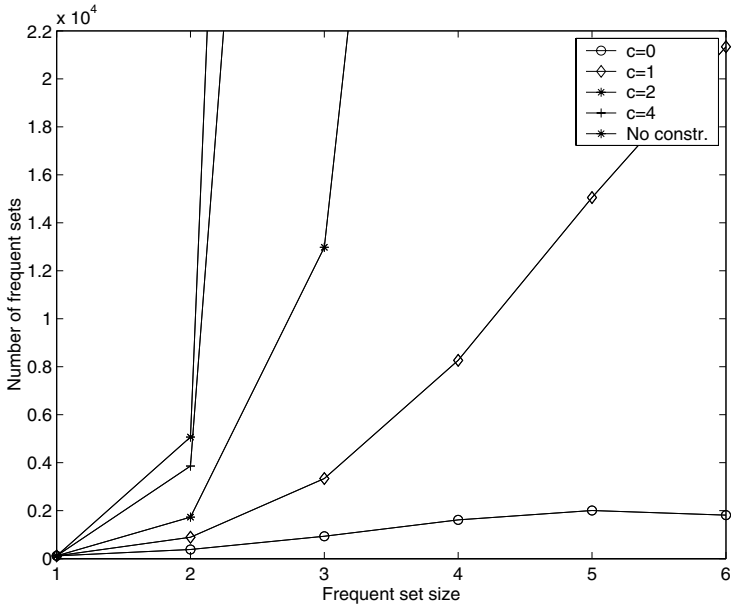


Fig. 3. Number of frequent sets with different values of constraining topological distance

important ones are property distances like distance in time, domain distances, and characteristics distances like the frequency of the type in the data set. The fourth type of constraints are the ones that are deduced from the previous mining results or that are set by the user, in order to reduce the amount of results and to focus on the most interesting phenomena.

Property distances are distance measures that are defined using event properties. Such a property can be, e.g., beginning or cancellation time of an event. These can be computed without seeing anything else than a data set or part of it. The distance between beginning or cancellation times of two events might be computed without knowing anything else about the domain.

Domain distances are defined by using additional domain knowledge, e.g., a control structure of a telecommunication network. In order to be able to compute distances over these structures, additional information of the domain in a form of a model is needed.

Characteristics distance is defined by the local statistical characteristics that can be computed from the data set. For example, frequencies of event types or average activity times of their instances can differ from each other so much that it is obvious that event types can not be related to each other. Another such a characteristic that can separate event types, is the distribution over time. An event type' s occurrences can be evenly distributed all over the data set from its beginning to its end. On the other hand, another event type might be as numerous in the data but it might be concentrated in a short peak.

Using of this kind of constraints that are based on knowledge available at the user organisation, improves quality of results of DM tasks. For example, according to our studies [8], as has been depicted in figure 3, introducing these constraints and domain specific distance measures reduced the size of the resulting set of association and episode rules. The average intrestingness of a single rule was also increased because purely accidental rules were not included in the result. Figure 3 shows how the number of frequent sets of different sizes varies, when the *distance constraint* c is changed. The distance constraint defines how many steps over the network topology there can be between sources of events so that events can be included in an occurrence of a frequent episode. When $c = 0$, then the events have to come from the same element. As can be seen, if the constraint is loosened the number of sets soon becomes overwhelming.

4.2 On-Line Support for Tactical Decision Making

In on-line support tasks described in section 3.2 the domain expert is alone. Tasks emerge constantly and there is no time for consultation nor money to hire analysis experts to assist the domain experts all the time in their daily routines. There are some network problems that repeat every now and then and in which a network expert can easily identify the problem and find the reason for it. However, there are plenty of situations in which the problem or its symptoms are practically unknown and where the cause is hidden somewhere in the network. Especially in these situations the tools that are available for the analysis should be able to parse and structure information from the data in such a way that a domain expert is able to find the reason for the trouble.

If we again think about the strengths and resources that a telecommunication operator organisation has for these situations, we find out that the domain knowledge, data about the network structure and its performance measurements and logs are the assets to be used. As was mentioned above, the organisations hardly have deep knowledge on DM or IDB methods. Therefore, the tools and methods have to be simple and usable. As the time requirements are strict, the tools have to be such that no or only little iteration is needed.

These kind of requirements set hard limitations for the developers. As the situations and problems, where the tools are needed, can be diverse, the information requirement identification is hard. The tools have to provide understandable results for a domain expert in different situations and a domain expert has to be able to focus the investigation on the essential information.

The requirements lead to solutions that are general in a way that the same method can be applied to data from multiple sources without big modifications in parameterisation. The user can then focus the analysis by selecting the data sources in such a way that the appropriate information is included in the data. The number of methods must not be large and they must be fairly simple to use and understand, since the user training for each method has to be provided for the domain experts. The methods have to provide information by using the same concepts and terminology that domain experts are using in their work. If the methods can extract such information out of the data and provide linkage

to legacy solutions that can be used to verify the findings, then the experts are most probably willing to accept the new tools to their daily use.

Comprehensive log compression for on-line log analysis support. As was mentioned above, frequent patterns capture the common value combinations that occur in the logs and often contain most of the volume of the data. A set of frequent patterns can be condensed furthermore by means of, e.g., closed frequent itemsets [17,3]. Closed sets form natural inclusion graphs between different frequent sets. This type of representation is quite understandable for an expert and can be used to create hierarchical views. These condensed representations can be extracted directly also from highly correlated and/or dense data, i.e., in contexts, where the approaches that compute the whole collection of frequent patterns are intractable [17,3,20,18]. The condensed representations can also be used to regenerate efficiently the whole collection of frequent patterns, possibly partially and on the fly.

Comprehensive Log Compression (CLC) [6] is a method based on the computation of a condensed representation of frequent patterns. We use this representation as an entry point to the data. The method provides a way to dynamically characterise and combine log data entries before they are shown to a human observer. It finds frequently occurring patterns from dense log data and links patterns to the data as a data directory. It is also possible to separate recurring data and analyse it separately. In many cases, this reduces the amount of data needed to be evaluated by an expert to a fraction of the original volume.

This type of representation is general w.r.t. different log types. Frequent patterns can be generated from most of the logs that have structure and contain repeating symbolic values in their fields. The CLC method summarises the most frequent value combinations in entries. This gives either a human expert or computationally more intensive algorithms a chance to continue with data that doesn't contain too common and trivial entries. Based on our experience with real-life log data, e.g., large application and firewall logs, the original data set of tens of thousands of rows can often be represented by just a couple of identified patterns and the exceptions not matching these patterns.

In figure 4 we show two days of firewall log data. The number of frequent sets computed with different threshold values varies in both examples around 10 000. We were able to reduce set of frequent sets into a set of about one hundred closed sets. From these closed sets we were able to select about 10 that covered more than 98% of lines in each data set. In the first data set (*Day 1*) there were 5 358 lines and in the second set (*Day 2*) there were 15 588 lines. Please, note that the scale of the y-axis is logarithmic.

The CLC-method extracts metainformation from the log transactions and uses it to summarise redundant value combinations without losing any essential information. The method first analyses the log data and searches for *closed itemsets*. When linked with the transactions, which support them, these sets and their inclusion graph can be used as navigational links to the dataset.

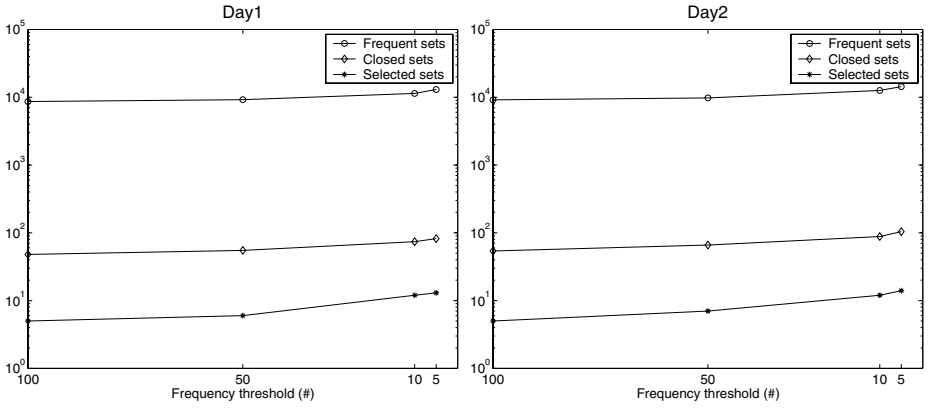


Fig. 4. Number of different types of results derived from two days of firewall log data

4.3 IDB for Data Management

DM and IDB methods can be integrated also to the lower levels of the analysis systems. This is actually one of the targets that have been set for the IDB research. The results can be used to speed up and simplify further analysis and minimise the storage space needed for the data and intermediate results. Requirements that can be set for this kind of integrated functionalities are, however, quite strict. The integrated functions have to work in feasible time and space with all kinds of available data, they have to be well tested, they have to work automatically without human operation, and they must not interfere with the legacy solutions that are used simultaneously.

Queryable log compression for data management. Closed sets can further be used to create so called *condensation formulae*. These formulae are used to identify and remove repetitive value combinations from log entries represented in database as transactions. The condensation formulae are stored together with the remaining parts of the data. If the original log file or its representation as a database table is needed, the formulae can be used to restore it completely – no information will be lost. This compression method is called *Queryable Log Compression* (QLC) [7]. It is an enhancement of the CLC method. In the QLC the data are compressed in such a way that it is possible to evaluate queries on the compressed data without a preceding decompression operation.

5 Discussion

In this article we have discussed several different aspects that affect the acceptance of advanced research results to industrial applications. Probably the most important thing is to understand the difference between the user view and the researcher/developer view: how they differ and what the requirements are that

users have to set for any new technology, method or system before it can be accepted.

During the *cInQ* project we made a couple of observations about the factors that help academic results to mature and to realise in products. These observations are related to research target setting, relations between technology and persons, and relations between structure and persons components in the user view.

In setting research objectives, challenging targets are needed to motivate researchers in their work. These targets are some technology generations ahead of the current solutions used in industrial applications. To overcome this gap between legacy systems and technologies developed as research results is a challenge. Because of this it can happen that the research targets are set high, researchers achieve almost the targets and the industry is able to adopt only basic steps leading to the new results. However, even these basic steps might be valuable for the industry.

As was found also in research on planning tool development [16], it is vital for any developed network analysis or management tool to provide immediate results for domain experts. This observation can be made also from our research history. If such results exists, and if it is easy for the domain expert to use the tool and to interpret its results, he is willing to spend his time on playing around with the tool and provide needed domain knowledge and ideas to the developers. Without this kind of knowledge sources, academic and industrial reseachers will be left out in the cold to play with demonstration data sets without a real connection to the application domain and its peculiarities.

Another observation that relates to the connection between the technology and persons components in the user view, was that the phase results of the research, e.g., demonstration prototypes have to be such that they are fairly easy to interface to the existing infrastructure. This is important because otherwise the users are not able to test results in their own environment with their own data, which is needed to create confidence in the new methods. If this is not done, the users are likely to abandon the results with arguments like "Looks nice but our data is much too complex!"

Close connection to existing systems helps researchers also to understand where the actual problems are. This is important especially in understanding what kind of data will be available and what their properties concerning volume, quality and frequency are. This might be used in demonstrating the benefits of the new technology. For example, the starting situation and its problems can be used to show the starting point of the research. It is then possible to create a contrast against this situation and show how new methods and ways of doing things improve the situation remarkably.

One requirement that relates to the structure component of the user view, is that in the user environment people know a lot about the domain and its operation. They might be totally ignorant with respect to DM or IDB techniques. For them the most important objective is to get the network cells working properly. If tools are too difficult to use or they require constant adjustment or multiple

iterations they will not be used if the expected results won't be transcendent when compared to results that can be acieved with conventional tools.

6 Conclusions

From an industrial perspective, basic research on new scientific methods becomes concrete through new innovations and evidence on how these new innovations improve the state of the art methods currently exploited in the companies. To prove the advantages of innovations, notable efforts should be spent so that the evaluation is not only done superficially, but that the methods and concepts are understood in a way that can lead to dissemination within the company – and beyond.

Let's take as a concrete example the cInQ EU project the work of which this paper refers to. During the first year of the cInQ project, the research concentrated on new innovations (algorithms, concepts, theories, etc.). The second year concentrated more on experiments and evaluation of the methods developed. For example, we made in-depth tests with different closed set algorithms in order to evaluate their performance compared to the earlier solutions. The motivation behind the tests was to show that the algorithms were faster than the existing ones and that the limits of tractable computation could be pushed forward.

In the spirit of the planned growing trend during the second cInQ year, the last year of the cInQ project concentrated on application, evaluation and dissemination of the cInQ project methods and results. During the third year we took up the basic questions presented already in the introduction: "Does the research answer the needs of the community, industry and/or the markets?" and "Do the new algorithms and research results bring added value w.r.t. the available and existing solutions?". This was because theoretically nice properties do not necessarily turn into exploitable solutions. How to convince research community and industry that IDB and condensed representations are useful (especially while the research area is still young and evolving), and to ensure successful continuation for the work?

From an industrial and exploitation perspective, the results of the cInQ project were promising. For example, biologists received new information from genes thanks to algorithms and methods developed; e.g., an original technique based on Galois connections that processes the transposed matrices while computing the frequent sets of genes [19]. Likewise, we were able to increase in-house competences and received also concrete evidence – based on the experiments and evaluations performed – on the practical usefulness of both constraint-based mining [8] and query-based approach (*Comprehensive Log Compression* (CLC) [6] and *Queryable Lossless Log Database Compression* (QLC) [7] methods building upon the *closed itemsets* [17,3] approach).

As the positive experiences from the cInQ project show, through fruitful combination of innovation and evaluation, the results of a basic research project can have possibilities to "sneak in" to research prototypes within companies, to generate more application-oriented projects (even bilateral) between academic and industrial partners, and provide added value to the research community in general.

However, as discussed in this paper, there is still a long way to go in understanding the needs of the end users, matching the developer and user perspectives, dealing with the challenges in industrial application due to, e.g., legacy systems, and aligning the basic research with exploitation. If we keep in mind these challenges and learn to act accordingly, we are one step further in building bridges between basic research and industrial application.

Acknowledgements

Part of this research was co-funded by *consortium on discovering knowledge with Inductive Queries* (cInQ). The cInQ project was funded by the Future and Emerging Technologies arm of the IST Programme (Contract no. IST-2000-26469). We also wish to thank all the partners and colleagues from the cInQ project for the co-operation.

References

1. Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In *Proceedings SIGMOD'93*, pages 207–216, Washington, USA, May 1993. ACM Press.
2. Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo. Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI Press, 1996.
3. Jean-François Boulicaut and Artur Bykowski. Frequent closures as a concise representation for binary data mining. In *Proceedings PAKDD'00*, volume 1805 of *LNAI*, pages 62–73, Kyoto, JP, April 2000. Springer-Verlag.
4. T. Bui, K. Higa, V. Sivakumar, and J. Yen. Beyond telecommuting: Organizational suitability of different modes of telework. In *Proceedings of the 29th Annual Hawaii International Conference on System Sciences*, pages 344–353, Maui, Hawaii, USA, 1996.
5. Usama M. Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery: An overview. In Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 1 – 34. AAAI Press, Menlo Park, CA, 1996.
6. Kimmo Hätönen, Jean François Boulicaut, Mika Klemettinen, Markus Miettinen, and Cyrille Masson. Comprehensive log compression with frequent patterns. In *Proceedings of Data Warehousing and Knowledge Discovery - DaWaK 2003 (DaWaK'03)*, Prague, Czech Republic, Sept 2003. Springer-Verlag.
7. Kimmo Hätönen, Perttu Halonen, Mika Klemettinen, and Markus Miettinen. Queryable lossless log database compression. In *Proceedings of the 2nd International Workshop on Knowledge Discovery in Inductive Databases - (KDID'03)*, Cavtat-Dubrovnik, Croatia, Sept 2003.
8. Kimmo Hätönen and Mika Klemettinen. Domain structures in filtering irrelevant frequent patterns. In Pier Luca Lanzi, Rosa Meo, and Mika Klemettinen, editors, *Database support for data mining applications*, number 2682 in LNCS. Springer-Verlag, 2004.

9. Kimmo Hätönen, Mika Klemettinen, Heikki Mannila, Pirjo Ronkainen, and Hannu Toivonen. Knowledge discovery from telecommunication network alarm databases. In *Proceedings of the 12th International Conference on Data Engineering (ICDE'96)*, pages 115 – 122, New Orleans, Louisiana, February 1996. IEEE Computer Society Press.
10. Kimmo Hätönen, Mika Klemettinen, Heikki Mannila, Pirjo Ronkainen, and Hannu Toivonen. TASA: Telecommunication alarm sequence analyzer, or "How to enjoy faults in your network". In *Proceedings of the 1996 IEEE Network Operations and Management Symposium (NOMS'96)*, pages 520 – 529, Kyoto, Japan, April 1996. IEEE.
11. P. Keen. Information systems and organizational change. *Communications of the ACM*, 24(1):24–33, 1981.
12. H Leavitt. Applying organizational change in industry: Structural, technological and humanistic approaches. In J. March, editor, *Handbook of Organizations*. Rand McNally, Chicago, Illinois, USA, 1965.
13. Sau Dan Lee and Luc De Raedt. Mining logical sequences. In Pier Luca Lanzi, Rosa Meo, and Mika Klemettinen, editors, *Database support for data mining applications*, number 2682 in LNCS. Springer-Verlag, 2003. (to appear).
14. Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3):259 – 289, 1997.
15. F. Niederman and J. Trower. Industry influence on IS personnel and roles. In *Proceedings of the 1993 Conference on Computer Personnel Research*, pages 226–233, St Louis, Missouri, USA, 1993.
16. Jukka K. Nurminen. *Modelling and implementation issues in circuit and network planning tools*. PhD thesis, Helsinki University of Technology, Systems Analysis Laboratory, P.O.Box 1100, FIN-02015 HUT, FINLAND, May 2003.
17. Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Efficient mining of association rules using closed itemset lattices. *Information Systems*, 24(1):25–46, January 1999.
18. Jian Pei, Jiawei Han, and Runying Mao. CLOSET an efficient algorithm for mining frequent closed itemsets. In *Proceedings SIGMOD Workshop DMKD'00*, Dallas, USA, May 2000.
19. François Rioult, Céline Robardet, Sylvain Blachon, Bruno Crémilleux, Olivier Gandrillon, and Jean-François Boulicaut. Mining concepts from large sage gene expression matrices. In *Proceedings of the 2nd International Workshop on Knowledge Discovery in Inductive Databases - (KDID'03) co-located with ECML-PKDD 2003*, Cavtat-Dubrovnik, Croatia, Sept 2003.
20. Mohammed Javeed Zaki. Generating non-redundant association rules. In *Proceedings SIGKDD'00*, pages 34–43, Boston, USA, August 2000. ACM Press.

How to Quickly Find a Witness

Daniel Kifer[1,⋆], Johannes Gehrke[1], Cristian Bucila[1], and Walker White[2]

[1] Cornell University
[2] University of Dallas

Abstract. The subfield of itemset mining is essentially a collection of algorithms. Whenever a new type of constraint is discovered, a specialized algorithm is proposed to handle it. All of these algorithms are highly tuned to take advantage of the unique properties of their associated constraints, and so they are not very compatible with other constraints. We present a more unified view of mining constrained itemsets such that most existing algorithms can be easily extended to handle constraints for which they were not designed a-priori. We apply this technique to mining itemsets with restrictions on their variance — a problem that has been open for several years in the data mining community.

1 Introduction

Constrained Itemset Mining is a very important data mining problem [15]. It can be stated as follows. Let $\mathcal{I}$ be a set of distinct "items" (where an item is an undefined primitive). A transaction t is a set of items (a nonempty subset of $\mathcal{I}$) and a database $\mathcal{D}$ is a multiset of transactions. In constrained itemset mining, we would like to find all subsets of $\mathcal{I}$ that satisfy a *constraint*, a user-defined property designed to tailor the output of the data mining algorithm to the user's preferences. Such constraints can be the traditional "minimum support constraint", where we are only interested in sets $X \subseteq \mathcal{I}$ such that there exist at least s transactions $t \in \mathcal{D}$ with $X \subseteq t$, or more complex constraints such as "the average price of the items has to be larger than c", or "the variance of the prices of the items has to be smaller than c". Three important classes of constraints have been studied: monotone, antimonotone, and convertible constraints [15,18]; each class has its own set of efficient mining algorithms [14,18,16,6,7,8]. Some of these algorithms have a certain degree of flexibility – they can efficiently mine constraints from several of these classes simultaneously.

For example, several algorithms can simultaneously mine monotone and antimonotone constraints [16,6,7,8], or mine convertible combined with either monotone or antimonotone constraints [18]. Unfortunately, as we will show later, the flexibility of these algorithms is very limited, especially when convertible constraints are involved.

We present a unified framework for constrained itemset mining that applies to any type of constraint. Our framework is based on the concept of efficiently finding a *witness*, which is a single itemset X on which we can test whether the constraint holds. This test will provide information about properties of other itemsets. That information

⋆ Supported by NSF.

J.-F. Boulicaut et al. (Eds.): Constraint-Based Mining, LNAI 3848, pp. 216–242, 2005.

can then be used for pruning the search space. The notion of a witness has conceptual implications. For example, we now can efficiently mine all three types of constraints *simultaneously* (by finding witnesses for each constraint), and we can also mine complicated constraints that are neither monotone, antimonotone, nor convertible. As a demonstration, we will introduce an efficient algorithm for finding a witness for constraints involving the variance of a set of items.

In developing this framework, we make the following contributions:

- We introduce the concept of a witness, which decouples the strategy for traversing the search space from the efficiency of pruning it (using constraints). This transforms the traversal strategy from a necessary restriction on an algorithm into an optimization heuristic. To illustrate the concept of a witness, we show a very efficient algorithm for finding a witness for a large class of functions which we call *stable functions*. (Section 2)
- We show how to efficiently find a witness for the constraints $\mathrm{var}(S) \leq c$ and $\mathrm{var}(S) \geq c$, and therefore show how to prune using those constraints - a problem that has been open in the literature for several years. (Section 3)
- We outline several heuristics that further improve the efficiency of finding witnesses. (Section 4)

For the remainder of this section, we take the reader on a short tour of the relevant issues motivating our approach. We introduce some terminology and helpful notation in Section 1.1, and then give an overview of our results in Section 1.2. Readers interested in the more technical aspects should continue on to Section 2.

1.1 Preliminaries

Let $\mathcal{I}$ be a set of distinct "items" (where an item is an undefined primitive). A transaction t is a set of items (a nonempty subset of $\mathcal{I}$) and a database $\mathcal{D}$ is a multiset of transactions. Given a function whose domain is $\mathcal{I}$, such as $\mathrm{price} : \mathcal{I} \rightarrow \mathbb{R}$, we extend it to sets of items in the natural way, e.g., $\mathrm{price}(S)$ is the multiset $\{\mathrm{price}(x) \ : \ x \in S\}$. We are also given a real-valued function whose domain is $2^{\mathcal{I}}$, the powerset of $\mathcal{I}$. An example of such a function is $\mathrm{support}(S)$, which is the number of transactions in $\mathcal{D}$ that are supersets of S. We will use such functions to define *constraints*. For example, if we want to find all sets of items that have support greater than some constant c, we say we are mining with the constraint $\mathrm{support}(S) > c$.

Let us now examine some classes of constraints.

Definition 1 (Antimonotone). *A constraint P is antimonotone if whenever $A \subseteq B \subseteq \mathcal{I}$ then $P(B) \Rightarrow P(A)$, or equivalently, $\neg P(A) \Rightarrow \neg P(B)$.*

Definition 2 (Monotone). *A constraint Q is monotone if whenever $A \subseteq B \subseteq \mathcal{I}$ then $Q(A) \Rightarrow Q(B)$, or equivalently, $\neg Q(B) \Rightarrow \neg Q(A)$.*

Note that both antimonotonicity and monotonicity are useful properties. Once we know that itemset A does not satisfy an antimonotone constraint P we don't need to look at supersets of A, and if itemset B satisfies P then we know that all subsets of B

satisfy P. Similarly, once we know that B does not satisfy a monotone constraint Q we don't need to look at B's subsets, and if A satisfies Q then so do all supersets of A.

These two classes of constraints have another useful feature: they are both closed under logical conjunction (AND). If P_1 and P_2 are antimonotone (resp., monotone) constraints then so is $P_1 \wedge P_2$ and we can use existing algorithms to prune with this compound constraint.

To define convertible constraints, we need to discuss the notion of a prefix. Fix an ordering on the elements of $\mathcal{I}$. We can therefore treat $S_1 \subseteq \mathcal{I}$ and $S_2 \subseteq \mathcal{I}$ as two sequences. Let ℓ_1 be the length of S_1, and ℓ_2 be the length of S_2. Then S_1 is a *prefix* of S_2 if $\ell_1 \leq \ell_2$ and the first ℓ_1 elements of S_2 are exactly S_1.

Definition 3 (Convertible [18]). *A constraint R is convertible monotone if there is an ordering ω_1 such that whenever S_1 is a prefix of S_2 then $R(S_1) \Rightarrow R(S_2)$ (i.e., $\neg R(S_2) \Rightarrow \neg R(S_1)$) and R is convertible antimonotone if there is an ordering ω_2 such that if T_1 is a prefix of T_2 then $R(T_2) \Rightarrow R(T_1)$ (i.e. $\neg R(T_1) \Rightarrow \neg R(T_2)$). R is convertible if it is both convertible monotone and convertible antimonotone.*

In order to prune with convertible constraints efficiently, an algorithm must examine itemsets in a restricted order. An example of a convertible constraint is $R \equiv \mathrm{avg}(\mathrm{price}(S)) > c$. If the items are sorted by price in ascending order then R is convertible monotone; if the items are sorted in descending order then R is convertible antimonotone. Similarly, the constraint $\mathrm{avg}(\mathrm{price}(S)) < c$ is also convertible.[1] But suppose we want to mine with the following constraint that involves the functions price and weight:

$$\Big(\mathrm{avg}(\mathrm{price}(S)) \leq c\Big) \wedge \Big(\mathrm{avg}(\mathrm{weight}(S)) \leq d\Big)$$

If we assume that price and weight are not correlated, then this conjunction of convertible constraints is *not* convertible. Thus existing algorithms for convertible constraints will not prune efficiently - they will use only one of these constraints and then post-process the output. This is an unfortunate situation, since many interesting predicates are conjunctions of convertible monotone or convertible antimonotone constraints.

1.2 Catching a Witness: An Overview

As an elementary example, suppose that $\mathcal{I} = \{a, b, c, d\}$ and that database D consists of the following sets: $\{a, b, c\}$, $\{b, c, d\}$, $\{b, c\}$,$\{b, d\}$. Assume we are interested in all subsets of $\mathcal{I}$ that have support ≥ 2. To find these sets, we must enumerate all possible subsets of $\mathcal{I}$ and then test this property for each of them. At one point we will consider the set $\{a\}$. It is included in only one set in D and therefore it is not interesting to us. We could add more elements to $\{a\}$, in fact we could add any subset of $\{b, c, d\}$ (call this set $\mathcal{F}(\{a\})$) to get another set in our enumeration. However, support is an antimonotone constraint and so any set containing $\{a\}$ will have support less than 2. Thus we can *prune* from consideration all sets X such that $\{a\} \subseteq X \subseteq \mathcal{F}(\{a\}) \cup \{a\}$ — 8 sets in all. Let us call this collection of sets $\mathcal{A}(\{a\})$. We say that $\{a\}$ *is a negative*

[1] Note that we can replace $<$ by $\leq$ and $>$ by $\geq$ without changing any of the properties stated so far.

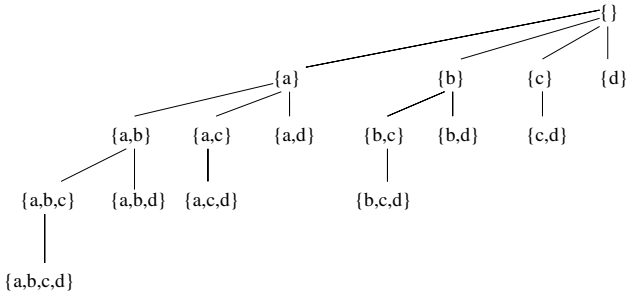


Fig. 1. An enumeration of a, b, c, d

witness for $\mathcal{A}(\{a\})$ because once we know that our constraint does not hold for $\{a\}$, we know it does not hold for any set in $\mathcal{A}(\{a\})$. If $\{a\}$ had been included in more than one transaction in D, then we could not conclude anything about *all* sets in $\mathcal{A}(\{a\})$ and we would have to examine them further. Thus having $\{a\}$ as a witness for $\mathcal{A}(\{a\})$ allows us to prune a large part of the search space. In general, if P holds for $\{a\}$ implies P holds for all sets in $\mathcal{A}(\{a\})$ then we call $\{a\}$ a positive witness for $\mathcal{A}(\{a\})$ with respect to P and a negative witness with respect to $\neg P$.

A property P can have both positive and negative witnesses. If X is a positive witness then $P(X) = \texttt{true}$ implies that $P(Y) = \texttt{true}$ for all $Y \in \mathcal{A}(X)$ and so we save time by not evaluating $P(Y)$. If X is a negative witness (then it is a positive witness for $\neg P$) then $P(X) = \texttt{false}$ implies that $P(Y) = \texttt{false}$ for all $Y \in \mathcal{A}(X)$ and so we save time by pruning $\mathcal{A}(X)$.

Let us investigate how witnesses actually work when mining with constraints. Each itemset mining algorithm enumerates candidate itemsets in some order, for example through a tree structure (see Figure 1 for an enumeration for a, b, c, d) that is traversed in a depth-first or breadth-first manner. When we examine a set X, we need to find a witness for the subtree rooted at X. In simple cases, such as mining with a single antimonotone constraint, X is this witness, whereas in other cases finding a witness is not so trivial.

For an example where X is not a witness for $\mathcal{A}(X)$, suppose that the prices of a, b, c, d are $1, 7, 6, 5$, respectively, and that we are interested in all sets whose average is at least 6. If we examine node $\{a\}$ in Figure 1 then clearly the average price of $\{a\}$ does not tell us much about the average price of nodes in $\mathcal{A}(\{a\})$, the subtree rooted at $\{a\}$. However, if we add to $\{a\}$ all items with price ≥ 6 we obtain the candidate witness $\{a, b, c\}$. Since we added as many items with price ≥ 6 as possible, if $\{a, b, c\}$ does not have an average ≥ 6, then no set in $\mathcal{A}(\{a\})$ can have an average ≥ 6, and thus $\{a, b, c\}$ is a negative witness. The average price of $\{a, b, c\}$ actually is less than 6 and so same is true for any set in $\mathcal{A}(\{a\})$. Thus this witness allows us to prune the complete subtree. Had the price of item a been 5 or higher, the witness would not give us enough information and we would have to traverse $\mathcal{A}(\{a\})$. Thus finding witnesses for constraints involving an average is rather straightforward: we add all items with value greater than the threshold to obtain the witness itemset, and then we test its average.

Now let us consider a more difficult case. Assume that our constraint states that the *variance* of the prices must be $\leq k$ (or $\geq k$) for some constant k. Now when we examine node $\{a\}$, good witnesses are elements in $\mathcal{A}(\{a\})$ that have maximal or minimal variance. But how do we find an element with maximal variance? Intuitively, the variance of a set is large if the elements are far away from the average. This motivates the following simple algorithm: we add to $\{a\}$ the item x that is furthest from the average of $\{a\}$, then add the item y that is furthest away from the average of $\{a, x\}$, etc. This simple algorithm overlooks the subtlety that we want to add elements that are furthest away from the average of the final witness, rather than the average of $\{a\}$ – but we do not know the average of the final witness a priori. Nevertheless, as we will show in Section 2, a variant of this algorithm actually finds the itemset with maximal variance.

Now let us consider the case where we want to find an itemset with minimal variance. Intuitively we want to include items that are close to the average of the final witness, but not items that are far from the average. Here we run into the same subtlety — the average we are talking about is the average of the witness, not the average of $\{a\}$. These subtleties present significant hurdles to the development of an efficient algorithm for finding a witness. As examples, assume that we are currently examining node X in a search tree. The following two approaches are doomed to fail:

First Algorithm

1. Start at $C = \mathcal{I}$ (all of the items).
2. Remove from C the element in $C \setminus X$ which is furthest away from the current average of C, and return true if this new set has variance $\leq k$
3. Repeat step 2.

Second Algorithm

1. Start at $C = X$.
2. Add to C the element closest to $\mathrm{avg}(C)$, and return true if this new set has variance $\leq k$
3. Repeat step 2.

We can construct an example where both algorithms fail to return the correct witness. Let X be $\{45, 55\}$, the set of two items with prices 45 and 55, respectively. Assume that the subtree rooted at X contains the following items: $1,000,000$ items with price 100; $999,999$ items with price 0; one item with price 30 and another item with price 15. From this example it is clear that there is only one set with minimal variance and we obtain it by adding to X all elements with price 100. Let k be slightly larger than the minimal variance but smaller than the variance of any other set containing X. The first algorithm will fail because it will add the item with price 30. The second algorithm will fail because the average of all prices is slightly less than 50, and thus the algorithm will remove all items with price $= 100$.

From this example we see that we can lower the variance by adding a "dense cluster" — many items with similar values. If we order the items on a line by price and slide an appropriately sized window, we may be able to find a good cluster that lowers the variance enough. In Section 3.2 we will explain the structure of such a window. However, the size of the window depends not only on the *values* of the elements in the

window, but also on the *number* of elements the window contains. In fact, as we slide the window, it can shrink: as the left endpoint of the window moves to the right, the right endpoint of the window might move to the left! In Section 3.2, using subtle reasoning about the structure of the space, we describe an algorithm that finds a witness in time *linear* in the number of items.

2 Witnesses

The execution of a typical data mining algorithm for antimonotone constraints looks like a tree. At the root is the empty set and all other nodes are non-empty sets of items. A child is a superset of its parent and contains one more item than its parent (see Figure 1 for an example). Let n be some node in the tree. Let $B(n)$ be the set of items associated with n, and let $\mathrm{Free}(n)$ be the collection of items that can be added to $B(n)$. $\mathrm{Free}(n)$ is the minimal set such that for any descendant n' of n, $B(n') = B(n) \cup J$ where $J \subseteq \mathrm{Free}(n)$. For example in Figure 1, if $B(n) = \{a, b\}$ then $\mathrm{Free}(n) = \{c, d\}$. Let $\mathcal{A}(n)$ be the collection of sets X such that $B(n) \subseteq X \subseteq B(n) \cup \mathrm{Free}(n)$. As is done in practice, we assume constraints have the following form: $f(X)\#c$ where $\#$ is either $<, \leq, >$ or $\geq$; c is a constant; X is a set; and f is a real-valued function whose domain is $2^{\mathcal{I}}$, the powerset of $\mathcal{I}$.

Definition 4 (Witness). *Given a fixed constant c, node n and a function $f : 2^{\mathcal{I}} \to \mathbb{R}$, a set $\mathcal{Y}_n \in \mathcal{A}(n)$ is called a* large witness *if*

$$f(\mathcal{Y}_n) \leq c \Rightarrow \forall X \in \mathcal{A}(n) : \ f(X) \leq c$$

A set $\mathcal{Z}_n \in \mathcal{A}(n)$ is called a small witness *if*

$$f(\mathcal{Z}_n) \geq c \Rightarrow \forall X \in \mathcal{A}(n) : \ f(X) \geq c$$

For a general predicate P, $W_n \in \mathcal{A}(n)$ is a positive witness if

$$P(W_n) = \texttt{true} \Rightarrow \forall X \in \mathcal{A}(n) : \ P(X) = \texttt{true}$$

and W_n is a negative witness if

$$P(W_n) = \texttt{false} \Rightarrow \forall X \in \mathcal{A}(n) : \ P(X) = \texttt{false}$$

The intuition behind this nomenclature is that a set in $\mathcal{A}(n)$ that maximizes f (over $\mathcal{A}(n)$) is a *large* witness and a set that minimizes f is a *small* witness. When it is unambiguous, the notational dependency on n will be dropped. We will use $\mathcal{Y}$ to represent a large witness and $\mathcal{Z}$ to represent a small witness. Note that

$$f(\mathcal{Y}) < c \Rightarrow \forall X \in \mathcal{A} : \ f(X) < c$$

and

$$f(\mathcal{Z}) > c \Rightarrow \forall X \in \mathcal{A} : \ f(X) > c$$

When we are mining for itemsets X that satisfy $f(X) \geq c$, if $f(\mathcal{Y}_n) < c$ then clearly we can prune out $\mathcal{A}(n)$ — we do not need to look at any set in that collection.

If $f(\mathcal{Y}_n) \geq c$ then we do not have enough information to prune and must examine the children of n. If $f(\mathcal{Z}_n) \geq c$ then we do not need to evaluate f on the sets in $\mathcal{A}(n)$ — we know the result will be greater than or equal to c. If $f(\mathcal{Z}_n) < c$ then we do not have enough information and must examine the children of n. Analogous statements are true when we have constraints $f(X)\#c$, where $\#$ is $>, <$, or $\leq$.

2.1 Comparison to Existing Methods

When mining with antimonotone constraints, such as $\text{support}(X) > c$, then for any node n, clearly $B(n)$ is a negative witness and $B(n) \cup \text{Free}(n)$ is a positive witness. For monotone constraints, such as $\text{support}(X) < c$, $B(n)$ is a positive witness and $B(n) \cup \text{Free}(n)$ is a negative witness. For the *function* $\text{support}(X)$, $B(n)$ is a large witness and $B(n) \cup \text{Free}(n)$ is a small witness (clearly a small or large witness is negative or positive depending on the inequality used in the constraint). Thus we have generalized pruning with monotone and antimonotone constraints. Most algorithms that prune with monotone and/or antimonotone constraints can easily be modified to search for witnesses in order to prune efficiently.

Witness-based pruning can also handle many convertible constraints. One of the most interesting convertible constraints is average (i.e., average price). Assuming all items have a price,

$$\mathcal{Y}_n = B(n) \cup \{x \in \text{Free}(n) : \text{price}(x) \geq c\}$$

is clearly a large witness, and

$$\mathcal{Z}_n = B(n) \cup \{x \in \text{Free}(n) : \text{price}(x) \leq c\}$$

is a small witness.

Naively, it may take $O(\text{Free}(n))$ time to find a witness and calculate its average. However, we just need to know the average of a witness and this can be maintained incrementally in constant time per node. Algorithm 1 shows this technique applied to a simple depth-first algorithm for antimonotone constraints. Note the $O(\mathcal{I})$ initialization step done once at the beginning of the algorithm. We can apply this technique in a straightforward manner to many other algorithms for mining monotone and antimonotone constraints, including DualMiner [8]. One advantage of this technique is that the modified algorithms can handle conjunctions of constraints. This is possible by simply searching for a witness for each constraint. Thus our technique can efficiently find sets of items X such that $R \equiv \text{avg\_price}(X) < c \wedge \text{avg\_weight}(X) < d$, whereas an algorithm designed for convertible constraints cannot prune with R — despite the fact that R is simply a conjunction of convertible constraints. Another advantage of our approach is that the presence of a conjunction of several constraints does not restrict the order in which nodes can be evaluated. This gives our technique an extra degree of freedom for optimization of traversal strategies with heuristics. Note that our technique can even be used to modify existing algorithms for convertible constraints.

Let us introduce some notation before we discuss some conceptual extensions. For convenience we will start identifying items x_i with their prices ($\text{price}(x_i)$). Since several items may have the same price we are now dealing with multisets and as a reminder

Algorithm 1 : AVGminer

Require: antimonotone P, node n,Free(n), $Zsum$, $Zcount$
1: **if** $n =$root **then**
2: Free(n) $\leftarrow \mathcal{I}$
3: $Zsum \leftarrow \sum_{x \in \text{Free}, \text{price}(x) \leq c} \text{price}(x)$
4: $Zcount \leftarrow \sum_{x \in \text{Free}, \text{price}(x) \leq c} 1$
5: **else if** $\neg P(n) \ \vee \ Zsum/Zcount > c$ **then**
6: RETURN (no set in $\mathcal{A}$ satisfies both constraints)
7: **else if** $P(n) \ \wedge \ \text{avg}(\text{price}(n)) \leq c$ **then**
8: OUTPUT $B(n)$
9: **end if**
10: $Temp \leftarrow$ Free(n)
11: **while** $Temp \neq \emptyset$ **do**
12: choose some $x \in Temp$; $Temp \leftarrow Temp \setminus \{x\}$
13: **if** $x \leq c$ **then**
14: $Zsum \leftarrow Zsum - x$; $Zcount \leftarrow Zcount - 1$
15: **end if**
16: create child n' such that $B(n') = B(n) \cup \{x\}$
17: AVGminer(P,n',$Temp$,$Zsum + x$,$Zcount + 1$)
18: **end while**

of this fact, $\oplus$ will represent multiset union and $\ominus$ will represent multiset set-difference. Therefore when we talk about a set in $\mathcal{A}$ we are really talking about a multiset.

The same witnesses that work for average also work for a more general class, the class of stable functions, as introduced in the following definition.

Definition 5. *A real-valued function f is* stable *if, for any c*

$$f(A), f(\{x\}) \geq c \Rightarrow f(A \oplus \{x\}) \geq c, \textit{and}$$
$$f(A), f(\{x\}) \leq c \Rightarrow f(A \oplus \{x\}) \leq c.$$

The predicates $f(S) \geq c$ and $f(S) \leq c$ are called stable constraints.

Examples of stable functions are average, median, and even linear combinations of moments. A linear combination of moments has the form

$$f(X) = \sum_j \frac{a_j}{n} \sum_{i=1}^{n} x_i^j = \frac{\sum_{i=1}^{n} \sum_j a_j x_i^j}{n} = \frac{\sum_{i=1}^{n} f(\{x_i\})}{n}$$

and is clearly stable. The following theorem shows how to find witnesses $\mathcal{Y}$ and $\mathcal{Z}$ for stable functions, such that $f(Y) \geq c \Rightarrow f(X) \geq c$ and $f(Z) \leq c \Rightarrow f(X) \leq c$, for all $X \in \mathcal{A}$.

Theorem 1. *Let n be a node and f a stable function. Then*

$$f(B(n) \oplus \{x \in \text{Free}(n) : \ f(x) \leq c\}) \leq c$$
if and only if $\quad \exists X \in \mathcal{A}(n)$ *such that* $f(X) \leq c$.

Also

$$f(B \oplus \{x \in \text{Free}(n) : \ f(x) \geq c\}) \geq c$$
$$\textit{if and only if} \qquad \exists X \in \mathcal{A}(n) \textit{such that} f(X) \geq c.$$

Proof. We only prove the first statement, as the second is similar. One direction is obvious. Assume there is some $X \in \mathcal{A}$ such that $f(X) \leq c$. Since f is stable, it follows by induction that $f(X \ominus \{x \in \text{Free} : \ f(x) > c\}) \leq c$. Then, $f(B \oplus \{x \in \text{Free} : f(x) \leq c\}) = f(X \ominus \{x \in \text{Free} : \ f(x) > c\} \oplus \{x \in \text{Free} \ominus X : \ f(x) \leq c\}) \leq c$. □

A stable function f is invertible if given $f(\{x_1, x_2, \ldots, x_k\})$ and x_i (for $1 \leq i \leq k$) we can compute $f(\{x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_k\})$. In this case we can use the same approach we used for average to maintain (in constant time per node) the value of f of a witness. For example, linear combinations of moments are invertible. It should be noted that the common convertible constraints are included in the class of invertible stable functions.

3 Mining Variance

We now apply our framework to solve an open problem: mining variance. Our solution essentially reduces the problem from finding itemsets to searching for a particular node in a lattice. The following key property will be used extensively.

Lemma 1. *If M is a multiset and $c, d \in \mathbb{R}$ such that*

$$|d - \text{avg}(M)| \geq |c - \text{avg}(M)|,$$

then we have

$$\text{var}(M \oplus \{d\}) \geq \text{var}(M \oplus \{c\}).$$

While we provide a formal proof below, this claim is intuitively obvious; the further away an element is from the average, the larger the variance. This leads to the following simple corollary.

Corollary 1. *If $c \in M$ and either $d \geq c \geq \text{avg}(M)$ or $d \leq c \leq \text{avg}(M)$ then* $\text{var}(M \ominus \{c\} \oplus \{d\}) \geq \text{var}(M)$.

Proof (of Lemma 1). Assume $|d - \text{avg}(M)| \geq |c - \text{avg}(M)|$. Let $n = |M|$. Given a constant k,

$$\text{var}(M \oplus \{k\}) = \frac{k^2 + \sum_M x_i^2}{n+1} - \frac{\left(k + \sum_M x_i\right)^2}{(n+1)^2}$$
$$= \frac{k^2 + \sum_M x_i^2}{n+1} - \frac{k^2 + 2k\sum_M x_i + \left(\sum_M x_i\right)^2}{(n+1)^2}$$

Thus

$$\begin{aligned}\operatorname{var}(M \oplus \{d\}) - \operatorname{var}(M \oplus \{c\}) &= \frac{d^2 - c^2}{n+1} - \frac{d^2 - c^2 + 2(d-c)\sum_M x_i}{(n+1)^2} \\ &= \frac{n}{(n+1)^2}\left[(d^2 - c^2) - 2(d-c)\operatorname{avg}(M)\right]\end{aligned}$$

The last equation is nonnegative if and only if

$$(d-c)(d+c) \geq (d-c)2\operatorname{avg}(M) \tag{1}$$

Case 1. If $d, c \geq \operatorname{avg}(M)$ then by hypothesis $d \geq c$ and so Equation (1) is satisfied.
Case 2. If $d, c \leq \operatorname{avg}(M)$ then by hypothesis $d \leq c$ and clearly $d + c \leq 2\operatorname{avg}(M)$. Therefore, Equation (1) is also satisfied.
Case 3. If $d \geq \operatorname{avg}(M) \geq c$ then $d - c \geq 0$ and by hypothesis $d - \operatorname{avg}(M) \geq \operatorname{avg}(M) - c$ and so $d + c \geq 2\operatorname{avg}(M)$. Thus Equation (1) is satisfied.
Case 4. Finally, if $d \leq \operatorname{avg}(M) \leq c$ then $d - c \leq 0$ and by hypothesis $\operatorname{avg}(M) - d \geq c - \operatorname{avg}(M)$ and so $2\operatorname{avg}(M) \geq d + c$. Even in this case Equation (1) is satisfied.
□

To find itemsets that satisfy $\operatorname{var}(X) > c$ we need to prune sets where $\operatorname{var}(X) \leq c$. Thus we need to find a witness $\mathcal{Y}$ such that $\operatorname{var}(\mathcal{Y}) \leq c \Rightarrow \operatorname{var}(X) \leq c$ for any $X \in \mathcal{A}$. The next subsection shows how this is done.

3.1 Finding Maximal Variance

An obvious choice for such a witness $\mathcal{Y}$ is a set in $\mathcal{A}(n)$ that has maximal variance. We begin by examining what such a witness looks like. For any set S let $\min_k(S)$ be the k smallest elements of S and $\max_k(S)$ be the k largest elements of S. Ties are broken according to some convention $>_\kappa$; that is if $a = b$ with $a \in X$ and $b \notin X$ then

- if $a > \operatorname{avg}(X)$ let $a >_\kappa b$
- if $a \leq \operatorname{avg}(X)$ then $a <_\kappa b$

We break all other ties arbitrarily.

Lemma 2. *Given a node n, then for any element X in $\mathcal{A}(n)$ with maximal variance, there exist two nonnegative integers L and R such that*

$$\begin{aligned}X = B(n) &\oplus \min_L\{y \in \text{Free} : \ y \leq \operatorname{avg}(X)\} \\ &\oplus \max_R\{y \in \text{Free} : \ y > \operatorname{avg}(X)\}\end{aligned}$$

Proof. We need only show the existence of L, as the existence of R is analogous. Let $S_1 = \{y \in X \ominus B : \ y \leq \operatorname{avg}(X)\}$ and let $F_1 = \{y \in \text{Free} : \ y \leq \max(S_1)\}$. If $S_1 = \emptyset$ then $L = 0$ and if $S_1 = F_1$ then $L = |S_1|$. Otherwise, let $m = \max(S_1)$ and choose $y = \min(F_1 \ominus S_1)$. If $y = m$ then $L = |S_1|$ and we are done by the tie-breaking convention. The only other possibility is $y < m$, in which case it is further away from $\operatorname{avg}(X)$ than m and, since $y \notin X$, by Corollary 1 we can replace m with y in X to increase the variance. Clearly this case can't happen and so a suitable L always exists.
□

In other words, in addition to B, X contains the L^{th} most extreme elements on the left and the R^{th} most extreme elements on the right.

A naive approach to finding a witness $\mathcal{Y}$ would look at all pairs of integers L, R and use Lemma 2, but that would result in an $O(|\operatorname{Free}|^2)$ algorithm. We need to find the elements that are furthest away *from the average of* $\mathcal{Y}$ without knowing what this average is. Because of this subtlety, it is surprising not only that a linear time algorithm exists, but also that this algorithm is greedy.

However, we have one precondition. Before we begin to mine, we must sort all elements by value. The sorted order of $\operatorname{Free}(n)$ can easily be maintained by most algorithms as they examine different nodes n. Thus we pay a one-time $O(|\mathcal{I}| \log |\mathcal{I}|)$ startup cost – which is not so bad considering how much time mining algorithms take – and a constant cost per node maintaining this order. Algorithm 2 shows the witness-search algorithm. It returns true if the witness has variance greater than c, false otherwise.

Algorithm 2 : Maximal Variance

Require: node n, $\operatorname{Free}(n)$ is sorted
1: $C_0 \leftarrow B(n), i \leftarrow 0$
2: **if** $\operatorname{var}(C_0) > c$ **then**
3: RETURN true
4: **end if**
5: $Temp \leftarrow \operatorname{Free}(n)$
6: **while** $Temp \neq \emptyset$ **do**
7: choose $x \in Temp$ with $|x - \operatorname{avg}(C_i)|$ largest
8: (ties broken arbitrarily)
9: $C_{i+1} \leftarrow C_i \oplus \{x\}$
10: **if** $\operatorname{var}(C_{i+1}) > c$ **then**
11: RETURN true
12: **else if** $\operatorname{var}(C_{i-1}) \geq \max\big(\operatorname{var}(C_i), \operatorname{var}(C_{i+1})\big), i \geq 1$ **then**
13: RETURN false
14: **end if**
15: $i \leftarrow i + 1$
16: **end while**
17: RETURN false

In algorithm 2 we keep adding elements that are furthest away from the *current* average until we find a $Y \in \mathcal{A}$ with $\operatorname{var}(Y) > c$ or we reach the stopping condition at line 12. The stopping condition essentially says that we get two chances to keep the variance growing.

In order to show that this algorithm is correct, we need to show two things: that it visits an element with maximal variance when a witness exists, and that the condition for returning "false" is correct. The first half of this correctness claim is covered by the theorem below.

Theorem 2. *Without any stopping conditions, Algorithm 2 will visit an element in* $\mathcal{A}(n)$ *with maximal variance.*

Proof. Let T be the set $B(n) \oplus \mathrm{Free}(n)$. Without stopping conditions, the execution of this algorithm produces a chain of sets $B(n) = C_0 \subset C_1 \subset \cdots \subset C_k = T$ and for all i, $|C_{i+1} \ominus C_i| = 1$. If T has maximal variance then we are done. If not, let j be the largest index such that C_j is a subset of an element with maximal variance but C_{j+1} is not. If C_j has maximal variance then we are done. Otherwise, let M be some superset of C_j that has maximal variance. Also let $c = C_{j+1} \ominus C_j$. By definition, c is chosen by the algorithm because it is the free element furthest away from $\mathrm{avg}(C_j)$.

Because of symmetry, we can assume $c \geq \mathrm{avg}(C_j)$. By the definition of c we know that $M \ominus C_j$ can only contain elements less than c. If some element is larger, it is further away from $\mathrm{avg}(C_j)$; if some element equals c, then we can just replace it with c without affecting variance, which violates the definition of C_j. This means that $c \geq \mathrm{avg}(M)$ since we can not add $M \ominus C_j$ to C_j and increase the average beyond c.

From Corollary 1, we know that $M \ominus C_j$ contains no element $\geq \mathrm{avg}(M)$. Otherwise we could replace it with c and the variance will not decrease. Therefore $\max(M \ominus C_j) = m < \mathrm{avg}(M)$.

Now we claim that $\mathrm{avg}(C_j) \geq m$. If this is not the case, then $m > \mathrm{avg}(C_j)$ and adding $M \ominus C_j$ to C_j would not raise the average past m. This implies $m \geq \mathrm{avg}(M)$, which cannot happen. Thus $\mathrm{avg}(C_j) \geq m$ and adding $M \ominus C_j$ to C_j would only lower the average. Since $C_j \subset M$, it follows that $\mathrm{avg}(C_j) \geq \mathrm{avg}(M)$. If $\delta = \min(M \ominus C_j)$ (which is $< \mathrm{avg}(M)$) it also follows that $\mathrm{avg}(C_j) \geq \mathrm{avg}(M \ominus \{\delta\})$. As c is the free element furthest away from $\mathrm{avg}(C_j)$, we see that

$$\begin{aligned} c - \mathrm{avg}(M \ominus \{\delta\}) &\geq c - \mathrm{avg}(C_j) \geq \mathrm{avg}(C_j) - \delta \\ &\geq \mathrm{avg}(M \ominus \{\delta\}) - \delta \geq 0 \end{aligned}$$

By Lemma 1 $\mathrm{var}(M \ominus \{\delta\} \oplus \{c\}) \geq \mathrm{var}(M)$, a contradiction. Therefore the greedy algorithm visits the node with the largest variance.

□

So all that is left is to show that if we reach the stopping condition, then no witness exists.

Theorem 3. *Let C_i be a multiset such that* $\mathrm{var}\, C_{i+1} \leq \mathrm{var}\, C_i$ *and* $\mathrm{var}\, C_{i+2} \leq \mathrm{var}\, C_i$. *Then for any $j \geq i$,* $\mathrm{var}(C_j) \leq \mathrm{var}(C_i)$.

The proof of Theorem 3 is rather involved, it is useful to first see an example motivating it. The implication of Theorem 3 is that if no set has variance greater than c, then we will find this out two iterations after we reach a node with maximal variance. The reason for this is that the variance does not grow monotonically, but instead zigzags. This is clear from the following example.

Example 1. Let $B = \{-40, -40, 40, 40\}$ and $\mathrm{Free} = \{-42, -42, 42, 42\}$. The chain of sets produced is

$$C_0 = \{-40, -40, 40, 40\} \qquad \mathrm{var}(C_0) = 1600$$

$$C_1 = C_0 \oplus \{42\} \qquad \mathrm{var}(C_1) = 1562.24$$

$$C_2 = C_1 \oplus \{-42\} \qquad \mathrm{var}(C_2) = 1654\frac{2}{3}$$

$$C_3 = C_2 \oplus \{42\} \qquad \mathrm{var}(C_3) = 1634\frac{2}{7}$$

$$C_4 = C_3 \oplus \{-42\} \qquad \mathrm{var}(C_4) = 1682$$

To prove Theorem 3, we need the following two lemmas.

Lemma 3. $\mathrm{var}(C_i) \geq \mathrm{var}(C_{i+1})$ *if and only if* $d = C_{i+1} \ominus C_i$ *satisfies the condition*

$$n(d - \mathrm{avg}(C_i))^2 \leq (n+1)\,\mathrm{var}(C_i), \text{ where } n = |C_i|$$

Alternatively, if C_i *has average* 0 *and sum of squares* Q*, then* $n^2 d^2 \leq (n+1)Q$.

Proof. Let $X = C_i$, $n = |X|$ and $Y = C_{i+1}$. Let $d = Y \ominus X$. First suppose that $\mathrm{avg}(X) = 0$ and without loss of generality assume that $d \geq 0$. Because of the way d is selected, $|d| \geq |c|$ for any $c \in T \ominus X$. Let Q be the sum of squares in X. Since $\mathrm{var}(X) \geq \mathrm{var}(Y)$,

$$\mathrm{var}(Y) - \mathrm{var}(X) = \frac{Q + d^2}{n+1} - \frac{d^2}{(n+1)^2} - \frac{Q}{n} = \frac{nd^2 - Q}{n(n+1)} - \frac{d^2}{(n+1)^2} \leq 0$$

This is true if and only if

$$\begin{aligned} &\frac{(n^2+n)d^2 - (n+1)Q - nd^2}{n(n+1)} \leq 0 \\ \Leftrightarrow\ &\frac{n^2d^2 - (n+1)Q}{n(n+1)} \leq 0 \\ \Leftrightarrow\ &n^2d^2 - (n+1)Q \leq 0 \\ \Leftrightarrow\ &n^2d^2 \leq (n+1)Q \end{aligned}$$

If X does not have average 0 then since variance is translation invariant, we can apply this result to

$$X' = \{x - \mathrm{avg}(X) :\ x \in X\}$$

and to $Q' = \sum_{x \in X} (x - \mathrm{avg}(X))^2$, $d' = d - \mathrm{avg}(X)$. Now Q' is just $n\,\mathrm{var}(X)$ and so

$$n(d - \mathrm{avg}(X))^2 \leq (n+1)\,\mathrm{var}(X)$$

□

To state the next Lemma, we use the following convention: when k is an integer and S is a multiset, we use $k \cdot S$ to denote k multiset unions of S with itself.

Lemma 4. *Let C be a set, a be an element and let $D \geq \mathrm{var}(C)$ and $p \in \mathbb{Z}^+$ be constants. If*

$$\mathrm{var}(C \oplus (p \cdot \{a\})) > D$$

then

$$(a - \mathrm{avg}(C))^2 > \frac{|C|+1}{|C|} D$$

Proof. Let $Q = \mathrm{var}(C)$. Let $C' = \{x - \mathrm{avg}(C) \ : \ x \in C\}$ and let $n = |C|$ and $b = a - \mathrm{avg}(C)$. Clearly $\mathrm{avg}(C') = 0$ and $\mathrm{var}(C') = Q$. The quantity nQ is the sum of squares of C' and $\mathrm{var}(C' \oplus (p \cdot \{b\})) = \mathrm{var}(C \oplus (p \cdot \{a\}))$.

$$\begin{aligned}
&\mathrm{var}(C \oplus (p \cdot \{a\})) \geq D \Rightarrow \mathrm{var}(C' \oplus (p \cdot \{b\})) \geq D \\
&\quad \Rightarrow \frac{nQ + pb^2}{n+p} - \frac{p^2 b^2}{(n+p)^2} \geq D \\
&\quad \Rightarrow (n+p)nQ + (n+p)pb^2 - p^2 b^2 \geq D(n+p)^2 \\
&\quad \Rightarrow (n+p)nQ + npb^2 \geq D(n+p)^2 \\
&\quad \Rightarrow (n+p)nD + npb^2 \geq D(n+p)^2 \\
&\quad \Rightarrow npb^2 \geq npD + p^2 D \\
&\quad \Rightarrow b^2 \geq D + pD/n \geq \frac{n+1}{n} D
\end{aligned}$$

□

Proof (of Theorem 3). Let $n = |C_i|$ and $e = C_{i+1} \ominus C_i$, $f = C_{i+2} \ominus C_{i+1}$. This theorem is obvious if $n < 2$ so we can assume that $n \geq 2$. Without loss of generality we can assume that $\mathrm{avg}(C_i) = 0$ and that $e > 0$.

Let Q be the sum of squares of C_i. Suppose there exists a $j > i$ such that $\mathrm{var}(C_j) > \mathrm{var}(C_i)$. Then let $J = C_j$. Let a be the largest value in $J \ominus C_i$ and b be the smallest value. Clearly $a = e$ and $b \leq f$. a satisfies the conditions in Lemma 3. If $a^2 \leq \frac{Q}{n}$ then this theorem is obviously true, so we can assume $a^2 > \frac{Q}{n}$. Since f is at least as far from $\mathrm{avg}(C_{i+1})$ as b, Lemma 1 implies that $\mathrm{var}(C_i \oplus \{a, b\}) \leq \mathrm{var}(C_i)$. Thus if $b \leq -\sqrt{\frac{Q}{n}}$ then by Lemma 1,

$$\mathrm{var}(C_i \oplus \{a, b\}) \geq \mathrm{var}\left(C_i \oplus \left\{\sqrt{\frac{Q}{n}}, -\sqrt{\frac{Q}{n}}\right\}\right) = \mathrm{var}(C_i)$$

So $b > -\sqrt{\frac{Q}{n}}$, which means that $|a| > |b|$ and $a + b > 0$.

From J we will inductively construct a multiset J^* such that $\mathrm{var}(J^*) \geq \mathrm{var}(J)$. Let $J^0 = J$ and given J^k, pick some element c from the set

$$H = \{x \in J^k \ominus C_i : \ a > x > b\}$$

If $c \geq \mathrm{avg}(J^k)$ then we know $a > c$ and so let $J^{k+1} = J^k \ominus \{c\} \oplus \{a\}$. Similarly, if $c \leq \mathrm{avg}(J^k)$ then we know $c > b$ and so let $J^{k+1} = J^k \ominus \{c\} \oplus \{b\}$. By Lemma

1, $\mathrm{var}(J^{k+1}) \geq \mathrm{var}(J^k)$. If H is empty and we cannot choose an element c, then let $J^* = J^k$. Clearly $J^* = C_i \oplus (p \cdot \{a, b\}) \oplus (q \cdot \{x\})$ for some integers p and q, where x is either a or b.

Now suppose $q \geq 1$ and x is the element b. If $p = 0$ then a is further away from $\mathrm{avg}\, J^* \ominus \{b\}$ than b. If $p \geq 1$ then

$$\mathrm{avg}\, J^* \ominus \{b\} \leq p(a+b)/(n+2p+q) \leq (a+b)/2$$

and so a is also further away from $\mathrm{avg}\, J^* \ominus \{b\}$ than b. By Lemma 1, $\mathrm{var}(J^* \ominus \{b\} \oplus \{a\}) \geq \mathrm{var}(J)$ and

$$J^* \ominus \{b\} \oplus \{a\} = C_i \oplus ((p+1) \cdot \{a, b\}) \oplus ((q-2) \cdot \{b\})$$

Note this is also true when $q = 1$ if we interpret $B \oplus (-1) \cdot A$ as $B \ominus A$. Using this argument repeatedly, we get the set

$$C_i \oplus ((p + \lfloor q/2 \rfloor) \cdot \{a, b\}) \oplus ((q \mod 2) \cdot \{a\})$$

which has variance $\geq \mathrm{var}(J^*)$. Thus, without loss of generality we can assume that x is the element a.

By hypothesis, $\mathrm{var}(C_i) - \mathrm{var}(C_{i+2}) \geq 0$, and so

$$\begin{aligned}
\mathrm{var}(C_i) - \mathrm{var}(C_{i+2}) &= \frac{Q}{n} - \frac{Q + a^2 + b^2}{n+2} + \frac{(a+b)^2}{(n+2)^2} \\
&= \frac{2Q - na^2 - nb^2}{n(n+2)} + \frac{(a+b)^2}{(n+2)^2} \quad \geq 0
\end{aligned}$$

which implies that $\frac{2Q}{n} - a^2 - b^2 + \frac{(a+b)^2}{n+2} \geq 0$.

Let $I = C_i \oplus (p \cdot \{a, b\})$. Then we have that

$$\begin{aligned}
\mathrm{var}(C_i) - \mathrm{var}(I) &= \frac{Q}{n} - \frac{Q + pa^2 + pb^2}{n+2p} + \frac{p^2(a+b)^2}{(n+2p)^2} \\
&= \frac{2pQ - npa^2 - npb^2}{n(n+2p)} + \frac{p^2(a+b)^2}{(n+2p)^2} \\
&= \frac{p}{n+2p}\left(\frac{2Q}{n} - a^2 - b^2 + \frac{p(a+b)^2}{n+2p}\right) \\
&= \frac{p}{n+2p}\left(\frac{2Q}{n} - a^2 - b^2 + \frac{p(n+2)}{n+2p}\frac{(a+b)^2}{n+2}\right) \\
&= \frac{p}{n+2p}\left(\frac{2Q}{n} - a^2 - b^2 + \frac{pn+2p}{n+2p}\frac{(a+b)^2}{n+2}\right) \\
&= \frac{p}{n+2p}\left(\frac{2Q}{n} - a^2 - b^2 + \frac{(a+b)^2}{n+2}\right) = \\
&\geq \frac{p}{n+2p}\left(\mathrm{var}(C_i) - \mathrm{var}(C_{i+2})\right) \geq 0
\end{aligned}$$

Thus $\mathrm{var}(I) \leq \mathrm{var}(C_i)$ and $J^* = I \oplus (q \cdot \{a\})$. By Lemma 4, it is only possible for $\mathrm{var}(J^*) > \mathrm{var}(C_i)$ if $(a - \mathrm{avg}(I))^2 > \frac{n+1}{n}\mathrm{var}(C_i)$. Since $a + b \geq 0$, we have $a \geq \mathrm{avg}(I) \geq 0$ and so $a \geq (a - \mathrm{avg}(I)) \geq 0$.

From Lemma 3

$$\mathrm{var}(C_i)\frac{n+1}{n} \geq a^2 \geq (a - \mathrm{avg}(I))^2$$

Therefore $\mathrm{var}(J^*)$ cannot be larger than $\mathrm{var}(C_i)$, and so when the variance of a set is not less than the variance of either its two successors, its variance is not less than the variance of any of its successors.

□

Given Theorems 2 and 3, we now know that Algorithm 2 is correct. However, it is important to note that it is also optimal.

Theorem 4. *If there exists a set X with $\mathrm{var}(X) \geq c$, then Algorithm 2 will find the shortest path to any node whose variance $\geq c$.*

Proof. Let C_i be the chain of sets from the proof of Theorem 2. We call a node X *quick* if $\mathrm{var}(X) \geq c$ and if $\forall Y \in (B, T)$, then $|Y| \leq |X| \Rightarrow \mathrm{var}(Y) < c$. We need to show that for some i, C_i is quick.

Quickness is a maximality property. Therefore, to complete the proof, we simply carry out the proof of Theorem 2, substituting "quick" for "maximal variance".

□

3.2 A Small Witness for Variance

Now that we know how to find a large witness $\mathcal{Y}$, we need an algorithm to find a witness $\mathcal{Z}$ such that $\mathrm{var}(\mathcal{Z}) > c \Rightarrow \mathrm{var}(X) > c$ for all $X \in \mathcal{A}$. This is a much more difficult problem. To see why, note that we indirectly used the following property to show that a greedy algorithm worked for finding maximal variance.

Lemma 5. *For any constant h, if $\mathrm{var}(Y) \geq h$ and $\mathrm{var}(X) \geq h$ then $\mathrm{var}(X \oplus Y) \geq h$.*

Proof. For convenience, let $A = \sum_{y_i \in Y} y_i$ and $B = \sum_{y_i \in Y} y_i^2$ and $n = |Y|$. Also let $C = \sum_{x_i \in X} x_i$ and $D = \sum_{x_i \in X} x_i^2$ and $m = |X|$. Since variance is invariant under translation, we can assume that the elements of X and Y are nonnegative. We know that

$$\mathrm{var}(Y) = \frac{B}{n} - \frac{A^2}{n^2} \geq h \Rightarrow B \geq nh + \frac{A^2}{n}$$
$$\mathrm{var}(X) = \frac{D}{m} - \frac{C^2}{m^2} \geq h \Rightarrow D \geq mh + \frac{C^2}{m}$$

Therefore

$$\begin{aligned} B + D &\geq nh + \frac{A^2}{n} + mh + \frac{C^2}{m} \\ &= h(n+m) + \frac{mA^2 + nC^2}{mn} \end{aligned}$$

$$
\begin{aligned}
&= h(n+m) + \frac{(mn+m^2)A^2 + (mn+n^2)C^2}{mn(m+n)} \\
&\geq h(n+m) + \frac{mnA^2 + mnC^2 + 2mnAC}{mn(m+n)} \\
&= h(n+m) + \frac{A^2 + C^2 + 2AC}{m+n} \\
&= h(n+m) + \frac{(A+C)^2}{m+n}
\end{aligned}
$$

This implies $h \leq \frac{B+D}{m+n} - \frac{(A+C)^2}{(m+n)^2} = \mathrm{var}(X \oplus Y)$.

□

This allowed us to add elements that had the largest effect on the variance without worrying too much about the structure of the set we were creating. The constraint $\mathrm{var}(X) < c$ does not have a similar property and this suggests that a greedy algorithm to find a witness Z does not exist. Thus the intuitive algorithms in Section 1.2 do not work. Instead, the following lemma describes what a witness should look like.

Lemma 6. *For any element X in (B, T) with minimal variance, there exist two non-negative integers L and R such that*

$$
\begin{aligned}
X = B(n) &\oplus \max_{L}\{y \in \mathrm{Free} :\ y \leq \mathrm{avg}(X)\} \\
&\oplus \min_{R}\{y \in \mathrm{Free} :\ y > \mathrm{avg}(X)\}
\end{aligned}
$$

Proof. The proof is analogous to that of Lemma 2 and again assumes that ties are broken according to the convention $>_\kappa$.

□

In other words, if we order the points in Free on a line, X contains $B(n)$ and only the points in some window of size $L + R$ over this line.

It is clear from Lemma 6 that if there exists a set X with $\mathrm{var}(X) < c$ then there exists a witness $\mathcal{Z}$ with $\mathrm{var}(\mathcal{Z}) < c$ and that $\mathcal{Z}$ is the multiset union of $B(n)$ and some window over $\mathrm{Free}(n)$. The next lemma states that this window does not have to be too big.

Lemma 7. *Let C be a set, $a \in C$, $n = |C|$ and let $D \geq \mathrm{var}(C)$. If $(a - \mathrm{avg}(C \ominus \{a\}))^2 > \frac{n}{n-1}D$ then $\mathrm{var}(C \ominus \{a\}) < D$*

Proof. We prove the lemma by contradiction. Assume $var(C \ominus \{a\}) \geq D$. Without loss of generality, assume $avg(C \ominus \{a\}) = 0$. Let Q be the sum of squares of $C \ominus \{a\}$. So, we have $Q \geq (n-1)D$ and $(n-1)a^2 > nD$. Then $var(C) = (Q + a^2)/n - a^2/n^2 = (nQ + (n-1)a^2)/n^2 > (nQ + nD)/n^2 \geq (n(n-1)D + nD)/n^2 = D$. Thus, $var(C) > D$, contradiction, therefore $var(C \ominus \{a\}) < D$.

□

We can derive an easy $O(|\,\mathrm{Free}\,|^2)$ search algorithm using Lemma 6, but it is possible to do better. The algorithm to determine if there is a set $\mathcal{Z}$ with $\mathrm{var}((Z)) \leq c$ is a

two-step sliding window algorithm. In the first step, we start with a window whose right endpoint is the largest element in Free. We slide the window to the left until the right endpoint is no longer greater than or equal to $\mathrm{avg}(B(n))$. If we have not found a witness, we repeat the same thing, but on the left hand side. We can reflect all points around the y-axis (i.e. multiply them by -1) without affecting the variance of any set, and so by symmetry we only need to describe the first step of the algorithm.

We can use Lemma 7 to define a suitable window. Note that the window size depends on the number of points in the window. We also have no guarantees that the window associated with the witness (by Lemma 6) is the same size as the algorithm's window. Therefore we must be careful about checking for witnesses to avoid a quadratic search algorithm. Once again, the algorithm assumes the elements of Free are maintained in sorted order. Let $\mathcal{F}$ be the array of elements of Free sorted in descending order. Given the index r of the right endpoint, we want the largest ℓ such that

$$T_{r,\ell} = \{\mathcal{F}[r], \mathcal{F}[r+1], \ldots, \mathcal{F}[\ell]\} \oplus B(n) \tag{2}$$

satisfies the following properties.

1. $\mathcal{F}[r] - \mathrm{avg}(T_{r,\ell} \ominus \mathcal{F}[r]) \leq \sqrt{ck/(k-1)}$ - where $|T_{r,\ell} \ominus \mathcal{F}[r]| = k-1$
2. $\mathcal{F}[r] - \mathcal{F}[\ell] \leq 2\sqrt{ck/(k-1)}$

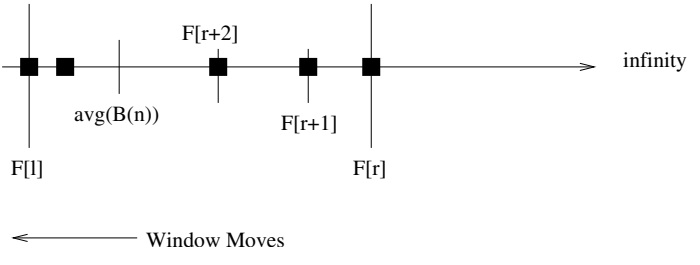


Fig. 2. The Window

The first condition states that we do not want the right endpoint to be further away from the average (without the endpoint) than allowable by the hypothesis of Lemma 7. Thus given a right endpoint, we know what the smallest allowable average is. Condition 2 states that we do not want the left endpoint to be further away from this quantity than is allowable by Lemma 7. The window defined by r and ℓ is W_r, our *target window for* r.

The window associated with a set M of minimal variance is a subset of W_r for some r. To see why, suppose this were not the case. Then one of the two conditions is false. This means that either the left endpoint or the right endpoint of M's window is too far from $\mathrm{avg}(M)$, so by Lemma 7 we can remove this endpoint and decrease the variance further. Hence all of these lemmas provide us with the following result.

Theorem 5. *Let $\mathcal{Z}$ be a witness which has a window associated to it as in Lemma 6. Then there exists a witness $\mathcal{Z}'$ whose window is a subset of the window of $\mathcal{Z}$, and the window of $\mathcal{Z}'$ is contained in a target window W_r for some r.*

For some choices of r, it may not be possible to get a window which brings the average close enough to $\mathcal{F}[r]$. In this case set ℓ to be the largest integer such that the set in equation (2) satisfies

1*. $\mathcal{F}[r] - \mathcal{F}[\ell] \leq \sqrt{ck/(k-1)}$ but
$\mathcal{F}[r] - \text{avg}(T_{r,\ell} \ominus \mathcal{F}[r]) > \sqrt{ck/(k-1)}$

The intuition behind this idea is that we add all the elements that are greater than or equal the minimum average allowed by Lemma 7. If this cannot get the average (without $\mathcal{F}[r]$) high enough, then no window will. But if this does move the average close enough, we can keep adding elements that satisfy condition 2. In any case we can move the left endpoint to the left until we reach ℓ and we will recognize ℓ as soon as we see it. Note that this does not change the truth of Theorem 5 since this added definition only enlarges windows that would have had length 0 otherwise.

The problem with target windows is that sliding this window to the left may cause the left endpoint to move to the right. In other words, it is possible that $W_{r+1} \subseteq W_r$ and therefore sliding this window over $\mathcal{F}$ may require an $O(|\mathcal{F}|^2)$ computation. For example, suppose $\mathcal{F}[r] = \mathcal{F}[r+1]$ and equality holds in condition 1. Sliding the window over would cause the average to move further away from $\mathcal{F}[r+1]$ and thus violate condition 1. Because of this our algorithm will maintain a window larger than the target window by simply leaving the left endpoint fixed in such cases. Furthermore, if the left endpoint is defined by condition 1*, the left endpoint will never move to the right.

Algorithms 3 and 4 show how to slide the window.

Algorithm 3 : SlideWindow

Require: r, ℓ
1: $r \leftarrow r + 1$
2: **if** $\text{var}(T_{r,\ell}) \leq c$ **then**
3: RETURN (true, r, ℓ)
4: **end if**
5: $\ell \leftarrow ExpandWindow(r, \ell)$
6: **if** $\text{var}(T_{r,\ell}) \leq c$ **then**
7: RETURN (true,r,ℓ)
8: **else**
9: RETURN (false,r,ℓ)
10: **end if**

Notice that ExpandWindow (Algorithm 4) checks the variance as it moves the left boundary. Since the left and right endpoints shift in one direction only, the variance can be computed in constant time by incrementally maintaining the number of elements in the window, their sum, and the sum of their squares. If the algorithm finds a witness, then it returns immediately and SlideWindow (Algorithm 3) will know this.

The main algorithm is shown in Algorithm 5, where we assume, for simplicity, that $\mathcal{F}[-1] = \infty$. This algorithm returns true if there is an element with $\text{var} \leq c$ and false otherwise.

As this algorithm is a little involved, we present the following illustrative example.

Algorithm 4 : ExpandWindow

Require: r, ℓ
1: **while** $\ell < |\mathcal{I}| - 1$ **do**
2: $k \leftarrow |T_{r,\ell+1}|$
3: **if** $\mathcal{F}[r] - \mathcal{F}[\ell+1] \leq \sqrt{ck/(k-1)}$ **then**
4: $\ell \leftarrow \ell + 1$ (Condition 1*)
5: **else if** $\mathcal{F}[r] - \text{avg}(T_{r,\ell+1} \ominus \mathcal{F}[r]) \leq \sqrt{ck/(k-1)}$, $\mathcal{F}[r] - \mathcal{F}[\ell+1] \leq 2\sqrt{ck/(k-1)}$ **then**
6: $\ell \leftarrow \ell + 1$
7: **else**
8: BREAK
9: **end if**
10: **if** $\text{var}(T_{r,\ell}) \leq c$ **then**
11: BREAK
12: **end if**
13: **end while**
14: RETURN ℓ

Algorithm 5 : SmallVar

1: $r \leftarrow -1, \ell \leftarrow 0$
2: **while** $\mathcal{F}[r] \geq \text{avg}(B(n))$ **do**
3: $(result, r, \ell) \leftarrow SlideWindow(r, \ell)$
4: **if** $result =$true **then**
5: RETURN true
6: **end if**
7: **end while**
8: Repeat with window to the left of $\text{avg}(B(n))$.
9: RETURN false

Example 2. Let $B = \{-20, 20\}$ and $\mathcal{F} = \text{Free} = \{30, 10, 0, -10, -30\}$. Also let $c = 200$. Our algorithm starts with $r = -1$, $\ell = 0$. Clearly $\mathcal{F}[-1] = \infty$ is larger than $\text{avg}(B(n))$, so we slide. We now set $r = 0$ and hence

$$T_{r,\ell} = \{30\} \oplus B = \{-20, 20, 30\}$$

As $\text{var}(T_{r,\ell}) = 700$, we expand the window.

We start with $k = 2$, $\mathcal{F}[r] = 30$, and $\mathcal{F}[\ell+1] = 10$. Then $\mathcal{F}[r] - \mathcal{F}[\ell+1] = 20$ while $\sqrt{ck/(k-1)} = 20$. So we increment ℓ by one. Furthermore, as $\text{var}(T_{r,\ell}) \approx 466.7 > 250$, we have not necessarily found a set of minimal variance yet. So we continue the loop.

Now $k = 3$, $\mathcal{F}[r] = 30$, and $\mathcal{F}[\ell+1] = 0$. Then $\mathcal{F}[r] - \mathcal{F}[\ell+1] = 30$ while $\sqrt{ck/(k-1)} \approx 17.3$, so line 3 of ExpandWindow is no longer true. Furthermore,

$$\text{avg}(T_{r,\ell+1} - \mathcal{F}[r]) = \text{avg}(\{-20, 0, 10, 20\}) = 2.5$$

and hence $\mathcal{F}[r] - \text{avg}(T_{r,\ell+1} - \mathcal{F}[r]) = 27.5$. This is not less than $\sqrt{ck/(k-1)} \approx 17.3$, so we are done with this iteration of ExpandWindow.

Again, $\mathcal{F}[0] = 30 > \text{avg}(B(n))$, so we slide $r = 1$, $\ell = 1$. As $\text{var}(T_{r,\ell}) \approx 433.3$, we expand the window. We start with $k = 2$, $\mathcal{F}[r] = 10$, and $\mathcal{F}[\ell + 1] = 0$. As $10 < \sqrt{ck/(k-1)} = 20$, we increment ℓ by one. As $\text{var}(T_{r,\ell}) \approx 291.7 > 250$, we continue the loop.

Now we have $\ell = 2$, $k = 3$. $\mathcal{F}[r] - \mathcal{F}[\ell + 1] = 20$, while $\sqrt{ck/(k-1)} \approx 17.3$. However,

$$\mathcal{F}[r] - \text{avg}(T_{r,\ell+1} - \mathcal{F}[r]) = 10 - \text{avg}(\{-20, -10, 0, 20\}) = 12.5$$

Furthermore, $\mathcal{F}[r] - \mathcal{F}[\ell+1] \leq 2\sqrt{ck/(k-1)}$, so now we have satisfied the condition on line 5 of ExpandWindow. We increment ℓ by one again. Now we have that

$$T_{r,\ell} = \{-20, -10, 0, 10, 20\}$$

As $\text{var}(T_{r,\ell}) = 250$, we continue with our loop.

However, it is easy to check that the set $T_{r,\ell} = \{-20, -10, 0, 10, 20\}$ is the one of minimal variance. Hence the remaining steps of the algorithm will determine that none of the itemsets in this lattice satisfy the constraint, and so we can prune them all.

Note that this algorithm runs in $O(|\mathcal{F}|)$ time because variance is computed once each time we move the right endpoint and once each time we move the left endpoint. Although SlideWindow is called $O(|\mathcal{F}|)$ times, it either does not move the left endpoint (hence doing a constant unit of work) or it moves the left endpoint to the left. Thus overall it does $O(|\mathcal{F}|)$ + $O(|\mathcal{F}|)$=$O(|\mathcal{F}|)$ units of work.

Correctness is given by the following theorem.

Theorem 6. *If there is some set in $\mathcal{A}$ with variance not greater than c then SmallVar (Algorithm 5) will find one such set.*

Lemma 8 (The Expanding Window). *If there exists a witness $\mathcal{Z}$ with $\text{var}(\mathcal{Z}) \leq c$ and and window defined by right endpoint a and left endpoint b then if $d > b$ and $\text{avg}(\mathcal{Z}) - \mathcal{F}[d] \leq \sqrt{c(|\mathcal{Z}| + 1)/|\mathcal{Z}|}$ then there is a witness $\mathcal{Z}'$ with $\text{var}(\mathcal{Z}') \leq c$ and window defined by endpoints a and d.*

Proof. If $d = b + 1$ then this is obvious by Lemma 4. If $d > b + 1$ then this is true by induction.

□

Proof (of Theorem 6).

The algorithm's window can be in 3 states.

State 1: The window satisfies condition 1*.
State 2: The window satisfies conditions 1 and 2.
State 3: The left endpoint didn't move and the previous window satisfied conditions 1 and 2 (otherwise the current window satisfies condition 1* and is in state 1).

It is clear that if the algorithm is in state 1, sliding the window will only move it to states 1 or 2. If it is in state 2 then it will only go to states 2 or 3. Finally, if it is in state 3, it can go to any other state.

We already know that there exists a witness $\mathcal{Z}$ that is characterized by Lemma 6 – it has an associated window. By Lemma 7 we can restrict ourselves to only look for witnesses whose right endpoints are close enough to the average of the rest of the witness (i.e. satisfy condition 1). Without loss of generality we can assume that the window is minimal in the sense that no witness has an associated window that is a proper subset of this. By symmetry, we can also assume that the right endpoint of this window is not less than $\mathrm{avg}(B(n))$. Thus by sliding the algorithm's window over to the left, at some point the right endpoint r of the algorithm window will also be the right endpoint of the minimal witness.

When this happens, by Theorem 5, the target window, W_r, contains the window of the minimal witness. When the algorithm window is in states 2 or 3 and has right endpoint r, it will contain the target window and therefore the window of the minimal witness. Since r is also the right endpoint of the window of the minimal witness, the algorithm window with right endpoint r can not satisfy condition 1* and so it will not be in state 1. Thus it is sufficient to prove two things.

(i) If the algorithm is in states 2 or 3 and a witness's window is contained within the algorithm's current window *and* both windows have the same right endpoint, then the algorithm will find this out.
(ii) If the window of a witness is inside the algorithm's window and the algorithm's window is in state 1, then when we slide the window it will still contain the window of a witness.

We prove property (ii) first. If the window satisfies condition 1* and a witness has its window inside the algorithm's window, then we test if $\mathrm{var}(T_{r,\ell}) \leq c$. If the variance is greater than c then we claim there is a witness $\mathcal{Z}$ whose window is inside of this window, but that the window of the witness has a right endpoint different from r. If there was a witness whose right endpoint was r then by condition 1*, $\mathcal{F}[r]$ is far away from $\mathrm{avg}\,\mathcal{Z} \ominus \{\mathcal{F}[r]\}$, and by Lemma 7 $\mathcal{Z} \ominus \{\mathcal{F}[r]\}$ is also a witness. Furthermore, the right endpoint of this witness's window will still be larger than $\mathrm{avg}(B(n))$ by the minimality assumption. At this point the algorithm would slide the window over and the window of $\mathcal{Z}$ would still be contained in the algorithm's window.

Property (i) follows by induction. The inductive hypothesis will also maintain the fact that if the algorithm window is in state 3 then if the window of a witness is inside the algorithm's window, then we can extend the left boundary of the witness's window to the left endpoint of the algorithm's window and so create a set with variance not greater than c.

The base case of the induction is the beginning of the algorithm. The first window must be either in state 1 or 2. If it is in state 1 then property (i) is vacuously true for the first algorithm window. If it is in state 2, let r be the right endpoint. ExpandWindow will initially start with the with right and left endpoints r and r and will check if we have a witness whose window has right endpoint r every time it moves the left endpoint. Thus we will know if a witness's window is inside the algorithm window and shares a right endpoint with it.

Let $g(r)$ be the largest integer such that

$$\mathcal{F}[r] - \mathcal{F}[g(r)] \leq \sqrt{ck/(k-1)}$$

where $k = |T_{r,g(r)}|$. Note that $g(r) = \ell$ if condition 1* holds and $g(r) \leq \ell$ for states 2 and 3. We will left r_1, ℓ_1 be the boundaries of the previous algorithm window and r_2, ℓ_2 be the boundaries of the current window. As we can treat any occurrence of state 1 as the initial state, the inductive step has five cases.

Case (I). The current window is in state 2 and the previous window is in state 1. Since the algorithm explicitly checks for the variance of $T_{r_2,\ell_1}, \ldots T_{r_2,\ell_2}$, we must show that we do not miss anything by not checking the variance of $T_{r_2,r_2}, \ldots, T_{r_2,\ell_1-1}$. Since $\ell_1 = g(r_1) \leq g(r_2)$ it is sufficient to show that we do not need to check the variance of $T_{r_2,r_2}, \ldots, T_{r_2,g(r_2)-1}$. Suppose there is a witness $\mathcal{Z}$ whose window has right endpoint r and left endpoint L between r_2 and $g(r_2) - 1$. Because the witnesses must satisfy condition 1, the distance between $\operatorname{avg} Z$ and $\mathcal{F}[g(r_2)]$ is not greater than the distance between $\mathcal{F}[r]$ and $\mathcal{F}[g(r_2)]$. Since $|\mathcal{Z}| \leq |T_{r_2,g(r_2)}|$ and

$$(|\mathcal{Z}| + 1)/|\mathcal{Z}| \geq (|T_{r_2,g(r_2)}| + 1)/|T_{r_2,g(r_2)}|$$

the $g(r_2)$ and $\mathcal{Z}$ satisfy the conditions of the Expanding Window Lemma. So $T_{r_2,g(r_2)}$ would also be a witness and would be explicitly checked by the algorithm.

Case (II). Both the current window and the previous window are in state 2. Once again we must show that we do not miss anything by not checking the variance of the sets $T_{r_2,r_2}, \ldots, T_{r_2,\ell_1-1}$. Suppose there is a witness $\mathcal{Z}$ whose window has right endpoint r and left endpoint L between r_2 and $\ell_1 - 1$. If $L \leq g(r_2)$ then we use the same arguments as in Case (I). Otherwise $L > g(r_2) \geq g(r_1)$ and so $T_{r_1,L}$ must satisfy Condition 1. But if $T_{r_2,L}$ were a witness then by Lemma 4 so is $T_{r_1,L}$ and this would have been discovered by the inductive hypothesis.

Case (III). The current window is in state 2 and the previous window is in state 3. Again we show that we do not miss anything by not checking the variance of $T_{r_2,r_2}, \ldots,$ T_{r_2,ℓ_1-1}. Suppose there is a witness $\mathcal{Z}$ whose window has right endpoint r and left endpoint L between r_2 and $\ell_1 - 1$. Then the window of the witness is bounded by r_1, ℓ_1 and so by the inductive hypothesis we can extend its left endpoint and so $T_{r_2,L}$ will have $\operatorname{var} \leq c$. This set will then be checked by the algorithm.

In the last two cases, the current state is 3. We must show that any witness whose window fits inside the algorithm's current window can have its left endpoint extended to the algorithm's window's left endpoint. From this it follows that if a witness has right endpoint equal to r_2, then the variance of T_{r_2,ℓ_2} is $\leq c$ and this is checked by the algorithm. If $\ell_2 \leq g(r_2)$ then we should actually be in state 3. Therefore $\ell_2 > g(r_2)$.

Case (IV). The current window is in state 3 and the previous window is in state 2. In this case $\ell_1 = \ell_2$ and T_{r_1,ℓ_2} satisfies both conditions 1 and 2 but T_{r_2,ℓ_2} does not. Since the fewer the elements in the window, the larger $2\sqrt{ck/(k-1)}$ is, condition 2 is automatically satisfied. Hence condition 1 must be false. Thus if there is a witness whose window has endpoints R, L where $r_2 \leq R \leq L < \ell_2$ and if $\operatorname{avg}(T_{R,L}) \geq \mathcal{F}[\ell_2]$ then

$$|\operatorname{avg}(T_{R,L}) - \mathcal{F}[\ell_2]| \leq \sqrt{ck/(k-1)}$$

where $k - 1 = |T_{r_1,\ell_2}|$. Since $k^* = |T_{R,L}| < |T_{r_1,\ell_2}|$, we have $|\operatorname{avg}(T_{R,L}) - \mathcal{F}[\ell_2]| \leq \sqrt{c(1 + |T_{R,L}|)/|T_{R,L}|}$ and we can use the Expanding Window Lemma. If, on the other

hand $\text{avg}(T_{R,L}) < \mathcal{F}[\ell_2]$ then $\mathcal{F}[\ell_2]$ is closer to $\text{avg}(T_{R,L})$ than $\mathcal{F}[R]$ is and because of our restriction on witnesses we can again use the Expanding Window Lemma to expand the left endpoint from L to ℓ_2.

Case (V). Both the current window and the previous window are in state 3. This follows trivially from the inductive hypothesis since $r_1 \leq r_2$ and $\ell_1 = \ell_2$ and so the current algorithm window is contained in the previous algorithm window.

□

3.3 Variance in Higher Dimensions

The definitions of average and variance easily extend to multi-dimensional spaces. For a d-dimensional vector space ($d > 1$), let $\overrightarrow{x} = (x_1, x_2, \ldots, x_d)$.

$$\text{avg}(S) = \frac{1}{|S|} \sum_{\overrightarrow{x} \in S} \overrightarrow{x} = \frac{1}{|S|} \left(\sum_{\overrightarrow{x} \in S} x_1, \ldots, \sum_{\overrightarrow{x} \in S} x_d \right)$$

and

$$\text{var}(S) = \frac{\sum_{\overrightarrow{x} \in S} ||\overrightarrow{x} - \text{avg}(S)||^2}{|S|}$$

where $||\overrightarrow{x} - \overrightarrow{y}||$ represents the Euclidean distance between $\overrightarrow{x}$ and $\overrightarrow{y}$. From Lemmas 2 and 6 we see that witnesses for variance in one-dimension have a nice structure because all the points can be ordered. This is no longer true in higher dimensions and so the algorithms for variance fall apart. To see this, let us extend Algorithm 2 to two-dimensions: given a node n, we keep adding to $B(n)$ the element in $\text{Free}(n)$ that is furthest from the current average. Suppose $B(n)$ consists of the point $(0, 0)$ and $\text{Free}(n)$ consists of the points $\{(5, 0),\ (0, 5),\ (3.01, 4.01)\}$. The point $(3.01, 4.01)$ is furthest away from $(0, 0)$ and is added to our candidate witness: $\{(0, 0),\ (3.01, 4.01)\}$ and its variance is 6.28505. After this the point $(5, 0)$ is added (the variance increases to 7.79782) and finally $(0, 5)$ is added and the variance becomes 9.70129. However, the algorithm never considers the set $\{(0, 0),\ (5, 0),\ (0, 5)\}$, which achieves the maximum variance 11.1111.

4 Heuristics

In practice, we do not always want to run a linear time (or greater) search algorithm to find a witness. Although a linear time algorithm may allow us to prune away an exponential number of sets, sometimes our negative witness satisfies the constraint. In those cases we cannot prune away $\mathcal{A}(n)$ and our time is wasted.

There are two techniques to deal with this problem. It may be possible to amortize the cost of the search by maintaining state that avoids redundant computation. For example, when we showed how to mine average, we maintained the average of the witness incrementally instead of recomputing it every time.

When amortization is not possible, we can use heuristics to tell us when to run the search algorithm. For example, if we are mining with a constraint $\text{var}(S) < c$ then we want to prune sets with variance greater than or equal to c. We can use the observation

that the higher the value of $\mathrm{var}(B(n))$, the less likely that $\mathrm{var}(S) < c$ for some $S \in \mathcal{A}$. Thus we can set some threshold τ on $\mathrm{var}(B(n))$, and if the variance is larger than the threshold we search for witness. A similar approach works for the constraint $var(S) > c$. A heuristic can also be based on some precomputed statistics.

When using such constraints we can also benefit from a heuristic which chooses the order in which elements are added to $B(n)$ to create children of $B(n)$. Thus we can try to arrange it so that we see many nodes n for which $B(n)$ has high variance. One such heuristic could be to order all items (in descending order) by their distance from the overall average of $\mathcal{I}$. This is very similar to the approach taken by the convertible algorithms [18].

We should note that in most cases amortization is possible by avoiding redundant computation. For example, suppose we have the constraint $\mathrm{var}(S) > c$ and that we are currently examining a node n. We run Algorithm 2 but find a set with variance $> c$. We cannot prune the subtree rooted at n but we can amortize the cost of the search. Let $a_1, a_2, \ldots, a_k$ be the elements that were added to $B(n)$ by the algorithm in that order. Because we cannot prune, we will eventually have to visit the nodes represented by the sets $B(n) \oplus \{a_1\}$, $B(n) \oplus \{a_1, a_2\}$, $B(n) \oplus \{a_1, a_2, a_3\}$, etc, in order to traverse the subtrees rooted at those nodes.

If we run the algorithm at those nodes we will get the same witness as when we ran it at n. Thus at those nodes we can choose not to run the algorithm. Since we visit these nodes anyway, the amortized cost of the search is at most a constant per node plus the cost of maintaining this information. By doing a depth-first traversal of nodes, we can arrange it so that the next k nodes that the algorithm traverses are these k nodes for which we already know the result of the witness search. In this case, maintaining extra state is constant per node. Otherwise we just need to maintain two numbers - the smallest a_i that is at least $\mathrm{avg}(B(n))$ and the largest a_j that is less than $\mathrm{avg}(B(n))$. Then whenever we come to a node of the form $B(n) \oplus J$ (where $J \subseteq \{a_1, a_2, \ldots, a_k\}$) we do not have to run the algorithm again since the same witness is also valid.

Similarly, if we had the constraint $\mathrm{var}(S) \leq c$, we run Algorithm 5 on a node n only to discover a set with variance $\leq c$. Again we cannot prune the subtree rooted an n. We let $a_1, \ldots, a_k$ be the consecutive sequence of points that define the window of the witness we have found. Clearly this would also be a witness when we examine a node represented by $B(n) \oplus J$ (where $J \subseteq \{a_1, \ldots, a_k\}$). We still have to traverse to these nodes to examine their subtrees, however we do not need to run the algorithm again. To maintain this state we need just two numbers – the left and right endpoints of the window of this witness.

5 Related Work

Agrawal et al. first introduced the problem of mining frequent itemsets as a first step in mining association rules [1]. They also considered item constraints such as an item must or must not be contained in an association rule. Agrawal and Srikant introduced the Apriori algorithm and some variations of it [3,2]. Srikant et al. generalized this mining problem to item constraints over taxonomies [20]. Other types of constraints were introduced later by Ng et al. [15,14]. These papers introduced the concepts of antimono-

tone and succinct constraints and presented methods for using them to prune the search space. These classes of constraints were also studied in the case of 2-variable constraints [12] and along with monotone constraints were further generalized and studied by Pei et al. [18,16]. Boulicant and Jeudy presented algorithms for mining frequent itemsets with both antimonotone and non-antimonotone constraints [6,7]. However they assume that the minimal itemsets satisfying the monotone constraint are easy to compute, that the minimum size of such itemsets is one, and that there is no gap in the sizes of itemsets that satisfy all the constraints - assumptions that frequently do not hold. This problem was also given a theoretical treatment by Gunopoulos et al. [10]. DualMiner is the first algorithm that simultaneously uses both monotone and antimonotone constraints for pruning the search space [8]. Some recent papers study the problem in the context of multi-attribute data of high dimensionality [19] or take another approach to the problem, such as not pushing the constraints deeply into the mining process, but enforcing the constraints in a final phase [11]. Another approach is a pre-processing algorithm called ExAnte which reduces the search space and the size of the transaction database [5]. This technique has also been pushed deeper into the mining process [4]. Other papers present specializations of previous algorithms, based on FP-trees [13] or based on projected databases [17].

References

1. R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In P. Buneman and S. Jajodia, editors, *Proc. SIGMOD 1993*, pages 207–216. ACM Press, 1993.
2. R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast Discovery of Association Rules. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, chapter 12, pages 307–328. AAAI/MIT Press, 1996.
3. R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In J. B. Bocca, M. Jarke, and C. Zaniolo, editors, *Proc. VLDB 1994*, pages 487–499. Morgan Kaufmann, 1994.
4. F. Bonchi, F. Giannotti, A. Mazzanti, and D. Pedreschi. Examiner: Optimized level-wise frequent pattern mining with monotone constraint. In *ICDM*, pages 11–18. IEEE Computer Society, 2003.
5. F. Bonchi, F. Giannotti, A. Mazzanti, and D. Pedreschi. Exante: Anticipated data reduction in constrained pattern mining. In N. Lavrac, D. Gamberger, H. Blockeel, and L. Todorovski, editors, *PKDD*, volume 2838 of *Lecture Notes in Computer Science*, pages 59–70. Springer, 2003.
6. J. Boulicaut and B. Jeudy. Using constraints during set mining: Should we prune or not, 2000.
7. J.-F. Boulicaut and B. Jeudy. Mining free itemsets under constraints. In *International Database Engineering and Application Symposium*, pages 322–329, 2001.
8. C. Bucila, J. E. Gehrke, D. Kifer, and W. White. Dualminer: A dual-pruning algorithm for itemsets with constraints. In *Proc. SIGKDD 2002*, Edmonton, Alberta, Canada, July 2002.
9. A. Delis, C. Faloutsos, and S. Ghandeharizadeh, editors. *SIGMOD 1999, Philadephia, Pennsylvania, USA*. ACM Press, 1999.
10. D. Gunopulos, H. Mannila, R. Khardon, and H. Toivonen. Data mining, hypergraph transversals, and machine learning. In *Proc. PODS 1997*, pages 209–216, 1997.

11. J. Hipp and U. Guntzer. Is pushing constraints deeply into the mining algorithms really what we want? *SIGKDD Explorations*, 4(1):50–55, 2002.
12. L. V. S. Lakshmanan, R. T. Ng, J. Han, and A. Pang. Optimization of constrained frequent set queries with 2-variable constraints. In Delis et al. [9], pages 157–168.
13. C. K.-S. Leung, L. V. Lakshmanan, and R. T. Ng. Exploiting succinct constraints using fp-trees. *SIGKDD Explorations*, 4(1):31–39, 2002.
14. R. T. Ng, L. V. S. Lakshmanan, J. Han, and T. Mah. Exploratory mining via constrained frequent set queries. In Delis et al. [9], pages 556–558.
15. R. T. Ng, L. V. S. Lakshmanan, J. Han, and A. Pang. Exploratory mining and pruning optimizations of constrained association rules. In L. M. Haas and A. Tiwary, editors, *Proc. SIGMOD 1998*, pages 13–24. ACM Press, 1998.
16. J. Pei and J. Han. Can we push more constraints into frequent pattern mining? In *ACM SIGKDD Conference*, pages 350–354, 2000.
17. J. Pei and J. Han. Constrained frequent pattern mining: A pattern-growth view. *SIGKDD Explorations*, 4(1):31–39, 2002.
18. J. Pei, J. Han, and L. V. S. Lakshmanan. Mining frequent item sets with convertible constraints. In *ICDE 2001*, pages 433–442. IEEE Computer Society, 2001.
19. C.-S. Perng, H. Wang, S. Ma, and J. L. Hellerstein. Discovery in multi-attribute data with user-defined constraints. *SIGKDD Explorations*, 4(1):56–64, 2002.
20. R. Srikant, Q. Vu, and R. Agrawal. Mining association rules with item constraints. In *Proc. KDD 1997*, 1995.

Relevancy in Constraint-Based Subgroup Discovery

Nada Lavrač[1,2] and Dragan Gamberger[3]

[1] Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia
[2] Nova Gorica Polytechnic, Vipavska 13, 5000 Nova Gorica, Slovenia
[3] Rudjer Bošković Institute, Bijenička 54, 10000 Zagreb, Croatia

Abstract. This chapter investigates subgroup discovery as a task of constraint-based mining of local patterns, aimed at describing groups of individuals with unusual distributional characteristics with respect to the property of interest. The chapter provides a novel interpretation of relevancy constraints and their use for feature filtering, introduces relevancy-based mechanisms for handling unknown values in the examples, and discusses the concept of relevancy as an approach to avoiding overfitting in subgroup discovery. The proposed approach to constraint-based subgroup mining, using the SD algorithm, was successfully applied to gene expression data analysis in functional genomics.

1 Introduction

One of the formulations of data mining [19] involves the specification of the language of patterns and a set of constraints that a pattern has to satisfy with respect to a given database. The constraints that a pattern has to satisfy can be divided in two parts: language constraints and evaluation/optimization constraints. The first concern the form of patterns (e.g., find if-then rules with a target class in the rule head), while the second concern the validity of patterns on a given dataset. The latter can be either evaluation constraints (e.g., find all rules with support above a given threshold) or optimization constraints (e.g., find three best rules with highest confidence).

Constraint-based data mining is now a recognized research topic [3]. The use of constraints enables more efficient induction as well as focusing the search for patterns likely to be of interest to the end-user. While constraint-based data mining research has been—until recently—mostly focused on mining frequent itemsets and association rules, mining frequent episodes and molecular fragments, this chapter focuses on constraint-based subgroup discovery, i.e., constraint-based mining of individual if-then rules of the form

$$Class \leftarrow Cond$$

where *Class* in the rule consequent is a property of interest which is the goal of investigation (the target class), and rule antecedent *Cond* is a conjunction of features (attribute–value pairs).

J.-F. Boulicaut et al. (Eds.): Constraint-Based Mining, LNAI 3848, pp. 243–266, 2005.

Having defined the pattern language of if-then rules, we proceed by informally defining the subgroup discovery task, while the formal definition of constraint-based subgroup discovery, involving the definition of language constraints and evaluation/optimization constraints, is the topic of Section 2.

The subgroup discovery task is informally defined as follows [26,7,16]: Given a population of individuals and a specific property of individuals that we are interested in, find population subgroups that are statistically 'most interesting', e.g., are as large as possible and have the most unusual distributional characteristics with respect to the property of interest.

In the particular task addressed in this chapter the goal of subgroup discovery is to uncover characteristic properties of population subgroups by building short rules which are of high quality. In our approach to subgroup discovery high quality, on the one hand, means that the distribution of classes of instances covered by the rule are statistically significantly different from the distribution in the training set in favour of large coverage of the target class instances, and on the other hand, it means avoidance to overfit the training set.

We restrict the subgroup discovery task to learning from class-labeled data, and induce individual rules (describing individual subgroups) from labeled training examples (labeled positive if the property of interest holds, and negative otherwise), thus targeting the process of subgroup discovery to uncovering properties of a selected target population of individuals with the given property of interest. Despite the fact that this form of rules suggests that standard supervised classification rule learning could be used for solving the task, the goal of subgroup discovery is to uncover individual rules/patterns, as opposed to the goal of standard supervised learning, aimed at discovering rulesets/models to be used as accurate classifiers of yet unlabeled instances [7].

This chapter introduces the constraint-based subgroup discovery task by defining the constraints used in the heuristic SD subgroup discovery algorithm [7].[1] We proceed by discussing constraint-based approaches used in data preprocessing: elimination of irrelevant features and handling of unknown values. Both data preprocessing steps are investigated within the concept of relevancy with the purpose of increasing the quality of induced rules. By reducing the total number of features—through the elimination of features that are less relevant—it enables more effective search for rules with good covering properties while preventing that inclusion of less relevant features or their conjunctions would degrade the quality of rules due to overfitting the training set.

We have successfully applied the proposed approaches to data preprocessing and constraint-based subgroup mining using the SD algorithm on a problem of gene expression data analysis in functional genomics. Gene expression monitoring by DNA microarrays (gene chips) provides an important source of information that can help in understanding many biological processes. The database we analyze consists of a set of gene expression measurements (examples), each

[1] Note that, in contrast with most constraint-based data mining approaches which exhaustively enumerate all solutions satisfying the given constraints, the SD algorithm performs heuristic search.

corresponding to a large number of measured expression values of a predefined family of genes (attributes). Each measurement in the database was extracted from a tissue of a patient with a specific disease; this disease is the class for the given example. The domain, described in [25,8] and used in our experiments, is a typical scientific discovery domain characterised by a large number of attributes compared to the number of available examples. As such, this domain is especially prone to overfitting, as it is a domain with 14 different cancer classes and only 144 training examples in total, where the examples are described by 16063 attributes presenting gene expression values.

While the standard goal of machine learning is to start from the labeled examples and construct models/classifiers that can successfully classify new, previously unseen examples, our main goal is to uncover interesting patterns/rules that can help to better understand the dependencies between classes (diseases) and attributes (gene expressions values). The experiments were performed separately for each cancer class so that a two-class learning problem was formulated for each cancer class as a target. For each of these tasks a complete procedure consisting of feature construction, handling of missing values, elimination of irrelevant features, and induction of subgroup descriptions in the form of rules was repeated. Using the SD subgroup discovery algorithm [7], for each class a single rule with maximal quality value was selected. The induced short rules, with 2–4 features in the rule consequent, were evaluated on an independent test set. Good prediction results for classes with relatively many training instances measured on an independent test set, as well as expert interpretation of induced rules prove the effectiveness of described methods for avoiding overfitting in scientific discovery tasks.

The paper is structured as follows. The constraint-based subgroup mining task is introduced in Section 2. In Section 3 the background is presented: related work on relevancy, our previous work on relevancy as an approach to feature filtering, as well as the ROC space and the TP/FP space providing a framework for the analysis of feature relevancy. Section 4 introduces new definitions of relevancy, reinterpreting feature relevancy and rule relevancy in the TP/FP space. Handling of unknown values within the relevancy concept, aimed at avoiding overfitting and inducing robust rules, is the topic of Section 5. Section 6 discusses the particular choice of the language of features and the interpretation of marginal values as unknown values in the functional genomics domain. Section 7 introduces the functional genomics domain in more detail, where the task is to distinguish between different cancer types. Experimental results show the benefits of proposed handling of unknown values and feature/rule relevancy filtering in this scientific discovery task.

2 Constraint-Based Subgroup Discovery with SD

Subgroup discovery is a form of supervised inductive learning of subgroup descriptions of the target class. As in all inductive rule learning tasks, the language bias is determined by the syntactic restrictions of the pattern language and the

vocabulary of terms in the language. In this work the hypothesis language is restricted to simple if-then rules of the form $Class \leftarrow Cond$, where $Class$ is the target class and $Cond$ is a conjunction of features. Features are logical conditions that have values *true* or *false*, depending on the values of attributes which describe the examples in the problem domain: subgroup discovery rule learning is a form of two-class propositional inductive rule learning, where multi-class problems are solved through a series of two-class learning problems, so that each class is once selected as the target class while examples of all other classes are treated as non-target class examples.

The goal of rule construction are rules with optimal covering properties on the available example set. A rule with ideal covering properties would be *true* for all target class (positive) examples and *false* for all non-target class (negative) examples. Target class examples covered by rule R are called *true positives* (TP), while non-target class examples covered by the rule are called *false positives* (FP).[2] All remaining non-target class examples not covered by the rule are called *true negatives* (TN). An ideal rule would be characterized by $TP = P$ and $TN = N$, where P is the set of positive examples, N the set of negative examples, and $E = P \cup N$.

In this work, subgroup discovery is performed by the SD algorithm, an iterative beam search rule learning algorithm [7]. The input to SD consists of a set of examples E and a set of features F that can be constructed for the given example set. The output of the SD algorithm is a set of rules with optimal covering properties on the given example set. The SD algorithm is implemented in the on-line Data Mining Server (DMS), publicly available at `http://dms.irb.hr`.[3]

2.1 The SD Algorithm

The goal of subgroup discovery algorithm SD (presented in [7] and—for completeness of this paper—outlined also in Figure 1) is to search for rules R that maximize $q_g(R) = \frac{TP}{FP+g}$, where TP are true positives, FP are false positives, and g is a *generalization parameter*. High quality rules will cover many target class examples and a low number of non-target class examples. The number of tolerated non-target class cases, relative to the number of covered target class cases, is determined by parameter g. For low g ($g \leq 1$), induced rules will have high specificity (low false alarm rate) since covering of every single non-target class example is made relatively very 'expensive'. On the other hand, by selecting a high g value ($g > 10$ for small domains), more general rules will be generated, covering also non-target class instances.

Algorithm SD takes as its input the complete training set E and the feature set L, where features $l \in L$ are logical conditions constructed from attribute values

[2] We should have used the notation $TP(R)$ and $FP(R)$ for positive and negative examples covered by rule R, but—for simplicity—argument R is occasionally omitted.

[3] The publicly available Data Mining Server and its constituent subgroup discovery algorithm SD can be tested on user submitted domains with up to 250 examples and 50 attributes. The variant of the SD algorithm used in gene expression data analysis was not limited by these restrictions.

Algorithm SD: Subgroup Discovery
Input: $E = P \cup N$ (E training set, $|E|$ training set size, P positive (target class) examples, N negative (non-target class) examples)
L set of all defined features (attribute values), $l \in L$
Parameter: g (generalization parameter, $0.1 < g$, default value 1)
min_support (minimal support for rule acceptance)
beam_width (maximal number of rules in *Beam* and *New_Beam*)
Output: $S = \{TargetClass \leftarrow Cond\}$ (set of rules R formed of *beam_width* best conditions *Cond*)

(1) **for** all rules in *Beam* and *New_Beam* ($i = 1$ to *beam_width*) **do**
initialize the rule condition to be empty, $Cond(i) \leftarrow \{\}$
initialize rule quality to zero, $q_g(R) \leftarrow 0$
(2) **while** there are improvements in *Beam* **do**
(3) **for** all rules in *Beam* ($i = 1$ to *beam_width*) **do**
(4) **for** all $l \in L$ **do**
(5) form new rule R by forming a new condition as a conjunction of the condition from *Beam* and feature l, $Cond(i) \leftarrow Cond(i) \wedge l$
(6) compute the quality of a new rule as $q_g(R) = \frac{TP}{FP+g}$
(7) **if** $\frac{TP}{|E|} \geq min\_support$ **and if** $q_g(R)$ is larger than the quality of any rule in *New_Beam* **and if** the new rule R is relevant **do**
(8) replace the worst rule in *New_Beam* with new rule R and reorder the rules in *New_Beam* with respect to their quality
(9) **end for** features
(10) **end for** rules from *Beam*
(11) $Beam \leftarrow New\_Beam$
(12) **end while**

Fig. 1. Heuristic beam search rule construction algorithm SD

describing the examples in E. If SD is used in the expert-guided framework, varying the value of g enables the expert to guide subgroup discovery in the TP/FP space, trying to minimize FP (plotted on the X-axis) and maximize TP (plotted on the Y-axis). See Section 3.3 for details on the relationship between the TP/FP space and the ROC (Receiver Operating Characteristic) space [23].

2.2 Constraints Used in the SD Algorithm

In the constraint-based data mining framework, a formal definition of subgroup discovery involves a set of constraints that induced subgroup descriptions have to satisfy. In the SD subgroup discovery algorithm the following constraints are used to formalize the SD constraint-based subgroup discovery task.

Language constraints

– Individual subgroup descriptions have the form of rules $Class \leftarrow Cond$, where $Class$ is the property of interest (the target class), and $Cond$ is a

conjunction of features (conditions based on attribute value pairs) defined by the language describing the training examples.

- For discrete (categorical) attributes, features have the form *Attribute* = *value* or *Attribute* ≠ *value*, for continuous (numerical) attributes they have the form *Attribute* > *value* or *Attribute* ≤ *value*. Note that features can have values *true* and *false* only, that every feature has its logical complement (for feature f_1 being $A_1 = v_1$ its logical complement $\overline{f_1}$ is $A_1 \neq v_1$, for $A_2 > v_2$ its logical complement is $A_2 \leq v_2$), and that features are different from binary valued attributes because for every attribute at least two different features are constructed.
  To formalize feature construction, let values v_{ix} ($x = 1..k_{ip}$) denote the k_{ip} different values of attribute A_i that appear in the positive examples and w_{iy} ($y = 1..k_{in}$) the k_{in} different values of A_i appearing in the negative examples. A set of features F is constructed as follows:
  - For discrete attributes A_i, features of the form $A_i = v_{ix}$ and $A_i \neq w_{iy}$ are generated.
  - For continuous attributes A_i, similar to [6], features of the form $A_i \leq (v_{ix} + w_{iy})/2$ are generated for all neighboring value pairs (v_{ix}, w_{iy}), and features $A_i > (v_{ix} + w_{iy})/2$ for all neighboring pairs (w_{iy}, v_{ix}).
  - For integer valued attributes A_i, features are generated as if A_i were both discrete and continuous, resulting in features of four different forms: $A_i \leq (v_{ix} + w_{iy})/2$, $A_i > (v_{ix} + w_{iy})/2$, $A_i = v_{ix}$, and $A_i \neq w_{iy}$.
- To simplify rule interpretation and increase rule actionability, subgroup discovery is aimed at finding short rules. This is formalized by a language constraint that every induced rule R has to satisfy: rule size (i.e., the number of features in *Cond*) has to be below a user-defined threshold: $size(R) \leq MaxRuleLength$ (in the experiments described in Section 7 this threshold was set to 4).

Evaluation/optimization constraints

- To ensure that induced subgroups are sufficiently large, each induced rule R must have high support, i.e., $sup(R) \geq MinSup$, where $MinSup$ is a user-defined threshold, and $sup(R)$ is the relative frequency of correctly covered examples of the target class in examples set E:

$$sup(R) = p(Class \cdot Cond) = \frac{n(Class \cdot Cond)}{|E|} = \frac{|TP|}{|E|}$$

- Other evaluation/optimization constraints have to ensure that the induced subgroups are highly significant (ensuring that the class distribution of examples covered by the subgroup description will be statistically significantly different from the distribution in the training set). This could be achieved in a straight-forward way by imposing a significance constraint on rules, e.g., by

requiring that rule significance $sig(R)$ is above a user-defined threshold.[4] Instead, in the SD subgroup discovery algorithm [7] the following rule quality measure assuring rule significance, implemented as a heuristic in rule construction, is used:

$$q_g(R) = \frac{|TP|}{|FP| + g} \tag{1}$$

It was shown in [7] that by using this optimization constraint (choose the rule with best $q_g(R)$ value in beam search of best rule conditions), rules with a significantly different distribution of covered positives and negatives, compared to the prior distribution in the training set, are induced. In the experiments described in Section 7, for every two-class problem the rule with the best $q_g(R)$ value for a fixed value $g = 5$ has been selected.

3 Background

This section provides the background for this research: some pointers to the related work on relevancy, the concept of feature relevancy based on p/n pairs of examples, as well as an introduction to the ROC space and the TP/FP space.

3.1 Related Work on Relevancy

The problem of attribute and feature relevancy has been addressed already in early inductive concept learning research [20]. This problem is actually encountered by every inductive learner. Usually, at each step of learning, the choice of the 'best' or 'most informative' attribute or feature needs to be made. This choice is frequently based on the distribution of positive and negative examples covered by the rule/hypothesis before and after attribute selection [24]. Whereas in most learning systems the selection of significant or informative features is part of the learning process, the theory of relevancy presented in this chapter is aimed at pointing out which features constitute a set of relevant features and which features are irrelevant and can be discarded, without even entering the 'best feature' competition. Such filtering of irrelevant features can thus be done

[4] To test significance, the likelihood ratio statistic is used as in CN2 [5] to measure the difference between the class probability distribution in the set of training examples covered by the rule and the class probability distribution in the set of all training examples, computed as follows: $2\sum_i n(Class_i.Cond).\log\frac{n(Class_i.Cond)}{n(Class_i)\cdot p(Cond)}$, where for each class $Class_i$, $n(Class_i.Cond)$ denotes the number of instances of $Class_i$ in the set where the rule body holds true, $n(Class_i)$ is the number of $Class_i$ instances, and $p(Cond)$ (i.e., rule coverage computed as $\frac{n(Cond)}{N}$) plays the role of a normalizing factor. Note that although for each generated subgroup description one class is selected as the target class, the significance criterion measures the distributional unusualness unbiased to any particular class; as such, it measures the significance of rule condition only: $sig(Class \leftarrow Cond) = sig(Cond)$.

in preprocessing of the set of training examples. Whereas most other algorithms only consider the 'local training set' (e.g., a subset of examples covered by the currently developed rule, or a subset of examples in the currently developed node of a decision tree) when deciding about the importance/relevance of attributes or features, we are concerned with finding 'globally relevant' features w.r.t. the entire set of training examples.

The problem of relevancy has recently attracted much attention in the context of feature subset selection in propositional learning [12,18]. An extensive discussion of different approaches to feature (attribute) subset selection can be found in [11], which distinguishes between filter and wrapper approaches, and introduces the notions of totally irrelevant, weakly relevant and strongly relevant features. In this categorisation, our work belongs to filter approaches which eliminate totally irrelevant features in preprocessing. Filtering approaches include, among others, different versions of the RELIEF algorithm [9,13], the FOCUS algorithm [1] and an approach to feature selection proposed in [22].

While relevancy of features has extensively been studied, relevancy of rules has only recently attracted much interest of researchers, especially in the context of rule filtering and suppression in rule postprocessing. Recent work by Morishita and Sese [21] shows how to efficiently prune rules via statistical metrics, by taking into the account convex optimization constraints. An effective approach to rule suppression has been implemented already in EXPLORA [10] to eliminate redundant subgroups. Rule/subgroup R_2 is evaluated as redundant relative to a rule R_1 with a higher quality $q(R_1)$ when $q(R_2) < \text{affinity}(R_2, R_1) \cdot q(R_1)$ and the *affinity* of two subgroups is defined as:

$$\text{affinity}(R_2, R_1) = \left(\frac{|R_1 \cap R_2|}{|R_2|} \right)^{\alpha} = \left(\frac{n(Cond_1 \cdot Cond_2)}{n(Cond_2)} \right)^{\alpha} \qquad (2)$$

where R_i stands for a rule of the form $Class \leftarrow Cond_i$. The parameter α (with default value 1) can be used to control the number of suppressions. The user can increase (or decrease) α to get fewer (or more) resulting subgroups.

3.2 Theory of Relevancy Based on p/n Pairs of Examples

The main aim of the theory of relevancy, described in [14,15], is to reduce the hypothesis space by the elimination of irrelevant features. Consider a two-class learning problem in which examples $e \in E$ are tuples of truth-values of features F. Training set E is represented as a table where rows correspond to training examples and columns correspond to features. A sample table is shown in Table 1. An element in the table has the value *true* when the example satisfies the condition (feature) in the column of the table, otherwise its value is *false*.

Definition 1: p/n pairs
A p/n pair is a pair of training examples where $p \in P$ and $n \in N$.

Definition 2: Coverage of p/n pairs
Let F denote a set of features. Feature $f \in F$ covers a p/n pair iff feature f has value true for p and value false for n.[5]

The notion of p/n pairs can be used to prove important properties of features if the hypothesis language $\mathcal{L}$ defining the feature set F is rich enough to allow for a complete and consistent rules R to be induced from the set of training examples E.[6] Let $F' \subseteq F$. It can be shown that a complete and consistent rule R can be found using only features from set F' iff for each possible p/n pair from the training set E there exists at least one feature $f \in F'$ that covers the p/n pair. The statement, formulated as a theorem for building complete and consistent hypotheses in classification rule learning, was proved in [15]. Its importance for the theory of relevance is manifold. First, it points out that when deciding about the relevancy of features it will be significant to detect which p/n pairs are covered by the feature. Second, it implies that useless features are those that do not cover any p/n pair. An important property of pairs of features can now be defined—coverage of features—which was defined in [14,15] as follows.

Definition 3: Coverage of features
Let $f \in F$. Let $E(f)$ denote the set of all p/n pairs covered by feature f. Feature f_{rel} covers feature f (i.e., f_{rel} is more relevant than f) iff $E(f) \subseteq E(f_{rel})$.

Example 1. Consider a domain with two positive examples, $P = \{p_1, p_2\}$, two negative examples $N = \{n_1, n_2\}$, and six features where $\overline{f_i}$ is a logical complement of f_i, illustrated in Table 1.

Table 1. Training examples represented as vectors of truthvalues of features

Examples		*Features*					
Ex.	*Cl.*	f_1	$\overline{f_1}$	f_2	$\overline{f_2}$	f_3	$\overline{f_3}$
p_1	$\oplus$	*false*	*true*	*true*	*false*	*false*	*true*
p_2	$\oplus$	*false*	*true*	*false*	*true*	*true*	*false*
n_1	$\ominus$	*true*	*false*	*true*	*false*	*true*	*false*
n_2	$\ominus$	*false*	*true*	*false*	*true*	*false*	*true*

In this example feature f_1 does not cover any p/n pair, $E(f_1) = \emptyset$, therefore it can be eliminated as irrelevant for rule learning. Its logical complement $\overline{f_1}$

[5] Notice that in the standard machine learning terminology we could reformulate the definition of coverage of p/n pairs as follows: feature f covers a p/n pair iff f covers (has value *true* for) the positive example p and does not cover (has value *false* for) the negative example n.

[6] The training set may include noise but there should be no contradictions, i.e. examples with same attribute values labeled by different class names.

covers two p/n pairs, $E(\overline{f_1}) = \{p_1/n_1, p_2/n_1\}$. Feature f_2 covers one p/n pair, $E(f_2) = \{p_1/n_2\}$ and its logical complement $\overline{f_2}$ covers only the pair built of p_2 and n_1. Although $\overline{f_2}$ is a logical complement of f_2, the sets of p/n pairs covered by f_2 and $\overline{f_2}$ are different, therefore both the feature and its complement are considered as relevant for rule learning. □

3.3 The ROC Space and the TP/FP Space

A point in the ROC space (ROC: Receiver Operating Characteristic) [23] shows classifier performance in terms of false alarm or *false positive rate* $FPr = \frac{|FP|}{|TN|+|FP|} = \frac{|FP|}{|N|}$ (plotted on the X-axis), and sensitivity or *true positive rate* $TPr = \frac{|TP|}{|TP|+|FN|} = \frac{|TP|}{|P|}$ (plotted on the Y-axis).

A point (FPr, TPr) depicting rule R in the ROC space is determined by the covering properties of the rule. The ROC space is appropriate for measuring the success of subgroup discovery, since rules/subgroups whose TPr/FPr tradeoff is close to the diagonal can be discarded as insignificant; the reason is that the rules with TPr/FPr on the diagonal have the same distribution of covered positives and negatives as the distribution in the training set. Conversely, significant rules/subgroups are those sufficiently distant from the diagonal. Subgroups that are optimal under varying TPr/FPr tradeoffs form a convex hull called the ROC curve. Figure 2 presents seven rules on the convex hull (marked by circles), including $X1$ and $X2$, while two rules $B1$ and $B2$ below the convex hull (marked by $+$) are of lower quality in terms of their TPr/FPr tradeoff.

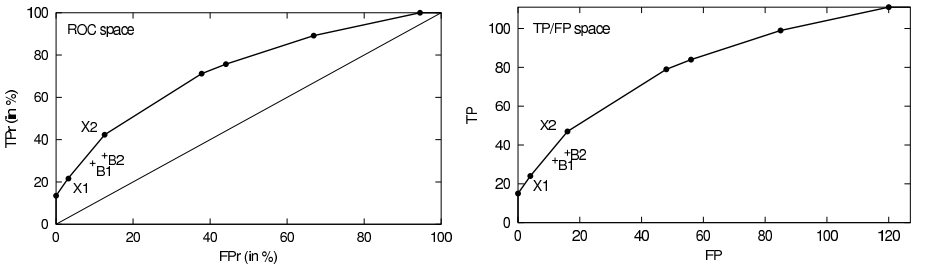


Fig. 2. The left-hand side figure shows the ROC space with a convex hull formed of seven rules that are optimal under varying TPr/FPr tradeoffs, and two suboptimal rules B1 and B2. The right-hand side presents the positions of the same rules in the corresponding TP/FP space.

It was shown in [16] that for rule R, the vertical distance from the (FPr, TPr) point to the ROC diagonal is proportional to the significance of the rule. Hence, the goal of a subgroup discovery algorithm is to find subgroups in the upper-left corner area of the ROC space, where the most significant rule would lie in point $(0, 1)$ representing a rule covering only positive and none of the negative examples ($FPr = 0$ and $TPr = 1$).

An alternative to the ROC space is the so-called TP/FP space (see the right-hand side of Figure 2), where FPr on the X-axis is replaced by $|FP|$ and TPr on the Y-axis by $|TP|$.[7] The TP/FP space is equivalent to the ROC space when comparing the quality of subgroups induced in a single domain. The reminder of this paper considers only this simpler TP/FP space representation.

4 Interpretation of Relevancy in the TP/FP Space

The concept of feature coverage introduced in this section is important as a relevancy constraint used in rule learning. The concept is not valid only for features but also for conjunctions of features and for complete rules.

Filtering based on absolute and relative relevancy introduced in this section can be applied in every domain. While the aim of absolute relevancy is to provide the minimal quality constraint required for every feature (rule), relative relevancy aims to ensure that only the best among available features will enter the rule construction process. The definition of relative irrelevancy is very useful because it does not depend on user-defined constraints. Relevancy-based filtering is therefore applicable in all feature-based machine learning applications [14]. It is useful also as a preprocessing filter for other symbolic learners such as decision tree learners, because complete attributes can be eliminated as irrelevant if all features generated for these attributes are detected as relatively or absolutely irrelevant.

4.1 Relative Relevancy

Let us now re-interpret the notions introduced in Sections 3.2 and 3.3 from the point of view of feature relevancy.

Definition 4: Coverage of features (revisited Definition 3)
Feature f_{rel} covers feature f (i.e., feature f_{rel} is more relevant than f) iff true positives of f are a subset of true positives of f_{rel} and true negatives of f are a subset of true negatives of f_{rel}, i.e., iff $TP(f) \subseteq TP(f_{rel})$ and $TN(f) \subseteq TN(f_{rel})$ (see Figure 3).

Definition 5: Relative relevancy
Feature f is relatively irrelevant iff there exists another feature f_{rel} such that f_{rel} covers f.

Theorem 1.
If feature f_{rel} covers feature f and feature g_{rel} covers g then $f_{rel} \wedge g_{rel}$ covers $f \wedge g$.

[7] The TP/FP space can be turned into the ROC space by simply normalizing the TP and FP axes to the [0,1]x[0,1] scale.

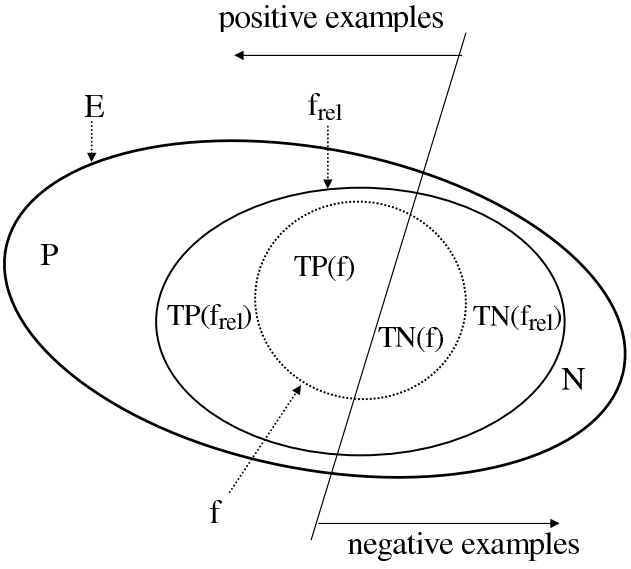


Fig. 3. The concept of relative relevancy illustrated by features f and f_{rel}. Feature f is relatively irrelevant because $TP(f) \subseteq TP(f_{rel})$ and $TN(f) \subseteq TN(f_{rel})$.

It is trivial to prove this theorem by first fixing one of the two conjuncts $g_{rel} = g$ and showing that $TP(f \wedge g) \subseteq TP(f_{rel} \wedge g)$ and $TN(f \wedge g) \subseteq TN(f_{rel} \wedge g)$. Next, the same relationship can be shown also for the case when g_{rel} covers g.[8] □

Relative relevancy of features is an important concept as feature f is not necessarily irrelevant because of its low $|TP|$ or $|TN|$ values but because there exists another more relevant feature with better covering properties. Therefore a relevancy filter using the concept of relative relevancy of features will never eliminate a feature that could potentially be relevant in conjunction with other features, as the more relevant feature which caused its elimination will take its role in the conjunction. Relative relevance ensures the quality of induced rules and, even more importantly from the point of view of avoiding overfitting, it ensures that rule learners will use only the best features available.

Consider now the simplest form of rules, whose conditions consist of a single feature. Suppose such rules are plotted in the TP/FP space, meaning that each feature represents a point in the TP/FP space. The more distant a feature is from the diagonal, the more significant is the feature. 'Good' features are those as close as possible to point $(0, P)$ in TP/FP space. The left-hand side of Figure 4 presents the concept of relative relevancy. As $|TP(f)| \leq |TP(f_{rel})|$, feature

[8] Theorem 1 can be proved also for the logical OR operation $f_{rel} \vee g_{rel}$. Consequently, if for feature f there exists another feature f_{rel} with the property that if in any rule f is substituted by f_{rel} the rule quality measured by the number of correct classifications $|TP|$ and $|TN|$ does not decrease, then f_{rel} can be always used instead of f, and feature f can be eliminated as irrelevant.

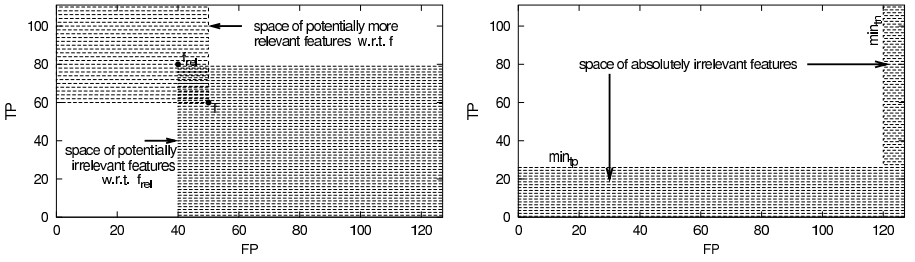


Fig. 4. The left-hand side figure presents the concept of relative relevancy while the right-hand side figure presents the concept of absolute relevancy

f_{rel} is plotted higher along the TP-axis. As $|TN(f)| \leq |TN(f_{rel})|$, therefore $|FP(f_{rel})| \leq |FP(f)|$, and feature f_{rel} is plotted more to the left (closer to the TP-axis) along the FP-axis than feature f.

Figure 4 shows feature f, a shaded area in the upper-left corner of f showing a part of the TP/FP space of features f_{rel} that are potentially more relevant than f, and a shaded area in the lower-right corner of f_{rel} showing the part of the space of features that are potentially irrelevant due to the existence of f_{rel}. Note that not all features left-up of f are more relevant and not all features right-down of f_{rel} are irrelevant, but only those that satisfy Definition 4.

4.2 Total Relevancy

In addition to irrelevant features defined through relative relevancy, also totally irrelevant features—those which are totally useless for distinguishing between the classes—can be eliminated in preprocessing.

Definition 6: Total irrelevancy
Feature f with $|TP(f)| = 0$ or $|TN(f)| = 0$ is totally irrelevant.

4.3 Absolute Relevancy

In order for a feature to be acceptable as a building block of rule conditions representing some genuine dependencies between classes and attribute values, the feature itself must have appropriate covering properties on the training set. These can be defined in terms of user-defined support constraints.

Definition 7: Absolute irrelevancy
Feature f that has either $|TP(f)| < MinTP$ or $|TN(f)| < MinTN$ is absolutely irrelevant, for $MinTP$ and $MinTN$ being user defined constraints.

For low values of $MinTP$ and $MinTN$, feature f with $|TP(f)| < MinTP$ is *true* for a small number of target class examples, and feature g with $|TN(g)| < MinTN$ is *false* for a small number of non-target class examples. Such small numbers may be due to statistical chance so that it seems reasonable not to

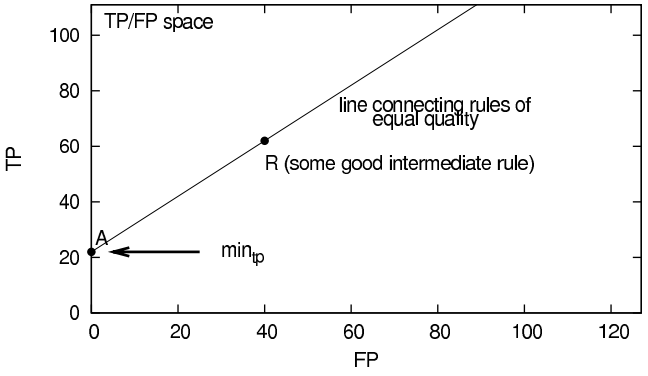


Fig. 5. The selection of the optimal $MinTP$ constraint based on the properties of a previously detected good rule R

use features with either of these properties in the rule construction process. The part of the TP/FP space of absolutely irrelevant features is represented by the shaded area of the right-hand side figure of Figure 4.

Although the significance of rules is proportional to their distance from the diagonal in the TP/FP space (Figure 2), this property is not appropriate as a quality criterion for features. As logical combinations of features lying on the diagonal or very near to it can result in very significant conjunctions of features (rules), only relative and absolute relevancy constraints defined in this work are considered as appropriate for feature filtering.

By conjunctions of features, the generated rule will have $|TP|$ equal or smaller than the smallest $|TP|$ value of the features forming a conjunctive subgroup description. In contrast, the $|TN|$ value of a rule will be at least as large as the largest $|TN|$ of the used features. This is the reason why $MinTP$ is typically selected higher than $MinTN$ (see the right-hand side figure of Figure 4) and it can be as large as the minimal estimated number of examples that must be covered by a subgroup of acceptably high quality for the domain. The problem with absolute irrelevancy is that both $MinTP$ and $MinTN$ are user defined constraints and that any value, regardless how high it is, can not guarantee that a feature is actually relevant. A practical suggestion is to start with their low values of these constraints and after that to experiment with higher values. The optimal point is just before a significant decrease of covering properties of induced rules can be noticed. A good starting values for gene expression domains are $MinTP = |P|/2$ and $MinTN = \sqrt{|N|}$ which have been used in all the experiments reported in Section 7. The selection of these constraints is not very critical for the final result because the majority of absolutely irrelevant features is detected also as relatively irrelevant. With mentioned $MinTP$ and $MinTN$ values in gene expression domains more than 90% of absolutely irrelevant features were detected as being also relatively irrelevant.

4.4 Analysis of Absolute Relevancy Constraints in the TP/FP Space

The major problem of the concept of absolute relevancy is the selection of appropriate $MinTP$ and $MinTN$ constraints. In cases when rules are built exclusively as conjunctions of features (as in the SD algorithm), the problem can be at least partially solved by the analysis of the $MinTP$ constraint in the TP/FP space.

Let us suppose that in the process of rule construction rule R (that could be also a single feature) represents the best solution detected so far or that we are able to estimate its properties based on previous experiments in the domain. The position of rule R in the TP/FP space is determined by its $TP(R)$ and $FP(R)$ values. In Figure 5 the line drawn through this point presents the line connecting all the points in the TP/FP space that have the same rule quality q_g as rule R. For various quality measures the slope of the line is different. For the $q_g(R)$ measure used in the SD algorithm [7] the slope is equal to $\frac{|TP(R)|}{|FP(R)|+g}$.[9] This line cuts the TP axis in point A with value $|FP(A)| = 0$ and some positive value $|TP(A)|$. Setting $MinTP = |TP(A)|$ is a good choice for the $MinTP$ constant because any conjunctive combination with a feature which has $|TP|$ value below $|TP(A)|$ can, in an ideal case, lead to a rule lying below point A and therefore have a lower quality than the already detected rule R. For the $q_g(R)$ measure this value is $g \cdot \frac{|TP(R)|}{|FP(R)|+g}$.[10] It can be noted that a better intermediate rule R (with higher $|TP(R)|$ and lower $|FP(R)|$ values) enables the selection of a higher $MinTP$ value, resulting in the elimination of more features and faster search without a decrease in the final rule quality. This property can be used so that the $MinTP$ value is adjusted dynamically to the best detected solution so far. The result is feature relevancy detection during the rule construction process. For very time consuming algorithms it can be useful to first detect a good R by a fast heuristic search algorithm in advance before starting the main rule construction process, ensuring that relevancy filtering can be done before starting the rule construction process.

The described analysis can not help us to estimate the optimal $MinTN$ value. In cases when rules are built by disjunctive instead of conjunctive connections of features, analogous reasoning is valid, which helps to select good $MinTN$ values but then $MinTP$ should be estimated and selected by the user.

4.5 Relevancy of Rules

The defined relations of relative and absolute relevancy are valid not only for rules consisting of a single feature but they can be applied to any logical combination of features that can be constructed in the rule induction process, as well as to complete rules. This property is very important because it can significantly reduce the time and space complexity of learning algorithms. In the SD

[9] For example, for the weighted relative accuracy measure, *WRAcc* [16], the slope of the line equals $\frac{|P|}{|N|}$.

[10] When the *WRAcc* rule quality measure is used, the optimal $MinTP$ value for rule R equals $|TP(R)| - |FP(R)| \cdot \frac{|P|}{|N|}$.

algorithm, the properties of relative and absolute relevancy are tested in each of its iterations for all the constructed conjunctive combinations of features. In SD, the $MinTP$ absolute relevancy constraint is implemented by the user-defined $MinSup$ constraint, while the $MinTN$ constraint is ensured by setting the absolute relevancy threshold for all the generated features.

5 Using the Concept of Relevancy in Handling of Unknown Values

The concept of relevancy can be used in handling of unknown values based on the guideline that by the elimination/replacement of unknown values the relevancy of features should not increase. By following this guideline, the approach proposed in this section contributes to preventing data overfitting, especially in domains with a large number of unknown attribute values. The proposed approach is different from typical procedures for handling unknown values such as considering the unknown value as an additional regular value or substituting of the unknown value by the most common or by a proportional fractional value [4].

To ensure that unknown value handling will not increase feature relevancy, an attribute with an unknown value in a positive example is—in all features constructed from this attribute—replaced by value *false*, while an unknown value occurring in a negative example is replaced by value *true* in all features constructed from the same attribute.

Table 2. Features generated from an attribute with value unknown (?) have value *false* if the example is positive, and value *true* if the example is negative. Feature values generated from unknown attribute values are presented in bold.

Examples		*Attributes*		*Features*					
Ex.	*Cl.*	X	Y	$X = A$	$X \neq A$	$X = P$	$X \neq P$	$Y > 3$	$Y \leq 3$
p_1	$\oplus$	A	5	*true*	*false*	*false*	*true*	*true*	*false*
p_2	$\oplus$	?	4	**false**	**false**	**false**	**false**	*true*	*false*
p_3	$\oplus$	P	?	*false*	*true*	*true*	*false*	**false**	**false**
n_1	$\ominus$	?	2	**true**	**true**	**true**	**true**	*false*	*true*
n_2	$\ominus$	A	?	*true*	*false*	*false*	*true*	**true**	**true**
n_3	$\ominus$	P	1	*false*	*true*	*true*	*false*	*false*	*true*

Example 2. Consider a domain with three positive examples, three negative examples, two attributes (one discrete and one continuous-valued), four features generated for the discrete attribute, and two (out of possibly many) features for the attribute with continuous values. The domain is presented in Table 2. It can be noticed that for known attribute values a feature and its complement always have different truth values, but for unknown attribute values all features have the same value: *false* if the example is positive and *true* if the example is negative. □

6 Using the Concept of Relevancy in Gene Expression Data Preprocessing

In some domains, like in the gene expression domain, there is a possibility to choose between different types of attributes and when confronted with this choice, the preference should be given to those leading to more relevant features.

6.1 Choice of the Language of Features

Gene expression scanners measure signal intensity as continuous values which form an appropriate input for data analysis. The problem is that for continuous valued attributes there can be potentially many boundary values separating the classes, resulting in many different features for a single attribute. There is also a possibility to use presence call (signal specificity) values computed from measured signal intensity values by the Affymetrix GENECHIP software. The presence call has discrete values A (absent), P (present), and M (marginal). Subgroup discovery as well as filtering based on feature and rule relevancy are applicable both for signal intensity and/or the presence call attribute values. Typically, signal intensity values are used [17] because they impose less restrictions on the classifier construction process and because the results do not depend on the GENECHIP software presence call computation. For subgroup discovery we prefer the later approach based on presence call values. The reason is that features presented by conditions like $Gene = P$ is *true* (meaning that *Gene* is present, i.e., expressed) or $Gene = A$ is *true* (meaning that *Gene* is absent, i.e., not expressed) are very natural for human interpretation and that the approach can help in avoiding overfitting, as the feature space is very strongly restricted, especially if the marginal value M is encoded as value unknown.

6.2 Handling Unknown Values and Feature Filtering

In the gene expression domain the M value is handled as an unknown value because we do not want to increase the relevance of features generated from attributes with M values. As for the other two values, A and P, it holds that two features for gene X, $X = A$ and $X \neq P$, are identical (see Table 2). Consequently, for every gene X there are only two distinct features $X = A$ and $X = P$. As suggested in Section 5, unknown values coming from marginal attribute values in positive examples are replaced by value *false*, while in negative examples they are replaced by value *true*.

Example 3. The approach applied in gene expression data analysis is illustrated in Table 3. The table presents five positive and four negative examples for one of the target classes in the gene expression domain. Only features generated from presence call values of three attributes (genes) are presented.

Observe that in this example, following Definition 5 of relative relevancy, feature $X = A$ is relatively irrelevant because of feature $Y = A$, and feature $X = P$ is relatively irrelevant because of feature $Z = A$. Consequently, both features generated for gene X can be eliminated as irrelevant. □

Table 3. Training examples represented as vectors of truthvalues of features. Notice that value M (marginal) is treated as an unknown attribute value.

Examples		*Genes*			*Features*					
Ex.	*Cl.*	X	Y	Z	$X=A$	$X=P$	$Y=A$	$Y=P$	$Z=A$	$Z=P$
p_1	$\oplus$	A	A	A	*true*	*false*	*true*	*false*	*true*	*false*
p_2	$\oplus$	P	P	A	*false*	*true*	*false*	*true*	*true*	*false*
p_3	$\oplus$	A	A	P	*true*	*false*	*true*	*false*	*false*	*true*
p_4	$\oplus$	P	P	A	*false*	*true*	*false*	*true*	*true*	*false*
p_5	$\oplus$	M	A	A	*false*	*false*	*true*	*false*	*true*	*false*
n_1	$\ominus$	A	P	P	*true*	*false*	*false*	*true*	*false*	*true*
n_2	$\ominus$	P	P	P	*false*	*true*	*false*	*true*	*false*	*true*
n_3	$\ominus$	M	M	A	*true*	*true*	*true*	*true*	*true*	*false*
n_4	$\ominus$	P	P	A	*false*	*true*	*false*	*true*	*true*	*false*

7 Experiments in Functional Genomics

The gene expression domain, described in [25,8] is a domain with 14 different cancer classes and 144 training examples in total. Eleven classes have 8 examples each, two classes have 16 examples and only one has 24 examples. The examples are described by 16063 attributes presenting gene expression values. In all the experiments we have used gene presence call values (A, P, and M) to describe the training examples. The domain can be downloaded from http://www-genome.wi.mit.edu/cgi-bin/cancer/datasets.cgi. There is also an independent test set with 54 examples. The standard goal of machine learning is to start from such labeled examples and construct classifiers that can successfully classify new, previously unseen examples. Such classifiers are important because they can be used for diagnostic purposes in medicine and because they can help to understand the dependencies between classes (diseases) and attributes (gene expressions values).

The experiments were performed separately for each cancer class so that a two-class learning problem was formulated where the selected cancer class was the target class and the examples of all other classes formed non-target class examples. In this way the domain was transformed into 14 inductive learning problems, each with the total of 144 training examples and between 8 and 24 target class examples. For each of these tasks a complete procedure consisting of feature construction, elimination of irrelevant features, and induction of subgroup descriptions in the form of rules was repeated. Finally, using the SD subgroup discovery algorithm [7], for each class a single rule R with maximal $q_g(R)$ value was selected, for $q_g(R) = \frac{|TP|}{|FP|+g}$ being the heuristic of the SD algorithm and $g = 5$ as the generalization parameter default value. The rules for all 14 tasks consisted of 2–4 features. The procedure was repeated for all 14 tasks with the same default parameter values. The induced rules were tested on the independent example set.

There are very large differences among the results on the test sets for various classes (diseases) and the precision higher than 50% was obtained for only 5 out

of 14 classes. There are only three classes (lymphoma, leukemia, and CNS) with more than 8 training cases and all of them are among those with high precision on the test set, while for only two out of eleven classes with 8 training cases (colorectal and mesothelioma) high precision was achieved. The classification properties of rules induced for classes with 16 and 24 target class examples (lymphoma, leukemia and CNS) are comparable to those reported in [25] (see Table 4), while the results on eight small example sets with 8 target examples were poor. An obvious conclusion is that the use of the subgroup discovery algorithm is not appropriate for problems with a very small number of examples because overfitting can not be avoided in spite of the heuristics used in the SD algorithm and the additional domain-specific techniques used to restrict the hypothesis search space. But for larger training sets the subgroup discovery methodology enabled effective construction of relevant rules.

Table 4. Covering properties on the training and on the independent test set for rules induced for three classes with 16 and 24 examples. Sensitivity is $\frac{|TP|}{|P|}$, specificity is $\frac{|TN|}{|N|}$, while precision is defined as $\frac{|TP|}{|TP|+|FP|}$.

Cancer	Training set			Test set		
	Sens.	Spec.	Prec.	Sens.	Spec.	Prec.
lymphoma	16/16	128/128	100%	5/6	48/48	100%
leukemia	23/24	120/120	100%	4/6	47/48	80%
CNS	16/16	128/128	100%	3/4	50/50	100%

7.1 Experiments in Feature Filtering

In the rest of this chapter experiments are performed on three classes with a sufficient number of training instances—lymphoma, leukemia, and CNS—for which induction of significant rules was possible. Table 5 shows the summary of results obtained by different experiments in eliminating irrelevant features. For absolute relevance default values $MinTP = |P|/2$ and $MinTN = \sqrt{|N|}$ as proposed in Section 4.3 were used.

Task 1. In the real domain with 16063 attributes both concepts of absolute and relative relevancy were very effective in reducing the number of features. About 60% of all features were detected as absolutely irrelevant while relative irrelevancy was even more effective as it managed to eliminate up to 75% of all the features. Their combination resulted in the elimination of 75%–85% of all the features. These results are presented in the first row of Table 5. The set of all features in these experiments was generated so that for each gene (attribute) two features were constructed ($Gene = A$ and $Gene = P$), followed by eliminating totally irrelevant features (with $|TP| = 0$ or $|TN| = 0$), which substantially reduced the total number of features.

Table 5. This table presents mean numbers of constructed features for the lymphoma, leukemia, and CNS domains. Presented are the total number of features (All), the number of features after the elimination of totally irrelevant features (Total), the number of features after the elimination of absolutely irrelevant features (Absolute), and the number of features after the elimination of absolutely and relatively irrelevant features (Relative). These three values are shown for the following training sets: the real training set with 16063 genes (with 32126 gene expression activity values, constructed as $Gene = A$ and $Gene = P$), a randomly generated set with 16063 genes, and a set with 32126 genes which is a combination of 16063 real and 16063 random attributes.

Tasks	All	Total	Absolute	Relative
Task 1 Real domain with 16063 att.	32126	23500	9628	4445
Task 2 Randomly generated domain with 16063 att.	32126	27500	16722	16722
Task 3 Combination of 16063 real and 16063 randomly generated attributes	64252	51000	26350	15712

Task 2. Another domain with 16063 completely randomly generated attribute values was also constructed, and the same experiments were repeated on this artificial domain as for the real gene expression domain. The results (repeated with five different randomly generated attribute sets) were significantly different: there were only about 40% of absolutely irrelevant features and practically no relatively irrelevant features. The results are presented in the second row of Table 5. Comparing the results for the real and for the randomly generated domain, especially large differences can be noticed in the performance of relative relevancy. It is the consequence of the fact that in the real domain there are some features that are really relevant; they cover many target class examples and a few non-target class examples and in this way they make many other features relatively irrelevant. The results prove the importance of relative relevancy for domains in which strong and relevant dependencies between classes and attribute values exist.

Task 3. The experiments with feature relevancy continued with another domain with 32126 attributes, generated as the combination of two previous domains with 16063 attributes each: the real and the randomly generated domain. The results are presented in the last row of Table 5. After the elimination of absolutely irrelevant features the number of features is equal to the sum of features that remained in the two independent domains with 16063 attributes. In contrast, relative relevancy was much more effective. Besides eliminating many features from the real attribute part it was now possible to eliminate also a significant part of features of randomly generated attributes.

Summary of the experiments. Figure 6 illustrates the results presented in Table 5 with one added domain with 32126 randomly generated attributes. From

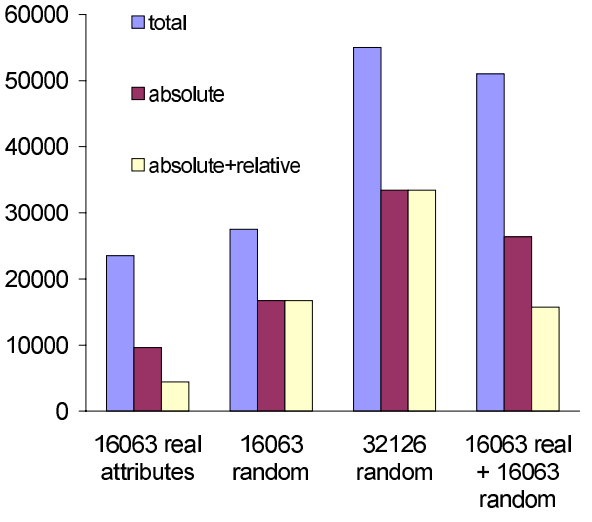


Fig. 6. Mean numbers of features for the three domains (lymphoma, leukemia, and CNS) for the following training sets: real training set with 16063 attributes of gene expression activity values, a randomly generated set with 16063 attributes, a randomly generated set with 32126 attributes, and a set which is a combination of 16063 real and 16063 random attributes

this analysis it is obvious that the elimination of features is very effective in real domains. The same result were confirmed in experiments with domains with only 8 target class examples. It is important that in domains which are combinations of real and random attributes the proposed feature filtering methodology is effective: in Task 3 less features remained after feature elimination (15712 features) than in Task 2 (16722 features). This proves that the presented methodology, especially relative relevancy, can be very useful in avoiding overfitting by reducing the hypothesis search space through the elimination of non-significant dependencies between attribute values and classes. This property is important because it can be assumed that among 16063 real attributes there are many of them which are irrelevant with respect to the target class.

7.2 Examples of Induced Rules

For three classes (lymphoma, leukemia, and CNS) with more than 8 training cases the following rules were induced by the constraint-based subgroup discovery approach involving relevancy filtering and handling of unknown values described in this chapter.

Lymphoma class:
(CD20_receptor EXPRESSED) AND
(phosphatidylinositol_3_kinase_regulatory_alpha_subunit NOT EXPRESSED)

Leukemia class:
(KIAA0128_gene EXPRESSED) AND
(prostaglandin_d2_synthase_gene NOT EXPRESSED)
CNS class:
(fetus_brain_mRNA_for_membrane_glycoprotein_M6 EXPRESSED) AND
(CRMP1_collapsin_response_mediator_protein_1 EXPRESSED)

The expert interpretation of the results yields several biological observations: two rules (for the lymphoma and leukemia classes) are judged as reassuring and one (the CNS class) has a plausible, albeit partially speculative explanation. Namely, the best-scoring rule for the lymphoma class in the multi-class cancer recognition problem contains a feature corresponding to a gene routinely used as a marker in diagnosis of lymphomas (CD20), while the other part of the conjunction (phosphatidylinositol, the PI3K gene) seems to be a plausible biological co-factor. The best-scoring rule for the leukemia class contains a gene whose relation to the disease is directly explicable (KIAA0128, Septin 6). Both M6 and CRMP1 appear to have multifunctional roles in shaping neuronal networks, and their function as survival (M6) and proliferation (CRMP1) signals may be relevant to growth promotion and CNS malignancy.

Both good prediction results on an independent test set (Table 4) as well as expert interpretation of induced rules prove the effectiveness of described methods for avoiding overfitting in scientific discovery tasks.

8 Conclusions

This chapter reinterprets the theory of relevancy, described in [14,15], as relevancy constraints applied in a constraint-based subgroup discovery. Although the target is the induction of rules presenting subgroup descriptions, the results concerning the concept of relevancy are more general and valid for any feature-based rule learner. The chapter presents the theory of feature relevancy in the context of ROC analysis and provides an experimental evaluation of the usefulness of feature elimination in a functional genomics domain. We have implemented domain dependent restrictions by using discrete instead of continuous attribute values, and domain independent restrictions by the elimination of irrelevant features. Interpretation of marginal gene values as unknown values helped in reducing the feature space and ensured the robustness of induced rules. The proposed subgroup discovery framework proved to be useful for solving scientific discovery tasks.

Acknowledgments

This work was supported by the Slovenian Ministry of Higher Education, Science and Technology, and the Croatian Ministry of Science, Education and Sport.

References

1. H. Almuallim and T.G. Dietterich. Learning with many irrelevant features, In *Proceedings of the 9th National Conference on Artificial Intelligence*, The MIT Press, 547–552, 1991.
2. R.J. Bayardo, R.Agrawal, and D.Gunopulos. Constraint-based rule mining in large, dense databases. In *Proc. of the 15th Conference on Data Engineering*, 188-197, 1999.
3. R.J. Bayardo, editor. *Constraints in Data Mining. Special issue of* SIGKDD Explorations, 4(1), 2002.
4. I. Bruha and F. Franek. Comparison of various routines for unknown attribute value processing. *Journal of Pattern Recognition and Artificial Intelligence* 10(8): 939–955, 1996.
5. P. Clark and T. Niblett. The CN2 induction algorithm. *Machine Learning*, 3(4): 261–283, 1989.
6. U.M. Fayyad and K.B. Irani. On the handling of continuous-valued attributes in decision tree generation. *Machine Learning* 8: 87–102, 1992.
7. D. Gamberger and N. Lavrač. Expert-guided subgroup discovery: Methodology and application. *Journal of Artificial Intelligence Research* 17: 501–527, 2002.
8. D. Gamberger, N. Lavrač, F. Železný, and J. Tolar. Induction of comprehensible models for gene expression datasets by the subgroup discovery methodology. *Journal of Biomedical Informatics* 37:269–284, 2004.
9. K. Kira and L.A. Rendell. A practical approach to feature selection, In *Proceedings of the 9th International Conference on Machine Learning*, Morgan Kaufmann, 249–256, 1992.
10. W. Klösgen. Explora: A multipattern and multistrategy discovery assistant. In *Advances in Knowledge Discovery and Data Mining*, 249–271, MIT Press, 1996.
11. R. Kohavi and G.H. John. Wrappers for feature subset selection. *Artificial Intelligence, Special Issue on Relevance*, 97: 273-324, 1997.
12. D. Koller and M. Sahami. Toward optimal feature selection. *Proceedings of the 13th International Conference on Machine Learning*, Morgan Kaufmann, 284–292, 1996.
13. I. Kononenko. Estimating attributes: Analysis and extensions of Relief, In *Proceedings of the 7th European Conference on Machine Learning*, LNAI 784, Springer, 171–182, 1994.
14. N. Lavrač, D. Gamberger, and P. Turney. A relevancy filter for constructive induction. *IEEE Intelligent Systems and their Applications* 13: 50–56, 1998.
15. N. Lavrač, D. Gamberger and V. Jovanoski. A study of relevance for learning in deductive databases. *Journal of Logic Programming* 40: 215–249, 1999.
16. N. Lavrač, B. Kavšek, P. Flach and L. Todorovski. Subgroup discovery with CN2-SD. *Journal of Machine Learning Research*, 5: 153–188, 2004.
17. J. Li and L. Wong. Geography of differences between two classes of data. In *Proc. of 6th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD2002)*, Springer, 325–337, 2002.
18. H. Liu and H. Motoda, editors. *Feature Selection for Knowledge Discovery and Data Mining*. Kluwer, 1998.
19. H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1(3): 241–258, 1997.
20. R.S. Michalski. A theory and methodology of inductive learning, In: R. Michalski, J. Carbonell and T. Mitchell (eds.) *Machine Learning: An Artificial Intelligence Approach*, Tioga, 83–134, 1983.

21. S. Morishita and J. Sese. Traversing itemset lattices with statistical metric pruning. In *Proceedings of the Nineteenth Symposium on Principles of Database Systems*, 226–236, 2000.
22. A.L. Oliveira and A.Sangiovanni-Vincentelli. Constructive induction using a non-greedy strategy for feature selection. In *Proceedings of the 9th International Conference on Machine Learning*, Morgan Kaufmann, 354–360, 1992.
23. F. Provost and T. Fawcett. Robust classification for imprecise environments. *Machine Learning*, 42(3): 203–231, 2001.
24. J.R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, (1993).
25. S. Ramaswamy et al. Multiclass cancer diagnosis using tumor gene expression signatures. In *Proc. Natl. Acad. Sci. USA*, 98(26): 15149–15154, 2001.
26. S. Wrobel. An algorithm for multi-relational discovery of subgroups. In *Proceedings of the 1st European Symposium on Principles of Data Mining and Knowledge Discovery*, Springer, 78–87, 1997.

A Novel Incremental Approach to Association Rules Mining in Inductive Databases*

Rosa Meo, Marco Botta, Roberto Esposito, and Arianna Gallo

Dipartimento di Informatica, Università di Torino, Italy
{meo, botta, esposito, gallo}@di.unito.it

Abstract. Constraints-based mining languages are widely exploited to enhance the KDD process. In this paper we propose a novel *incremental* approach to extract itemsets and association rules from large databases. Here incremental is used to emphasize that the mining engine does not start from scratch. Instead, it exploits the result set of previously executed queries in order to simplify the mining process. Incremental algorithms show several beneficial features. First of all they exploit previous results in the pruning of the itemset lattice. Second, they are able to exploit the mining constraints of the current query in order to prune the search space even more. In this paper we propose two incremental algorithms that are able to deal with two — recently identified — types of constraints, namely item dependent and context dependent ones. Moreover, we describe an algorithm that can be used to extract association rules from scratch in presence of context dependent constraints.

1 Introduction

The problem of mining association rules and, more generally, that of extracting frequent sets from large databases has been widely investigated in the last decade [1,2,3,4,5,6]. These researches addressed two major issues: on one hand, performance and efficiency of the extraction algorithms; on the other hand, the exploitation of user preferences about the patterns to be extracted, expressed in terms of constraints. Constraints are widely exploited also in data mining languages, such as in [5,7,8,9,10,11] where the user specifies in each data mining query, not only the constraints that the items must satisfy, but also different criteria to create groups of tuples from which itemsets will be extracted. Constraint-based mining languages are also the main key factor of inductive databases [12], proposed in order to leverage decision support systems. In inductive databases, the user explores the domain of a mining problem submitting to the system many mining queries in sequence, in which subsequent queries are very often a refinement of previous ones. This might be, if not properly addressed, a huge computational workload. This problem becomes even more severe considering that these queries are typically instances of iceberg queries [13], well-known to be expensive on very large databases such as common data warehouses. In such systems the intelligent exploitation of user constraints becomes the key factor for a successful exploration of the problem search space [14]. The same occurs also in "dense" datasets, in which the volume of

* This work has been funded by EU FET project cInQ (IST-2000-26469).

J.-F. Boulicaut et al. (Eds.): Constraint-Based Mining, LNAI 3848, pp. 267–294, 2005.

the result set (the frequent itemsets) compared to the input data is particularly large. In these cases, constraints are exploited to make a problem tractable that otherwise would require a non affordable computational workload [15].

In this application context, in order to speed up the query execution time, it makes sense to exploit the effort already done by the DBMS with previous queries. In fact, inductive databases can materialize the result of (some of the) previous queries. In this way, previous results are available to the mining engine which can "reuse" some of the information contained in them in order to reduce the workload. Indeed, since nowadays the storage space is critic to a lesser extent, our aim is to reduce as much as possible the computational work of the data mining engine. Furthermore, we suppose that the mining engine works in an environment similar to a data warehouse, in which database content updates occur rarely and in known periods of time. This greatly simplifies the task, since previous results are considered up to date and can be usefully exploited to speed up the execution of current queries. Therefore, we suggest that the execution plan of a constraint-based query should take into consideration also the results of previous queries, already executed and readily available. The necessity of storing and exploitiong a collection of query results, has been recognized previously also in [16] in which they propose a rule query language for the postprocessing of rules based on their statistical properties or elements.

We present here an incremental approach (originally proposed in [17]) that computes the result of a query starting from the result of a previous, more general query. The new result is computed by enforcing on the previous result set the new constraints. We notice that several "incremental" algorithms have been developed in the data mining area [18,19,20,21], but they address a different issue: how to efficiently revise the result set of a mining query when the database source relations get updated with new data. In this Chapter we show that the new incremental query evaluation technique is beneficial and reduces the system response time. First of all because previous results allow pruning of the itemset lattice. Secondly, because the mining constraints of the current query allow to prune the search space even more. Of course, we assume that the system relies on an optimizer who is entitled to recognize query equivalence and query containment relationships in order to identify the most convenient result from which starting the incremental computation. [22] describes a prototype of such an optimizer and shows that its execution time is negligible (in the order of milliseconds) for most practical applications.

Our main contributions here, are two incremental algorithms that provide a fast solution to the case of query containment. The first one exploits the somewhat implicit assumption made in almost all previous works in constraint-based mining: properties on which users define constraints are functionally dependent on the item to be extracted, i.e., the property is either always true or always false for all occurrences of a certain item in the database. In this case, it is possible to establish the satisfaction of the constraint considering only the properties of the item itself, that is, separately from the context of the database in which the item is found (e.g., the purchase transaction). In [22], we characterized the constraints that are functionally dependent on the item extracted and called them *item dependent* (ID) constraints. The exploitation of these constraints proves to be extremely useful for incremental algorithms. Indeed, ID constraints allow the selection

of the valid itemsets in advance, based on their characteristics, that hold separately from the transactions in which itemsets are found. Similar reasoning occurs also for *succinct* constraints [23] with the difference that these latter consist in properties to be evaluated on each single item separately from the other items of the itemset. In other words, they do not foresee aggregate properties on itemsets.

In [22], another class of constraints, namely the *context dependent* (CD) constraints, was introduced as well. Context dependent constraints occur very often in many important application domains, such as in business (e.g., analysis of stock market data) [24] in science (e.g., meteorological forecast) [25] but also in the traditional applications of data mining, such as in market basket analysis [26].

In order to offer a first grip to the intuition about this novel concept, let us explain it by means of a simple example which applies to the analysis of stock market data. To this purpose, we assume that the database to be analyzed contains the attributes:

`date` — the date of interest;
`stock` — the name of the stock;
`price` — a categorical attribute assuming values in {`increased`, `not varied`, `decreased`}.

In such a context, the user may be interested in whether there exists any (negative) correlation between groups of stock items. For instance, it may want to associate stocks for which `price=increased` with ones for which `price=decreased` instead. As an example of the result, he/she may find useful to discover that $\ll$*when price of AT&T and Microsoft stocks increase, then the price of Sun Microsystems decreases with a probability of* 78% $\gg$.

In this case, the stock price, which is the feature on which constraints are evaluated, does not depend only on the stock, but it also depends on another variable (time). Time and stock together provide the context in which price is determined. Therefore, in contrast to ID constraints, the satisfaction of CD constraints cannot be decided without reading the contextual information present in the database transaction. While, in the simplest situations, the problem may be solved by filtering the database relation before the mining process, such a filtering is not possible when different predicates are given for the body part and the head part of the rules or when constraints on aggregates must hold on the sets. In this latter cases, CD constraints proved to be very difficult to be dealt with. In fact, a CD constraint is not necessarily satisfied by a certain itemset in all its instances in the database. And this fact has big influence on the possibilities of pruning that constraints allow on the lattice search space. In fact, even if an itemset, satisfying a CD constraint *within a transaction*, satisfies one of the well studied properties of monotonicity or anti-monotonicity over the itemset lattice, the same properties do not necessarily hold for that itemset in the whole database. Unfortunately, most of the state of the art algorithms [4,23,27], are based instead on the principle that those properties hold for a certain pattern database wide.

As far as incremental mining is concerned, the presence of a CD constraint in a query implies that one needs to carefully check whether the constraints are satisfied by scanning the transaction table. In the following, we present a new algorithm which is able to deal with context dependent constraints. We show that incremental algorithms are valuable tools even in this setting.

Despite the lack of studies on algorithms dealing with contextual characteristics of itemsets, CD constraints have revealed to be of a certain importance in the extraction of knowledge from databases. For instance, in [26] the authors claim it is important to identify the contextual circumstances in which patterns hold. They propose to reason on circumstances organizing them in a lattice and searching there the most general circumstances in which patterns hold. Their work, however, restricts reasoning on circumstances to conjunctive statements and makes no use of available query results. On the contrary, CD constraints proposed here can be organized with no restriction and queries are allowed to contain general boolean predicates. Secondly, the proposed approach makes a significant usage of available results of previous queries (if the incremental approach results effective with respect to a conventional execution, i.e., by scratch). The incremental option for a data mining algorithm is of course preferable in an inductive database system, since it allows the exploitation of all the available informations in the system in order to speed up the response time.

The rest of the paper is organized as follows. Section 2 presents some preliminary definitions, discusses the properties of queries of the containment relationship. Section 3 and Section 4 present two incremental algorithms able to deal with item dependent and context dependent constraints respectively. Section 5 shows a first set of experimental assessments of the incremental algorithms. In order to fully evaluate the validity of the incremental algorithms in general and with a fair comparison, since in literature there are no algorithms that extract association rules with context dependent constraints, we propose in Section 6 a baseline miner algorithm (called CARE) that is non-incremental and is able to extract association rules with item or context dependent constraints. Finally, in Section 7 we study the worthiness of the incremental approach using the CARE algorithm as a baseline miner. Section 8 draws some conclusions.

2 Preliminary Definitions and Notation

Let us consider a database instance D and let T be a database relation having the schema TS=$\{A_1, A_2, \ldots, A_n\}$. A given set of functional dependencies Σ over the attribute domains $dom(A_i)$, $i = 1..n$ is assumed to be known.

As a running example, let us consider a fixed instance of a market basket analysis application in which T is a `Purchase` relation that contains data about customer purchases. In this context, TS is given by {`tr, date, customer, product, category, brand, price, qty`}, where: `tr` is the purchase transaction identifier, `customer` is the customer identifier, `date` is the date in which the purchase transaction occurred, `product` is the purchased product identifier, `category` is the category to which the product belongs, `brand` is the manufacturer of the product, `price` is the product price, and `qty` is the quantity purchased in transaction `tr`. The Σ relation is {`product` $\rightarrow$ `price`, `product` $\rightarrow$ `category`, `product` $\rightarrow$ `brand`, {`tr, product`} $\rightarrow$ `qty`, `tr` $\rightarrow$ `date`, `tr` $\rightarrow$ `customer`}. It should be noted, however, that the validity of the framework is general, and that it does not depend on either the mining query language or the running database example.

Of course, the above schema could also be represented over a set of relations and dimensions adopting the usual data warehouse star schema. However, the non-normalized

form is more readily explained as well as more common in a data mining environment. In fact, data mining practitioners usually preprocess the data warehouse in order to obtain a database schema similar to the one introduced.

As above mentioned, the system tries to exploit past results in order to react more promptly to user requests. In order to work, such a system must be able to recognize that syntactically different queries are, actually, similar. The following definition, which introduces the notion of *grouping equivalence*, allows the system to recognize that two ways of partitioning the database are equivalent. Clearly, this could be done by building the partitions and checking whether they are identical. However, this approach is clearly too costly. Instead, our approach is to exploit known domain knowledge in order to obtain an answer without actually accessing the database.

Definition 1. *Grouping equivalence relationship.*

Two sets of attributes K_1 and K_2 are said to be `grouping equivalent` *if and only if for any relation T defined on TS:*

$$\forall t_1, t_2 \in T : t_1[K_1] = t_2[K_1] \Leftrightarrow t_1[K_2] = t_2[K_2]$$

where $t_1[K_1]$ is the projection of the tuple t_1 on the attributes in K_1.

Put in other words: K_1 and K_2 are grouping equivalent if and only if $K_1 \leftrightarrow K_2$ (where $\leftrightarrow$ denotes a bidirectional functional dependence).

Example 1. *As we pointed out, the grouping equivalence relation has been introduced as a tool to distinguish whether two set of attributes partition the database in the same way. The following table reports a case where the set of attributes* {`tr`} *and* {`date`, `customer`} *are grouping equivalent, while, for instance* {`date`} *and* {`product`} *are not.*

tr	data	customer	product
1	10/1/2005	CustomerA	Milk
1	10/1/2005	CustomerA	Bread
1	10/1/2005	CustomerA	Beer
2	10/1/2005	CustomerB	Meat
2	10/1/2005	CustomerB	Biscuits
3	12/1/2005	CustomerA	Milk
3	12/1/2005	CustomerA	Biscuits

Let us notice, however, that this example is an oversimplification of what expressed in Definition 1. In fact, here it is reported only a single database instance and we asked to check the grouping equivalence relationship on it. The definition, instead, requires the relation to hold for all *database instances of a given database schema. This actually implies that the relation must be checked using known functional dependencies.*

Sets of attributes that are grouping equivalent form a grouping equivalence class E. As an example of the usefulness of this concept, let us notice that if two queries differ only for the way the relations are grouped and the grouping attributes used in

the two queries are in the same equivalence class, then the two queries are bound to be equivalent.

We assume to know about a set of grouping equivalence classes $E_1 \ldots E_j$.

Example 2. *In the* `Purchase` *example, the following non trivial equivalence class may be found:*

E_1= {{`tr`}, {`date, customer`}, {`tr, date`}, {`tr, customer`}, {`tr, date, customer`}}.

In writing a mining query, the user must specify, among the others, the following parameters:

- The *item attributes*, a set of attributes whose values constitute an item, i.e., an element of an itemset. The language allows one to specify possibly different sets of attributes, one for the antecedent of association rules (body), and one for the consequent (head).
- The *grouping attributes* needed in order to decide how tuples are grouped for the formation of each itemset. The grouping attributes, for the sake of generality and expressiveness of the language, can be decided differently in each query according to the purposes of the analysis.
- The *mining constraints* specify how to decide whether an association rule meets the user needs. In general, a mining constraint takes the form of a boolean predicate which refers to elements of the body or of the head (possibly on the values of any of the attributes in TS, e.g., kind of product, price or quantity). Since we want to allow different constraints on the body and on the head of the association rules, we admit two separate constraint expressions for each part of the rule.
- An expression over a number of *statistical measures* used to reduce the size of the result set and to increase the relevance of the results. This evaluation measures are evaluated only on the occurrences of the itemsets that satisfy the mining constraints.

Usually in market basket analysis, when the user/analyst wants to describe by means of itemsets the most frequent sales occurred in purchase transactions, the grouping attribute is `tr` (the transaction identifier) and the itemsets are formed by the projection on `product` of sets of tuples selected from one group. However, for the sake of generality and of the expressive power of the mining language, grouping can be decided differently in each query. For instance, if the analyst wants to study the buying behavior of customers, grouping can be done using the `customer` attribute, or if the user wants to study the sales behaviour over time he/she can group by `date` or by week or month in the case these attributes were defined.

Users may exploit the mining constraints in order to discard uninteresting itemsets and to improve the performances of the mining algorithm.

More formally, a mining query for the extraction of association rules is described as the 7-tuple:

$$Q = (T, G, I_B, I_H, \Gamma_B, \Gamma_H, \Xi)$$

where: T is the database table; G is the set of grouping attributes; I_B and I_H are the set of item attributes respectively for the body and the head of association rules; Γ_B and

Γ_H are boolean expressions of atomic predicates specifying constraints for the body and for the head of association rules; Ξ is an expression on some statistical measures used for the evaluation of each rule.

We define an atomic predicate to be an expression in the form:

$$\mathtt{A_i}\theta\mathtt{v_{Ai}}$$

where θ is a relational operator such as $<, \leq, =, >, \geq, \neq$, and $\mathtt{v_{Ai}}$ is a value from the domain of attribute A_i.

Ξ is defined to be a conjunction in which each term has the form

$$\xi\theta\mathtt{v}$$

where ξ is a statistical measure for the itemset evaluation, v is a real value, and θ is defined as above.

For the sake of simplicity, in this paper we focus on the *support count* and *confidence* statistical measures. The extension to other measures should be straightforward. The support count is the number of distinct groups containing both the itemsets in the association rule. Confidence is the ratio between the association rule support and support of the body.

Example 3. *The query*

$$Q=(\texttt{Purchase}, \{\texttt{tr}\}, \{\texttt{product}\}, \{\texttt{product}\},\ \texttt{price}>100, \texttt{price}\geq 200,\ \textit{support count}\geq 20 \textit{ AND confidence}\geq 0.5)$$

over the Purchase *relation (first parameter) extracts rules formed by* products *in the body (third parameter) associated to* products *in the head (fourth parameter), where all the products in the rule have been sold in the same transaction (second parameter). Moreover, the* price *of each product in the body must be greater than 100 (fifth parameter) and the* price *of each product in the head must be greater or equal to 200 (sixth parameter). Finally, the support count of the returned rules must be at least 20 and the confidence of the rules at least 0.5. Even if the query syntax we gave is best suited for the purposes of this paper, it is quite unfriendly when it comes to understandability. Many languages have been proposed in the literature that can easily express the kind of constraints we introduced in this paper. For instance, query Q could be expressed in the* Minerule *[8] language as follows:*

MINERULE Q
SELECT DISTINCT $1..n$ `product` AS BODY, $1..n$ `product` AS HEAD, SUPPORT, CONFIDENCE
WHERE BODY.`price`> 100 AND HEAD.`price`≥ 200
FROM `Purchase`
GROUP BY `tr`
EXTRACTING RULES WITH SUPPORT COUNT:20, CONFIDENCE: 0.5

Now that we have seen how constraint-based mining queries are formed, let us define two particular types of constraints: the *item dependent* constraints and the *context dependent* ones. In the following, we will denote by $X \rightarrow Y$ a functional dependency (FD) between two attribute sets X (LHS) and Y (RHS) in the database schema TS.

Definition 2. *Dependency set.*
A dependency set *of a set of attributes X contains all the possible RHS that can be obtained from X following a FD $X \rightarrow Y$ in Σ (direct or transitive) such that there is no $X' \subset X$ such that $X' \rightarrow Y$.*

As we did for equivalence classes, we assume to know about a set of dependency sets.

Example 4. *The dependency set of* {product} *is* {category, price, brand}.*The dependency set of* {tr} *is* {customer, date} *while the dependency set of* {tr, product} *is* {qty}. *As a consequence, one can safely assume that:*

- *the value of* product *can be used to determine the values of attributes* category, price, *and* brand*;*
- *the value of* tr *and* product *uniquely determines the value of* qty.

Definition 3. *Context dependent and item dependent constraints.*
Given a query

$$Q = (T, G, I_B, I_H, \Gamma_B, \Gamma_H, \Xi)$$

let us consider an atomic predicate $\mathtt{P}(\mathtt{A}) \in \Gamma_B$ *(respectively* Γ_H*).* $\mathtt{P}(\mathtt{A})$ *is defined to be an* item dependent constraint *if and only if* A *belongs to the dependency set of I', where $I' \subseteq I_B$ (respectively, $I' \subseteq I_H$). If* $\mathtt{P}(\mathtt{A})$ *is not an item dependent constraint, then it is a* context dependent constraint.
A query Q is said to be item dependent if all atomic predicates in Γ_B and Γ_H are item dependent constraints. If Q is not item dependent, then it is context dependent.

We notice that an itemset $\mathcal{I}$ satisfies an item dependent constraint either in any database group in which it occurs, or in none. This immediately implies the following:

Lemma 1. *Statistics for itemsets with item dependent constraints.*
An itemset $\mathcal{I}$ that satisfies an item dependent constraint in a mining query has a statistical measure that is a function of the total number of groups in which $\mathcal{I}$ occurs in the given database instance.

On the contrary, a context dependent constraint might possibly be satisfied by some, but not all, occurrences of itemset $\mathcal{I}$. Then, the statistical measure cannot be evaluated on the number of groups in which the itemset appears. In fact, this number may differ from (i.e., be larger than) the number of groups in which the itemset satisfies the constraint.

We now give some properties that allow to identify the existence of the containment relationship between two mining queries. These conditions provide the theoretical ground supporting the algorithms that we discuss in Section 3 and 4. Through all the discussion we will denote with Q the query issued by the user at the present time and that we want to "optimize", and with Q^i the ones in the past queries repository.

To begin with, we notice that not all Q^i are suitable candidates for testing the containment relationship. In fact, many of them may be built using features which immediately hinder the possibility of finding the relationship to hold. In the following, we

present a simple test which can be used to filter out those queries very efficiently. Further work is probably needed under this viewpoint, but for the moment we consider this aspect as future work. For the time being, we define the concept of candidate queries for containment of Q (that is those queries that may contain Q) as follows.

Definition 4. *Candidates for query containment of Q.*
Let us consider

$$Q^i = (T, G^i, I_B^i, I_H^i, \Gamma_B^i, \Gamma_H^i, \Xi^i)$$
$$Q = (T, G, I_B, I_H, \Gamma_B, \Gamma_H, \Xi)$$

A previous query Q^i is a candidate for containment of Q, if the following conditions hold:

1. *G^i is in the same grouping equivalence class of G*
2. *I_B^i is in the same grouping equivalence class of I_B*
3. *I_H^i is in the same grouping equivalence class of I_H*

Therefore, a first criterion for selecting the candidate queries for containment of Q can be based on testing the above three conditions. That is, we require that the queries are "grouping equivalent" (i.e., both the queries partition the input data in the same groups) and that they use an "equivalent" description for the items in the body and in the head part of the association rules.

Let us denote by R the set of association rules returned by Q and by R^i the set of association rules returned by Q^i.

Theorem 1. *Properties of Query Containment.*
Given

$$Q^i = (T, G^i, I_B^i, I_H^i, \Gamma_B^i, \Gamma_H^i, \Xi^i)$$
$$Q = (T, G, I_B, I_H, \Gamma_B, \Gamma_H, \Xi)$$

Let Q^i be a candidate for containment of Q.
Let the following hypothesis of entailment between constraints of Q and Q^i be fulfilled:

$$\Gamma_B \Rightarrow \Gamma_B^i$$
$$\Gamma_H \Rightarrow \Gamma_H^i$$
$$\Xi \Rightarrow \Xi^i$$

Under these conditions, $R \subseteq R^i$. Furthermore, the support count (`sup_count`*) of an association rule $r \in R$ is upper bounded by the support count (*`sup_count`^{i}*) of the same rule in R^i.*

Proof. Assuming that Q^i is a candidate for containment of Q implies that Q and Q^i partition the database in the same groups and extract from them the same sets of potential association rules. That is, if the two queries did not contain any constraint, then their result sets would be identical. Let us call the result set of the unconstrained query

R'. In order to prove the containment relation, it is left to show that any rule r which is selected from R' by Q is also selected by Q^i. This is actually immediate, in fact, for any such rule: $r \in R \Rightarrow \Gamma_B(r) \wedge \Gamma_H(r) \wedge \Xi(r) \Rightarrow \Gamma_B^i(r) \wedge \Gamma_H^i(r) \wedge \Xi^i(r) \Rightarrow r \in R'$. Here, the first and the last implications hold since, by definition, a rule belongs to the result set of a query Q' if and only if it belongs to the unconstrained version of Q' and satisfies all the constraints in it. The middle entailment, instead, is directly implied by the assumptions of the theorems.

Moreover, any itemset which satisfies Γ_B in a database group also satisfies Γ_B^i in the same group. The same holds for the head constraints. In addition, there might exist some database groups in which Γ_B^i and Γ_H^i are satisfied, but Γ_B or Γ_H are not. Therefore, the support count of any rule r in R^i is bound to be not lower than the support count of r in R.

The following lemma specializes the previous theorem for queries with item dependent constraints.

Corollary 1. *Query containment with item dependent constraints.*

Under the same hypotheses of Theorem 1, let us also assume that the queries are item dependent. Then, a rule $r \in R \cap R^i$ has the same support count in both Q and Q^i. A rule r, such that $r \in R^i$ but $r \notin R$, has a support count equal to zero in Q.

Proof. By definition, an item dependent constraint is satisfied by all the occurrences of a given itemset in the database or by none. Thus, an association rule in R that satisfies Γ_B and Γ_H satisfies also Γ_B^i and Γ_H^i in the same number of groups (and thus satisfies both Ξ and Ξ^i). Therefore its support count will be the same in both the result sets.

However, a rule in R^i which does not satisfy Γ_B and Γ_H will not satisfy them in all the database groups in which it occurs. Thus, it will have a support count equal to zero in Q.

3 An Incremental Algorithm for Item Dependent Constraints

In a previous work [22], we showed that item dependent constraints are particularly desirable from the viewpoint of the optimization of languages for data mining. In particular, we showed that we can obtain the result of a newly posed query Q by means of set operations (unions and intersections) on the results of previously executed queries. We qualify this approach to itemset mining as *incremental* because instead of computing the itemsets from scratch it starts from a set of previous results. In this paper, we are interested in studying the situation of query containment, that is, to consider situations in which query Q imposes a more restrictive set of constraints with respect to a previous query, here denoted with Q^i. In this paper, we show that item dependent constraints can also be exploited to simplify the problem of incremental mining. In fact, it turns out that, in order to retrieve the desired rules, it suffices to identify the rules in the previous results that satisfy the new constraints. As the results in Section 2 imply, this is not generally true in a situation involving context-dependent constraints. In fact, in the latter case, one needs to carefully update the statistical measures of the rules as well (see Section 4).

In Section 2, we showed that under the item dependency assumption, whenever a query Q^i is found to contain Q, it is rather easy to extract the new results from past ones. In fact, it suffices to search in R^i those rules which satisfy the requirements of Q and to copy them verbatim (along with their support counts) into the new result set.

A sketch of this incremental approach is reported in Algorithm 1. The algorithm is very simple: it checks which of the rules in R^i satisfy the constraints in Q and updates R accordingly. It is important to notice that testing Γ_B and Γ_H is a feasible and efficient operation. In fact, since the constraints are item dependent, their evaluation does not require to access the whole (possibly huge) facts table. On the contrary, it merely requires to access the dimension tables and to check the constraints using the informations found therein. Since those tables does usually fit into the main memory or in the DBMS buffer memory, this rarely becomes a demanding operation. In addition, the Ξ constraint is also easily checked by using the statistical measures stored together with the rules in the past result.

Algorithm 1: Item Dependent (ID) incremental algorithm

Data : $R^i = \{b \rightarrow h\}$: old result set;
$Q = (T, G, I_B, I_H, \Gamma_B, \Gamma_H, \Xi)$: the query issued by the user;
Result : R : the set containing the rules satisfying Q
$R \leftarrow \emptyset$;
foreach $r \in R^i$ **do**
  if $\Gamma_B(r) \wedge \Gamma_H(r) \wedge \Xi(r)$ **then**
    $R \leftarrow R \cup \{r\}$;
  end
end

4 An Incremental Algorithm for Context Dependent Constraints

In this section we propose an incremental algorithm which is able to construct the result of a new mining query Q starting from a previous result R^i even when the mining constraints are not item dependent. At the best of our knowledge this is the first attempt to write a mining algorithm able to deal with context dependent constraints [3,4,11,28,29,30].

The algorithm is best described by considering two separate steps. In the first one, the algorithm reads rules from R^i and builds a data structure which keeps track of them. We call this structure the *BHF* (Body-Head Forest) and describe it in Section 4.1. We notice that since the BHF is built starting from a previous result set and represent only rules found therein, this corresponds to a first pruning of the search space. In fact, subsequent operations will simply disregard rules that do not appear in it (the correctness of this approach is implied by Theorem 1).

In the second step, the algorithm considers two relations $T_b = \{< i, g > | i \in I_B, g \in G, \Gamma_B \; is \; true\}$ and $T_h = \{< i, g > | i \in I_H, g \in G, \Gamma_H \; is \; true\}$, containing the items and the group identifiers (GIDs) that satisfy the mining constraints in query Q. T_b and T_h are obtained by evaluating the constraints on the fact table. Their role is

to keep track of the context in which the itemsets appear. In fact, the context dependent constraints require that their validity is checked group by group. The two relations fullfil this purpose. We notice that this is another point in which the search space is pruned. In fact, the constraints are evaluated on the database and the items which do not satisfy the mining constraints are removed, once and for all, from the input relations.

Finally, the algorithm updates the counters in the BHF data structure accordingly to the itemsets found in T_b and T_h. The counters are then used to evaluate the statistical measures needed to evaluate whether the constraint Ξ is satisfied.

4.1 Description and Construction of the BHF

A BHF is a forest containing a distinguished tree (the body tree) and a number of ancillary trees (head trees). The body tree is intended to summarize the itemsets which will be found in the body part of the rules. Importantly, in any tree, an itemset B is represented as a single path and vice versa. In the node corresponding to the end of a path, it is stored a head tree and the (body) support counter.

Analogously, the head tree (associated to the itemset B) is intended to summarize the itemsets that will appear in the head part of the rules (having the body equal to B). A head tree is similar in structure to a body tree with the exception that there are no head trees associated to the end of any path. A path in a head tree corresponds to an itemset H and is associated to a counter which stores the support of the rule.

Figure 1 gives a schematic representation of a BHF.

In the following, we will make use of the following notation: given a node n belonging to a body tree or to a head tree, we denote with n.child(i) the body (respectively the head) tree rooted in the node n in correspondence of the item i. For instance, in the root node of the BHF reported in Figure 1, there are four items, and three non-empty children; root.child(a) denotes the body node containing the items c, d, and z. In a similar way we denote the head tree corresponding to a particular item i in a node n using the notation n.head(i). We also assume that both body elements and head elements are sorted in an unspecified but fixed order. We denote with $B[k]$ (respectively with $H[k]$) the k-th element of the body B (respectively head H) w.r.t. this ordering. Finally, in many places we adopt the standard notation used for sets in order to deal with BHF nodes. For instance, we write $i \in n$ in order to specify that item i is present in node n; we write $n \cup i$ in order to denote the node obtained from n by the addition of item i.

Procedure insertRule

Data : root : the BHF root node
$B \rightarrow H$: the rule to be inserted

headTree $\leftarrow$ insertBody(root, B, 1) ;
insertHead(headTree, H, 1);

Procedure insertRule describes how a rule is inserted in the BHF structure. The procedure consists in two steps. In the first one, the body B of the rule is inserted in the body tree (see Function insertBody) and the node n corresponding the end of the path

associated to B is determined. In the second one, the head is inserted and attached to n (see Procedure insertHead).

We notice that the hierarchical structure of the BHF describes a compressed version of a rule set. In fact, two rules $B_1 \rightarrow H_1$ and $B_2 \rightarrow H_2$ share a sub path in the body tree provided that B_1 and B_2 have a common prefix. Analogously they share a sub path in a head tree provided that $B_1 \equiv B_2$ and H_1 and H_2 have a common prefix.

Function insertBody

Data : n : a BHF node
B : an itemset
k : an integer

if $B[k] \notin n$ **then**
$n \leftarrow n \cup B[k]$
end
if $k < \text{size}(B)$ **then**
insertBody(n.child($B[k]$), $B, k+1$)
else
return n.head($B[k]$)
end

Procedure insertHead

Data : n : a BHF node
H : an itemset
k : an integer

if $H[k] \notin n$ **then**
$n \leftarrow n \cup H[k]$
end
if $k < \text{size}(H)$ **then**
insertHead(n.child($H[k]$), $H, k+1$)
end

4.2 Description of the Incremental Algorithm

Here, we assume that a BHF has been initialized using the rules in the previous result set R^i (but with their support count equal to zero: it will adjusted in the following step). We will show how the BHF is updated and the rules are extracted in order to build the novel result set R.

In the following we will denote with:

- $T_b^{ITEM}[g] \equiv \{i \mid (g,i) \in T_b\}$ and with $T_h^{ITEM}[g] \equiv \{i \mid (g,i) \in T_h\}$ the set of items in group g that satisfy the body and the head constraints, respectively.
- $T_b^{GID} \equiv \{g \mid (g,i) \in T_b\}$ and with $T_h^{GID} \equiv \{g \mid (g,i) \in T_h\}$ the set of GIDs in T_b and in T_h, respectively.
- $T_b^{GID}[i] \equiv \{g \mid (g,i) \in T_b\}$ and with $T_h^{GID}[i] \equiv \{g \mid (g,i) \in T_h\}$ the set of group identifiers in which item i satisfies the body and the head constraints, respectively.

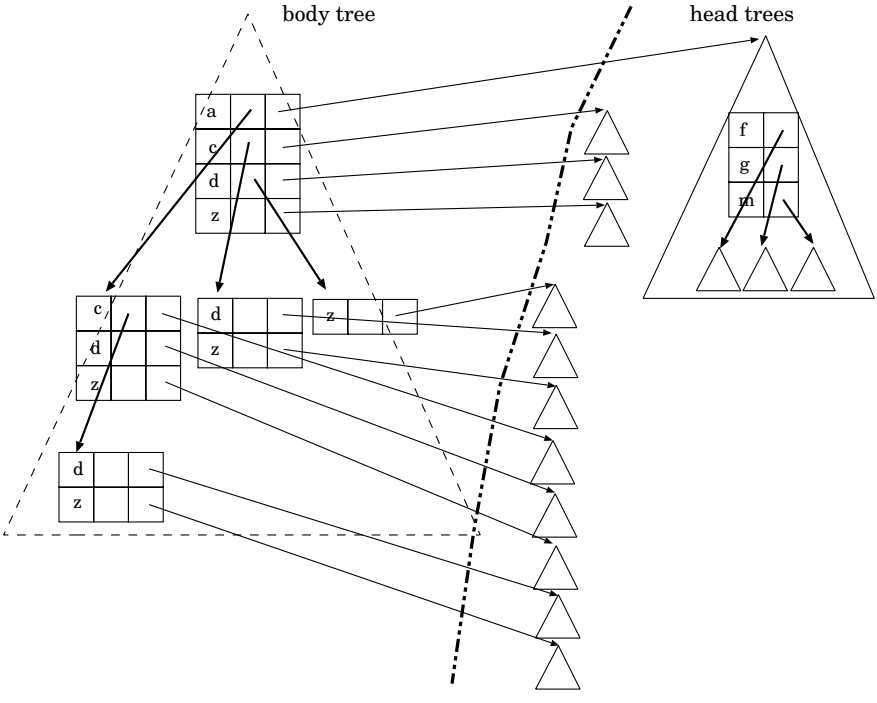


Fig. 1. Example of BHF

- τ the support threshold chosen by the user
- $r.body$ the body of rule r and with $r.head$ the head of rule r

For the sake of readability, we reported in Algorithm 5 a simplified version of the incremental algorithm which has the advantage of making its intended behavior clear. We believe it is self explanatory. Instead, the implemented version greatly improves on the simple, reported one. Let us now assume that the working of Algorithm 5 is clear. We now try to give an idea of how the implemented version works and improve on it. The main difference is that in order to avoid the checking of each and every rules in the BHF (see Procedure incrRuleSupp), the algorithm performs a depth first search in the BHF. In this way it is able to find all the rules which need their support to be updated for a given group g without actually enumerate all possible rules.

Let us illustrate the way the implemented algorithm proceeds by means of an example. Let us assume that the BHF in Figure 1 is given, and for group g it holds $T_b^{ITEM}[g] = \{a, c, z\}$ and $T_h^{ITEM}[g] = \{f, l\}$. In order to update the support counters in the BHF tree, the algorithm proceeds as follows. The root of the body tree is examined in order to check which of the items it contains are in $\{a, c, z\}$. The item "a" is found and its support counter is therefore incremented (we recall that the supports of the body part of the rules are needed in order to evaluate the rules confidence values). Since $T_h^{ITEM}[g] \neq \emptyset$, the head tree associated to "a" is examined. As a result, the algorithm increments the support counters associated to the items in $T_h^{ITEM}[g]$. That is

the ones corresponding to the rules having "a" in their body and any subset of $\{f, l\}$ in their head.

Once the updating of the counters in the head tree rooted in "a" is complete, the algorithm examines the rest of the body tree. In the root node of the sub tree rooted in "a", the algorithm searches whether it contains items belonging to $\{c, z\}$. It finds item "c" and increments its support. As in the previous case, the algorithm examines the head tree associated to this item and updates the support counters accordingly.

Then, the sub tree rooted in "c" is examined in a similar way. Whenever a body tree node does not contain any items in $T_b^{ITEM}[g]$, the algorithm backtracks to its ancestor looking for items in $T_b^{ITEM}[g]$ that have not been "visited" yet in that node.

Algorithm 5: Context Dependent (CD) incremental algorithm

```
Data   : T_b, T_h
Result : R_2
for all GID g ∈ T_b^GID do
    incrRuleSupp(BHF, T_b^ITEM[g], T_h^ITEM[g] )
end
for all rule r ∈ BHF do
    if Ξ(r) then
        R_2 ← R_2 ∪ r
    end
end
```

5 Results

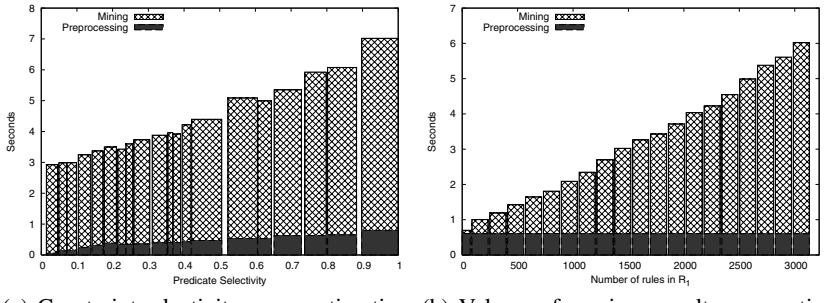
The two incremental algorithms presented in this paper have been assessed on a database instance, describing retail data, generated semi-automatically. We generated a first approximation of the fact table (`purchases`) using the synthetic data generation program described in [31]. The program has been run using parameters $|T| = 25$, $|I| = 10$, $N = 1000$, $|\mathcal{D}| = 10,000$, i.e., the average transaction size is 25, the average

Procedure `incrRuleSupp`

```
Data   : a BHF,
         T_b, T_h
Result : It updates the support counters in the BHF
for all r ∈ BHF do
    if r.body⊆ T_b^ITEM[g] then
        r.body.support++;
        if r.head⊆ T_h^ITEM[g] then
            r.support++;
        end
    end
end
```

size of potentially large itemsets is 10, the number of distinct items is 1000 and the total number of transactions is 10.000. Then, we updated this initial table by adding some attributes which provide the details (and the contextual information) of each purchase. We added some item dependent features (such as "category of product" and "price") and some context dependent features (such as "discount" and "quantity"). The values of the additional attributes have been generated randomly using uniform distributions on the respective domains[1].

We note here how a single fact table suffices for the objectives of our experimentation. While, in fact, the characteristics of the database instance (e.g., total database volume and data distribution) are determinant in order to study the behavior of mining algorithms, this is not so when we are up to study incremental algorithms. Indeed, as simple complexity considerations point out, the important parameters from the viewpoint of the performance study of incremental algorithms are the selectivity of the mining constraints (which determine the volume of data to be processed from the given database instance) and the size of the previous result set.

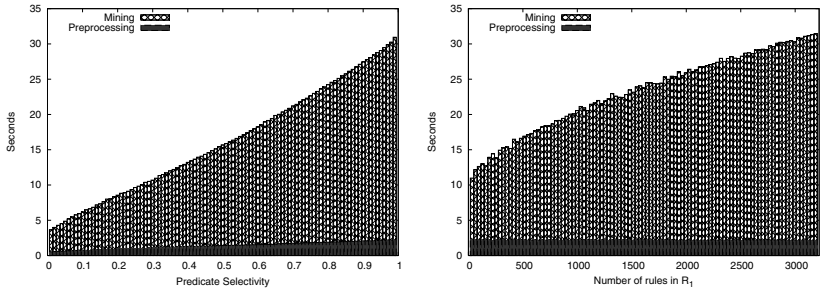


(a) Constraint selectivity vs execution time (b) Volume of previous result vs execution time

Fig. 2. Empirical evaluation of the item dependent (ID) incremental algorithm

In Figure 2(a) we report the performances of the item dependent incremental algorithm (ID) as the selectivity of the mining constraints changes. We experimented different constraints on the item dependent attributes, letting the constraints selectivity vary from 0% to 100% of the total number of items. In Figure 2(a) we sampled twenty points. Figure 2(b) tests the same algorithm, but it lets vary the number of rules in the previous result set. Again we sampled twenty points (in the range $0 \ldots 3220$). The two figures report the total amount of time needed by the algorithm to complete. In particular, the bars, which represent the single experiments, are divided in two components: the preprocessing time (spent in querying the database to retrieve and store in main memory the items that satisfy the constraints), and the core mining time (needed by the algorithm to read the previous result set and to filter out those rules that do not satisfy the constraints any more).

[1] The dataset can be downloaded from `www.cinq-project.org`



(a) Constraint selectivity vs execution time (b) Volume of the previous result vs execution time

Fig. 3. Empirical evaluation of the CD incremental algorithm

Figures 3(a) and 3(b) report the performances of the context dependent (CD) algorithm. The figures report again the total execution time, specifying how much time was spent for preprocessing and for the core mining task. It is worth noticing, that the CD incremental algorithm performs a greater amount of work with respect to the ID algorithm because the problem it solves is far more complex. In fact, in the preprocessing phase the algorithm must retrieve all the group/item pairs satisfying the constraints and access to them in order to build and update the BHF data structure. Only then, it can retrieve the results from the BHF structure.

A couple of points are worth noting. The execution times of both algorithms increase almost linearly with the increase of the two parameters (constraint selectivity and previous results), but, as it was expected, the item dependent incremental algorithm runs much faster than its counterpart.

Moreover, as the experiments in Section 7 will show, both the algorithms are faster than CARE, a new algorithm (that we introduce in the following section) which, nevertheless, is capable of solving a class of more difficult problems - mining association rules in presence of context-dependent constraints.

CARE, as the most of the algorithms, operates starting from scratch. We emphasize here, that, at the best of our knowledge, CARE is the only mining algorithm capable of dealing with context dependent constraints. Hence, comparisons with other mining algorithms on the field of context dependent constraints are difficult to be made. Our guess is that any state of the art mining algorithm should be able to outperform CARE, on problems defined in terms of item dependent constraints, due to the lower generality of the problem they face.

Interestingly, nevertheless, thanks to the good performances of incremental algorithms also on problems with context dependent constraints, one has always the choice of avoiding the use of CARE. This is done by running first the mining algorithm of his choice (on the problem defined by the query but *without* the context dependent constraints) and then applying the incremental algorithm on top of it (with the addition of context dependent constraints). The issue of which choice is the most promising is out of the scope of this paper. However, we believe that the answer is likely to depend on the problem at hand. In particular, whenever the mining constraints select a very small

part of the original dataset, CARE is likely to be very fast. On the contrary, whenever the result set is small, the incremental algorithm seems more efficient.

6 The Constrained Association Rules Extractor Algorithm

CARE (Constrained Association Rules Extractor) is an algorithm which has been designed to extract association rules in presence of context dependent constraints. Context dependent constraints do not allow the classical two phases algorithms (first, find frequent itemsets, second, extract association rules), because frequency count and constraint satisfaction interact. In principle, a levelwise algorithm, such as Apriori [31], might be used, provided that at each iteration the constraint is checked on the database. We decided to develop CARE starting from Partition [32], since it has been shown to perform better than Apriori on many databases.

Even if CARE can work with item dependent constraints, its main purpose is to provide a first and simple solution to the problem posed by CD constraints, thus providing a baseline comparison for incremental algorithms. However, it is necessary to notice that CARE is still under development in order to overcome its limitations: it is not able to deal with cross mining constraints, such as `BODY.feature1 < HEAD.feature1`, or aggregate constraints, such as `sum(BODY.quantity) > val`. Its main features are the following:

- in contrast with any published algorithm we are aware of, CARE uses two separate structures for storing items to be put in the body and items to be put in the head of an association rule. This is needed because constraints on the *body items* might be different from constraints on the *head items*.
- It maintains for each item a list of group identifiers (*gidlists*) of the transactions in which the item appears, as done, for instance, in Partition [32]. This allows, on one hand, to scan the database only twice, and on the other hand, to keep all needed information to combine body items and head items. In a future implementation, on dense databases, bitmaps will be used to efficiently store and operate on gidlists.

In a constrained mining framework, items must be frequent and satisfy the additional mining constraints. In the general settings we are considering, items in the body of a rule should satisfy a mining constraint Γ_B, whereas items in the head of a rule should satisfy a possibly different mining constraint Γ_H. Such constraints can be evaluated a posteriori, i.e., at the time rules are extracted from frequent itemsets, only if the constraints are item dependent.

In the general case of context dependent constraints (e.g., `BODY.qty > 2` $\wedge$ `HEAD.qty = 1`), items might satisfy the constraints Γ_B or Γ_H in some transactions and not in others, thus influencing the frequency with which an item that satisfies the mining constraints occurs. Moreover, as already said, since the same item in a transaction might satisfy only one of the two constraints Γ_B and Γ_H, it is necessary to maintain separate structures for storing *body itemsets* and *head itemsets*. In particular, there is a structure (named BH) for the items satisfying the constraints on the body. Each entry of this structure contains a triple $< i, \mathrm{Bgids}, \mathrm{HH} >$, where i is an item, Bgids is a list of group identifiers (gidlists) in which the item is found in the source table, and HH is a

structure containing the items satisfying the constraints on the head. In this way, every item that can be in the body of a rule is associated with the set of items that can be put in the head of such a rule. This structure is very similar to the BHF previously described, as it only contains the root nodes of the forest. In the following, we will use set notation for the abstract description of the algorithms, while in the actual implementation of these structures we used hashmaps, for efficiency reasons.

Algorithm 7: The CARE algorithm

Data Structures:
Data : T_b, T_h;
ϵ minimum support threshold;
Result : R
BH = $\{ < i, \text{Bgids}, \text{HH} > | i \in I_B,$
$\text{Bgids} = T_b^{GID}[i], |\text{Bgids}| \geq \epsilon,$
$\text{HH} = \{< j, \text{Hgids} > | j \in I_H,$
$\text{Hgids} = \text{Bgids} \cap T_h^{GID}[j], |\text{Hgids}| \geq \epsilon\}$
$\}$
$R = \text{buildRules}(\text{BH}, \epsilon)$;
return R;

CARE works in two steps: in the first step, BH is initialized by scanning the tables T_b and T_h. Then, the rules are extracted through a recursive process.

The full sketch of CARE is reported in Algorithm 7.

Rules are extracted from the BH structure by first creating a body itemset (see function buildRules that in turn calls buildBody), then creating the corresponding head itemsets for that body (see function buildHead), and repeating the process recursively for all items in the BH structure. Note that each itemset in the body (head) is obtained by union of the current body (head) with another item in the BH structure (HH structure). The gidlist of the candidate itemset is obtained by intersection of the respective gidlists of the two "ancestor" itemsets and its cardinality is finally tested to verify the support constraint.

Function buildRules

Data : BH the BH structure ;
ϵ minimum support threshold;
Result : R
while $(\text{BH} \neq \emptyset)$ **do**
  let $e =< i, \text{Bgids}, \text{HH} > \in \text{BH}$;
  $\text{BH} = \text{BH} - \{e\}$;
  $R = R \cup \text{buildBody}((\{i\}, \text{Bgids}), \text{BH}, \text{HH}, \epsilon)$;
end
return R;

Function buildBody

Data : (CurrentBodyItemset, CurrentGids) current body information ;
BH the body-head structure ;
ϵ minimum support threshold;
Result : R the rules extracted
$R = \text{buildHead}(\text{CurrentBodyItemset}, \text{HH}, (\emptyset, \text{CurrentGids}), \epsilon)$;
while $(\text{BH} \neq \emptyset)$ **do**
  let $e =< i, \text{Bgids}, \text{HH} >\in \text{BH}$;
  $\text{BH} = \text{BH} - \{e\}$;
  $\text{newgids} = \text{Bgids} \cap \text{CurrentGids}$;
  if $|\text{newgids}| \geq \epsilon$ **then**
    $\text{newBody} = \text{CurrentBodyItemset} \cup \{i\}$;
    $R = R \cup \text{buildHead}(\text{newBody}, \text{HH}, (\emptyset, \text{newgids}), \epsilon)$;
    $R = R \cup \text{buildBody}((\text{newBody}, \text{newgids}), \text{BH}, \epsilon)$;
  end
end
return R;

Function buildHead

Data : CurrentBodyItemset current body items ;
HH the Head structure ;
(CurrentHeadItemset, CurrentGids) current head information ;
ϵ minimum support threshold;
Result : R the rules extracted
$R = \emptyset$;
while $(\text{HH} \neq \emptyset)$ **do**
  let $e =< i, \text{Hgids} >\in \text{HH}$;
  $\text{HH} = \text{HH} - \{e\}$;
  $\text{newgids} = \text{Hgids} \cap \text{CurrentGids}$;
  if $|\text{newgids}| \geq \epsilon$ **then**
    $\text{newHead} = \text{CurrentHeadItemset} \cup \{i\}$;
    $R = R \cup \{\text{CurrentBodyItemset} \rightarrow \text{newHead}\}$;
    $R = R \cup \text{buildHead}(\text{CurrentBodyItemset}, \text{HH}, (\text{newHead}, \text{newgids}), \epsilon)$;
  end
end
return R;

Let us illustrate the process through a simple example. Suppose we are given the source table reported in Table 1 and that we want to extract rules that satisfy the following constraints:

- `BODY.qty`$\geq$`1` $\wedge$ `HEAD.qty`$\geq$`5`
- minimum support count = 2

After reading the source table filtered by *body constraints* and *head constraints*, CARE builds the BH structure as reported in Table 2.

Table 1. Example of a source table

gid	item	qty
1	A	2
	B	7
	C	8
2	A	1
	C	6
3	A	1
	B	6
	C	1

Table 2. The BH structure built from the source table in Figure 1

body item	gidlist	HH structure	
A	1,2,3	head item	gidlist
		B	1,3
		C	1,2
C	1,2,3	head item	gidlist
		B	1,3

It should be noted that there is no entry for body item B, because, even though item B is frequent, there are no frequent items for the head associated to B, i.e., no rules with B in the body can be extracted.

Afterwards, buildRules is called and rules are extracted in the following order:

- $A \rightarrow B$
- $A \rightarrow C$
- $AC \rightarrow B$
- $C \rightarrow B$

i.e., for every body, buildHead is called to build the corresponding heads. Then, buildBody is called recursively to build larger and larger bodies. Finally, the next item in the BH structure is taken into consideration and the process repeated.

Of course, a number of optimizations might be implemented by accurately computing gidlist intersections and storing intermediate results. However, the current implementation is sufficiently efficient to be used for comparison with the incremental algorithms presented in the following sections.

7 Comparison Between the CD Incremental Algorithm and CARE

In this section we compare the performances of the CD incremental algorithm with CARE. In the experiments, we want to observe how the dimensions of the problem impact on the performances of the two algorithms. Thus, we designed the experiments by varying one dimension at a time, so that the influence of each dimension is observed

separately with respect to the other ones. The dimensions are (as already pointed out in Section 5): the selectivity of the mining predicates, the support threshold, and the volume (number of rules) of the previous result set. For each experiment, we report the running time of the two algorithms. We notice that, in general, the problem parameters have a different impact on the two algorithms. For instance, the support threshold is probably the parameter which has the highest impact on CARE running time, but the same time, it affects the running time of the incremental algorithm only in a marginal way. Moreover, the size of the result set of a previous (more general) query usually is not an interesting parameter for mining algorithms although it is probably the most important one for the incremental algorithm.

The two algorithms have been assessed on the `purchases` database we introduced in the previous section. Some preliminary considerations on the influence that the typology of the dataset has on the incremental algorithms are necessary.

`purchases` is a sparse dataset. In sparse data, roughly speaking, the volume of results of a mining query, compared with the volume of the original database, is reduced and lower with respect to dense datasets at equal conditions (such as support threshold and constraints selectivity). We believe the main results on the behavior of the incremental algorithm we present in this Section with experiments on a sparse dataset should be still valid on a dense dataset with some differences. First, on a dense dataset the impact of I/O operations for reading the source database is less important if compared with the I/O required for reading the previous result set. This would constitute a disadvantage for the incremental algorithm. Second, in a dense dataset, the larger previous result set would allow a minor pruning on the search space which, on the other side, would be larger because data in a dense dataset are much more correlated. As a conclusion, the volume of the previous result set and that one of the search space should be two issues that should counterbalance each other. However, more insights on dense datasets are reserved for further work.

Extensive preliminary results on `purchases` showed that the incremental algorithm is substantially faster than CARE when a reduced previous result set is available. Indeed, this latter one allows incremental algorithms to do much pruning on the itemsets search space which finally results in a decisive advantage for the incremental algorithm. Hence, here, we want to test the incremental algorithms in a more stressing situation. We assumed that we have a single previous query available, which contained a very low support threshold (namely: 0.0085) and very loose predicates. As a result, the incremental algorithms have at their disposal in the BHF data structures a previous, very large (and thus not filtering) result set, composed of 158,336 rules. In the experiments, the current query is representative of a typical business-value scenario with a medium volume (6056 rules): find all the rules which appeared in 1.2 percent of the transactions and that contained costly to average priced items ("price ≥ 1000").

In the first pool of experiments, we varied just the size of the previous result set. We started from a previous result set which contained 7,931 rules (i.e., 5% of the total), and increased this number repeatedly until the whole volume of the 158,336 rules is reached. Figure 4 reports the result of the experiments. The number (N) of rules contained in the previous result set is plotted against the total running time of the two algorithms (y-axis). Here, we stress the fact that the two algorithms solve the same problem, but

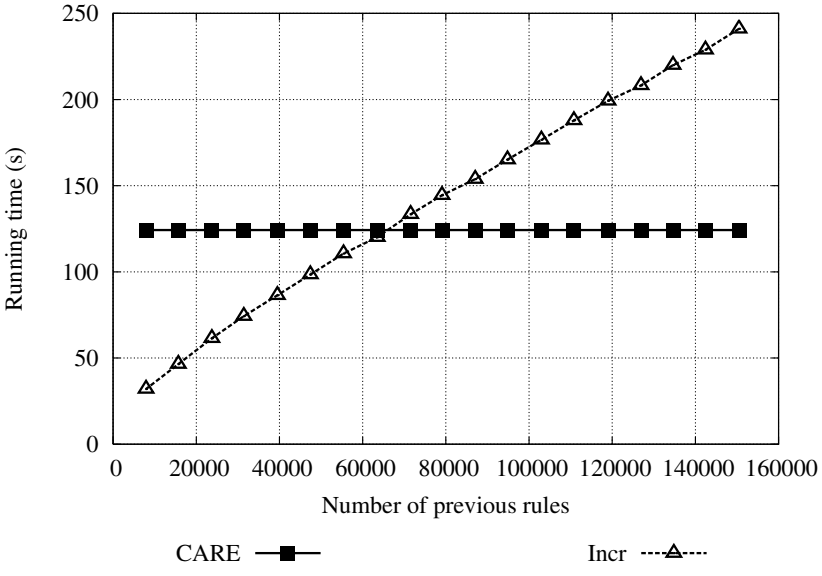


Fig. 4. CARE vs Incremental algorithm, as the cardinality of the previous query result set varies

while the incremental algorithm exploits the result of a previous query, CARE starts from scratch. As we can see, the running time of the incremental algorithm is very low in correspondence to low values of N (the number of rules in the previous result set). Then it increases linearly with N. We want to remark here this linear behavior is desirable, i.e., the scalability of incremental algorithms with respect to one of the main dimensions of the problem. On the other side, since the running time of CARE does not change with the tested parameter, the time spent by the incremental algorithm to solve the problem is bound to overcome the running time of CARE in the limit (i.e., when only a large volume of the previous result is available). In the experiment, this happens for $N = 65,998$ (accordingly to the line which interpolates the displayed data), but it should be noted that this value highly depends on the running time of CARE, i.e., on the support value and on the constraints of the current query. One may wonder why the incremental algorithm does not succeed in showing a better behavior than CARE in all situations (since the incremental algorithm has at its disposal more information than CARE). The answer is that the incremental algorithm has been thought and optimized in order to be extremely fast whenever a good (i.e., restricted and thus filtering) previous answer could be found in the system memory. In order to achieve the desired filtering of the search space, it mainly enforces the initial pruning provided by the results it reads from the disk. This, on one side, allows the algorithm to benefit from a very fast support counters update schema (and allows also a single pass over the database). On the other side, however, it forces the algorithm to read from disk another complete source of information that reveals a choice less competitive when its volume is large (and the pruning not sufficient). In conclusion, the incremental algorithm is a desirable strategy when a reduced previous result is available (if compared with the search space).

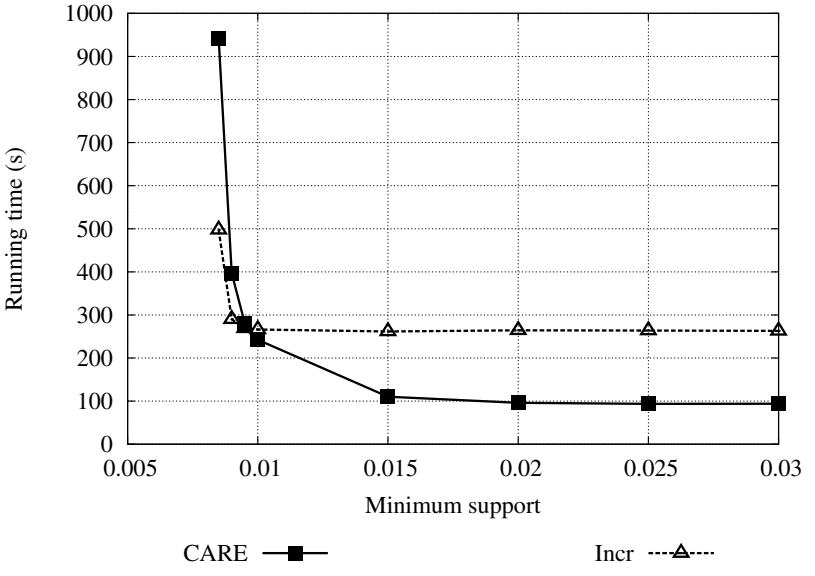


Fig. 5. CARE vs Incremental algorithm, as the support threshold of the current query varies

The trade off between the two algorithms is made evident once again by the results reported in the second set of experiments, shown in Figure 5, where the support threshold (x-axis) is plotted against the running time of the algorithms (y-axis). The previous result set contains the total volume of the large, above mentioned, previous result (158,336 rules). The mining constraints in current query are the same as the ones in the previous set of experiments but the support threshold in the current query varies between 0.0085 (the value set in the adopted, previous query) and 0.03. As it was expected, the incremental algorithm is much less affected by changes of the support threshold than CARE. In particular, we can see that its running time drops almost immediately to about 260 seconds and then it does not change very much. The reason for the drop in the execution time is that as the support threshold decreases, the number of rules given as output increases. Hence, it turns out that, in the current settings, the algorithm takes about 260 seconds to update the 158,336 support counters in BHF structure, while the rest of the time is spent in saving the result on the database. On the contrary, CARE by exploiting the antimonotonicity property of the constraints on minimum support, outperforms the incremental algorithm as the support threshold becomes high enough (reaching the lower bound of 100 seconds). Notably, according to the results reported in Figure 4, the chances are that if the previous query had contained less than 50,000 rules (instead of the $158,336$ present in the current setting), then the incremental algorithm would have outperformed CARE (with execution times lower than 100) no matter the value of the support threshold!

In the last pool of experiments, we set the support threshold equal to 0.0085, the previous results contain 158,336 rules, and let the selectivity of the constraints vary from 0.9010 to 0.9849. Figure 6 reports the results. As usual, the y-axis reports the total

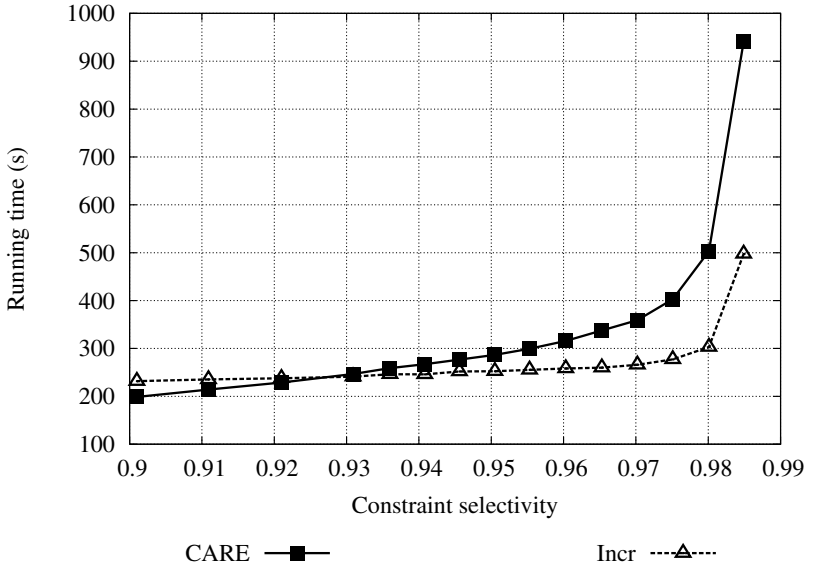


Fig. 6. CARE vs Incremental algorithm, as the constraints in the current query vary

running time of the algorithms, while the x-axis reports the percentage of the source table selected by the mining constraints (i.e., it reports the ratio between the number of rows that satisfy the constraints and the cardinality of the source table). As we can see the running time of both algorithms increases with the percentage of selected rows. Interestingly, this result suggests that the incremental algorithm, despite being faster on larger datasets, does not run as fast when the size of the database become smaller. This is probably due to the overhead the incremental algorithm suffers in order to build the BHF data structure. In order to check this hypothesis, we plotted in Figure 7 the total time needed by the algorithms in order to build the result once the preprocessing steps were completed[2]. As it can be seen, the "core" operations are consistently cheaper in the case of the incremental algorithm (this is expected, since it avoids the costly operations needed to manage the gid lists which are necessary in case of context dependent constraints).

In summary, the use of the incremental algorithm, is very often a winning choice when a suitable past result can be found. However, the choice between an incremental

[2] Which steps of each algorithm contribute to the preprocessing is hard to be stated objectively since the two algorithms make rather different choices in their early steps. However, we decided to consider as preprocessing the steps that are performed before the exploration of the itemsets search space occurs, which is usually a typical operation of the core data mining algorithm. In the case of CARE, we considered the time spent in reading the database as a preprocessing step since it is needed to fill the data structures used later on. In the case of the incremental algorithms we considered as pre-processing the time spent in building the BHF data structure, but not the time spent reading the DB since this one is interleaved with the "navigation" and management of the search space.

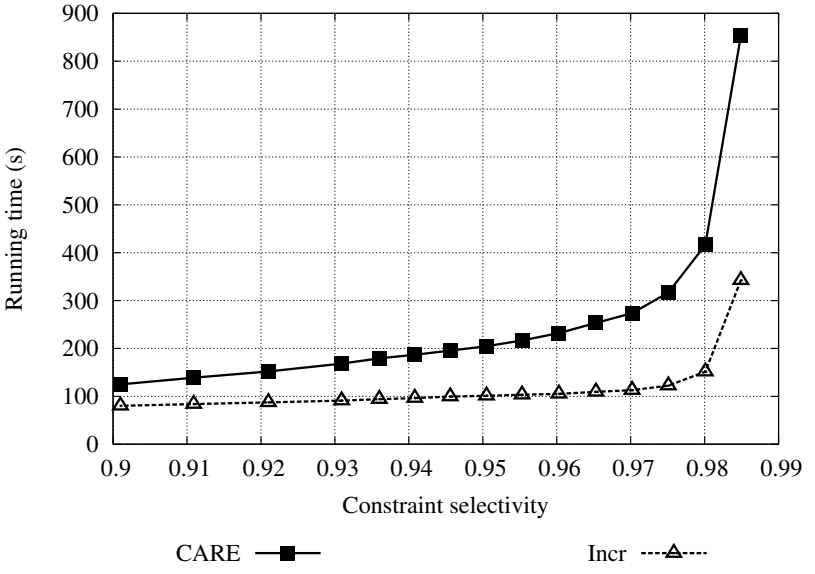


Fig. 7. CARE vs Incremental algorithm, as the constraints of the current query vary (times reported without the preprocessing time)

strategy and a non-incremental one should be made taking into account how the change of the problem dimensions affect the two strategies. In particular, it seems to be clear that whenever a very small previous result can be found, the incremental algorithm is hardly outperformed: it searches a small space and it builds the information needed to find the rules very efficiently. However, when the size of the previous result set grows larger, a "traditional" miner may win, especially when the support threshold is high. In this case, in fact, one looses both the advantages of the incremental algorithm: the algorithm will spend a large part of the time in building the BHF structure out of the previous result, and will probably search a larger space w.r.t. the space searched by algorithms which exploit the antimonotonicity of support.

8 Conclusions

In this paper we proposed a novel "incremental" approach to constraint-based mining which makes use of the information contained in previous results to answer new queries. The beneficial factors of the approach are that it uses both the previous results and the mining constraints, in order to reduce the itemsets search space.

We presented two incremental algorithms. The first one deals with item dependent constraints, while the second one with context dependent constraints. We note how the latter kind of constraints has been identified only recently and that there is very little support for them in current mining algorithms. However, the difficulty to solve mining queries with context dependent constraints can be partially overcome by combining

the "traditional" algorithms proposed so far in the literature, and the context dependent incremental algorithm proposed in this paper.

Moreover, we described a non-incremental algorithm (CARE) for the extraction of constrained association rules, in order to provide a direct comparison for the incremental ones. CARE is specifically designed to deal with context dependent constraints on both the body and the head of association rules and is, to the best of our knowledge, the only one of this type.

In Section 5 and in Section 7, we evaluated the incremental algorithms on a pretty large dataset. The results show that the approach reduces drastically the overall execution time. Whenever we have a small previous result to exploit, or when the support threshold is small, we believe the improvement is absolutely necessary in many practical data mining applications, in data warehouses and inductive database systems.

References

1. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases. In: Proc.ACM SIGMOD Conference on Management of Data, Washington, D.C., British Columbia (1993) 207–216
2. Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., Verkamo, A.I.: Fast discovery of association rules. In Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R., eds.: Knowledge Discovery in Databases. Volume 2. AAAI/MIT Press, Santiago, Chile (1995)
3. Srikant, R., Vu, Q., Agrawal, R.: Mining association rules with item constraints. In: Proceedings of 1997 ACM KDD. (1997) 67–73
4. Ng, R.T., Lakshmanan, L.V.S., Han, J., Pang, A.: Exploratory mining and pruning optimizations of constrained associations rules. In: Proc. of 1998 ACM SIGMOD Int. Conf. Management of Data. (1998) 13–24
5. Tsur, D., Ullman, J.D., Abiteboul, S., Clifton, C., Motwani, R., Nestorov, S., Rosenthal, A.: Query flocks: A generalization of association-rule mining. In: Proceedings of 1998 ACM SIGMOD Int. Conf. Management of Data. (1998)
6. Chaudhuri, S., Narasayya, V., Sarawagi, S.: Efficient evaluation of queries with mining predicates. In: Proc. of the 18th Int'l Conference on Data Engineering (ICDE), San Jose, USA (2002)
7. Imielinski, T., Virmani, A., Abdoulghani, A.: Datamine: Application programming interface and query language for database mining. KDD-96 (1996) 256–260
8. Meo, R., Psaila, G., Ceri, S.: A new SQL-like operator for mining association rules. In: Proceedings of the 22st VLDB Conference, Bombay, India (1996)
9. Han, J., Fu, Y., Wang, W., Koperski, K., Zaiane, O.: DMQL: A data mining query language for relational databases. In Proc. of SIGMOD-96 Workshop on Research Issues on Data Mining and Knowledge Discovery (1996)
10. Wang, H., Zaniolo, C.: User defined aggregates for logical data languages. In: Proc. of DDLP. (1998) 85–97
11. Perng, C.S., Wang, H., Ma, S., Hellerstein, J.L.: Discovery in multi-attribute data with user-defined constraints. ACM SIGKDD Explorations **4** (2002) 56–64
12. Imielinski, T., Mannila, H.: A database perspective on knowledge discovery. Communications of the ACM **39** (1996) 58–64
13. Fang, M., Shivakumar, N., Garcia-Molina, H., Motwani, R., Ullman, J.: Computing iceberg queries efficiently. In: Proceeding of VLDB '98. (1998)
14. Sarawagi, S.: User-adaptive exploration of multidimensional data. In: Proc. of the 26th Int'l Conference on Very Large Databases (VLDB), Cairo, Egypt (2000) 307–316

15. Jeudy, B., Boulicaut, J.F.: Optimization of association rule mining queries. Intelligent Data Analysis **6** (2002) 341–357
16. Tuzhilin, A., Liu, B.: Querying multiple sets of discovered rules. In: KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. (2002)
17. Baralis, E., Psaila, G.: Incremental refinement of mining queries. In: Proc. of First International Conference on Data Warehousing and Knowledge Discovery. Volume 1676 of Lecture Notes in Computer Science., Springer (1999) 173–182
18. Cheung, D.W., Han, J., Ng, V.T., Wong, C.Y.: Maintenance of discovered association rules in large databases: an incremental updating technique. In: ICDE96 12th International Conference on Data Engineering, New Orleans, Louisiana, USA (1996)
19. Lee, S.D., Cheung, D., Kao, B.: A general incremental technique for maintaining discovered association rules. In: Proceedings of the 5th International Conference On Database Systems For Advanced Applications, Melbourne, Australia (1997) 185–194
20. Thomas, S., Bodagala, S., Alsabti, K., Ranka, S.: An efficient algorithm for the incremental updation of association rules in large databases. In: KDD. (1997) 263–266
21. Labio, W., Yang, J., Cui, Y., Garcia-Molina, H., Widom, J.: Performance issues in incremental warehouse maintenance. In: Proceedings of Twenty-Sixth International Conference on Very Large Data Bases. (2000) 461–472
22. Meo, R., Botta, M., Esposito, R.: Query rewriting in itemset mining. In: Proceedings of the 6th International Conference On Flexible Query Answeringd Systems. LNAI 3055, Springer (2004)
23. Leung, C.K.S., Lakshmanan, L.V.S., Ng, R.T.: Exploiting succinct constraints using fp-trees. ACM SIGKDD Explorations **4** (2002) 40–49
24. Lu, H., Feng, L., Han, J.: Beyond intratransaction association analysis: mining multidimensional intertransaction association rules. ACM Trans. Inf. Syst. **18** (2000) 423–454
25. Feng, L., Dillon, T.S., Liu, J.: Inter-transactional association rules for multi-dimensional contexts for prediction and their application to studying meteorological data. Data Knowledge Engineering **37** (2001) 85–115
26. Grahne, G., Lakshmanan, L.V.S., Wang, X., Xie, M.H.: On dual mining: From patterns to circumstances, and back. In: Proceedings of the 17th International Conference on Data Engineering. (2001)
27. Bucila, C., Gehrke, J., Kifer, D., White, W.M.: Dualminer: a dual-pruning algorithm for itemsets with constraints. In: Proceedings of 2002 ACM KDD. (2002) 42–51
28. Bayardo, R., Agrawal, R., Gunopulos, D.: Constraint-based rule mining in large, dense databases. In: Proceedings of the 15th Int'l Conf. on Data Engineering, Sydney, Australia (1999)
29. Lakshmanan, L.V.S., Ng, R., Han, J., Pang, A.: Optimization of constrained frequent set queries with 2-variable constraints. In: Proceedings of 1999 ACM SIGMOD Int. Conf. Management of Data. (1999) 157–168
30. Raedt, L.D.: A perspective on inductive databases. ACM SIGKDD Explorations **4** (2002) 69–77
31. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Proceedings of the 20th VLDB Conference, Santiago, Chile (1994)
32. Savasere, A., Omiecinski, E., Navathe, S.: An efficient algorithm for mining association rules in large databases. In: Proceedings of the 21st VLDB Conference, Zurich, Switzerland (1995)

Employing Inductive Databases in Concrete Applications

Rosa Meo[1], Pier Luca Lanzi[2], Maristella Matera[2],
Danilo Careggio[1], and Roberto Esposito[1]

[1] Università di Torino, Dipartimento di Informatica,
corso Svizzera 185, I-10149, Torino, Italy
{meo, careggio, esposito}@di.unito.it
[2] Politecnico di Milano, Dipartimento di Elettronica e Informazione,
Piazza Leonardo da Vinci, 32, I-20133, Milano, Italy
{lanzi, matera}@elet.polimi.it

Abstract. In this paper we present the application of the inductive database approach to two practical analytical case studies: Web usage mining in Web logs and financial data. As far as concerns the Web domain, we have considered the enriched XML Web logs, that we call *conceptual logs*, produced by specific Web applications. These ones have been built by using a conceptual model, namely WebML, and its accompanying CASE tool, WebRatio. The Web conceptual logs integrate the usual information about user requests with meta-data concerning the Web site structure. As far as concerns the analysis of financial data, we have considered the trade stock exchange index *Dow Jones* and studied its component stocks from 1997 to 2002 using the so-called *technical analysis*. Technical analysis consists in the identification of the relevant (graphical) patterns that occur in the plot of evolution of a stock quote as time proceeds, often adopting different time granularities. On the plots the correlations between distinctive variables of the stocks quote are pointed out, such as the quote trend, the percentage variation and the volume of the stocks exchanged. In particular we adopted *candle-sticks*, a figurative pattern representing in a condensed diagram the evolution of the stock quotes in a daily stock exchange. In technical analysis, candle-sticks have been frequently used by practitioners to predict the trend of the stocks quotes in the market.

We then apply a data mining language, namely `MINE RULE`, to these data in order to identify different types of patterns. As far as Web data is concerned, recurrent navigation paths, page contents most frequently visited, and anomalies such as intrusion attempts or a harmful usage of the resources are among the most important patterns. As far as concerns the financial domain, we searched for the sets of stocks which frequently exhibited a positive daily exchange in the same days, so as to constitute a collection of quotes for the constitution of the customers' portfolio, or the candle-sticks frequently associated to certain stocks, or finally the most similar stocks, in the sense that they mostly presented in the same dates the same typology of candle-stick, that is the same behaviour in time.

J.-F. Boulicaut et al. (Eds.): Constraint-Based Mining, LNAI 3848, pp. 295–327, 2005.

The purpose of this paper is to show that the exploitation of the nuggets of information embedded in the data and of the specialised mining constructs provided by the query languages, enables the rapid customization of the mining procedures following to the users' need. Given our experience, we also claim that the use of queries in advanced languages, as opposed to ad-hoc heuristics, eases the specification and the discovery of a large spectrum of patterns.

1 Introduction

In this paper we present two case studies, the mining of the Web log of a University Department and the analysis of financial data, Dow Jones index of the market stock exchange in a long period of time. We conducted the analysis by means of the exploitation of a query language for association rule mining – `MINE RULE` – and used it in a fashion typical to inductive databases. This choice has two purposes: (i) analyzing the concrete domain data and extracting interesting, usable and actionable patterns; (ii) evaluating the usability and expressive power of the mining query language by itself and inside the context of an inductive database approach.

For the analysis of Web domain, we have adopted *WebML conceptual logs* [11], that are Web logs enriched with novel information with respect to standard Web logs. Indeed, they have been obtained by integration of (ECFL) Web server logs with information on the Web design application and information on the Web pages content.

For the analysis of the financial domain, we adopted technical analysis, a typology of analysis that relies upon the study of the quotes plots in their temporal evolution. In particular, we adopted a typology of pattern named *candle-stick*. We searched in the data many configurations of these patterns, and some of them have been considered for a long time by the practitioners of the field a predictive tool for the determinant changes in the market stock quotes.

Inductive databases were proposed in [14] to leverage decision support systems by means of the application of specific mining languages to the analysis of data stored in databases. Thus, the user/analyst can interact with the system during the knowledge discovery process in a similar way as with a DBMS, that is, by specifying interesting data mining patterns with queries that in turn retrieve and store in the database the discovered information.

The adoption of this case study had also purpose to verify and experiment the suitability of some knowledge discovery (KDD) scenarios for inductive databases. KDD scenarios have been produced as a set of characteristic queries solving some frequently asked questions (mining problems) by users/analysts in order to recover from frequently occurring problems. We devolped some of these KDD scenarios inside the *cInQ* project (http://www.cinq-project.org), an european funded project and a consortium of universities and industries launched in order to evaluate the feasibility of inductive databases and of mining languages to extract interesting and actionable patterns from real data-sets.

The case studies are nowadays domains of extreme importance.

For the first case study, the analysis of Web logs, we identified three main typologies of user needs for which the association rules could constitute an aid: (i) the identification of frequent crawling paths by the users (Web structure and Web usage analysis), (ii) the identification of user communities and (iii) the identification of critical situations (anomalies, security attacks, high traffic) in which the information system could be placed.

The first topic allows the customization and construction of adaptive Web sites and recommender systems. The second one is important for customer relationship management and business applications (e.g, e-commerce). The third one is essential to the management of computer security for a reliable, efficient and available information system and its Web front-end and also to the credibility of Internet iteself.

To solve the first need the association rules that are searched for are those ones that identify sets of pages most frequently accessed by a consistent number of users. Furthermore, these sets of pages could have related information content. In these cases the discovered crawling paths could identify the most valuable information content and therefore could suggest a potential restructuring of the design of the Web site (e.g., by making easier the search for those pages by the creation of ad-hoc indexes or by fast access paths). This aspect is strictly correlated to the construction of a dynamical and adaptive Web site that is able to learn by previously submitted requests, changes accordingly to the user in order to fit best to its probable, future requests. Furthermore, the acquired knowledge allows the enhancement of the Web site with the potentialities of a recommender system, that suggests the user the preferable paths and interesting contents he/she could search for, on the basis of the frequently observed user visits and of the user profiles.

The second topic is very important for customer relationship management and many business applications, e.g., e-commerce applications and targeted marketing.

The third topic is also very important and the application of data mining in this field could give a consistent aid in information system administration as well as in computer security management. Indeed, nowadays we continuously observe the verification of dangerous situations in which our information systems are placed under attacks or are blocked by a highly congested traffic. Therefore, the searched association rules will try to give a descriptive profile of situations that frequently end in generation of errors by the Web server (situations that mostly correspond to hackers' attacks) such as repeated sequences of operations or services or the usage of a certain browsers or operating systems. Another kind of useful association rules tries to provide a profile of critical users, either because they request frequently a large volume of data or because they are often associated to certain typologies of errors returned by the system.

We had in addition the motivations to augment the quality of a Web application site, which is nowadays very important for a company (especially for e-commerce applications) since it reflects the immage of the company and might constitute its success or failure.

The second case study, the stock trading, had also very concrete motivations. Of course, the analysis of financial data and a better understanding of the stocks market behavior and evolution are of extreme importance in stock trading: it might allow investors to regain faith in the market, reducing the investors' risks, their losses and increasing the gains. An efficient data mining activity on this type of data is very important, since nowadays trading mainly occurs on-line and real-time, and is a very demanding process since it requires the real time operativity from every internet access point on huge volumes of stream data. Furthermore, the capability to make in advance the right prediction has enormous importance since the negotiation activity of stocks is a very rapid process, and involves huge flows of money: in this context, the right operation at the right time can make the investors gain or save huge volumes of money.

In this context, the main user needs are coarsely the following.

(i) Detect the best performing stocks in order to compose the portfolio. Possibly this consists also in the identification of the most similar stocks, once again, in order to acquire better knowledge on the stocks quotes behavior.

(ii) Detect the patterns configurations, in the stock quotes, that most frequently were associated to particular situations in the market, such as a high instability, and allowed to predict a consistent variation in stocks quotes.

(iii) Verification of some principles in technical analysis, principles that usually have acquired the consent of practitioners, but not always have gained the approval of statisticians and experts in economics. One of some principles, originally proposed by Dow Jones, but also confirmed by the most rigorous economists is the fact that a consistent increase in the volumes exchanged of a stock corresponds also to a notable variation in the prices of the same stock.

The paper is organized as follows. Section 2 provides an overview of the first application presented: the analysis of Web logs. Section 3 describes the `MINE RULE` operator briefly and the Web Log case study. Section 4 describes the second application: the analysis of financial data by the adoption of concepts of technical analysis (candle-sticks). In these latter Sections we provided also the KDD scenarios made of queries that allowed us to obtain useful results, and for each of them we provided a detailed description. Sections 5 and 6 respectively provide an evaluation of obtained results and some guidelines of using inductive databases in the analyzed application domains. Finally Section 7 draws the conclusions.

2 Application 1: Web Log Analysis

Current Web applications are very complex and highly sophisticated software products, whose quality, as perceived by users, can heavily determine their success or failure. A number of methods have been proposed for evaluating the quality of Web applications. In particular, Web usage mining methods are employed to analyze how users exploit the information provided by the Web site. For instance, showing those navigation patterns which correspond to high Web usage, or those which correspond to early leaving [17,27].

Web usage mining is mainly performed on the server side, and therefore is based on information found in log files, containing data about single user page access requests, represented in one of the available standard formats [25,4,25]. Accordingly, Web Usage Mining approaches rely heavily on the preprocessing of log data as a way to obtain high level information regarding user navigation patterns and ground such information into the actual data underlying the Web application [26,9,5]. Preprocessing generally includes four main steps:

- *Data cleaning*, for removing information that is useless for mining purposes, e.g.: requests for graphical contents, such as banners or logos; navigations performed by robots and webspiders.
- *Identification of user sessions*, for extracting full users navigation paths from the poor information available in Web logs. This step can be very demanding [26], especially due to the adoption of proxy servers by applications, which do not allow the unique identification of users from Web logs.
- *Content and structure information retrieval*, for mapping users page requests into the actual information of visited pages.
- *Data formatting*, for preparing data obtained through the previous three steps for being processed by the actual mining algorithms.

Notwithstanding the preprocessing effort, in most of the cases the information extracted is usually insufficient and with much loss of the available information at the Web design level, such as the complete knowledge about the structure and content organization of a Web application [23,6].

The approach we present in this paper is different with respect to other methods so far proposed, in that Web Usage Mining is directly integrated into the Web application development process. We use a conceptual model for Web application design (*WebML*) and its supporting case tool (*WebRatio*) for the development of Web applications that are able to produce rich Web logs (*conceptual logs*). Conceptual logs provide the mining phase with some of the necessary information with no loss and no additional cost: the content and hypertext structure of the application (originally determined by the application design) which can be now easily tailored on specific mining techniques. Furthermore, this specification is accompanied by the parameters that allow the instantiation of dynamic pages, the identifier of the unit of information in pages, the structure of the Web site recorded as a further specification of the typology of unit (e.g., content unit or index unit) and last but not least, the identifier of the user crawling session which allows to determine the main relevant activities performed on the application Web site by the users.

In the rest of this section, we shortly illustrate the main features of the adopted model, WebML (Web Modeling Language) [2,3], and of the rich logs that WebML-based applications produce.

2.1 WebML and WebRatio

WebML (Web Modeling Language) is a visual language for specifying the content structure of a Web application, as well as the organization and presentation of

such a content in a hypertext [2,3]. It mainly consists of the Data Model and the Hypertext Model.

The *WebML Data Model* adopts the Entity-Relationship primitives for representing the organization of the application data.

The *WebML Hypertext Model* allows the description of how data, specified in the data model, are published through elementary units, called *content units*, whose composition makes up *pages*; it also specifies how content units and pages are interconnected to constitute *site views*, i.e., the application front-end. The WebML Hypertext Model includes:

The WebML Hypertext Model includes:

- The *composition model*, concerning the definition of pages and their internal organization in terms of elementary pieces of publishable content, called *content units*. Content units offer alternative ways of arranging data dynamically extracted from entities and relationships of the data schema.
- The *navigation model*, describing links between pages and content units, which have to be provided to facilitate information location and browsing.
- The *content management model*, which consists of a set of units for specifying operations for creating and updating content.

Besides the visual representation, WebML primitives are also provided with an XML-based representation, suitable to specify those properties that would not be conveniently expressed by a graphical notation.

WebRatio is the CASE tool supporting the WebML-based development (`http://www.webratio.com`). The core of WebRatio is a code generator, based on XML and XSL technologies, which is able to process WebML conceptual schemas, by translating their visual specification into concrete page templates, and generate automatically the application code. The resulting Web applications feature a three-tier architecture, in which all the relevant dimensions of a dynamic application are covered: data extraction from the data sources, code for managing the business logic, and page templates for the automatic generation of the front-end.

3 Mining Conceptual Logs

Conceptual logs are standard log files integrated with information available from the Application Server logs, from WebML application runtime, and of conceptual schema of the underlying Web application. In this Section we describe the typology of information contained in these Web logs. We also present the KDD scenarios for this specific application domain, i.e., the sequences of queries in a constraint-based mining language (`MINE RULE`) which allowed us to obtained useful, interesting and actionable patterns for Web administrators, Web application designers and application analysts.

3.1 DEI Web Application Conceptual Logs

The Web site of DEI (Department of Electronic and Information) collects one fourth of the overall clickstream directed to Politecnico di Milano, Italy. We

collected the Web logs of the first consecutive 3 months in 2003. The original Web log stored by the Web server (`Apache http server`), was 60 MBytes large and is constituted by a relation that has the following information:

RequestID: the identifier of the request made by the user of a Web page;
IPcaller: IP address from which the request is originated; very often it is a proxy IP, that masks the real identification of the user.
***Date*:** date of the request,
***TS*:** time stamp of the request,
***Operation*:** the kind of operation request (for instance, `get` or `put`)
***Page URL*:** URL of the page to be transfered as a consequence of the request,
***Protocol*:** transfer protocol used (such as `TCP/IP`),
***Return Code*:** code returned by the Web server to the user,
***Dimension*:** dimension in bytes of the page,
***Browser*:** name of the browser from which the request is originated,
***OS Type*:** type of the Operating System.

To this main, standard information collected by Web server, WebML and WebRatio design applications add other information.

***Jsession*:** identifier of the user crawling session that spams over the single page requests. User crawling sessions are identified by an enabled Java browser by the Java thread identifier of a Web crawling.
***Page*:** identifier of the page generated by the application server. Very often a page is generated dynamically but this identifier is always the same for each page.
***UnitID*:** identifier of an atomic piece of information contained in a page. This identifier gives information on the type of content of a page.
***OID*:** identifier of an object (for instance, a professor, a course, a pubblication) whose content is shown in a page. This object identifier is used by the application server to instantiate in different ways dynamic pages according to the object itself. For instance, all professor pages obey to the same template that shows personal data, photo, description of the curriculum vitae of the person and of its research area. Instead, the real information that is shown for each person changes accordingly to the professor, and therefore to the `OID` parameter that identifies the person.
***Order*:** ordering number in which content units are presented in the page.

The Web Log contained almost 353 thousands user sessions, constituted each by an average of 12 page requests, for a total of more than 4.2 millions of page requests. The total number of pages (dynamic, instantiated by means of OIDs) was 38554.

3.2 MINE RULE

`MINE RULE` is an SQL-like operator for mining association rules in relational databases. A detailed description can be found in [20]. This operator extracts a

set of association rules from the database and stores them back in the database in a separate relation.

Let us explain the operator with the aid of a simple example on `WebLogTable`, containing the information of the conceptual log described in Section 3.1. The following `MINE RULE` statement extracts rules that aim to provide a description of the situations that generate frequently an error in the Web server (a favorite situation for attacks). $WebLogTable$ has been grouped by $RequestId$; requested rules associate values of $\langle Operation, Browser, PageURL \rangle$ with values of *Return Code*. Selected rules will have a value of returned code corresponding to an error (`WHERE` clause). Rules will have a support and a confidence greater than the minimum requested values (respectively 0.2 and 0.4).

```
MINE RULE SituationsRuturnCodes AS
SELECT DISTINCT 1..n Operation, Browser, Page Url AS BODY,
                1..n Return Code AS HEAD, SUPPORT, CONFIDENCE
    WHERE HEAD.Return Code LIKE '%error%'
FROM WebLogTable
GROUP BY RequestId
EXTRACTING RULES WITH SUPPORT:0.2, CONFIDENCE:0.4
```

This statement extracts each rule as an association of attribute values occurring within single tuples. In other statement cases, rule elements are constituted by values of the same attribute (e.g., `Page URL`) occurring in different tuples (e.g., requests) of the same group (e.g., date).

The main features of `MINE RULE` are:

- *Selection of the relevant set of data* for a data mining process. This feature is applied at different granularity levels (row level or at the group level, with the *group condition*).
- Definition of the *structure of the rules* (single or multi-dimensional association rules) and cardinality of the rule body and head.
- Definition of *constraints applied at different granularity levels*. Constraints belong to different categories: constraints applied at the rule level (*mining conditions* instantiated by a `WHERE` clause), constraints applied at a group level (instantiated by a `HAVING` predicate) and constraints applied at the *cluster* level (*cluster conditions*). For lack of space we will not make use of cluster condition in this paper.
- Definition of *rule evaluation measures*. Practically, the language allows to define support and confidence thresholds.[1] Support of a rule is computed on the total number of groups in which it occurs and satisfies the given constraints. Confidence is analogously computed (ratio between the rule support and the support of the body satisfying the given constraints).

3.3 Analysis of Web Logs with MINE RULE

We have imported into a relational DBMS (`mysql`) conceptual logs obtaining a table named `WebLogTable`. In this Section we describe in detail the KDD

[1] Theoretically, also other measures, based on body and head support, could be used.

scenarios, composed of a sequence of pre-processing, mining and post-processing queries that we have designed for discovery of useful patterns in the Web logs. These queries can be conceived as a sort of *template* that can be used to gather descriptive patterns from Web logs, useful to solve some frequent, specific or critical situations.

Analysis of Users that Visit the Same Pages. This analysis aims at discovering Web communities of users on the basis of the pages that they frequently visited.

Pre-processing Query. The mining request could be preceded by a pre-processing query selecting only those page requests that occurred frequently (above a certain threshold) thus allowing to neglet the rare page requests.

Mining Query. This query finds the associations between sets of users (IP addresses) that have all visited a certain number of same pages. In particular this number of pages is given in terms of support of the rules. (In this example, support is computed over the requested pages, since grouping is made according to the requested pages). It is an open issue whether the discovered regularities among IP addresses occur because these IP addresses have been commonly used by the same users in their pages crawling. Indeed, this phenomenon could put in evidence the existence of different IP addresses dynamically assigned to the same users.

```
MINE RULE UsersSamePages AS
SELECT DISTINCT 1..n IPcaller AS BODY, 1..n IPcaller AS HEAD,
                                              SUPPORT, CONFIDENCE
FROM WebLogTable
GROUP BY Page Url
EXTRACTING RULES WITH SUPPORT:0.001, CONFIDENCE:0.4
```

In order to instantiate in a meaningful way the preceding query, we had to perform some exploratory analysis of the source table, which is necessary to derive some meaningful values for the support threshold. Indeed, since the number of groups in which the source table is partitioned is decided by the grouping clause in the specific `MINE RULE` query, we had to launch a standard SQL query to derive the total number of page Urls contained in the `WebLogTable`. This number was 38554. Therefore, the minimum support threshold must be higher than 1/38554, otherwise every possible sets of user's IP addresses that accidentally requested one single common page would be recovered by the system. With the value of 0.001 for the minimum support we extracted 421 rules which decreases to 151 with a value of minimum confidence equal to 0.4. As a further work it would be interesting to extract some condensed representation of the set of frequent rules, such as a set of non redundant rules as proposed in [29,22].

Examples of some obtained results. In practice, in our experiments we discovered that the most frequently co-occurring IP addresses belong to Web crawlers engines or big entities, such as universities.

A similar query would occur if we wish to discover user communities which share the same user profile in terms of usage of the network resources. In this case, we would add constraints (in the mining condition, for instance) on the volume of the data transferred as a consequence of a user request. In this case examples of discovered patterns are the requests of frequent download of materials for courses, or documentation provided in the users' home pages.

Post-processing Query. As a post-processing query instead we could be interested in finding those pages that have been all visited most frequently by certain sets of users. This is a query that crosses-over extracted patterns and original data. With this request we could also discard from the discovered patterns those ones belonging to Web crawlers engines.

These post-processing queries are possible because `MINE RULE` system stores the discovered rules in the database. The main table, contains the rules, in terms of identifiers of body and head itemsets and of the rules statistical measures (which, in the current implementation are simply support and confidence). The secondary table, contains the details of the elements of rule body and head itemsets (in terms of the body and head schemas specified in the `SELECT` clause of the query).

The following two query scenarios aim at performing Web structure mining.

Most Frequent Crawling Paths

Pre-processing Query. As in previous case, the mining request can be preceded by a pre-processing selecting only those page requests that occurred frequently.

Mining Query. This query returns sequences of pages (ordered by date of visit) frequently visited.

```
MINE RULE FreqSeqPages AS
SELECT DISTINCT 1..2 Page Url AS BODY, 1..1 Page Url AS HEAD,
                                    SUPPORT, CONFIDENCE
WHERE BODY.Date = HEAD.Date and BODY.TS < HEAD.TS
FROM WebLogTable
GROUP BY IPcaller
EXTRACTING RULES WITH SUPPORT:0.002, CONFIDENCE:0.4
```

You can notice that in this case we grouped by user (`IPcaller`) and searched for sets of pages frequently occurring in the visits of a sufficient number of users (support). Notice also that we used a mining condition to constrain the temporal ordering between pages in antecedent and consequent of rules, thus ensuring the discovery of sequential patterns. We limited the search to ordered sets pages requestes in the same day by the same user IP, temporally ordered (see condition on time stamps). We also counted the total number of distinct groups in order to evaluate a meaningful value for the support threshold and we obtained that it was equal to 406. Thus minimum threshold was setted higher than $1/406 = 0.002$. With this value we obtained 7415 rules which reduced to 1773 setting the confidence threshold.

Examples of some obtained results. In practice, examples of resulting interesting patterns showed that requests of a research center page, or research expertise area were later followed by the home page of a professor. We interpreted this as a hint to the fact that people preferred to reach the personal pages by mean of a secondary access point (the research center or reserach area index) instead of the more direct index to the personal home pages. This was perhaps a sign that the general index of the global institution on people home pages was too slow inducing requests to be preferentially directed to other more little and more efficient indices.

Post-processing Query. A post-processing query can discover the sets of IP addresses originating the frequently occurring page requests. Again, this query crosses over patterns and data. We discovered by this query some publicly available IPs, of some well-known internet providers in Italy.

Units that Occur Frequently Inside Users Crawling Sessions. One of the main advantages gained by the conceptual web logs is the knowledge of the information content of the pages. These content units can give us a more precise information on the reasons why certain pages are frequently requested by the users. The following query extracts associations between two sets of content units that appeared together in at least a certain number of crawling sessions.

```
MINE RULE UnitsSessions AS
SELECT DISTINCT 1..n UnitID AS BODY, 1..n UnitID AS HEAD,
                                        SUPPORT, CONFIDENCE
FROM WebLogTable
GROUP BY Jsession
EXTRACTING RULES WITH SUPPORT:0.05, CONFIDENCE:0.4
```

Examples of some obtained results. With this query we discovered that the units that most frequently co-occurred in visits are the structural components of the Web site, such as indexes, overview page of the personnel, map pages of the site, and so on. The results of this query could be used by the Web application designers to restructure the site in order to make easier the search for those units that frequently co-occur in user visits.

Anomalies Detection. This query tries to determine the associations between pages and users that caused a bad authentication error when making access to those pages. Therefore, this query wants to determine those pages that could be effectively used by callers as a way to enter illegally into the information system.

Pre-processing Query. The mining request was preceded by a pre-processing query selecting only those page requests that occurred a sufficient number of times. This discards those requests that have been mistakenly submitted by the user (a wrongly typed password), that if not repeated many times, cannot be considered an intrusion attempt.

```
MINE RULE BadAuthentication AS
SELECT DISTINCT 1..1 IPcaller AS BODY, 1..n Page Url AS HEAD,
                                        SUPPORT, CONFIDENCE
WHERE BODY.IPcaller = HEAD.IPcaller
FROM WebLogTable WHERE Return Code='bad authentication'
GROUP BY Date
EXTRACTING RULES WITH SUPPORT:0.03, CONFIDENCE:0.4
```

You can notice that `WHERE Return Code='bad authentication'` is effectively a pre-processing operation that selects only the portion of interest of Web logs. In this query we grouped source data by date, thus identifying patterns (association of users to page requests) that are frequent in time. Notice that mining condition `WHERE BODY.IPcaller = HEAD.IPcaller` ensures that page requests (head) effectively were originated by the callers associated to them (body).

The total number of obtained rules was 80 which decreased to 72 with the confidence threshold.

Examples of some obtained results. Some examples of retrieved patterns are provided by those attempts of change of passwords, or downloading of some reserved information.

High Traffic Users

Pre-processing query. Similarily to previous queries, also this data mining query could be preceded by a pre-processing step, selecting only the frequent page requests. Indeed, rare page requests can be neglected.

Mining query. This query returns the associations between two sets of user IP addresses from which a request of pages is characterized by a large volume of data. This constraint is enforced by means of a preprocessing predicate `WHERE dimension>=1024` that selects only those requests generating high volume of traffic on the network.

```
MINE RULE HighTrafficUsers AS
SELECT DISTINCT 1..n IPcaller AS BODY, 1..n IPcaller AS HEAD,
                                          SUPPORT, CONFIDENCE
FROM WebLogTable
     WHERE dimension>=1024
GROUP BY date
EXTRACTING RULES WITH SUPPORT:0.03, CONFIDENCE:0.4
```

Notice that we grouped the input relation by date thus identifying the users that request pages characterized by a high volume frequently in time.

Post-processing query. A cross-over query can discover those pages originating the frequently occurring page requests.

Examples of some obtained results. As examples of discovered patterns there are the requests of frequent download of materials for courses, or documentation provided in user home pages.

Errors Correlated to Usage of an Operating System. This query returns associations between the operating system and the error code frequently returned by the Web server.

```
MINE RULE OSErrors AS
SELECT DISTINCT 1..1 OStype AS BODY, 1..n Return Code AS HEAD,
                                              SUPPORT, CONFIDENCE
WHERE BODY.OStype=HEAD.OStype
FROM WebLogTable  WHERE Return Code LIKE '%error%'
GROUP BY Date
EXTRACTING RULES WITH SUPPORT:0.01, CONFIDENCE:0.4
```

Notice the pre-processing predicate (`WHERE Return Code ..`) that selects only the page requests that result in some errors. The total number of retrieved rules was 296. This query is similar to query named `BadAuthentication` for the discovery of anomalies and can be useful to test the reliability and robustness of a new Web application.

Users that Visit Frequently Certain Pages. This request aims at discovering if recurrent requests of a set of pages from a certain IP exist. This puts in evidence the *fidelity* of the users to the service provided by the Web site.

```
MINE RULE UsersPages AS
SELECT DISTINCT 1..1 ipaddress AS BODY, 1..2 Page Url AS HEAD,
                                          SUPPORT, CONFIDENCE
WHERE BODY.ipaddress = HEAD.ipaddress
FROM WebLogTable
GROUP BY RequestId
EXTRACTING RULES WITH SUPPORT:0.001, CONFIDENCE:0.4
```

As in previous experiments we setted the minimum support threshold after a prior inspection of the total number of received requests (which was equal to 4.2 millions). However, the mining condition reduced the number of effective groups from which a valid association rule was present. Also, the support threshold helped to reduce the volume of the result which finally contained only 421 rules.

Examples of some obtained results. Examples of patterns we discovered are provided by the pages that allow the download of material, such as course slides and research papers published publicly in the personal home pages. Other similar queries, on content units (instead of pages) are also a useful suggestion and allow to acquire a lower granularity in discovering the user crawling paths. Patterns resulting from this request confirm the previous results (the most recurrent

requests are download of materials from the Web site). This observation gave to system administrators useful informations to manage the bandwidth of the network in more optimized ways.

3.4 Query Execution Times

Figure 1 reports for completeness the execution times of queries in the experiments on Web log. You can observe one bar representing the execution time of each component of the system in execution (parser, optimizer of the execution plan, pre-processing phase, data mining itemset and rules extraction phases). With this experiment we can also compare the relative impact on execution times of the various components. In another Chapter of this book we discussed in detail the algorithms and the data structures adopted by the `MINE RULE` system for executing some of the queries. In particular, that Chapter can be consulted to obtain more information on the particular strategy adopted to execute the queries when mining conditions are boolean expressions of terms in the form `[BODY|HEAD].<Attribute> <ComparisonOperator> <Attribute-Value>` and no clustering condition is present. That Chapter discusses mainly on the opportunity to develop a constraint incremental evaluation strategy exploiting previous queries results, stored in the database. It compares this incremental strategy

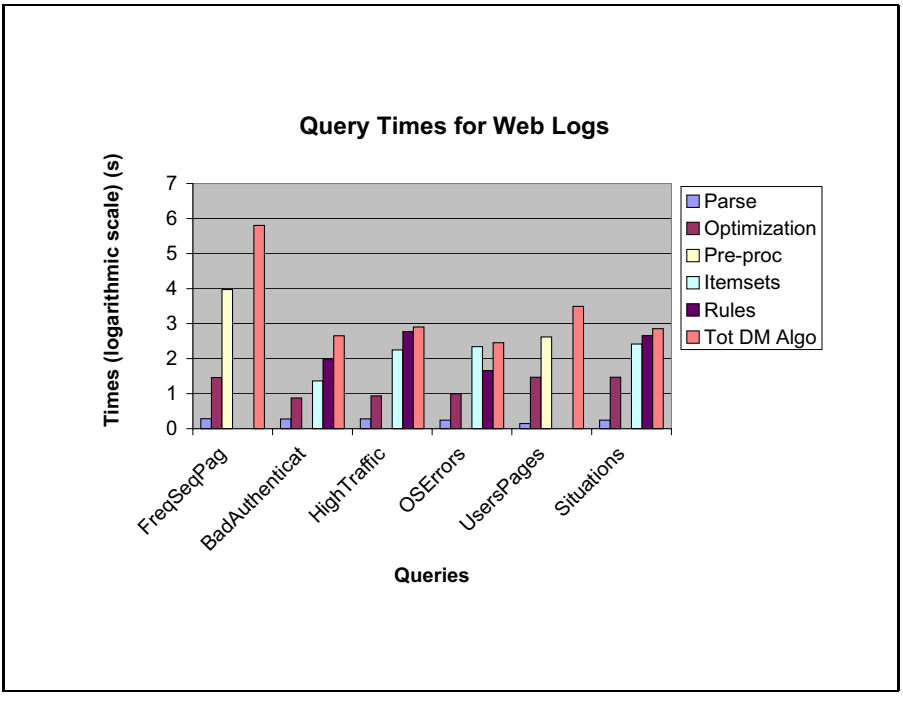


Fig. 1. Query Execution Times in Experiments on Web Logs

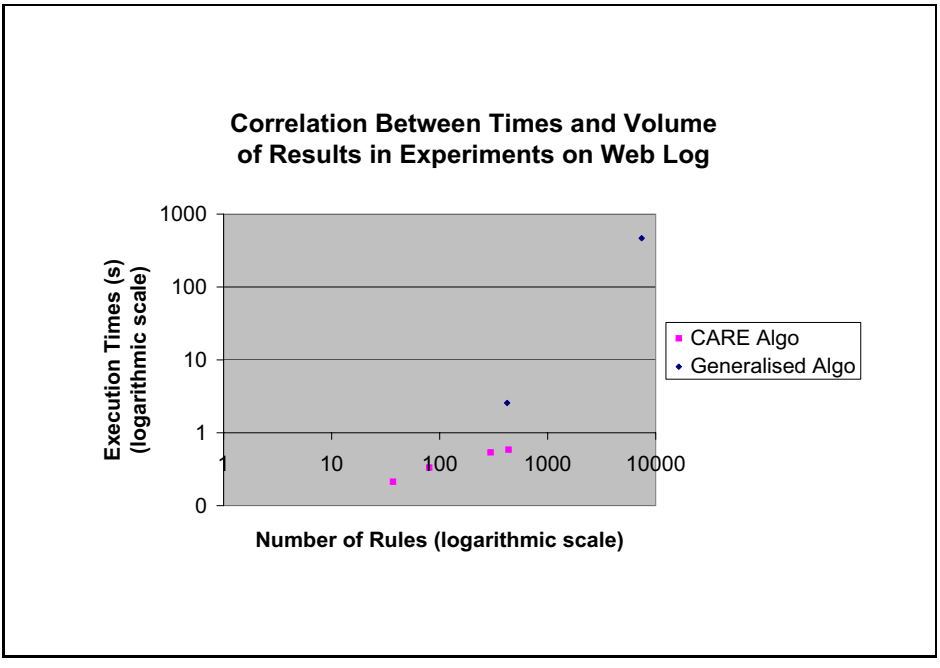


Fig. 2. Correlation Between Times and Volume of Results in Experiments on Web Log

with a strategy that works from scratch (the algorithm is called *CARE*). Still, in this Chapter we present only queries executed from scratch. Indeed, since the various scenarios have just the purpose to address typical domain-specific situations they are un-correlated (one with respect to the others). Therefore, they cannot be solved with the proposed incremental strategy.

[1] can be consulted to obtain details on the execution algorithm that addresses the general cases of a `MINE RULE` statement (called generalised algorithm). In fact, `MINE RULE` can be instantiated in very different queries, comprising different constraint typologies and even retrieving multi-dimensional rules. For instance, the generalised algorithm takes care when (i) body or head schemas are different or defined on a list of attributes; when (ii) "cross" mining conditions are present (i.e., conditions between `BODY` and `HEAD`, under the form `BODY.<Attribute> <ComparisonOperator> HEAD.<Attribute>`); (iii) conditions on clusters or aggregate functions are specified. Therefore, a rich set of different algorithms has been implemented in the system in order to better exploit the specificity of each query (in terms of constraints typologies and properties, regularities in the selected source data, such as functional dependencies between the attributes in the rules and in the mining condition).

Moreover, in Figure 2 it is possible to observe the diagram showing the correlation between query execution times and number of rules in the results (scales are logarithmic). You can immediately observe that they are approximately linearly correlated. However, the trend in execution times of the generalised algorithm

is much more severe, because of course it addresses a general and more complex problem at the expenses of efficiency.

4 Application 2: Financial Data Analysis

4.1 Dow Jones Stocks Exchange Index

Dow and Jones, two economists of the XX century, with a set of articles appeared in 1900-02 in Wall Street Journal, defined a set of few stocks whose value could have been used with the purposes of monitoring the evolution of the USA economy. Initially stocks were grouped in two sets: transportation companies and industrial companies (energy production, mineral extraction, and so on). Nowadays, the index named *Dow Jones 30* contains 30 stocks of companies still strictly connected to production activities in USA, such as Microsoft, Intel, AT&T, General Motors, Mc Donalds', etc... and is still used as a meter of judgement on the evolution and wealth of the american economy because it is grounded on some big companies whose core activity is the production of consumers' goods or services. However, it is very much dynamic and in a temporal interval of few years it can change a significant part of its constituting stocks.

In Figure 3 we report the daily percentage variation of Dow Jones index from 1896 till 2003.

4.2 Technical Analysis

As opposed to fundamental analysis, which is based on the study of the corporations' activity under a macro economic view [10], technical analysis is based on the a-posteriori study of the stocks quote trend. Technical analysis [24,13,15,8,24] has been founded at the beginning of the XX century by some economists such as Dow, Hamilton and Rhea.

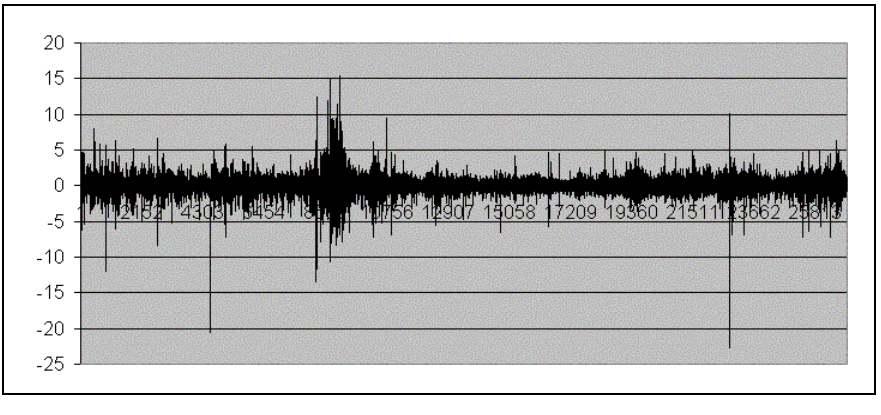


Fig. 3. Dow Jones index percentual daily change from 1896 to 2003

Some Foundation Hypothesis of Technical Analysis

- Quote is considered as the synthesis of the investors' confidence in the stock intrinsic value [18];
- investors (psycological) reactions to certain events are always the same and will repeat in the future [16];
- current trend in quotes will continue until some event makes it change. This is useful for the detection of patterns that determine the inversion of the quote trend.

Technical analysis is based on the feeling that the quote of a stock is based on the investors' faith in that stock value at the moment, and this determines that quotes have a cyclic evolution, able to reach a maximum value or a minimum value in a certain period of time (named as *resistance* and *support* respectively). The ability to determine these values is very important for the stock market analyzers because helps investors to determine the best time to sell or buy the stocks [19].

Another important event that analyzers try to determine is the point in time when the quotes of a stock change their trend (from positive to negative or viceversa). Indeed, for an investor who wishes to sell his/her stocks, the best time to sell them is the point in which the trend from positive becomes negative. At this point, the stock reached its maximum value and afterwards it will start to decrease its value. Therefore, if the investor sells in this point in time he/she will be able to make the best profit from the sold. Analogously for the purchase: the best point in time to buy a stock is when the trend from negative turns into positive. At this time the stock value has reached the minimum value and from this time on, it will start to increase making the customer spend more for the same stocks.

4.3 Japanese Candle-Sticks

Candle-sticks have been originally proposed by Japanese market analyzers to study the rice market. A single candle-stick represents the synthesis of the exchanged stocks occurred in one period of time (such as a date or a week) for a given stock. Graphically it is similar to box-plot used for exploratory and descriptive data analysis: it is constituted by a box located in a `time x quote` dimension plot, whose horizontal borders identify the open and close value of the stock in that time period. A candle-stick is colored as follows.

Black candle-stick: represents a time period in which the open value is higher than the close value. This identifies a period in which the stock lost part of its value.

White candle-stick: represents a time period in which the open value is lower than the close value. This identifies a period in which the stock gained part of its value.

Two vertical lines might depart from the box borders: the lower line represents the minimum value reached in the period while the higher the maximum. If some

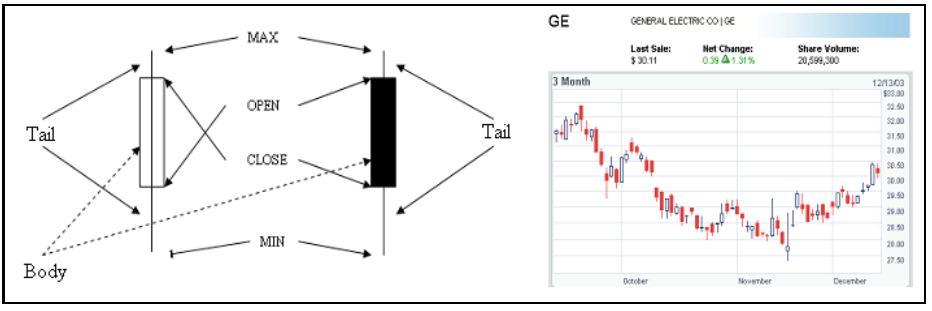


Fig. 4. Two candle-sticks and their usage in daily stock quotes plots

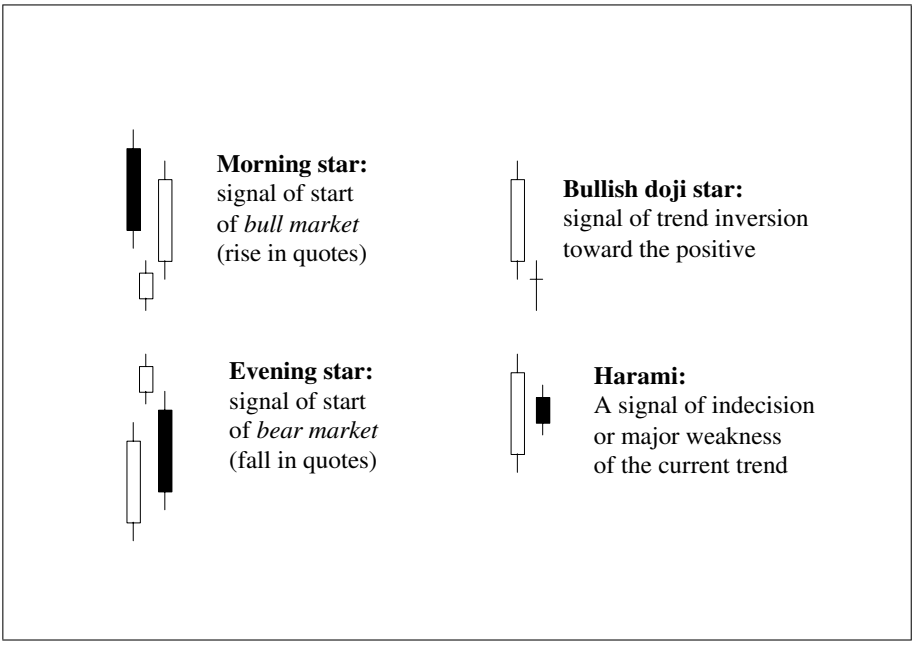


Fig. 5. Some representative candle-sticks and their predictive meaning

of these lines are absent it means that the minimum or the maximum value is reached with the open or close value (Figure 4 on the left).

The right part of Figure 4 shows an example of box-plots positioned in a `time x quote` graph. This offers an immediate and visual representation of the evolution of the value of a stock in the time, with candle-sticks providing the quality (positive or negative) of the stock exchange period.

For a long time, technical analysts have used candle-sticks to grasp with a visual representation the evolution of quotes in time. They have elaborated many configurations of the different candle-sticks in time, some of which are

reported in Figure 5 with a brief explanation of their meaning. Some technical analysts considered these patterns as an "alarm" able to predict in conjunction to other observational elements (such as the volume of exchanges) the immediate evolution of stocks quotes. This is based on the principle that the quote of a stock is determined by the sell/buy law by investors themselves. Indeed, their willingness to buy or sell is determined by the market reactions to certain events which tend to repeat in the future. Therefore, the quote of a stock represents the "summa" of the willingness of the investors to buy the stocks and on their confidence in the stock intrinsic value.

4.4 Analysis of Dow Jones 30 Stocks Exchange with MINE RULE

We downloaded data on the daily market exchanges of stocks in Dow Jones 30, from 1997 till 2002, transformed it in relational form and imported in a DBMS (mysql). Since the index is very dynamic and some stocks were changed during this period, some of the analysis were performed on a subset of this time period (such as one or two years, only). The data set we collected is 2.5 MB large, and contains the following information for every date:

date: the date of the trade exchange;
ticker: the symbol identifying the stock;
open: the opening value of the stock;
close: the closing value of the stock;
min: the minimum value of the stock;
max: the maximum value of the stock;
volume: the total number of exchanged stocks of a ticker in the date.

In the following we report the KDD scenarios we developed for this financial application domain.

Frequent Candle-Stick Sequences. The following statement returns the pairs of candle-sticks that have been found in two successive dates by a relevant number of different stocks.

```
MINE RULE frequent-candle-sticks-sequences AS
SELECT DISTINCT 1..1 candle-stick AS BODY,
                1..1 candle-stick AS HEAD, SUPPORT, CONFIDENCE
WHERE BODY.date=HEAD.date-1
FROM dj30quotes
GROUP BY ticker
EXTRACTING RULES WITH SUPPORT:0.30, CONFIDENCE:0.40
```

As you will be able to see in Section 4.5 in which the execution times of the queries are reported, this query had one of the worst performance (though the number of retrieved rules with these support and confidence thresholds is low: only 78). However, notice the mining condition is checking all the possible consecutive dates in the temporal interval of 5 years, which is a quite large domain.

Examples of some obtained results. We launched this statement on the stock quotes of different years, separately, and compared the results. We noticed for example that in years 1997, 1998 and 1999, the following candle-stick sequence, `a black candle-stick immediately followed by a white one`, has been found much more frequently than in later years, 2000, 2001 and 2002 (between 23% and 76% of all the stocks). In fact, in the first years investors obtained high profits (Dow Jones index roughly doubled its value) and fostered good faith in the market; on the contrary, in later years, great losses were experienced. We believe that this sequence can be interpreted as a signal of trading which is going to be soon saturated (intuitively, that everyone is willing to buy). However, in years 2000-2002, we found that sequences made by `three consecutive candle-sticks including only black candle-sticks and white candle-sticks with both the tails` (i.e., such as that the minimum and maximum laid outside of the body of the candle-stick) were surprisingly much more frequent than in previous years (their frequency was almost doubled). Our interpretation to this is that this type of candle-stick sequence might be a signal of a "nervous" market, which is a sign of indecision and might constitute a suggestion of refrain from a new investment.

Pairs of Stocks with Similar Behavior. This statement searches for the stocks with a similar behavior in time. The similarity of behavior is decided according to the common candle-sticks types the two stocks show in the same dates.

```
MINE RULE similar-tickers AS
SELECT DISTINCT 1..1 ticker AS BODY, 1..1 ticker AS HEAD,
                                          SUPPORT, CONFIDENCE
WHERE BODY.candle-stick = HEAD.candle-stick
FROM dj30quotes
GROUP BY date
EXTRACTING RULES WITH SUPPORT:0.30, CONFIDENCE:0.40
```

Examples of some obtained results. In Figure 6 we show the plot of quotes in 2002 of the pair of stocks in the result that are most similar: `Hewlett-Packard` and `Microsoft Corp`. You can notice actually how much they are similar. Another set of very similar tickers is composed by `Home Depot Inc.`, `Walt Disney-Disney C.`, `JP Morgan Chase Co.` and `American Express Inc`.

Verification of Price Percentage Variation by Volumes. The main aim of the following queries is the verification of one of the most well-known principles in stock trading [18]: increasing volumes in the exchanges of a stock is a signal of a broader participation among investors; that is, contextually to increasing volumes one could expect a great movement in prices.

Pre-processing query. We preceded the real mining query with a pre-processing query with the purpose to identify the high volumes. This query computes for

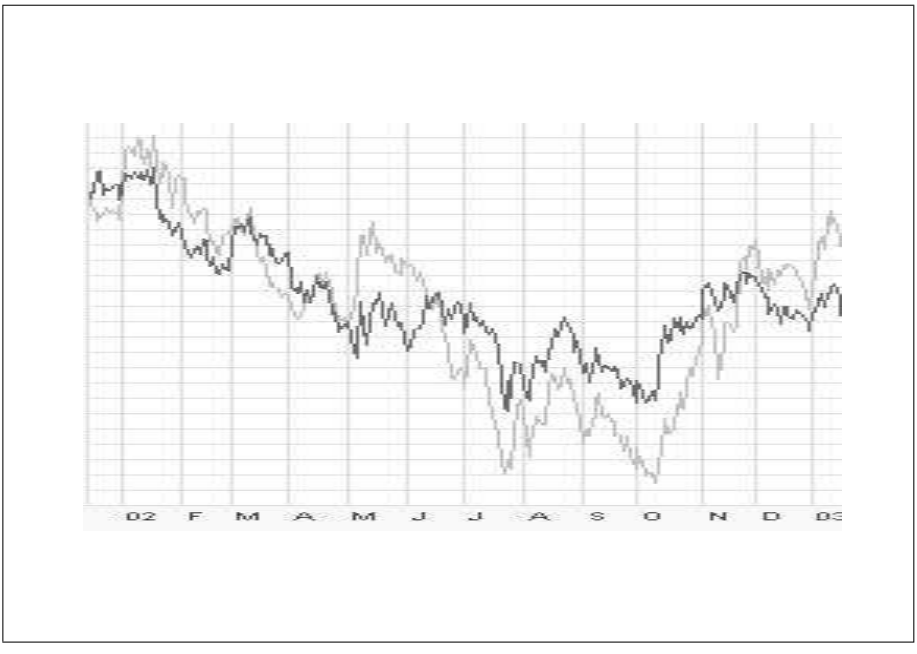


Fig. 6. Comparison between plot of Hewlett-Packard and Microsoft stock daily quotes in 2002

each stock and each date the ratio r between the exchanged volume (i.e. the number of exchanged stocks) in the day and the average volume exchanged in the whole year. In this way we could label each stock exchange date with a boolean attribute called `high_volume`.

We considered for the identification of high volume dates, for any single stock, different values of the ratio r. In this example, we experimented a value of $r = 300\%$ uniform for all the stocks and selected in source relation only the stock exchanges characterized by a high volume.

Furthermore, with the aid of another query, we found also the percentage change in quotes during the day computed with respect to the close price of the previous day, called `varp`, defined as follows:

$$varp = \frac{(close - open)}{open}$$

Analogously to what is done with high volumes, we are interested in high variations in prices, and labeled each date with `high_varp` if

$$varp >= 5\% \vee varp <= -5\%$$

Both `high_volume` and `high_varp` will be used in the following mining query.

Mining query. This statement searches for frequent associations (in time, i.e., occurred in a large number of weeks of the year) between a high volume and a high percentage variation of price of the same stock in the same date.

```
MINE RULE VarpByVols AS
SELECT DISTINCT 1..1 high_volume AS BODY, 1..1 high_varp AS HEAD,
                                                SUPPORT, CONFIDENCE
WHERE BODY.ticker=HEAD.ticker AND BODY.date=HEAD.date
FROM dj30quotes
GROUP BY week
EXTRACTING RULES WITH SUPPORT:0.001, CONFIDENCE:0.001
```

Examples of some obtained results. From the obtained results, we can say that 86% of the exchanges in which the price percentage variation (`varp`) is above 5% or below −5% occurs with a volume greater than 125% of the daily volume, averaged on the whole year. This confirms the initial hypothesis we wanted to test.

The above query needed a certain amount of preparation tests, since we had to discover the value of the ratio r and of the percentage variation *varp* that best confirm the evidence. In Section 5 we will discuss more on the KDD process that was necessary to prepare the source data with a discretization step (obtain the boolean derived attributes `high_volume` and `high_varp`).

Stocks with White Candle-Stick in the Same Date. The following search is motivated by the necessity of identifying a set of stocks suitable to constitute the investors' portfolio.

Pre-processing query. Similarly to previous query, we performed a pre-processing query selecting the stock exchanges occurred with high volumes. (We considered again the same values used previously).

Mining query. We launched the following mining query on the stocks in 1997, the first year in the observed time period, and searched for the pair of stocks that frequently had a white candle-stick in the same dates with a relevant percentage of price variation.

```
MINE RULE white-candle-stick-pairs AS
SELECT DISTINCT 1..1 ticker AS BODY, 1..1 ticker AS HEAD,
                                      SUPPORT, CONFIDENCE
WHERE BODY.candle-stick-type='white' AND
      HEAD.candle-stick-type='white' AND
      BODY.varp>5 AND HEAD.varp>5
FROM dj30quotes
GROUP BY date
EXTRACTING RULES WITH SUPPORT:0.01,CONFIDENCE:0.40
```

Examples of some obtained results. This query returns 870 rules. The ones that show the highest support and confidence are constituted by six stocks such as `UBS AG`, `General Electric`, `Honeywell`, `Intel`, `Merck & Co.` and `Procter & Gamble`. We can notice how among all the rules, many of them are redundant. What is really meaningful in this case is the condensed representation of itemsets that occur in the same situations (dates): the concepts, in Formal Concept Analysis [28,21,22]. Further work on the MINE RULE system, which actually does not provide support for a condensed representation of itemsets, is still needed to improve the representation of itemsets in a way that is more meaningful.

Post-processing. This step was used to evaluate the portfolio composed by previously selected stocks. This portfolio was monitored for the following 4 years and outperformed Dow Jones index from 5% to 11% in any year. Furthermore, it gained in each single year from the 5% to 29% of the total investment. This is a very useful result and is a first demonstration of the practical usefulness of these techniques.

Discovery of Frequent Doji Star Candle-Sticks in Time. The following statement searches for the pairs of successive dates in which most of the stocks show a *doji star candle-stick*. Specialized literature on technical analysis considers this pattern as a signal of reversal of trend followed by a signal of indecision of the market. It can be viewed as an alarm signal. Indeed, we discovered this pattern in spring and in autumn of 2002. An example of this pattern can be observed in Figure 5 under the name of *bullish doji star* that is very similar to the doji star pattern with the exception that the first candle-stick is white instead of black. In the following `MINE RULE`, you can observe the mining condition

$$HEAD.open + 0.003 * HEAD.open > HEAD.close \; AND$$
$$HEAD.open - 0.003 * HEAD.open < HEAD.close$$

which serves to search for the cross pattern with a tolerance between the open and the close value of a 0.3%. In fact, a perfect match would be very unprobable. Notice, how this tolerance in the comparison between open and close values plays a similar role to soft comparison in fuzzy query languages, since it allows us to test a predicate under some weaker conditions, necessary in the stock financial domain in which a certain amount of noise is always present.

```
MINE RULE freq-doji-star-candle-sticks-in-time AS
SELECT DISTINCT 1..1 date AS BODY, 1..1 date AS HEAD,
                                    SUPPORT, CONFIDENCE
WHERE BODY.date=HEAD.date-1 AND
      BODY.close>BODY.open AND BODY.close<HEAD.open AND
      (HEAD.open +0.003*HEAD.open > HEAD.close AND
       HEAD.open -0.003*HEAD.open < HEAD.close)
FROM dj30quotes
```

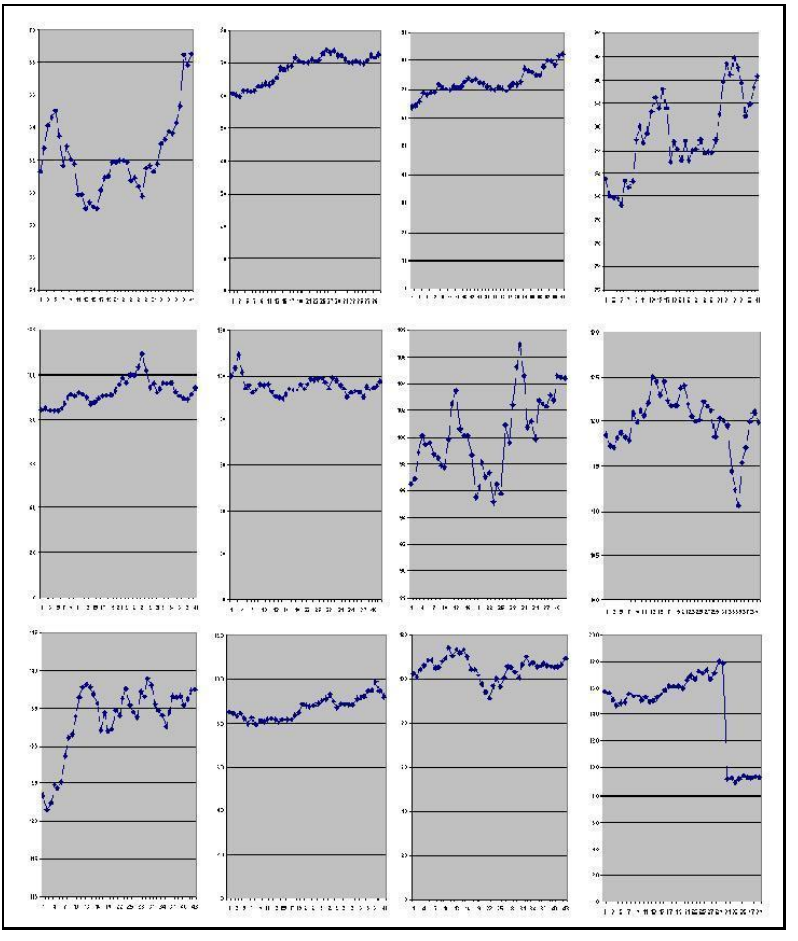


Fig. 7. Quotes plot of Microsoft stock in the time period (2 months) immediately following a bullish doji star occurrence

```
GROUP BY ticker
EXTRACTING RULES WITH SUPPORT:0.30, CONFIDENCE:0.40
```

86 total rules were discovered for all the tickers. In Figure 7 we report the stock trend immediately following the detection of a bullish doji star in `Microsoft` stock and indeed, you can notice how the trend is usually significantly positive, especially in the first part of the evolution. A bullish doji star is the opposite: it corresponds to a signal of indecision followed by a reversal in the market trend. We also checked for the presence of this pattern in a stock, like `Microsoft Corp.`, that is the stock characterized by the highest capitalization in the market.

Discovery of Morning Star Candle-Sticks. The following statements search for the dates in which most of the stocks show a *morning star* candle-stick

pattern. As you might recall, this pattern is composed by three candle-sticks, so that we need in this case two separate `MINE RULE` statements: the first one, called `1st-part-morning-star` searches for the first part of the pattern (first two candles); the second one, `2nd-part-morning-star` is needed for the second part (second and last candle). Finally, a simple `SQL` query joins the results looking for the complete pattern where the first part of the pattern is followed by the second part for the same stock and in an immediately successive date. Of course, the intermediate candle must be the same in the two parts of the pattern. Of course this methodology could be extended for patterns of arbitrary length and thus to the discovery of sequential patterns.

```
MINE RULE 1st-part-morning-star AS
SELECT DISTINCT 1..1 date AS BODY, 1..1 date AS HEAD,
                                    SUPPORT, CONFIDENCE
WHERE BODY.date=HEAD.date-1 AND BODY.close<BODY.open AND
      BODY.close>HEAD.close AND HEAD.close>HEAD.open
FROM dj30quotes
GROUP BY ticker
EXTRACTING RULES WITH SUPPORT:0.30, CONFIDENCE:0.40

MINE RULE 2nd-part-morning-star AS
SELECT DISTINCT 1..1 date AS BODY, 1..1 date AS HEAD,
                                    SUPPORT, CONFIDENCE
WHERE BODY.date=HEAD.date-1 AND BODY.close>BODY.open AND
      BODY.close<HEAD.open AND HEAD.close>HEAD.open
FROM dj30quotes
GROUP BY ticker
EXTRACTING RULES WITH SUPPORT:0.30, CONFIDENCE:0.40
```

90 and 131 occurrences, respectively of the first and second part of the morning-star pattern, were discovered in the five years.

Post-processing. A post-processing standard SQL query performs the join between the result of the `1st-part-morning-star` query and of the `2nd-part--morning-star` query taking care that the head of the first part coincides with the body of the second part. This guarantees that the two parts of a morning-star candle-stick pattern are effectively found in two consecutive days. In 2002, only 29 occurrences of the complete pattern were discovered. (We tested this query only on one year because the intermediate table for the 5 years was too large to perform the join in the database with a reasonable time).

One final observation concerns how some fuzzy conditions, similar to what done for query `freq-doji-star-candle-sticks-in-time` could be useful to gain a certain amount of flexibility in evaluating the time interval between the occurrence of the first and the second part of the candle-stick pattern.

```
SELECT D1 ticker, D1.date
FROM dj30quotes2002 D1, dj30quotes2002 D2, dj30quotes2002 D3,
```

```
     1st-part-morning-star F, 2nd-part-morning-star S
WHERE D1.ticker=D2.ticker AND D2.ticker=D3.ticker AND
      D1.date=F.body AND D2.date=F.head AND
      S.body=D2.date AND D3.date=S.head
```

Examples of some obtained results. The results confirm that these candle-stick patterns are quite rare. (In 2002, they were present mainly in August and December, a period in which the market raised again).

4.5 Query Execution Times

Figure 8 reports for completeness the execution times of queries in the experiments on Dow Jones 30. You can notice how the execution times of the join query has evaluation times comparable to the extraction of rules by `MINE RULE`.

Moreover, in Figure 9 it is possible to observe the diagram showing the correlation between query execution times and the number of rules in the results (scales are logarithmic). You can immediately observe that they are clustered around a central point with the exception of some outliers. If we go to observe with more attention of which queries they consists of, we can see that the best query (first outlier for the generalised algorithm, working with a cross-condition between body and head) is `similar_tickers` which has a simple

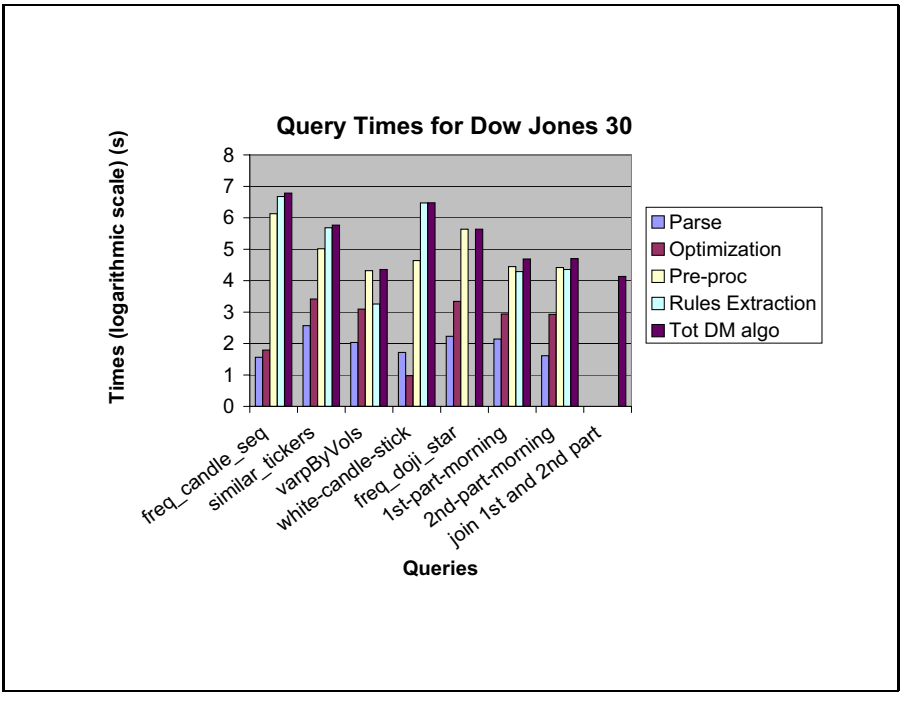


Fig. 8. Query Execution Times in Experiments on Dow Jones 30

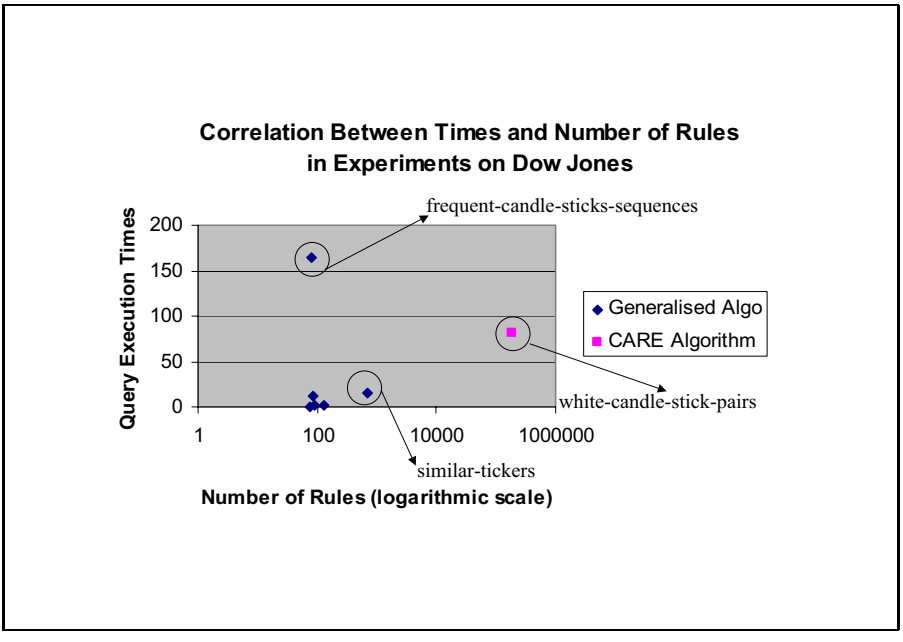


Fig. 9. Correlation Between Times and Volume of Results in Experiments on Dow Jones

mining condition (evaluated on equality on a little set of values: the 10 different candle-sticks). Instead, the worst query for the generalised algorithm is `frequent_candle_sticks_sequences`: it must evaluate the cross mining condition on the set of different trading dates, which is pretty big in a time interval of some years (4764 values). Finally you can observe again that `CARE` algorithm works much faster than the generalised algorithm. Indeed, in this experiment, `white-candle-stick-pairs` evaluated 189660 candidate rules in a lower time than the time required by the generalised algorithm to retrieve only 78 rules (but on different mining conditions). However, we must observe again, as we did already with the experiments on Web log, that this confirms the fact that this is the price to pay to have the possibility to treat more general conditions in queries. For instance, cross mining conditions (i.e., a comparison between body and head features) could be very important in practical applications.

5 Evaluation of Discovered Knowledge

In previous Section we reported the sequences of queries submitted to `MINE RULE` system in order to discover useful knowledge in practical applications, such as the analysis of Web logs for Web usage mining, and of financial data (Dow Jones, in particular). We can draw now some conclusions on the discovered patterns.

In order to be useful, discovered patterns must be:

1. interesting for the user/analyst
2. actionable, in the sense that immediate actions/decisions can be taken as a consequence of their observation.

In constrained based mining the first point is immediately fulfilled by the retrieved patterns, because by definition extracted patterns satisfy user defined constraints. Indeed, constraints are provided by the analyst to the system in order to identify the desired patterns and discard all the remaining ones. Desired patterns could be the interesting ones for many reasons. First of all because they occur frequently, and therefore they refer to a statistically relevant number of events occurred in the application domain. Secondarily because some user constraints are able to discriminate the properties of the desired pattern class with respect to some contrasting classes. However, notice that sometimes it is not easy to identify the right constraint (or at least the right constant value in a comparison). For instance, in some of the examples, as in `freq-doji-star-candle-sticks-in-time` we adopted a sort of "fuzzy" constraints. In other cases, such as in `VarpByVols`, a preparatory session was necessary. We generated derived attributes (for the ratio between daily exchange and average volume and for the price percentage variation) and discretised them in two boolean attributes (`high_volume` and `high_varp`). The discretization was tested several times (using different thresholds) and the results used later, during the mining step.

The second point is more difficult to establish in an absolute sense. Usually, a post-processing phase, following the proper mining phase is necessary to establish if the retrieved patterns are actionable. Generally, a cross-over query that retrieves the original data in which a pattern occurs is necessary to reveal in which way or which data representing real life entities are involved in the phenomena described by the patterns themselves. For instance, in the Web application domain, if patterns describing users attempts to change passwords, or putting in evidence which user configuration settings more frequently are correlated to system errors are employed to identify the users causing the problem, those patterns are immediately actionable because are self explaining. Other patterns, such as patterns on the users' most frequent crawling paths, are more difficult to translate immediately in some actions on the Web applications because might indicate a high value path or content in the Web application or non perfectly adequate tools (such as search engines or indexes) in the site, as well. In this latter case, this might involve a new design of the application, the hypertext structure and of main the content areas that organise the Web site. Finally, the discovered patterns could also identify two content areas that are contextually correlated but could not require the involvement of a new Web site design. This is the case in which it is necessary providing the users with some additional suggestions on the pages where the user could find some content in which he/she could be interested in (see recommending systems). As regards the financial domain, an example of an actionable pattern is constituted by the pairs of stocks exhibiting frequently a white candle-stick because they can be used to

suggest the composition of users' portfolios. However, we should observe that in this case we discovered many rules that Another example, is the detection of a bullish doji star pattern which is a suggestion for investors to perform a sell or buy of stocks.

6 Guidelines for Using an Inductive Database in the Web Mining Application Domain

In previous Sections we have described the process we underwent for mining a Web site. This process is briefly summarised here in order to give the user an abstraction on this process, of the difficulties in which he might be involved applying an inductive database to the analysis of a Web application load, to the identification of the profile of the users in terms of frequent visits and to the usability of the Web site.

These steps are very much alike to the traditionally well-known KDD process for the discovery of knowledge from a database:

1. customization and storage of the Web log
2. preparation of the data
3. individuation and selection of the interesting data
4. mining phase
5. post-processing of the result
6. interpretation

The *customisation and storage of the Web log* corresponds to the step of loading and integration of the data that are relevant to the analysis. This step comprises the memorization of all the elements that result useful during the analysis, such as user identification, user session identification, Web crawlers robot exclusions (because they automatically spam all the Web sites and thus show a typology of interaction with the Web application that is not leaded by the real information content displayed by the pages).

Preparation of the data is a step that is often necessary for increasing the performance of the following mining phase. It consists in the selection of the data that probably will be involved in the interested patterns. Therefore this phase allows pruning of large volumes of data that will not contribute to the searched patterns. For instance, if we are looking for frequent crawling paths by users, we can immediately discard all those requests referring to pages that have been requested a little number of times. On the contrary, if we want to identify the `top k pages` in the user preferences, we should probably discard requests to those pages that, although the most frequently selected, do not provide the user the final content he is really looking for. These pages are probably some indexes and the map or overview pages of the Web site – in conclusions, those elements that structurally are needed to the crawling of the Web site itself.

Individuation and selection of the interesting data consists in definition of the constraints (and of their parameters values) that define the interesting patterns for the user and will probably later result in actionable patterns.

Some examples are provided, for instance, by the patterns that describe the profile of those users whose requests frequently provide the greatest traffic load over the net; the parameter values in this case express the volume in bytes that defines a congested traffic situation.

The mining phase consists in the execution of the mining query provided by the user and incorporating user constraints on the prepared and selected data. It results in a set of patterns that will later be post-processed in the following phase either for visualisation or for evaluation purposes.

Post-processing can consist in pattern selection, for eliminating pattern redundancies and increase the quality of the result, or further queries over both patterns and data. Cross-over queries are often necessary in order to evaluate pattern actionability. For instance, if a pattern describes the top k most frequently occurred pair of pages, we probably would also be interested in the pages themselves, in their content and discover in which way they are related to each other. Probably also an interesting issue is the profile of the users who frequently requested (at least one of) them. All these questions can be answered by some post-processing queries that do a cross-over between discovered patterns and the available data both in the Web log and in the Web site.

Interpretation phase inspects both the results of the mining phase and the results of the post-processing phase and decide how to translate in practice the results obtained by previous phases (pre-processing, mining, post-processing) and the deployment. Results of this phase consists either in actions over the Web application design or in the decision of performing further queries, starting again to execute some steps of the discovery cycle, either from the first step, the second or the third step (customisation and storage of the Web log, preparation or selection of interesting data).

7 Conclusions

We presented the application of inductive databases to two practically important case studies. The first one is the analysis of Web logs of the main Web application of a University Department. The Web log was a conceptual log, obtained by integration of standard (ECFL) Web server logs with information on Web design application and Web pages content information. The second one is the analysis of stocks quotes from the Dow Jones index, from 1998 till 2003. We adopted Japanese candle-sticks, a descriptive pattern proposed in technical analysis to determine in conjunction with other relevant events the main occurrences in time of certain relevant events in the evolution of stocks quotes.

With both these applications we applied and evaluated the usability and expressive power of a mining query language – MINE RULE. The purpose was to verify its feasiblility in practice to solve real case problems and experiment the suitability of some KDD scenarios, developed for inductive databases.

KDD scenarios have been previously produced as a set of characteristic queries solving some frequently asked questions (mining problems) from inductive database end users/analysts in order to recover from frequently occurring

problems in their environment. We showed the possibility to employ these scenarios by means of the mining query languages.

The result of the queries provides us also an evaluation of the expressive power of the designed mining languages for inductive databases in `CInQ` project on the development of inductive databases (EU project IST-2000-26469), e.g., `MINE RULE`. It proved to be simple but yet a powerful query language for mining, because with the aid of few *template* queries the user is able to afford the main critical problems respectively in Web usage mining and in financial technical analysis. Indeed, this mining language is provided with expressive constraints and querying predicates (on single or aggregate properties) that are suitable to extract the description patterns needed to analyse the actual data.

In the Web domain application obtained patterns can be exploited for the definition of effective navigation and composition of hypertext elements to be adopted for improving the Web site usability. We also obtained some concrete examples of interesting or suspicious event that are useful to the end-users (Web and system administrators).

In the financial domain, obtained patterns can be effectively used to detect stock quotes evolutionary similarities, to select the stocks for the formation of the investors portfolios and to study the stock trade behavior in general.

The examples of query we provided show that the mining language is powerful, and at the same time versatile because its operational semantics seems to be the basic one. The grouping clause (that corresponds to described entities), the rule attributes selection (that corresponds to observed entities) and the support computation (that corresponds to statistical relevance of the rules) allow the users, in some of their combinations to specify completely different and new problems simply by adoption of a different choice of attributes in grouping, rules and conditions. The results of the different mining queries can then be composed with standard SQL queries (in pre- and post-processing phase) forming a sequence of queries that constitute a KDD scenario for the application of an inductive database to the particular domain analyzed. Indeed these experiments allow us to claim that Mannila and Imielinski's initial view on inductive databases [14] was correct: <<There is no such thing as real discovery, just a matter of the expressive power of the query languages>>.

References

1. M. Botta, R. Meo, and C.Malangone. Association rules extraction with mine rule operator. Technical report, RT73-2003, Dipartimento di Informatica, University of Torino, Italy, April 2003.
2. Stefano Ceri, Piero Fraternali, and Aldo Bongio. Web modeling language (webml): a modeling language for designing web sites. In *Proc. of WWW9 Conference*, May 2000.
3. Stefano Ceri, Piero Fraternali, Aldo Bongio, Marco Brambilla, Sara Comai, and Maristella Matera. *Designing Data-Intensive Web Applications*. Morgan Kaufmann, San Francisco, CA, 2002.
4. Apache Cocoon. Cocoon. http://xml.apache.org/cocoon/.

5. R. Cooley. *Web Usage Mining: Discovery and Application of Interesting Patterns from Web Data*. PhD thesis, University of Minnesota, 2000.
6. R. Cooley, P.N. Tan, and J. Srivastava. *Discovery of Interesting Usage Patterns from Web Data*. LNCS/LNAI. Springer Verlag, 2000.
7. G. Das, K.-I. Lin, H. Mannila, G. Renganathan, and P. Smyth. Rule discovery from time series. In *Proceedings of the 1997 ACM SIGKDD International Conference*, ACM SIGKDD, 1997.
8. D.Brown and R.Jennings. On technical analysis. *Review of Finance Studies*, 2:527–551, 1989.
9. Federico Michele Facca and Pier Luca Lanzi. Mining interesting knowledge from weblogs: A survey. Technical Report 2003.15, Dipartimento di Elettronica e Informazione. Politecnico di Milano., April 2003.
10. J. Farrell. *Portfolio Management: Theory and Application*. McGraw-Hill, 1997.
11. P. Fraternali, M. Matera, and A. Maurino. Conceptual-level log analysis for the evaluation of web application quality. In *Proceedings of LA-Web'03, Santiago, Chile, November 2003*. IEEE Computer Society, 2003.
12. T.-C. Fu, F.L. Chung, V. Ng, and R. Luk. Pattern discovery from stock time series using self-organizing maps. In *Proceedings of the 1997 ACM SIGKDD International Conference*, ACM SIGKDD, 2001.
13. G.Ramazan. The predictability of security returns with simple trading rules. *The Journal of Empirical Finance*, 5:347–359, 1998.
14. T. Imielinski and H. Mannila. A database perspective on knowledge discovery. *Coomunications of the ACM*, 39(11):58–64, November 1996.
15. A. Ito. Empirical evaluation of technical analysis: A synthesis. Technical report, International University of Japan, November 1999.
16. M.C. Jensen. Random walks and technical theories: Some additional evidence. *The Journal of Finance*, (25):469–482, 1970.
17. R. Kohavi and R. Parekh. Ten supplementary analyses to improve e-commerce web sites. In *Proceedings of the Fifth WEBKDD Workshop: Webmining as a premise to effective and intelligent Web Applications*, ACM SIGKDD, Washington, DC, USA, 2003. Springer-Verlag.
18. L.Blume, D.Easley, and M.O'Hara. Market statistics and technical analysis: the role of trading volumes. *The Journal of Finance*, 49:153–181, 1994.
19. A. W. Lo, H. Mamaysky, and J. Wang. Foundations of technical analysis: Computational algorithms, statistical inference, and empirical implementation. *The Journal of Finance*, LV(4):1705–1765, August 2000.
20. R. Meo, G. Psaila, and S. Ceri. An extension to SQL for mining association rules. *Journal of Data Mining and Knowledge Discovery*, 2(2), 1998.
21. Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Efficient mining of association rules using closed itemset lattices. *Inf. Syst.*, 24(1):25–46, 1999.
22. Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Mining bases for association rules using closed sets. In *Proceedings of the 16th International Conference on Extending Databases*, IEEE, 2000.
23. P. Pirolli, J. Pitkow, and R. Rao. Silk from a sow's ear: Extracting usable structures form the web. In *Proc. of CHI 96 Conference*. ACM Press, April 1996.
24. M. Pring. *An introduction to Technical Analysis*. McGraw-Hill, 1997.

25. John R. Punin, Mukkai S. Krishnamoorthy, and Mohammed J. Zaki. Logml: Log markup language for web usage mining. In R. Kohavi, B. Masand, M. Spiliopoulou, and J. Srivastava, editors, *WEBKDD 2001 - Mining Web Log Data Across All Customers Touch Points, Third International Workshop, San Francisco, CA, USA, August 26, 2001. Revised Papers*, volume 2356 of *Lecture Notes in Computer Science*, pages 88–112. Springer, 2002.
26. Jaideep Srivastava, Robert Cooley, Mukund Deshpande, and Pang-Ning Tan. Web usage mining: Discovery and applications of usage patterns from web data. *SIGKDD Explorations*, 1(2):12–23, 2000.
27. M. Teltzrow and B. Berendt. Web-usage-based success metrics for multi-channel businesses. In *Proceedings of the Fifth WEBKDD Workshop: Webmining as a premise to effective and intelligent Web Applications*, ACM SIGKDD, Washington, DC, USA, 2003. Springer-Verlag.
28. R. Wille. Concept lattices and conceptual knowledge systems. *Computers and Mathematics with Applications*, 23:493, 1992.
29. Mohammed Zaki. Mining non-redundant association rules. *Data Mining and Knowledge Discovery*, 9:223–248, 2004.

Contribution to Gene Expression Data Analysis by Means of Set Pattern Mining

Ruggero G. Pensa[1], Jérémy Besson[1,2],
Céline Robardet[3], and Jean-François Boulicaut[1]

[1] INSA Lyon, LIRIS CNRS UMR 5205,
F-69621 Villeurbanne cedex, France
{Ruggero.Pensa, Jeremy.Besson, Jean-Francois.Boulicaut}@insa-lyon.fr
[2] UMR INRA/INSERM 1235,
F-69372 Lyon cedex 08, France
[3] INSA Lyon, PRISMA,
F-69621 Villeurbanne cedex, France
Celine.Robardet@insa-lyon.fr

Abstract. One of the exciting scientific challenges in functional genomics concerns the discovery of biologically relevant patterns from gene expression data. For instance, it is extremely useful to provide putative synexpression groups or transcription modules to molecular biologists. We propose a methodology that has been proved useful in real cases. It is described as a prototypical KDD scenario which starts from raw expression data selection until useful patterns are delivered. It has been validated on real data sets. Our conceptual contribution is (a) to emphasize how to take the most from recent progress in constraint-based mining of set patterns, and (b) to propose a generic approach for gene expression data enrichment. Doing so, we survey our algorithmic breakthrough which has been the core of our contribution to the IST FET cInQ project.

1 Introduction

Thanks to a huge research and technological effort, one of the challenges for molecular biologists is to discover knowledge from data generated at very high throughput. Indeed, different techniques (including microarray [1] and SAGE [2]) enable to study the simultaneous expression of (tens of) thousands of genes in various biological situations or experiments. Such data can be seen as expression matrices in which the expression level of genes (the attributes or columns) are recorded in various biological situations (the objects or rows). A toy example of a gene expression matrix is in Fig. 1a. Exploratory data mining techniques are needed that can, roughly speaking, be considered as the search for interesting bi-sets, i.e., sets of biological situations and sets of genes which are associated in some way. Indeed, it is interesting to look for groups of co-regulated genes, also known as synexpression groups [3], for which a reasonable assumption is that they participate in a common function within the cell. The association between

J.-F. Boulicaut et al. (Eds.): Constraint-Based Mining, LNAI 3848, pp. 328–347, 2005.

a set of co-regulated genes and the set of biological situations that gives rise to this co-regulation is called a transcription module and their discovery is a major goal in functional genomics since it paves the way to a better understanding of gene regulation networks.

The use of hierarchical clustering (see, e.g., [4]) is quite popular among practitioners. Genes are grouped together according to similar expression profiles. The same can be done on biological situations. Thanks to the appreciated vizualization component introduced with [4], biologists can identify sets of genes that are co-regulated in some sets of situations. Global patterns like partitions provide an a priori interesting "global picture" of similarity structures in the whole data. The results of most of the clustering algorithms are non overlapping groups of genes. It means that a given gene belongs to one and only one cluster while we already know genes which clearly participate to various biological functions. Furthermore, their heuristic nature can lead to different results. Co-clustering or bi-clustering techniques do not change fundamentally the problem: the benefit comes from an assessment of the association between both partitions, i.e., sets of genes and sets of situations but we still get non overlapping partitions based on a local optimization process [5,6]. In other terms, we get a global pattern which capture some more or less expected phenomena.

A complementary approach is to look for collections of local patterns in the gene expression data. Heuristic statistical methods have been proposed to identify a priori interesting bi-sets from raw numerical data (see, e.g., [7,8]). A promising direction of research is to consider complete constraint-based mining techniques on boolean gene expression data sets. The completeness assumption means that every pattern from the pattern language which satisfies the defined constraints has to be returned (e.g., every frequent set, every closed set) and, in this case, we use non heuristic methods. In these data sets, boolean gene expression properties are encoded, e.g., over-expression, strong variation, co-regulation. We get boolean data sets which are also called in some application domains transactional data sets.

Let $\mathcal{O}$ denotes a set of objects or rows (e.g., biological situations) and $\mathcal{P}$ denotes a set of properties or columns (e.g., genes). For instance, expression properties can be encoded into a boolean matrix $\mathbf{r} \subseteq \mathcal{O} \times \mathcal{P}$. $(o_i, g_j) \in \mathbf{r}$ denotes that gene j has the encoded expression property in situation i. For deriving a boolean context from raw gene expression data, we generally apply discretization operators that, depending of the chosen expression property, compute thresholds from which it is possible to decide between wether the true or the false value must be assigned. On our toy example in Fig. 1, $\mathcal{O} = \{h_1, h_2, h_3, h_4, d_1, d_2, d_3, d_4\}$ and $\mathcal{P} = \{g_1, g_2, \ldots, g_8\}$. A value "1" for a biological situation and a gene means that the gene is up (greater than $|t|$) or down (lower than $-|t|$) regulated in this situation. Using threshold $t = 0.4$ for Fig. 1a leads to the boolean matrix in Fig. 1b.

Local pattern discovery tasks can be performed when searching for putative synexpression groups or transcription modules. To compute synexpression groups, we can extract the so-called frequent itemsets (sets of genes) from the de-

rived boolean contexts. Notice that sets of genes that are frequently co-regulated can be post-processed into association rules [9,10].

In our boolean toy example (Fig. 1), the genes from $\{g_2, g_5\}$ are in relation with $\{h_1, h_2, h_4, d_3\}$.

The relevancy of the extracted patterns can be improved by considering the frequent closed itemsets which are the frequent maximal sets of genes whose encoded expression properties are shared by a same set of biological situations. For instance, $\{g_2, g_4, g_5, g_7\}$ is a closed itemset because g_4 and g_7 are the other genes which are in relation with each element from $\{h_1, h_2, h_4, d_3\}$. Formally these local patterns are the set components of formal concepts [11]. A formal concept is a maximal set of genes associated to a maximal set of situations, e.g., $(\{h_1, h_2, h_4, d_3\}, \{g_2, g_4, g_5, g_7\})$ in the data from Fig. 1b. Such patterns can indeed be considered as putative transcription modules [12,13,14].

	Genes							
Sit.	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8
h_1	0.1	-0.5	0.3	0.7	3	0.2	6.1	-0.1
h_2	0.2	-0.6	0.4	0.5	1.2	0.1	4.2	-0.5
h_3	0.2	-0.3	0.9	0.1	0.4	5	0.5	-0.1
h_4	2.1	-0.7	-0.2	0.6	4.1	0.3	5.3	-0.3
d_1	0.2	-0.8	0.2	-0.5	0.4	6.3	0.4	-0.6
d_2	2.3	-0.4	0.1	0.7	-5.1	0.4	5.8	-0.2
d_3	1.2	-0.6	0.1	0.6	3.6	0.3	6.2	-0.1
d_4	1.6	0.1	0.3	0.6	2.8	0.4	4.9	0.1

(a)

	Genes							
Sit.	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8
h_1	0	1	0	1	1	0	1	0
h_2	0	1	0	1	1	0	1	1
h_3	0	0	1	0	0	1	1	0
h_4	1	1	0	1	1	0	1	0
d_1	0	1	0	1	0	1	0	1
d_2	1	0	0	1	1	0	1	0
d_3	1	1	0	1	1	0	1	0
d_4	1	0	0	1	1	0	1	0

(b)

Fig. 1. A gene expression matrix (a) and a derived boolean context (b)

This paper is a methodological paper. It abstracts our practice in several real-life gene expression data analysis projects to disseminate a promising practice within the scientific community. Our methodology covers the whole KDD process and not just the mining phase. Starting from raw gene expression data, it supports the analysis and the discovery of transcription modules via a constraint-based bi-set mining approach from computed boolean data sets. The generic process is described within the framework of inductive databases, i.e., each step of the process can be formalized as a query on data and/or patterns that satisfy some constraints [15,16]. It leads us to a formalization of boolean gene expression data enrichment. We already experimented a couple of practical instances of this approach and it has turned to be crucial for increasing the biological relevancy of the extracted patterns.

Details about each step of the method and the algorithms or solvers which have been developed in the context of the CInQ project have been already published. Therefore, we avoid most of the technical details, just emphasizing the main algorithmic principles and the methodological added-value of our "in silico" approach for transcription module discovery.

The main publications which are associated to this method are:

- Preprocessing numerical gene expression data to encode boolean gene expression properties [9,17].
- Using AC-Miner [18] for computing frequent closed sets and interesting association rules between boolean gene expression properties [9];
- Computing putative transcription modules as formal concepts with a AC-Miner-like algorithm [12,13];
- Using D-Miner for computing putative transcription modules as formal concepts under monotonic constraints [19,14];
- Boolean gene expression data enrichment [20,14].
- Post-processing putative transcription modules [21].

2 Classical Approaches in Gene Expression Data Analysis

From a technical point of view, traditional gene expression data analysis is based on similarities between expression profiles. The expression profile of a gene, is the sequence of its expression values in different biological situations. For example, in the drosophila melanogaster data set (see [22]), the expression levels of about 4 000 genes are measured for a number of time points during the drosophila life cycle. Studying the expression profile of each gene, it is possible to observe the behavior of such a gene during the whole life cycle. A typical analysis task, is to compare expression profiles two by two, noticing the principal differences and similarities between two expression profiles. This is clearly not feasible when thousands of genes are involved. An important contribution to gene expression data analysis is due to Eisen et al. (see [4]). They consider a technique based on hierarchical clustering which enables to compare expression profiles of thousands of genes simultaneously. Genes sharing similar expression profiles are grouped together in the same subtree structure of the resulting dendrogram. This supports the analysis for finding putatively cooperating genes. Dually, biological situations can be processed with the same clustering algorithm. The resulting structure enables to associate groups of genes to groups of situations in which these genes are co-expressed. For instance, in Fig. 2, we can observe dendrograms for the data set in Fig. 1a. Such an approach can be used for identifying some patterns like putative transcription modules.

One major problem concerning such a technique is that searching transcription modules is not that simple. For instance, most of the traditional clustering algorithms, including [4], provide non overlapping (bi-)clusters: one gene (resp. one situation) is associated to only one cluster. Moreover, similarities are computed by considering the whole collection of gene or situation vectors. From the biological point of view, we know that a gene can participate in various biological functions, in different cells and environmental conditions, and at the same time, it is not influenced by the whole set of situations. Therefore, traditional unsupervised clustering techniques are not really oriented to the discovery of transcription modules and synexpression groups, even though they remain useful for exploratory analysis of gene expression data sets.

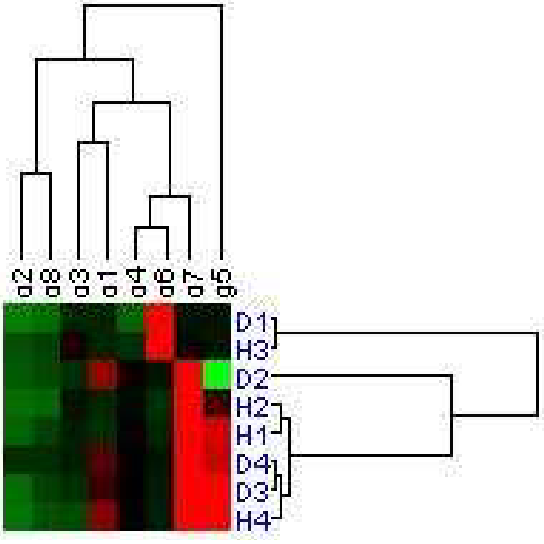


Fig. 2. Dendrograms obtained after a hierarchical clustering on the data from Fig. 1a

A solution can come from local patterns, i.e., patterns which hold in part of the data. For example, the signature algorithm (see [7,8]) enables to find some putative transcription modules starting from a set of known genes. These techniques however heuristically compute some a priori interesting patterns. It makes sense to look at the recent breakthrough concerning complete algorithms for local set pattern mining.

3 A KDD Approach for Gene Expression Analysis

We introduce our KDD-based methodology for gene expression data analysis. It exploits our results in several domains like constraint-based data mining, preprocessing of gene expression raw data, and postprocessing of pattern collections. It has been proved useful for supporting the search of putative transcription modules.

We decided to work on Boolean gene expression data sets instead of numerical data sets. Boolean gene expression data sets encode boolean gene expression properties. The main advantage is that beside encoding techniques based on raw value discretizations, an expert knowledge can be used for assessing the encoding (e.g., checking that computed property is consistent with some available knowledge). A second advantage is that we can add other boolean properties of genes within the same context (e.g., the fact that a gene is or not associated to a given transcription factor). The main drawback is that many different point of views can be considered on a phenomenon like over-expression and the proposed encoding techniques have parameters (i.e., thresholds) that can not be

fixed easily. Of course, if the boolean data do not capture well the chosen property, then most of the patterns extracted from it will be irrelevant. Therefore, we have designed a method for fixing encoding method parameters. Once boolean gene expression data sets are available, we have considered the extraction of set patterns like closed sets, association rules, and formal concepts. The number of discovered patterns can be huge and it happens that the computation turns to be untractable. To increase both the relevancy and the tractability of this task, we have considered user-defined constraints which can be pushed into the extraction phase. The final step consists in post-processing the extracted patterns by deducing new information on data, and exploiting it for further mining tasks. We have also designed a technique to visualize similarities between extracted patterns by means of a user-friendly graphical representation. This post-processing has been proved useful to support pattern interpretation by biologists.

3.1 Pre-processing

We assume that raw expression data, i.e., a function that assigns a real expression value to each couple $(o, g) \in \mathcal{O} \times \mathcal{P}$ is available and that some tasks have been selected by the molecular biologists. A typical example concerns the discovery of putative transcription modules that involve at least a given set of genes that are already known to be co-regulated in a given class of biological situations, e.g., diabetic ones.

Due to the lack of space, we do not consider the typical data manipulation statements that are needed, e.g., for data normalization, data cleaning, gene and/or biological situation selection according to some background knowledge (e.g., removing housekeeping genes from consideration).

Discretization. This step concerns gene expression property encoding and is obviously crucial. The simplest case concerns the computation of a boolean matrix $\mathbf{r} \subset \mathcal{O} \times \mathcal{P}$ which encode a simple expression property for each gene in each situation, e.g., over-expression[1]. Different algorithms can be applied and parameters like thresholds have to be be chosen. For instance, [9] introduces three techniques for encoding gene over-expression:

- "Mid-Ranged". The highest and lowest expression values in a biological situation are identified for each gene and the mid-range value is defined. Then, for a given gene, all expression values that are strictly above the mid-range value give rise to value 1, 0 otherwise.
- "Max - X% Max". The cut off is fixed w.r.t. the maximal expression value observed for each gene. From this value, we deduce a percentage X of this value. All expression values that are greater than the (100 - X)% of the Max value give rise to value 1, 0 otherwise.

[1] Not only it is possible to consider several attributes per gene for one property, e.g., one for "strong overexpression" and one for "suspected strong-expression" but also one can decide to encode various properties per gene like "up-regulation" and "down-regulation".

– "X% Max". For each gene, we consider the biological situations in which its level of expression is in X% of the highest values. These genes are assigned to value 1, 0 for the others.

These techniques give different points of view on the over-expression biological phenomenon and it is unclear which one performs better. The impact of the chosen technique and the used parameters on both the quantity and the relevancy of the extracted patterns is crucial. For instance, the density of the discretized data depends on the discretization parameters and the cardinalities of the resulting sets (collections of itemsets, association rules or formal concepts) can be very different. We clearly need a method to evaluate different boolean encoding (different techniques and/or various parameters) of the same raw data and thus a framework to support user decision about the discretization from which the mining process can start. Our thesis is that a good discretization might preserve some properties that can be already observed from raw data. Let E denote a gene expression matrix. Let $\{Bin_i,\ i = 1..b\}$ denote a set of different discretization operators and $\{\mathbf{r}_i,\ i = 1..b\}$ a set of boolean contexts obtained by applying these operators, i.e. $\forall i = 1..b,\ \mathbf{r}_i = Bin_i(E)$. Let $S : \mathbb{R}^{n,m} \longmapsto \mathbb{R}$ denote an evaluation function that measure the quality of the discretization of a gene expression matrix. We say that a boolean context $\mathbf{r}_i$ is more valid than another context $\mathbf{r}_j$ w.r.t the S measure if $S(\mathbf{r}_i) > S(\mathbf{r}_j)$. In [17], we studied an original method for such an evaluation. We suggest to compare the similarity between the dendrogram generated by a hierarchical clustering algorithm (e.g., [4]) applied to the raw expression data and the dendrograms generated by the same algorithm applied to each derived boolean matrix. Given a gene expression matrix E and two derived boolean contexts $\mathbf{r}_i$ and $\mathbf{r}_j$, we can choose the discretization that leads to the dendrogram which is the most similar to the one built on E. The idea is that a discretization that preserves the expression profile similarities is considered more relevant. A simple measure of similarity between dendrograms has been studied and experimentally validated on various gene expression data sets.

Let $\mathcal{O} = \{o_1, \ldots, o_n\}$ denote the set of n objects. Let T denote a binary tree built on $\mathcal{O}$. Let $\mathcal{L} = \{l_1, \ldots, l_n\}$ denote the set of n leaves of T associated to $\mathcal{O}$ for which, $\forall i \in [1 \ldots n]$, $l_i \equiv o_i$. Let $\mathcal{B} = \{b_1 \ldots b_{n-1}\}$ denote the set of the $n-1$ internal nodes of T generated by a hierarchical clustering algorithm starting from $\mathcal{L}$. By construction, we consider $b_{n-1} = r$, where r denotes the root of T. Let us define the two sets:

$$\delta(b_i) = \{b_j \in \mathcal{B} \mid b_j \textit{ is a descendent of } b_i\}$$
$$\tau(b_i) = \{l_j \in \mathcal{L} \mid l_j \textit{ is a descendent of } b_i\}.$$

We want to measure the similarity between a tree T and a reference tree T_{ref} built on the same set of objects $\mathcal{O}$. For each node b_i of T, we define the following score (denoted S_B and called **BScore**):

$$S_B(b_i, T_{ref}) = \sum_{b_j \in \delta(b_i)} a_j$$

$$a_j = \begin{cases} \frac{1}{|\tau(b_j)|}, & if \quad \exists b_k \in T_{ref} \mid \tau(b_j) = \tau(b_k) \\ 0, & otherwise \end{cases} \quad (1)$$

To obtain the similarity score of T w.r.t. T_{ref} (denoted S_T and called **TScore**), we consider the **BScore** value on the root, i.e.:

$$S_T(T, T_{ref}) = S_B(r, T_{ref}) \quad (2)$$

As usually, it is interesting to normalize the measure to get a score between 0 (for a tree which is totally different from the reference) and 1 (for a tree which is equal to the reference). In the **TScore** measure, since its max value depends on the tree morphology, we can normalize by $S_T(T_{ref}, T_{ref})$:

$$\overline{S_T}(T, T_{ref}) = \frac{S_T(T, T_{ref})}{S_T(T_{ref}, T_{ref})} \quad (3)$$

$\overline{S_T}(T, T_{ref}) = 0$ means that T is totally different from T_{ref}, i.e., there are no matching nodes between T and T_{ref}. Indeed, $\overline{S_T}(T, T_{ref}) = 1$ means that T is totally similar to T_{ref}, i.e., every node in T matches with a node in T_{ref}. Given two trees T_1 and T_2 and a reference T_{ref}, if $\overline{S_T}(T_1, T_{ref}) < \overline{S_T}(T_2, T_{ref})$, then T_2 is said to be more similar to T_{ref} than T_1 according to **TScore**.

We can apply this technique to both the situation and gene trees. Indeed, we obtain two different similarity scores. To consider a unique **TScore**, we can compute the mean between the two scores. However, in order to force the general similarity score to be equal to 0 when at least one of the two scores is equal to 0, we prefer to use the square root of the product of the two similarity scores:

$$\overline{S_{AT}}(T_g, T_s, T_{ref}) = \sqrt{\overline{S_T}(T_g, T_{ref}) \cdot \overline{S_T}(T_s, T_{ref})}$$

where T_g and T_s denote respectively the dendrograms for genes and situations.

Let us apply this technique to the gene expression matrix in Fig. 1a. We decide to evaluate the set of discretization operators Bin_i, where $i = 1..10$, and such that values in the matrix whose absolute value is greater than $i \times 10^{-1}$ are coded with a "1" in the boolean matrix, while the other expression values are coded with a "0" (e.g., for $i = 5$, the threshold is set to $5 \times 10^{-1} = 0,5$). Therefore, we can obtain ten different boolean contexts and we process each of them with the same hierarchical clustering algorithm. Then we compare the resulting gene and situation dendrograms with those obtained by clustering the original real expression matrix from Fig. 1a. The results are presented in Fig. 3. We can observe that for a threshold of 0.4 the square root of the product between the gene similarity score and the situation similarity score is maximal. If we discretize the raw data from Fig. 1a with such a threshold, we obtain the boolean context given in Fig. 1b.

Boolean Gene Expression Data Enrichment. We can mine boolean gene expression matrices for frequent sets of genes and/or situations, association rules between genes and/or situations, formal concepts, etc. In the following, we focus

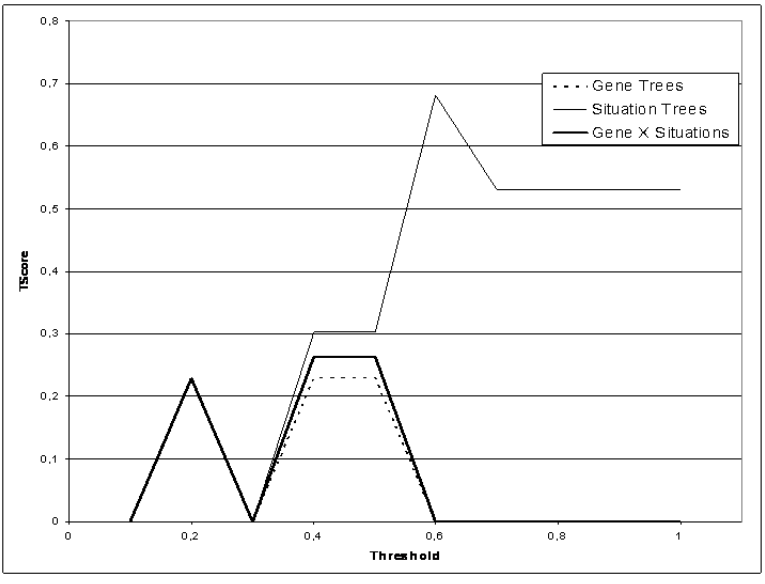


Fig. 3. Similarity scores w.r.t. threshold values

on mining phases that compute formal concepts. When the extractions are feasible, many patterns are discovered (up to several millions) while only a few of them are interesting. It is however extremely hard to decide of the interestingness characteristics a priori. We now propose a powerful approach for improving the relevancy of the extracted formal concepts by boolean data enrichment. It can be done a priori with some complementary information related to genes and/or situations. For instance, we can add information about the known functions of genes as it is recorded in various sources like Gene Ontology [23]. Other information can be considered like the associated transcription factors. A simple way to encode this kind of knowledge consists in adding a row to $\mathbf{r}$ for each gene property. Dually, we can add some properties to the situations vectors. For instance, if we know the class of a group of situations (e.g. diabetic vs. non diabetic individuals) we can add a column to $\mathbf{r}$. We can also add boolean properties about, e.g., cell type or environmental features. Enrichment of boolean data can be performed by more or less trivial data manipulation queries from various bioinformatics databases. $\mathbf{r}' \subset \mathcal{O}' \times \mathcal{P}'$ will denote the relation of the enriched boolean context.

In Fig. 4a, we add three gene properties tf_1, tf_2 and tf_3. A value "1" for a gene and a property means that this gene has the property. For instance, tf_1 could mean that the gene is regulated by a given transcription factor. Dually, in Fig. 4b, we consider two classes of situations, namely c_H and c_D. A value "1" for a situation and a class means that this situation belongs to the class but this could be interpreted in terms of situation properties as well. For instance, c_D (resp. c_H) could mean whether biological situations are diabetic (resp. healthy) ones. In

the data in Fig. 4b, a formal concept like $(\{d_2, d_3, d_4, tf_1, tf_3\}, \{g_1, g_4, g_5, g_7, c_D\})$ informs us about a "maximal rectangle of true values" that involves four genes, regulated by two transcription factors tf_1 and tf_2 in three situations that are of class c_D. This could reveal sets of genes that are co-regulated in diabetic situations but not in healthy ones. We will discuss later how iterative enrichment enables to improve the relevancy of the extracted patterns.

	Genes							
Sit.	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8
h_1	0	1	0	1	1	0	1	0
h_2	0	1	0	1	1	0	1	1
h_3	0	0	1	0	0	1	1	0
h_4	1	1	0	1	1	0	1	0
d_1	0	1	0	1	0	1	0	1
d_2	1	0	0	1	1	0	1	0
d_3	1	1	0	1	1	0	1	0
d_4	1	0	0	1	1	0	1	0
tf_1	1	0	0	1	1	0	1	1
tf_2	0	1	0	1	1	0	1	0
tf_3	1	1	0	1	1	0	1	0

(a)

	Genes									
Sit.	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	c_H	c_D
h_1	0	1	0	1	1	0	1	0	1	0
h_2	0	1	0	1	1	0	1	1	1	0
h_3	0	0	1	0	0	1	1	0	1	0
h_4	1	1	0	1	1	0	1	0	1	0
d_1	0	1	0	1	0	1	0	1	0	1
d_2	1	0	0	1	1	0	1	0	0	1
d_3	1	1	0	1	1	0	1	0	0	1
d_4	1	0	0	1	1	0	1	0	0	1
tf_1	1	0	0	1	1	0	1	1	1	1
tf_2	0	1	0	1	1	0	1	0	1	1
tf_3	1	1	0	1	1	0	1	0	1	1

(b)

Fig. 4. Two examples of enriched boolean microarray contexts

3.2 Pattern Extraction

Constraint-Based Extraction of Formal Concepts. We consider here formal concept extraction from eventually enriched boolean contexts.

Definition 1 (Bi-set). *A bi-set (T, G) is a couple of sets such that $T \subseteq \mathcal{O}$ and $G \subseteq \mathcal{P}$. We often use the term rectangle to denote bi-sets: clearly, a bi-set defines a combinatorial rectangle in the boolean matrix, i.e., up to permutations over rows and columns.*

Definition 2 (1-rectangle). *A bi-set (T, G) is a 1-rectangle in $\mathbf{r}$ (constraint $\mathcal{C}_{1R}(T, G)$) iff $\forall t \in T$ and $\forall g \in G$ then $(t, g) \in \mathbf{r}$. When a bi-set (T, G) is not a 1-rectangle, we say that it contains 0 values.*

Definition 3 (Formal concept). *A bi-set (T, G) is a concept in $\mathbf{r}$ iff (T, G) is a 1-rectangle and $\forall T' \subseteq \mathcal{O} \backslash T$, $T' \neq \emptyset$, $(T \cup T', G)$ is not a 1-rectangle and $\forall G' \subseteq \mathcal{P} \backslash G$, $G' \neq \emptyset$, $(T, G \cup G')$ is not a 1-rectangle. A concept (T, G) is thus a maximal 1-rectangle. We denote the associated constraint as $\mathcal{C}_{Concept}(T, G, \mathbf{r})$.*

Thanks to the mathematical properties of formal concepts [11] (e.g., each formal concept is built on closed sets for both dimensions), a first approach to extract the complete collection of formal concepts consists in computing the whole collection of closed itemsets and their associated objectsets. This can be

done by slightly modifying existing algorithms for extracting closed sets (see, e.g., [24] for a survey). Indeed, in some applications, we can use frequent closed set mining with a 0 frequency threshold. In our biological contexts, the number of genes (items) is very large (up to thousands) and it is often impossible to use these algorithms to perform this task. However, in many gene expression data sets, the number of biological situations, i.e., of objects, is quite small. As a result, a simple transposition of the matrix solves the problem [12,13]. When the number of objects increases, this technique is however no more tractable.

To overcome this problem (i.e., working on boolean gene expression matrices whose none of the two dimensions is small enough), we have been considering the definition and the use of constraints which enable to reduce both the search space and the solution space. It is indeed possible to consider formal concepts whose one set component is large enough [25]. We have studied the possibility to enforce constraints on both components.

Definition 4 (Constraints on formal concepts). *Assume that* (T, G) *is a formal concept in* $\mathbf{r}$.
Minimal size constraints:
(T, G) *satisfies the constraint* $\mathcal{C}_t(\mathbf{r}, \sigma_1, T)$ *iff* $|T| \geq \sigma_1$.
(T, G) *satisfies the constraint* $\mathcal{C}_g(\mathbf{r}, \sigma_2, G)$ *iff* $|G| \geq \sigma_2$.
Syntactical constraints:
(T, G) *satisfies the constraint* $\mathcal{C}_{Inclusion}(\mathbf{r}, X, G)$ *iff* $X \subseteq G$.
(T, G) *satisfies the constraint* $\mathcal{C}_{Inclusion}(\mathbf{r}, X, T)$ *iff* $X \subseteq T$.
Minimal area constraint:
(T, G) *satisfies the constraint* $\mathcal{C}_{area}(\mathbf{r}, \sigma, (T, G))$ *iff* $|T| \times |G| \geq \sigma$.

These constraints are quite obvious to interpret for end-users, here molecular biologists. Properties of constraints have been studied extensively and monotonicity properties can lead to major optimizations.

Definition 5 (Monotonic and anti-monotonic constraints). *Let* $\preceq$ *be a partial order on a set* $\mathcal{S}$. *A constraint* $\mathcal{C}$ *on* $\mathcal{S}$ *is said monotonic (resp. anti-monotonic) w.r.t.* $\preceq$ *iff* $\forall\ s_1,\ s_2 \in \mathcal{S}$, *if* $s_1 \preceq s_2$ *and* $\mathcal{C}(s_1)$ *(resp.* $\mathcal{C}(s_2)$*) is satisfied then* $\mathcal{C}(s_2)$ *(resp.* $\mathcal{C}(s_1)$*) is also satisfied.*

Let us now define our partial order on bi-sets.

Definition 6 (Partial order). *The partial order* $\preceq$ *on bi-sets is defined as follows:* $(T_1, G_1) \preceq (T_2, G_2)$ *iff* $T_1 \subseteq T_2$ *and* $G_1 \subseteq G_2$.

Given this partial order, the constraints introduced in Definition 4 are monotonic. We have proposed the D-Miner algorithm for computing every formal concept which satisfies a given monotonic constraint [19]. It generates the formal concept candidates w.r.t. the chosen partial order such that the defined constraints can be pushed deeply into the extraction phase. More precisely, D-Miner first computes a list H of 0-rectangles composed of an object and the items which are not in relation with it. Then, it builds a tree whose root is the bi-set $(\mathcal{O},\mathcal{P})$. Each node (T, G) is recursively split using an element (a, b) of H,

such that $a \cap T \neq \emptyset$ and $b \cap G \neq \emptyset$, until H is empty: the left child is $(T \setminus a, G)$ whereas the right one is $(T, G \setminus b)$. Another constraint denoted $\mathcal{C}_{left}$ has to be pushed to avoid the computation of sub-concepts such that each leaf of the tree is finally a formal concept. Constraint $\mathcal{C}_{left}$ is used to check that all the children of $(T \setminus a, G)$ contain at least one item in b. To illustrate this process, we consider in Fig 5 the extraction of formal concepts (T, G) from $\mathbf{r}_2$ (see Table 1) with an area larger than 4, i.e., satisfying $\mathcal{C}_{area}(\mathbf{r}_2, 4, ((T, G))$. Underlined bi-sets are the leaves which do not satisfy either $\mathcal{C}_{left}$ or $\mathcal{C}_{area}$.

Table 1. Context $\mathbf{r}_2$ (left) and its corresponding H list

	g_1	g_2	g_3
t_1	0	0	1
t_2	1	0	1
t_3	0	0	1
t_4	1	0	1

(t_1, g_1g_2)
(t_2, g_2)
(t_3, g_1g_2)
(t_4, g_2)

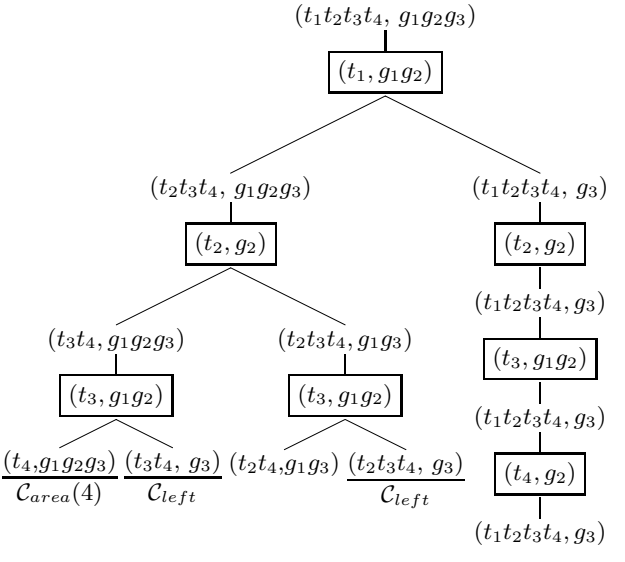


Fig. 5. Formal concept computation on $\mathbf{r}_2$

In $\mathbf{r}_2$, we have only two formal concepts with an area greater or equal to 4: $(\{t_2, t_4\}, \{g_1, g_3)\}$ and $(\{t_1, t_2, t_3, t_4\}, \{g_3\})$.

It is quite useful to use these constraints in enriched contexts. For instance we can search for potentially interesting bi-sets that involve a minimum number of genes (more than γ) to ensure that the extracted formal concepts are not due to noise. They must also be made of enough (say ≥ 3) biological situations from

$\mathcal{D} = \{d_1, .., d_4\}$ and few (say ≤ 1) biological situations of $\mathcal{H} = \{h_1, .., h_4\}$ or vice versa. In other terms, the potentially interesting bi-sets (T, G) are formal concepts that verify the following constraints as well:

$$(|T_H| \geq 3 \ \wedge \ |T_D| \leq 1 \ \wedge \ |G| \geq \gamma) \tag{4}$$
$$\vee (|T_D| \geq 3 \ \wedge \ |T_H| \leq 1 \ \wedge \ |G| \geq \gamma) \tag{5}$$

where T_H and T_D are the subsets of T that concern respectively the biological situations from $\mathcal{H}$ (e.g., healthy individuals) and the ones from $\mathcal{D}$ (e.g., diabetic patients). Notice that this constraint which is the disjunction of Equation 4 and Equation 5 is a disjunction of a conjunction of monotonic and anti-monotonic constraints on $2^{\mathcal{O}}$ and $2^{\mathcal{P}}$. Using D-MINER, we push the monotonic ones, i.e.:

$$q_1 : \mathcal{C}_{Concept}(T, G, \mathbf{r}) \wedge \mathcal{C}_t(\mathbf{r}, 3, T_H) \wedge \mathcal{C}_g(\mathbf{r}, \gamma, G).$$

$$q_2 : \mathcal{C}_{Concept}(T, G, \mathbf{r}) \wedge \mathcal{C}_t(\mathbf{r}, 3, T_D) \wedge \mathcal{C}_g(\mathbf{r}, \gamma, G).$$

Applying the previously defined constraints to the data set in Fig. 4a (using D-MINER, then post-processing the pattern collection to check non monotonic ones), we get the two following formal concepts:

$(\{h_1, h_2, h_4, d_3, tf_2, tf_3\}, \{g_2, g_4, g_5, g_7\})$ for q_1 with $\gamma = 1$
$(\{h_4, d_2, d_3, d_4, tf_1, tf_3\}, \{g_1, g_4, g_5, g_7\})$ for q_2 with $\gamma = 1$

In this example, g_1 and g_2 are putative interesting genes, each of them characterizes only one class of situations represented in the data set. Moreover, all these genes are regulated by the same transcription factor tf_3. This could mean that they are involved in the same biological function of the cell.

Another way to proceed, is to consider the class properties c_H and c_D that we added into the boolean context in Fig. 4b. We can easily perform an extraction of formal concepts under the following constraints:

$$q_3 : \mathcal{C}_{Concept}(T, G, \mathbf{r}) \wedge \mathcal{C}_{Inclusion}(\mathbf{r}, c_H, G) \wedge \mathcal{C}_g(\mathbf{r}, \gamma, G) \wedge \mathcal{C}_t(\mathbf{r}, \gamma\prime, T_H).$$

With $\gamma = 3$ and $\gamma\prime = 3$, two formal concepts satisfy such a constraint:

$(\{h_1, h_2, h_4, tf_2, tf_3\}, \{g_2, g_4, g_5, g_7, c_H\})$
$(\{h_1, h_2, h_4, tf_1, tf_2, tf_3\}, \{g_4, g_5, g_7, c_H\})$

Then, we can ask for a second collection with all the formal concepts (T, G) such that the class attribute c_D is included in G:

$$q_4 : \mathcal{C}_{Concept}(T, G, \mathbf{r}) \wedge \mathcal{C}_{Inclusion}(\mathbf{r}, c_D, G) \wedge \mathcal{C}_g(\mathbf{r}, \gamma, G) \wedge \mathcal{C}_t(\mathbf{r}, \gamma\prime, T_D).$$

The formal concepts resulting from the execution of the second query, with $\gamma = 3$ and $\gamma\prime = 3$, are:

$(\{d_2, d_3, d_4, tf_1, tf_3\}, \{g_1, g_4, g_5, g_7, c_D\})$
$(\{d_2, d_3, d_4, tf_1, tf_2, tf_3\}, \{g_4, g_5, g_7, c_D\})$

Notice that gene g_2 appears only in the first class of patterns, while g_1 appears only in the second class. In other words, using queries q_3 and q_4 we focus on the same putative interesting genes (and the same situations) obtained with queries q_1 and q_2. The difference is that we use here only monotonic constraints that can be efficiently pushed by D-Miner.

Let us compare these results with a classical gene expression data analysis approach. If we observe the dendrogram obtained by applying a hierarchical clustering algorithm to the raw data set (see Fig. 2), we can notice that only gene g_4 and g_7 are grouped together. Other genes belonging to the pattern extracted before are relatively far (w.r.t. the height of the branches), from g_4 and g_7. It is interesting to notice that genes g_2 and g_5 are considered as not belonging to the same cluster of g_4 and g_7, even for a relatively "high" cut.

3.3 Post-processing and Iteration

Formal concept extraction, even constraint-based mining, can produce large numbers of patterns, especially in the first iteration of the KDD process, i.e., when very few information can be used to further constrain the bi-sets to be delivered. Notice also that, from a practical perspective, not all the specified constraints can be pushed into the mining algorithm: some of these constraints have to be checked in a post-processing phase. For instance, we can exploit non monotonic constraints defined in Equation 4 and Equation 5 (i.e., $|T_D| \leq 1$ and $|T_H| \leq 1$) that can not be pushed within D-Miner.

KDD processes are clearly complex iterative processes for which obtained results can give rise to new ideas for more relevant constraint-based mining phases (inductive queries) or data manipulations. When a collection of patterns has been computed, it can be used for deriving new boolean properties. In particular, let us assume that we got two sets of patterns that can characterize two classes of genes and, dually, two classes of situations. Therefore, we can define two new class properties related to genes and their dual class properties related to situations. The boolean context $\mathbf{r}'$ can then be extended towards $\mathbf{r}'' \subset \mathcal{O}'' \times \mathcal{P}''$. Considering our running example, we can associate a new property p_H (resp. p_D) for the genes not belonging to the formal concepts which are returned by q_4 (resp. q_3). It leads to the enriched boolean context given in Fig. 6. New constraints on the classes can be used for the next mining phase. New set size constraints can be defined as well. As a result, a new iteration will provide a new collection of formal concepts which is more relevant according to the user current task. Each time a collection of formal concepts is available, we can decide either to analyze it by hand, e.g., studying each genes separately, or looking for new boolean data enrichment and revisited constraints for the next iteration. Also, genes to which we can associate new functions, are the best candidates to be chosen for iterating the KDD process and take advantage of larger seed sets of genes.

In any cases, at the end of the process, we have a set of putative interesting genes and a set of putative interesting situations. Iterations can be stopped when we have a set of putative interesting genes that can be easily studied by hand. A priori knowledge is very important at this point. In our running example, we did

	Genes									
Sit.	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	c_H	c_D
h_1	0	1	0	1	1	0	1	0	1	0
h_2	0	1	0	1	1	0	1	1	1	0
h_3	0	0	1	0	0	1	1	0	1	0
h_4	1	1	0	1	1	0	1	0	1	0
d_1	0	1	0	1	0	1	0	1	0	1
d_2	1	0	0	1	1	0	1	0	0	1
d_3	1	1	0	1	1	0	1	0	0	1
d_4	1	0	0	1	1	0	1	0	0	1
tf_1	1	0	0	1	1	0	1	1	1	1
tf_2	0	1	0	1	1	0	1	0	1	1
tf_3	1	1	0	1	1	0	1	0	1	1
p_H	0	1	1	0	0	1	0	1	1	1
p_D	1	0	1	0	0	1	0	1	1	1

Fig. 6. A new enriched boolean context

not introduced any additional information about known genes, i.e., genes that are already known as being directly involved in the analyzed problem. However, studying interactions between genes whose functions are already identified, and new putative interesting genes discovered by means of our methodology, can help biologists to suggest putative functions for new genes.

An other important problem concerns the postprocessing of formal concept collections. We need efficient techniques to support the subjective search for interesting patterns. In [21], we introduced an "Eisen-like" visualization technique, that enables to group similar formal concepts by means of a hierarchial clustering algorithm. We defined a distance between two formal concepts and then a distance between two clusters of formal concepts. For the first step, we use the symmetrical set difference Δ between two sets S_i and S_j: $S_i \Delta S_j = S_i \cup S_j \setminus S_i \cap S_j$.

Definition 7. *(Distance between two formal concepts) Assume that* $c_i = (T_i, G_i)$ *and* $c_j = (T_j, G_j)$ *are two formal concepts, the distance* d *between* c_i *and* c_j *is defined as*

$$d(c_i, c_j) = \frac{1}{2}\frac{|T_i \Delta\, T_j|}{|T_i \cup T_j|} + \frac{1}{2}\frac{|G_i \Delta\, G_j|}{|G_i \cup G_j|} \tag{6}$$

where $|S|$ *denotes the cardinality of* S*.*

To compute the distance between two clusters of formal concepts, we associate a pseudo-concept to each cluster. A pseudo-concept is a unique representation for all the formal concepts within a cluster. It is composed of two fuzzy sets, one set of genes and one set of biological situations: a degree of membership α_i (a real number between 0 and 1) is associated to each element e_i of the referential set (i.e., $\mathcal{O}$ or $\mathcal{P}$). Value 0 (resp. value 1) denotes that the element does not belong (resp. belongs) to the set.

Definition 8. *(Pseudo-concept) A pseudo-concept is denoted by* $(T', G', N) \subseteq \mathcal{O}' \times \mathcal{P}' \times \mathbb{N}$ *with* $\mathcal{O}' = \mathcal{O} \times [0;1]$ *and* $\mathcal{P}' = \mathcal{P} \times [0;1]$. *The weight* N *denotes the number of formal concepts represented by the pseudo-concept.*

It is possible to generalize the distance d for measuring the similarity between pseudo-concepts. The classical fuzzy set operators (indexed with f) are used:

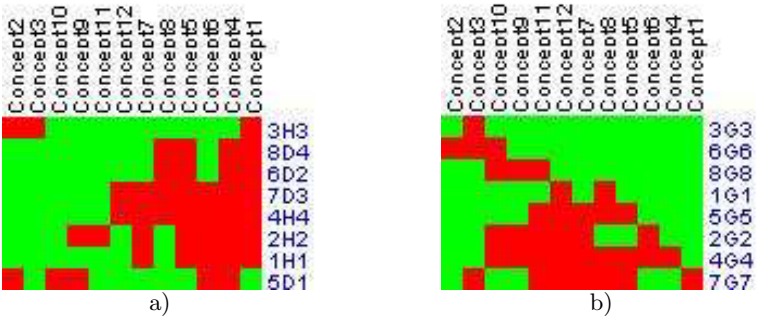
$$S_1 \cup_f S_2 = \{(o, \max\{\alpha_1, \alpha_2\}) \mid o \in \mathcal{O},\ (o, \alpha_1) \in S_1 \text{ and } (o, \alpha_2) \in S_2\}$$
$$S_1 \cap_f S_2 = \{(o, \min\{\alpha_1, \alpha_2\}) \mid o \in \mathcal{O},\ (o, \alpha_1) \in S_1 \text{ and } (o, \alpha_2) \in S_2\}$$
$$S_1 \setminus_f S_2 = \{(o, \alpha_1 - \alpha_2) \mid o \in \mathcal{O},\ (o, \alpha_1) \in S_1 \text{ and } (o, \alpha_2) \in S_2\}$$
$$|S_1|_f = \sum_{o \in \mathcal{O}} \alpha,\ (o, \alpha) \in S_1$$

Thanks to this approach, we can reduce the impact of concept multiplication in noisy boolean data and support the post-processing of tens of thousands of formal concepts.

Example 1. In the boolean context from Fig. 1b, twelve formal concepts (with at least one gene and one situation) can be extracted:

Concept1 : $(\{h_1, h_2, h_3, h_4, d_2, d_3, d_4\}, \{g_7\})$
Concept2 : $(\{h_3, d_1\}, \{g_8\})$
Concept3 : $(\{h_3\}, \{g_3, g_6, g_}7)$
Concept4 : $(\{h_1, h_2, h_4, d_1, d_2, d_3, d_4\}, \{g_4\})$
Concept5 : $(\{h_1, h_2, h_4, d_2, d_3, d_4\}, \{g_4, g_5, g_7\})$
Concept6 : $(\{h_1, h_2, h_4, d_1, d_3\}, \{g_2, g_4\})$
Concept7 : $(\{h_1, h_2, h_4, d_3\}, \{g_2, g_4, g_5, g_7\})$
Concept8 : $(\{h_4, d_2, d_3, d_4\}, \{g_1, g_4, g_5, g_7\})$
Concept9 : $(\{h_2, d_1\}, \{g_2, g_4, g_8\})$
Concept10 : $(\{d_1\}, \{g_2, g_4, g_6, g_8\})$
Concept11 : $(\{h_2\}, \{g_2, g_4, g_5, g_7, g_8\})$
Concept12 : $(\{h_4, d_3\}, \{g_1, g_2, g_4, g_5, g_7\})$

By applying a hierarchical clustering associated to a simple visualization technique (using Treeview from [4]), we provide the pictures (rectangles) in Fig. 7.



a) b)

Fig. 7. Situation (a) and gene (b) rectangles resulting of a hierarchical clustering of concepts

A dark-colored cell in the rectangle means that the related gene (or situation) is present in the related formal concept. Notice that groups of similar formal concepts can be identified by looking for relatively dense red zones either in the situation rectangle or in the gene rectangle.

Thanks to this graphical approach and in contrast to the dendrograms obtained with a simpler approach (see Fig. 2), we can notice a strong correlation between genes involved in previously extracted patterns (g_1,g_2,g_4,g_5,g_7), and a disposition of situations which is more consistent w.r.t. their class values.

4 Biological Validations

The method and the techniques we have considered in the previous sections have been applied with success to different real-life data sets and problems. In some experiments, we have considered well-documented gene expression data sets (i.e., containing accurate biological knowledge) to validate the methods by rediscovery (see, e.g., [17,20]). We have also applied this approach to original gene expression data sets from which new biological knowledge has been extracted. For instance, in [9], the authors have used closed sets (and more precisely some association rules derived from them) to derive biologically relevant knowledge from human SAGE data [26]. The selection on the SAGE data concerns the expression level of 822 genes measured in 74 biological situations (cancerous and not cancerous tissues belonging to various human organs). After an over-expression encoding by means of the "Max - X% Max" method (see Section 3.1), homogeneous closed sets of genes have been studied in detail and, among others, it enabled to suggest a putative function for an EST-encoded protein.

A successful application of constraint-based extraction of formal concepts (see Section 3.2) to an original microarray data set has been described in [14]. Each DNA microarray contains the RNA expression level of about 20 000 genes before and after a perfusion of insulin in human skeletal muscle [27]. It is a nice example of gene expression data enrichment: the considered context encodes information about different gene properties that are biologically relevant (expression level for healthy people and for diabetic patients, regulation by known transcription factors). The set $\mathcal{O}$ of situations was thus partitioned into the set $\mathcal{H}$, the set $\mathcal{D}$ and a set of transcription factors $\mathcal{F}$. After a typical data preprocessing (e.g., removing genes whose none of their transcription factors are known), the final boolean context contained 104 objects (94 transcription factors and 10 biological situations, 5 for healthy individuals and 5 for diabetic patients) and 304 genes. Even though a formal concept discovery from such a boolean context has turned out to be very hard, pushing monotonic constraints has enabled to get significant results. Potentially interesting bi-sets (T, G) were considered as the formal concepts satisfying the following constraints:

$$(|T_H| \geq 4 \wedge |T_D| \leq 2 \wedge |G| \geq \gamma) \tag{7}$$

$$\vee\, (|T_D| \geq 4 \wedge |T_H| \leq 2 \wedge |G| \geq \gamma) \tag{8}$$

The authors have considered in details one of the extracted formal concept which is particularly interesting as it contains genes which are either up-regulated or down-regulated after insulin stimulation, this being based on the homology of their promotor DNA sequences (associated transcription factors) [14]. This is indeed a kind of results we hardly get with classical approaches like [4].

5 Conclusion

We have considered data mining methods and tools which can support knowledge discovery from gene expression data. A prototypical KDD scenario which takes the most from recent progress in constraint-based set pattern mining has been described. Importantly, some of our results on algorithms have been indeed motivated by the gene expression data mining task. For instance, it has motivated the design of D-Miner because of the failure of available algorithms for closed set mining on biological data sets of interest. Concrete instances of this scenario have been considered in several real-life gene expression data analysis problems, including the whole human SAGE [13] data and the microrray data described in [27]. We better understand the crucial issues of boolean gene expression property encoding. Also, boolean gene expression data enrichment appears to be a powerful technique for supporting the iterative search of relevant patterns w.r.t. a given analysis task. The perspectives of this research include the need for fault-tolerant formal concept mining, i.e., strong associations which might however accept some exceptions, but also the multiple uses of the extracted patterns. For instance, local patterns like formal concepts could be used in complementarity with (bi-)clustering techniques, typically to support accurate (bi-)cluster characterization.

Acknowledgements. Most of the results reported in this paper have been obtained during the cInQ IST-2000-26469 European project funded by the European Union. Our research on methodological approaches to gene expression data analysis is also partially funded by CNRS ACI MD46 Bingo. Finally, we would like to thank our colleagues in molecular biology, Olivier Gandrillon and Sophie Rome who have provided such nice challenges for data mining.

References

1. DeRisi, J., Iyer, V., Brown, P.: Exploring the metabolic and genetic control of gene expression on a genomic scale. Science **278** (1997) 680–686
2. Velculescu, V., Zhang, L., Vogelstein, B., Kinzler, K.: Serial analysis of gene expression. Science **270** (1995) 484–487
3. Niehrs, C., Pollet, N.: Synexpression groups in eukaryotes. Nature **402** (1999) 483–487
4. Eisen, M., Spellman, P., Brown, P., Botstein, D.: Cluster analysis and display of genome-wide expression patterns. Proc. Natl. Acad. Sci. USA **95** (1998) 14863–14868

5. Robardet, C., Feschet, F.: Efficient local search in conceptual clustering. In: Proceedings DS'01. Number 2226 in LNCS, Springer-Verlag (2001) 323–335
6. Dhillon, I., Mallela, S., Modha, D.: Information-theoretic co-clustering. In: Proceedings ACM SIGKDD 2003, ACM (2003) 1–10
7. Ihmels, J., Friedlander, G., Bergmann, S., Sarig, O., Ziv, Y., Barkai, N.: Revealing modular organization in the yeast transcriptional network. Nature Genetics **31** (2002) 370–377
8. Bergmann, S., Ihmels, J., Barkai, N.: Iterative signature algorithm for the analysis of large-scale gene expression data. Physical Review **67** (2003)
9. Becquet, C., Blachon, S., Jeudy, B., Boulicaut, J.F., Gandrillon, O.: Strong association rule mining for large gene expression data analysis: a case study on human SAGE data. Genome Biology **12** (2002) See http://genomebiology.com/2002/3/12/research/0067.
10. Creighton, C., Hanash, S.: Mining gene expression databases for association rules. Bioinformatics **19** (2003) 79 – 86
11. Wille, R.: Restructuring lattice theory: an approach based on hierarchies of concepts. In Rival, I., ed.: Ordered sets. Reidel (1982) 445–470
12. Rioult, F., Boulicaut, J.F., Crémilleux, B., Besson, J.: Using transposition for pattern discovery from microarray data. In: Proceedings ACM SIGMOD Workshop DMKD'03, San Diego (USA) (2003) 73–79
13. Rioult, F., Robardet, C., Blachon, S., Crémilleux, B., Gandrillon, O., Boulicaut, J.F.: Mining concepts from large SAGE gene expression matrices. In: Proceedings KDID'03 co-located with ECML-PKDD 2003, Catvat-Dubrovnik (Croatia) (2003) 107–118
14. Besson, J., Robardet, C., Boulicaut, J.F., Rome, S.: Constraint-based concept mining and its application to microarray data analysis. Intelligent Data Analysis journal **9** (2005) 59–82
15. Boulicaut, J.F., Klemettinen, M., Mannila, H.: Modeling KDD processes within the inductive database framework. In: Proceedings DaWaK'99. Volume 1676 of LNCS., Florence, I, Springer-Verlag (1999) 293–302
16. De Raedt, L.: A perspective on inductive databases. SIGKDD Explorations **4** (2003) 69–77
17. Pensa, R., Leschi, C., Besson, J., Boulicaut, J.F.: Assessment of discretization techniques for relevant pattern discovery from gene expression data. In: Proceedings 4th ACM SIGKDD Workshop BIOKDD'04, Seattle (USA), ACM (2004) 24–30
18. Boulicaut, J.F., Bykowski, A., Rigotti, C.: Free-sets: a condensed representation of boolean data for the approximation of frequency queries. Data Mining and Knowledge Discovery journal **7** (2003) 5–22
19. Besson, J., Robardet, C., Boulicaut, J.F.: Constraint-based mining of formal concepts in transactional data. In: Proceedings PAKDD'04. Volume 3056 of LNAI., Sydney (Australia), Springer-Verlag (2004) 615–624
20. Pensa, R., Besson, J., Boulicaut, J.F.: A methodology for biologically relevant pattern discovery from gene expression data. In: Proceedings DS'04. Volume 3245 of LNAI., Padova (Italy), Springer-Verlag (2004) 230–241
21. Robardet, C., Pensa, R., Besson, J., Boulicaut, J.F.: Using classification and visualization on pattern databases for gene expression data analysis. In: Proceedings PaRMa'04 co-located with EDBT 2004. Volume 96 of CEUR Workshop Proceedings., Heraclion - Crete, Greece (2004)
22. Arbeitman, M., Furlong, E., Imam, F., Johnson, E., Null, B., Baker, B., Krasnow, M., Scott, M., Davis, R., White, K.: Gene expression during the life cycle of drosophila melanogaster. Science **297** (2002) 2270–2275

23. Ashburnerand, M., Ball, C., Blake, J., Botstein, D., et al.: Gene ontology: tool for the unification of biology. the gene ontology consortium. Nature Genetics **25** (2000) 25–29
24. Goethals, B., Zaki, M.: Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations FIMI 2003, Melbourne, USA (2003)
25. Stumme, G., Taouil, R., Bastide, Y., Pasqier, N., Lakhal, L.: Computing iceberg concept lattices with TITANIC. Data & Knowledge Engineering **42** (2002) 189–222
26. Lash, A., Tolstoshev, C., Wagner, L., Schuler, G., Strausberg, R., Riggins, G., Altschul, S.: SAGEmap: A public gene expression resource. Genome Research **10** (2000) 1051–1060
27. Rome, S., Clément, K., Rabasa-Lhoret, R., Loizon, E., Poitou, C., Barsh, G.S., Riou, J.P., Laville, M., Vidal, H.: Microarray profiling of human skeletal muscle reveals that insulin regulates 800 genes during an hyperinsulinemic clamp. Journal of Biological Chemistry (2003) 278(20):18063-8.

Boolean Formulas and Frequent Sets

Jouni K. Seppänen and Heikki Mannila

HIIT Basic Research Unit, Lab. Computer and Information Science,
FI-02015 Helsinki University of Technology, Finland
{Jouni.Seppanen, Heikki.Mannila}@hut.fi

Abstract. We consider the problem of how one can estimate the support of Boolean queries given a collection of frequent itemsets. We describe an algorithm that truncates the inclusion-exclusion sum to include only the frequencies of known itemsets, give a bound for its performance on disjunctions of attributes that is smaller than the previously known bound, and show that this bound is in fact achievable. We also show how to generalize the algorithm to approximate arbitrary Boolean queries.

1 Introduction

Algorithms for mining frequent itemsets continue to be a subject of recent data mining research [GZ03] long after the original publications [AIS93, AMS+96]. Less attention has been received by the question of how one can utilize the frequent itemsets one has mined. The original motivation was provided by association rules, but we claim that the collection of frequent itemsets is good for much more than rule mining: they give us a picture of the joint distribution of the data, and can therefore be used to approximately evaluate Boolean queries over the original data.

The simple idea of approximating exponentially long inclusion-exclusion sums using a small collection of frequent itemsets was considered in [MT96]. Thus the frequent sets can be seen as a condensed representation of the data. In this paper we give a more thorough presentation of the issues involved. We sharpen Theorem 5 in [MT96], which shows that the approximation error for a disjunctive query is bounded by $2^{b-2}/b$, where b is the size of the negative border. Here we prove a bound of $\binom{b}{\lceil b/2 \rceil}/b$, and give a family of examples for which this bound is reached. Our main contribution is the generalization of the discussion to arbitrary Boolean formulas.

Related work includes using maximum entropy to approximate the joint distribution [PMS00, PS01] and linear programming to find upper and lower bounds for queries [BSH04]. These approaches share the problem that they require exponential space in the number of attributes involved. There has also been much work on reducing the size of the itemset collection, such as free-sets [BBR00] and non-derivable sets [CG02]. However, most such work concentrates on algorithms for discovering itemsets, not on using the itemsets obtained to evaluate queries.

The rest of this paper is structured as follows. We start in Section 2 from the almost trivial case of estimating conjunctive queries, introducing notation and

J.-F. Boulicaut et al. (Eds.): Constraint-Based Mining, LNAI 3848, pp. 348–361, 2005.

showing the general idea of the results that follow. In Section 3 we discuss the much more interesting case of disjunctive queries, and in Section 4 we generalize our results to arbitrary Boolean queries.

2 Conjunction of Attributes

The goal of this section is to introduce our notation and the basic task in a simple setting. Throughout the paper, we will denote by U a set of *attributes* or *items*, and by r a *binary relation* over U: that is, r is a multiset of *tuples* $T \subset U$. *Itemsets* are arbitrary subsets of U, denoted by X or Y. We will denote by $g(X)$ the fraction of tuples in r that are equal to X, and call this quantity the *exact frequency* of X. By contrast, the *frequency* $f(X)$ is the fraction of tuples that contain all attributes in X. Thus

$$f(X) = \sum_{Y \supset X} g(X).$$

An itemset is called *frequent* if its frequency is at least a predefined threshold σ. The collection of σ-frequent itemsets is denoted $\mathcal{F}_\sigma$. As is well known, this collection is *downward closed*: given itemsets $X \subset Y \in \mathcal{F}_\sigma$, we have $X \in \mathcal{F}_\sigma$. We will in general denote by $\mathcal{F}$ an arbitrary downward closed collection of itemsets.

The task that we are interested in is estimating the result of a *Boolean query* φ. For now, we let φ be simply a conjunction of attributes: let $C \subset U$ be any itemset, and let

$$\varphi = \bigwedge C = \bigwedge_{A \in C} A.$$

We say that a tuple $T \in r$ *supports* φ, denoted $T \models \varphi$, if T includes every attribute in the query. The *frequency* of the query $f(\varphi)$ is the fraction of tuples in r that support φ. In the case of conjunctions, one sees immediately that a tuple T supports φ if and only if $C \subset T$, and thus the frequency of the query φ is equivalent to the frequency of the itemset C.

We can now present the task Approximate Query$(\mathcal{F}_\sigma, \varphi)$: given the collection $\mathcal{F}_\sigma$ of frequent itemsets, and a Boolean query φ, find an estimate $\hat{f}(\varphi)$ that should be close to $f(\varphi)$. Any solution to this task will be evaluated on its worst-case accuracy, i.e., how far from the actual frequency it can be. The measure of accuracy that we will use is the maximum absolute value of the error $e(\varphi) = f(\varphi) - \hat{f}(\varphi)$.

It is of course easy to come up with a simple and fairly accurate solution, which we name Truncate Sum (for reasons that will become clear later): $\hat{f}(\varphi) = f(C)$ if $C \in \mathcal{F}_\sigma$, otherwise $\hat{f}(\varphi) = 0$. Again, C is the set of attributes in the conjunction φ. If $C \in \mathcal{F}_\sigma$, we have $\hat{f}(\varphi) = f(\varphi)$ and therefore $e(\varphi) = 0$. Otherwise, we know that $0 \leq f(C) < \sigma$, which implies that $|e(\varphi)| \leq \sigma$. Thus our bound for the maximum absolute error is σ. We have shown the following result:

Proposition 1. *For a conjunction of attributes* φ, Truncate Sum *yields results to* Approximate Query *that have maximal absolute error* σ.

3 Disjunction of Attributes

In this section we consider the case where φ is a disjunction of attributes. The main results are that the worst-case error of Truncate Sum can be bounded by an expression that depends exponentially on the size of the negative border (which will be defined later), and that this worst-case bound cannot be decreased at all: that is, there are collections of frequent sets where the bound holds with equality.

We now investigate queries of the form

$$\varphi = \bigvee D = \bigvee_{A \in D} A.$$

A tuple $T \in r$ supports the query if at least one of the attributes in D appears in T: in logical notation, $T \models \varphi$ if $D \cap T \neq \emptyset$. The frequency of φ is, again, the fraction of tuples that support φ. A basic result in combinatorics is that this frequency can be obtained by the inclusion-exclusion principle:

$$f(\varphi) = \sum_{X \subseteq D} [X \neq \emptyset] \, (-1)^{|X|+1} f(X). \tag{1}$$

Here, and in the sequel, we avoid long sum conditions in subscripts by using the "Iverson notation"

$$[P] = \begin{cases} 1, & P \text{ is true,} \\ 0, & P \text{ is false,} \end{cases}$$

popularized by Knuth [Knu92].

The inclusion-exclusion principle is fine if we know the frequency of every itemset X that is a subset of D. If we do not, our approach is to compute the sum over all itemsets whose frequencies we do know:

$$\hat{f}(\varphi) = \sum_{X \subseteq D} [\emptyset \neq X \in \mathcal{F}_\sigma] \, (-1)^{|X|+1} f(X). \tag{2}$$

The exponentially long sum (1) is truncated to the terms (2) that we know; thus the name Truncate Sum. The error made in the approximation is

$$e(\varphi) = \sum_{X \subseteq D} [X \in \mathcal{G}] \, (-1)^{|X|+1} f(X),$$

where by $\mathcal{G}$ we denote the family of non-frequent sets, i.e., the complement of $\mathcal{F}_\sigma$.

An intuition for this estimate is provided by the well-known Bonferroni inequalities, which state that if our collection $\mathcal{F}_\sigma$ happens to contain exactly the itemsets of size at most k, then the error is bounded by the sum of frequencies of itemsets of size $k+1$. [GS96] The proof is simple, although not entirely trivial: the error consists of exponentially many terms, which happen to mostly cancel out. We would like to prove analogues of the Bonferroni inequalities for our more general case.

We start from a simple upper bound that is not very interesting in itself but will be used in the proof of Theorem 1. Recall that $\mathcal{G}$ is the complement of $\mathcal{F}_\sigma$.

Lemma 1. *For a disjunction of attributes* $\varphi = \bigvee D$,

$$|e(\varphi)| \leq \sum_{X \in \mathcal{G}} \binom{|X|}{\lceil |X|/2 \rceil} g(X).$$

Proof. We write the frequency as a sum over tuples: $f(X) = |r|^{-1} \sum_{T \in r} [T \supset X]$, and therefore

$$e(\varphi) = |r|^{-1} \sum_{T \in r} \sum_{X \subset T} [X \in \mathcal{G}] \, (-1)^{|X|+1}.$$

One way of proving the Bonferroni inequalities is based on pairing up most of the tuples in $\mathcal{G}$; we proceed similarly.

We first introduce some notation: $t = |T|$, and $t' = \lceil t/2 \rceil$. It is well known that the power set $\mathcal{P}(T)$ can be written as a union of $\binom{t}{t'}$ disjoint *chains*, where a chain means a collection C of sets where given any two sets $X, Y \in C$, either $X \subset Y$ or $Y \subset X$. [Bol88, Theorem 1 of Section 4] The construction of Bollobás yields chains that are symmetric and consist of consecutive sets: if we write $C = \{ X_1, X_2, \ldots, X_k \}$ with $X_1 \subset X_2 \subset \ldots X_k$, then $|X_1| + |X_k| = d$ and $|X_{j+1}| = |X_j| + 1$ for all $1 \leq j < k$. Thus, if d is *odd*, each chain C is of even length, and the alternating sum $\sum_{X \in C} (-1)^{|X|+1}$ is zero. If d is *even*, the chains from the construction are of odd length. However, we can remove one attribute A from T, perform the construction on $T \setminus \{ A \}$ to obtain a collection of $\binom{d-1}{d'-1}$ chains, and then add to the collection a duplicate of each chain with A added to every set: the result is a partition of $\mathcal{P}(T)$ into $2\binom{t-1}{t'-1} = \binom{t}{t'}$ chains, each of which consists of an even number of consecutive sets.

We can thus assume that there is a partition $T = C_1 \cup C_2 \cup \cdots \cup C_m$ of T into $m = \binom{t}{t'}$ disjoint chains, such that $\sum_{X \in C_j} (-1)^{|X|+1} = 0$ for each chain C_j. Now

$$\sum_{X \subset T} [X \in \mathcal{G}] \, (-1)^{|X|+1} = \sum_{j=1}^{m} \sum_{X \in C_j} [X \in \mathcal{G}] \, (-1)^{|X|+1}.$$

Every chain C_j that is wholly contained in either $\mathcal{F}_\sigma$ or $\mathcal{G}$ contributes 0 to this sum. Every other chain contributes either 0 or ± 1. Therefore,

$$\left| \sum_{X \subset T} [X \in \mathcal{G}] \, (-1)^{|X|+1} \right| \leq m = \binom{t}{t'}.$$

The claim follows by observing that $g(X) = |r|^{-1} \sum_{T \in r} [T = X]$. □

Recall that the Bonferroni inequalities, which apply to the case where $\mathcal{F}_\sigma$ consists of all itemsets of size at most k, give an error bound related to the itemsets of size $k+1$. An analogue of the size $k+1$ itemsets that is both intuitively appealing and practically useful is the *negative border* Bd^- [MT96], defined as the family of minimal non-frequent sets:

$$Bd^- = \{ X \in \mathcal{G} \mid Y \in \mathcal{F}_\sigma \, \forall Y \subsetneq X \}.$$

Note that if $\mathcal{F}_\sigma$ consists of sets of size at most k, then Bd^- is exactly the family of sets of size $k+1$. The practical usefulness stems from the famous Apriori algorithm, which computes the family $\mathcal{F}_\sigma$ and finds Bd^- as a byproduct of its stopping condition [AMS+96].

We will prove a lemma connecting the negative border to the error $e(\varphi)$. First, we need to define some more notation: $\mathcal{G}_D$ is the set of non-frequent subsets of D,

$$\mathcal{G}_D = \{\, X \mid X \in \mathcal{G}, X \subset D \,\},$$

and the *negative border relative to* D, denoted Bd^-_D, consists of the minimal sets in $\mathcal{G}_D$. Note that $Bd^-_D \subset Bd^-$, since if X is minimal in $\mathcal{G}_D$, all subsets of X are in $\mathcal{F}_\sigma$, and therefore X is minimal also in $\mathcal{G}$. We will also need the concept of *exact frequency of* X *relative to* D, defined for $X \subset D$ as the fraction of tuples whose intersection with D is X, which we can write as

$$g_D(X) = f\Big(\bigwedge_{A \in X} A \wedge \bigwedge_{A \in D \setminus X} \neg A\Big).$$

Lemma 2. *Consider the query* $\varphi = \bigvee D$. *If* $Bd^-_D \neq \{\, \emptyset \,\}$, *the algorithm* Truncate Sum *has an error of*

$$e(\varphi) = \sum_{\emptyset \neq \mathcal{E} \subset Bd^-_D} (-1)^{|\mathcal{E}|+|\bigcup \mathcal{E}|} g_D\big(\bigcup \mathcal{E}\big). \tag{3}$$

To illustrate the lemma, consider some simple examples. If Bd^-_D consists of a single set $B \neq \emptyset$, the error is an inclusion-exclusion sum

$$e(\varphi) = \sum_X [B \subset X \subset D]\, (-1)^{|X|+1} f(X),$$

which is of course exactly the expression for $(-1)^{|B|+1} g_D(B)$.

Likewise, if Bd^- is the two-set family $\{\, B_1, B_2 \,\}$ with $B_j \cap D \neq \emptyset$ for $j = 1, 2$, we obtain

$$e(\varphi) = \sum_X [B_1 \subset X \subset D \text{ or } B_2 \subset X \subset D]\, (-1)^{|X|+1} f(X).$$

We can use inclusion-exclusion to decompose the condition on X:

$$\begin{aligned}[B_1 \subset X \subset D \text{ or } B_2 \subset X \subset D] \\ \qquad = [B_1 \subset X \subset D] + [B_2 \subset X \subset D] - [B_1 \cup B_2 \subset X \subset D]\end{aligned}$$

Thus we can break the formula for $e(\varphi)$ into three components, which sum up to $g_D(B_1)$, $g_D(B_2)$ and $-g_D(B_1 \cup B_2)$. The proof of the lemma is a straightforward extension of this idea.

Proof of Lemma 2. The error is given by

$$e(\varphi) = \sum_X [X \in \mathcal{G}_D]\,(-1)^{|X|+1} f(X). \tag{4}$$

We can rewrite the condition $X \in \mathcal{G}_D$ in terms of the minimal sets in $\mathcal{G}_D$ as follows: We have $X \in \mathcal{G}_D$ if and only if $X \supset B$ for some $B \in Bd_D^-$. We apply inclusion-exclusion on the Iverson function:

$$\begin{aligned}[X \in \mathcal{G}_D] &= \sum_{B \in Bd_D^-} [B \subset X \subset D] - \sum_{B_1, B_2 \in Bd_D^-} [B_1 \cup B_2 \subset X \subset D] + \cdots \\ &= \sum_{\emptyset \neq \mathcal{E} \subset Bd_D^-} (-1)^{|\mathcal{E}|+1} \left[\bigcup \mathcal{E} \subset X \subset D\right].\end{aligned}$$

Plugging this in the error sum (4) and changing the order of summation, we obtain

$$e(\varphi) = \sum_{\emptyset \neq \mathcal{E} \subset Bd_D^-} (-1)^{|\mathcal{E}|+1} \sum_X \left[\bigcup \mathcal{E} \subset X \subset D\right] (-1)^{|X|+1} f(X).$$

It now suffices to show that for $Y \subset D$,

$$(-1)^{|Y|} g_D(Y) = \sum_X [Y \subset X \subset D]\,(-1)^{|X|} f(X),$$

for then letting $Y = \bigcup \mathcal{E}$ yields (3). This is an easy exercise in inclusion-exclusion: given a tuple $T \in r$, write $T = R \cup S$ with $R \subset D$, $S \subset U \setminus D$. The tuple will contribute $(-1)^{|R|}$ to all terms corresponding to $X \subset R \cup S$. In the case $R = Y$, the contribution is exactly $(-1)^{|Y|}$; otherwise, the contributions cancel out. □

Based on the lemma, we can prove an analogue to the Bonferroni inequalities that gives, however, rather larger bounds than the Bonferroni case.

Theorem 1. *For a disjunction of attributes $\varphi = \bigvee D$, the absolute error $|e(\varphi)|$ of* Truncate Sum *is bounded by*

$$\binom{|Bd_D^-|}{\lceil |Bd_D^-|/2 \rceil} |Bd_D^-|^{-1} \sum_{X \in Bd_D^-} f(X).$$

Proof. Arrange the sum (3) in the form

$$e(\varphi) = \sum_{X \in \mathcal{G}_D} \nu(X) g_D(X).$$

For the coefficients $\nu(X)$ we have

$$\nu(X) = (-1)^{|X|} \sum_{\mathcal{E} \subset Bd_X^-} \left[\bigcup \mathcal{E} = X\right] (-1)^{|\mathcal{E}|}.$$

In this sum, the condition $\left[\bigcup \mathcal{E} = X\right]$ defines an upwards-closed subfamily of the powerset of Bd_X^-. We know from Lemma 1 that the absolute value of this alternating sum is bounded by $\binom{m}{m'}$ with $m = |Bd_X^-|$ and $m' = \lceil m/2 \rceil$.

Arrange also the sum $\sum_{X \in Bd_D^-} f(X)$ in the form

$$\sum_{X \in \mathcal{G}_D} \mu(X) g_D(X).$$

We have for the coefficients $\mu(X)$

$$\mu(X) = \sum_{Y \in Bd_D^-} [Y \subset X] = |Bd_X^-| > 0$$

for all $X \in \mathcal{G}_D$. The ratio $|\nu(X)|/\mu(X)$ is bounded by $\binom{m}{m'}/m$, and this bound is largest for $X = D$. Thus

$$\begin{aligned} |e(\varphi)| &\leq \sum_{X \in \mathcal{G}_D} |\nu(X)| g_D(X) \leq \sum_{X \in \mathcal{G}_D} \left(\max \frac{|\nu(X)|}{\mu(X)} \right) \mu(X) g_D(X) \\ &\leq \binom{|Bd_D^-|}{\lceil |Bd_D^-|/2 \rceil} |Bd_D^-|^{-1} \sum_{X \in Bd_D^-} f(X). \end{aligned} \tag{5}$$

□

Using the inequality $f(X) < \sigma$ for $X \in Bd^-$, we can obtain a form of the bound that is independent of the actual frequencies of sets in the border.

Corollary 1. *For a disjunction of attributes $\varphi = \bigvee D$,*

$$e(\varphi) \leq \binom{|Bd^-|}{\lceil |Bd^-|/2 \rceil} \sigma.$$

Thus, the bound depends superpolynomially on the size of the border. A natural question is whether the bound can be decreased. We will next show that the answer is negative: the bound is in the worst case tight. The example will have a small negative border. The key part in the proof is constructing the family $\mathcal{F}_\sigma$ so that when Lemma 1 is used in the proof of Theorem 1, equality holds. This is the case when the minimal families $\mathcal{E} \subset Bd_X^-$ that satisfy the condition $\bigcup \mathcal{E} = X$ are exactly of size $\lceil |Bd_X^-|/2 \rceil$.

Theorem 2. *There exists a set U, a relation r over U, and a downward-closed collection of itemsets $\mathcal{F}$ such that for the disjunctive query $\varphi = \bigvee U$ the absolute error of* Truncate Sum *is*

$$|e(\varphi)| = \binom{|Bd^-|}{\lceil |Bd^-|/2 \rceil} |Bd^-|^{-1} \sum_{X \in Bd^-} f(X).$$

Proof. Choose integer parameters $p > k > 1$; p will be the number of sets in the negative border, and we will see later that choosing $p = 2k + 1$ suits our purposes well. We will need $n = \binom{p}{k}$ attributes: let $U = [n] = \{1, \ldots, n\}$. We will set up Bd^- so that for all families $\mathcal{E} \subset Bd^-$, $|\mathcal{E}| \leq k$ implies $\bigcup \mathcal{E} \neq U$, and $|\mathcal{E}| > k$ implies $\bigcup \mathcal{E} = U$. To achieve this, we first enumerate all the k-element subsets of $[p]$; there are n of them, and we will name them $K_1, K_2, \ldots, K_n$ in any arbitrary order. Then for all $q \in [p]$, we define W_q as the set of those i such that $q \notin K_i$. Let $Bd^- = \{W_q \mid q \in [p]\}$. Note that Bd^- is an antichain, since all sets W_q have the same number of elements; thus we can define $\mathcal{F}$ as the downward-closed collection of sets that are not supersets of any sets in Bd^-, and Bd^- will automatically be the negative border corresponding to $\mathcal{F}$.

We must now prove the assertion that for $\mathcal{E} \subset Bd^-$, $\bigcup \mathcal{E} = U$ if and only if $|\mathcal{E}| > k$. Given any collection $\mathcal{E}$ of border sets, we can write $\mathcal{E} = \{W_q \mid q \in Q\}$ for some index set $Q \subset [p]$. If $|\mathcal{E}| = |Q| \leq k$, some set K_i must be a superset of the index set Q, since we have enumerated all k-element subsets of $[p]$. But then we have that $i \notin \bigcup \mathcal{E}$, and thus $\bigcup \mathcal{E} \neq U$. Conversely, if $\bigcup \mathcal{E} \neq U$, there must be some $i \notin \bigcup \mathcal{E}$, and therefore for all $q \in Q$ we must have $q \in K_i$, because $i \notin W_q$. But this means that $Q \subset K_i$, and therefore $|\mathcal{E}| = |Q| \leq |K_i| = k$. We have thus shown that $\bigcup \mathcal{E} = U$ if and only if $|\mathcal{E}| > k$.

We will let $\varphi = \bigvee U$ over all the attributes. Thus, the terms $g_D(X)$ will be the usual exact frequencies $g(X)$ and the family Bd_D^- will be the usual negative border Bd^-. We will also let $g(U) = 1$ and $g(X) = 0$ for all $X \notin \mathcal{F}$, $X \neq U$. We can let $g(X)$ be some sufficiently high number for all $X \in \mathcal{F}$ so that $\mathcal{F} = \mathcal{F}_\sigma$ for some σ.

Now we are in a position to apply Lemma 2. The sum over $\mathcal{E} \subset Bd^-$ becomes a sum over those $\mathcal{E}$ for which $\bigcup \mathcal{E} = U$, since $g(\bigcup \mathcal{E}) = 0$ otherwise. By the construction, these are exactly those $\mathcal{E}$ such that $|\mathcal{E}| > k$. Thus

$$e(\varphi) = \sum_{\mathcal{E} \subset Bd^-} (-1)^{|\mathcal{E}|+|r|} [|\mathcal{E}| > k] = (-1)^{|r|} \sum_{j=k+1}^{n} (-1)^j \binom{p}{j}.$$

It is an easy proof by induction that

$$\sum_{j=k+1}^{n} (-1)^j \binom{p}{j} = (-1)^{k+1} \binom{p-1}{k}.$$

If we now let $p = 2k + 1$, we have

$$|e(\varphi)| = \binom{2k}{k} = \binom{|Bd^-|}{|Bd^-|/2}.$$

Since we have $f(X) = 1$ for all $X \in Bd^-$, the frequency sum of sets in the border is $\sum_X [X \in Bd^-] f(X) = |Bd^-| = |Bd_D^-|$. This completes the proof. □

While the construction creates a small number of sets in the border, there are of course many sets that are "almost" in the border, which is not true in the usual Bonferroni situation. The following theorem is another analogue of the Bonferroni inequalities.

Theorem 3. *Define the* thick negative border Bd^-_* *as the family of itemsets that are not frequent but that have at least one frequent subset. Then*

$$e(\varphi) \leq \sum_{X \in Bd^-_*} f(X).$$

Proof. Again, we will write $e(\varphi)$ as a sum over all tuples $T \in r$. We will show that the contribution made by T toward $e(\varphi)$ is bounded by the number of sets in Bd^-_* that include T, which implies the claimed upper bound.

First of all, if $T \in \mathcal{F}_\sigma$, the contribution is zero. Otherwise, the contribution is

$$\sum_{X \subset T} [X \notin \mathcal{F}_\sigma] \, (-1)^{|X|+1}. \tag{6}$$

Select any attribute $A \in T$, and delete from the sum (6) all pairs $X, Y \notin \mathcal{F}_\sigma$ such that $Y = X \cup \{ A \}$. What we have left is

$$\sum_{X \subset T} [X \notin \mathcal{F}_\sigma] \, [X \setminus \{ A \} \in \mathcal{F}_\sigma] \, (-1)^{|X|+1}.$$

All sets fulfilling both conditions of the sum are in $Bd^-_* \cap \mathcal{P}(T)$, and thus the absolute value of the contribution is bounded by $|Bd^-_* \cap \mathcal{P}(T)|$. Summing these inequalities for all contributions yields

$$|e(\varphi)| \leq \sum_{T \in R} |Bd^-_* \cap \mathcal{P}(T)| \leq \sum_{T \in R} |Bd^-_*| = \sum_{X \in Bd^-_*} f(X). \qquad \square$$

We have proved two theorems for upper-bounding the absolute error: Theorems 1 and 3. Both theorems are problematic in practice: the bound of Theorem 1 grows exponentially, and the thick border of Theorem 3 can be very large. It would be useful to find a bound for Truncate Sum in-between these two theorems. Note that the construction of Theorem 2 creates a large number of maximal frequent sets. By analogy with the negative border, one can define the *positive border* Bd^+ as the collection of these sets. For the construction, Bd^+ is large and Bd^- is small; in many practical cases, Bd^+ is smaller and Bd^- larger. The set $Bd^+ \cup Bd^-$ is worth investigating, and we conjecture (again [Man02]) that

$$e(\varphi) \leq \sum_X [X \in Bd^- \cup Bd^+] \, f(X).$$

4 General Queries

In this section we will generalize the preceding discussion: we will define Truncate Sum for arbitrary Boolean formulas φ, and prove counterparts of Lemma 2 and Theorem 1. The bounds provided by these results can be even larger than the disjunction-specific bounds of the previous section.

Let now φ be an arbitrary Boolean formula, i.e., an expression consisting of negation $\neg$, conjunction $\wedge$, disjunction $\vee$ and attributes $A \in U$. We define the semantics of such formulas in the usual way: $T \models \varphi$ if φ is true when the attributes are substituted by their values in T. The goal remains the same: to approximate $f(\varphi)$, the fraction of tuples supporting φ, given the collection $\mathcal{F}_\sigma$ of σ-frequent itemsets.

The support of the query formula can obviously be written as

$$f(\varphi) = \sum_X [X \models \varphi]\, g(X).$$

We denote the coefficients $\zeta(X) = [X \models \varphi]$. What we want to do is write

$$f(\varphi) = \sum_X \xi(X) f(X)$$

with suitable new coefficients $\xi(X)$, and then truncate the sum, obtaining

$$\hat{f}(\varphi) = \sum_X [X \in \mathcal{F}_\sigma]\, \xi(X) f(X).$$

To compute the new coefficients, we can use inclusion-exclusion: since

$$g(X) = \sum_Y [Y \supset X]\, (-1)^{|Y \setminus X|} f(Y),$$

we have

$$\begin{aligned} f(\varphi) &= \sum_X \zeta(X) g(X) = \sum_X \sum_Y \zeta(X)\, [Y \supset X]\, (-1)^{|Y \setminus X|} f(Y) \\ &= \sum_Y f(Y) \sum_X \zeta(X)\, [Y \supset X]\, (-1)^{|Y \setminus X|}. \end{aligned}$$

The required coefficients are thus given by

$$\xi(Y) = \sum_X [X \subset Y]\, (-1)^{|Y \setminus X|} \zeta(X).$$

Next we prove a generalization of Lemma 2.

Lemma 3. *When φ is an arbitrary Boolean formula with exact-frequency coefficients $\zeta(X) = [X \models \varphi]$ and the border Bd^- does not contain the empty set,*

$$e(\varphi) = \sum_X \nu(X) g(X),$$

where

$$\nu(X) = (-1)^{|X|} \sum_{\emptyset \neq \mathcal{E} \subset Bd_X^-} (-1)^{|\mathcal{E}|+1} \sum_Y \left[X \setminus \bigcup \mathcal{E} \subset Y \subset X\right] (-1)^{|Y|} \zeta(Y).$$

Proof. The error is

$$\begin{aligned} e(\varphi) &= \sum_X [X \in \mathcal{G}]\, \xi(X) f(X) \\ &= \sum_{X,Y} [X \in \mathcal{G}]\, f(X)\, [Y \subset X]\, (-1)^{|X \setminus Y|} \zeta(Y). \end{aligned}$$

Again we apply inclusion-exclusion on the condition $X \in \mathcal{G}$:

$$[X \in \mathcal{G}] = \sum_{\emptyset \neq \mathcal{E} \subset Bd^-} (-1)^{|\mathcal{E}|+1} \left[X \supset \bigcup \mathcal{E}\right],$$

obtaining

$$\begin{aligned} e(\varphi) &= \sum_{\emptyset \neq \mathcal{E} \subset Bd^-} (-1)^{|\mathcal{E}|+1} \sum_{X,Y} \left[X \supset \bigcup \mathcal{E}\right] f(X)\, [Y \subset X]\, (-1)^{|X \setminus Y|} \zeta(Y) \\ &= \sum_{\emptyset \neq \mathcal{E} \subset Bd^-} (-1)^{|\mathcal{E}|+1} \sum_Y (-1)^{|Y|} \zeta(Y) \sum_X \left[X \supset \bigcup \mathcal{E} \cup Y\right] (-1)^{|X|} f(X) \\ &= \sum_{\emptyset \neq \mathcal{E} \subset Bd^-} (-1)^{|\mathcal{E}|+1} \sum_Y (-1)^{|Y| + |\bigcup \mathcal{E} \cup Y|} \zeta(Y) g\big(\bigcup \mathcal{E} \cup Y\big). \end{aligned}$$

Regrouping the terms yields

$$e(\varphi) = \sum_X g(X) (-1)^{|X|} \sum_{\emptyset \neq \mathcal{E} \subset Bd_X^-} (-1)^{|\mathcal{E}|+1} \sum_Y \left[X \setminus \bigcup \mathcal{E} \subset Y \subset X\right] (-1)^{|Y|} \zeta(Y),$$

which is the claim. □

To see that this generalizes Lemma 2, let $\varphi = \bigvee D$. Then $\zeta(X) = [X \cap D \neq \emptyset]$. Consider the sum over Y:

$$\sum_Y \left[X \setminus \bigcup \mathcal{E} \subset Y \subset X\right] (-1)^{|Y|} [Y \cap D \neq \emptyset]. \tag{7}$$

We may assume that $\bigcup \mathcal{E} \subset X$, since the outer sum is taken over $\mathcal{E} \subset Bd_X^-$. Furthermore, if $\bigcup \mathcal{E}$ contains any attribute A that is not in D, we can pair up terms corresponding to $Y \ni A$ and $Y \setminus \{A\}$, and thus show that the sum (7) is 0. On the other hand, if X contains any attributes that are in D but not in $\bigcup \mathcal{E}$, the Iverson function $[Y \cap D \neq \emptyset]$ is always 1, and since $\bigcup \mathcal{E} \neq \emptyset$, and the sum (7) is seen to compute the difference in number of even and odd subsets of $\bigcup \mathcal{E}$, which is of course 0.

Assume now that $X = \bigcup \mathcal{E} \cup Z$ with $\bigcup \mathcal{E} \subset D$ and $Z \cap D = \emptyset$. We thus have for $Z \subset Y \subset X$ that $[Y \cap D \neq \emptyset] = [Y \neq Z]$, and the sum (7) becomes $-(-1)^{|Z|} = (-1)^{|X \setminus \bigcup \mathcal{E}|+1} = (-1)^{|X|+|\bigcup \mathcal{E}|+1}$, since $\bigcup \mathcal{E} \subset X$.

We have shown for all X that

$$\sum_{\emptyset \neq \mathcal{E} \subset Bd_X^-} (-1)^{|\mathcal{E}|+1} \sum_Y \left[X \setminus \bigcup \mathcal{E} \subset Y \subset X\right] (-1)^{|Y|} \zeta(Y)$$
$$= \sum_{\emptyset \neq \mathcal{E} \subset Bd_X^-} \left[\bigcup \mathcal{E} \subset D\right] \left[(X \setminus \bigcup \mathcal{E}) \cap D = \emptyset\right] (-1)^{|\mathcal{E}|+|\bigcup \mathcal{E}|}.$$

The result of Lemma 2 follows by noting that for $X \subset D$

$$g_D(X) = \sum_Y [Y \cap D = \emptyset]\, g(X \cup Y)$$

and rearranging terms.

The coefficients $\nu(X)$ used in the statement of the lemma have already played a role in proving Theorem 1: the key part was showing that $|\nu(X)| \leq 2^{|Bd_X^-|}$ for disjunctions φ. A natural question then is, how large can $|\nu(X)|$ be for general queries? To answer this question, we rearrange the sum as

$$\nu(X) = (-1)^{|X|} \sum_Y [Y \subset X]\, (-1)^{|Y|} \zeta(Y) \sum_{\emptyset \neq \mathcal{E} \subset Bd_X^-} (-1)^{|\mathcal{E}|+1} \left[X \setminus \bigcup \mathcal{E} \subset Y\right]. \quad (8)$$

Denote by S the innermost sum. We can rewrite it in the form

$$S = \sum_{\emptyset \neq \mathcal{E} \subset Bd_X^-} \left[X \setminus Y \subset \bigcup \mathcal{E}\right] (-1)^{|\mathcal{E}|+1},$$

which is seen to be an inclusion-exclusion sum over the upwards-closed subfamily

$$\left\{ \mathcal{E} \subset Bd_X^- \;\middle|\; \bigcup \mathcal{E} \supset X \setminus Y \right\} \quad (9)$$

of the powerset of Bd_X^-. Applying Lemma 1 to this sum, we have for $|S|$ an upper bound of $\binom{p}{\lceil p/2 \rceil}$, where $p = |Bd_X^-|$. Combining this with the fact that $\zeta(Y)$ is always 0 or 1, we obtain

$$|\nu(X)| \leq 2^{|X|-1} \binom{|Bd_X^-|}{\lceil |Bd_X^-|/2 \rceil}.$$

We thus have the following analogue of Theorem 1.

Theorem 4. *For an arbitrary query φ, the absolute error $|e(\varphi)|$ of* Truncate Sum *is bounded by*

$$2^{|U|-1} \binom{|Bd^-|}{\lceil |Bd^-|/2 \rceil} |Bd^-|^{-1} \sum_{X \in Bd^-} f(X).$$

The bound in the general case is even larger than the one in the disjunction case. How close to the bound can we come? Consider the sum (8). The form of the alternating sum over Y suggests that a parity-like function would be a difficult case: if $\zeta(Y) = 1$ if and only if $|Y|$ is even, the sum becomes

$$\nu(X) = (-1)^{|X|} \sum_{Y \subset X} \big[|Y| \text{ even}\big]\, S,$$

where S is the inclusion-exclusion sum mentioned in the proof of Theorem 4. The bound for $|S|$ used Lemma 1, where it is easy to see that equality holds if the upwards-closed family (9) consists of those sets $\mathcal{E} \subset Bd_X^-$ that have $|\mathcal{E}| = \lceil |Bd_X^-|/2 \rceil$. But for $Y = \emptyset$ exactly this is achieved by the construction in the proof of Theorem 2. For larger sets $Y \subset X$, S is smaller; however, this suffices to show that if the statement of Theorem 4 is to be strengthened, one cannot simply decrease the general bound for $|S|$, but more careful analysis of the double sum (8) is required.

5 Conclusion and Future Work

We have described the Approximate Query problem and analyzed the Truncate Sum algorithm, expanding upon the foundations in [MT96]. The results are disappointing in a sense: for the simple-looking query class of disjunctions of attributes, the behavior is not even polynomial in the size of the border. However, this is a worst-case situation that may not be very realistic in practical, sparse datasets. In the proof of Theorem 1, the key inequality (5) is based upon bounding the ratio $|\nu(X)|/\mu(X)$. However, the ratio is multiplied by the quantity $g_D(X)$, the exact frequency of X when the data is projected to the attributes in D, and in sparse data it is reasonable that this quantity should vanish for most large itemsets X. This observation suggests a modified algorithm: when mining the frequent itemsets, remove from the data those tuples where the ratio would be large, and store them separately; if the data is sparse, there should not be too many of these tuples. Queries can be computed exactly for the difficult, dense tuples, and approximated for the easy part of the data condensed into the frequent itemset representation.

More generally, assume that there is space for storing some extra information along with the frequent itemsets. The question then is, what is a good class of information to store in order to approximate a wide variety of queries?

Another avenue for future research is to use the information inherent in frequent itemsets in some way other than truncating the inclusion-exclusion sum. In the Bonferroni case, Linial and Nisan have shown that if the frequencies are known for sets X with $|X| \geq \Omega(\sqrt{|D|})$, there are good approximations to $f(\bigvee D)$ using multipliers other than ± 1, and if the frequencies are known only for sets X with $|X| \leq O(\sqrt{|D|})$, no approximation can be very good [LN90]. It would be interesting to extend this approach to the general case of frequent itemsets that do not form such a level family, and to queries more general than disjunctions.

References

[AIS93] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In *SIGMOD '93*, pages 207–216, 1993.

[AMS+96] Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo. Fast discovery of association rules. In Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, chapter 12, pages 307–328. AAAI Press, 1996.

[BBR00] Jean-François Boulicaut, Artur Bykowski, and Christophe Rigotti. Approximation of frequency queries by means of free-sets. In *PKDD '00*, volume 1910 of *LNCS*, pages 75–85. Springer, 2000.

[Bol88] Béla Bollobás. *Combinatorics: set systems, hypergraphs, families of vectors and combinatorial probability*. U Cambridge, 1988.

[BSH04] Artur Bykowski, Jouni K. Seppänen, and Jaakko Hollmén. Model-independent bounding of the supports of Boolean formulae in binary data. In Rosa Meo, Pier Luca Lanzi, and Mika Klemettinen, editors, *Database Support for Data Mining Applications: Discovering Knowledge with Inductive Queries*, volume 2682 of *LNAI*, pages 234–249. Springer, 2004.

[CG02] Toon Calders and Bart Goethals. Mining all non-derivable frequent itemsets. In *PKDD '02*, volume 2431 of *LNAI*, pages 74–85. Springer, 2002.

[GS96] Janos Galambos and Italo Simonelli. *Bonferroni-type Inequalities with Applications*. Probability and its Applications. Springer, 1996.

[GZ03] Bart Goethals and Mohammed J. Zaki, editors. *Proceedings of the Workshop on Frequent Itemset Mining Implementations (FIMI–03)*, volume 90 of *CEUR-WS*, Melbourne, Florida, 2003. http://CEUR-WS.org/Vol-90/.

[Knu92] Donald E. Knuth. Two notes on notation. *Am. Math. Monthly*, 99(5):403–422, 1992.

[LN90] Nathan Linial and Noam Nisan. Approximate inclusion-exclusion. *Combinatorica*, 10(4):349–365, 1990.

[Man02] Heikki Mannila. Local and global methods in data mining: Basic techniques and open problems. In P. Widmayer, F. Triguero, R. Morales, M. Hennessy, S. Eidenbenz, and R. Conejo, editors, *Automata, Languages and Programming*, volume 2380 of *LNCS*, pages 57–68. Springer, 2002.

[MT96] Heikki Mannila and Hannu Toivonen. Multiple uses of frequent sets and condensed representations. In *KDD '96*, pages 189–194, Portland, Oregon, August 1996. AAAI Press.

[PMS00] Dmitry Pavlov, Heikki Mannila, and Padhraic Smyth. Probabilistic models for query approximation with large sparse binary datasets. In *UAI*, 2000.

[PS01] Dmitry Pavlov and Padhraic Smyth. Probabilistic query models for transaction data. In *KDD '01*, 2001.

Generic Pattern Mining Via Data Mining Template Library*

Mohammed J. Zaki, Nilanjana De, Feng Gao, Paolo Palmerini**, Nagender Parimi, Jeevan Pathuri, Benjarath Phoophakdee, and Joe Urban

Computer Science Department, Rensselaer Polytechnic Institute, Troy NY 12180

Abstract. Frequent Pattern Mining (FPM) is a very powerful paradigm for mining informative and useful patterns in massive, complex datasets. In this paper we propose the Data Mining Template Library, a collection of generic containers and algorithms for data mining, as well as persistency and database management classes. DMTL provides a systematic solution to a whole class of common FPM tasks like itemset, sequence, tree and graph mining. DMTL is extensible, scalable, and high-performance for rapid response on massive datasets. A detailed set of experiments show that DMTL is competitive with special purpose algorithms designed for a particular pattern type, especially as database sizes increase.

1 Introduction

Frequent Pattern Mining (FPM) is a very powerful paradigm which encompasses an entire class of data mining tasks, namely those dealing with extracting informative and useful patterns in massive datasets representing complex interactions between diverse entities from a variety of sources. These interactions may also span multiple-scales, as well as spatial and temporal dimensions. FPM is ideally suited for categorical datasets, which include text/hypertext data (e.g., news articles, web pages), semistructured and XML data, event or log data (e.g., network logs, web logs), biological sequences (e.g. DNA/RNA, proteins), transactional datasets, and so on. FPM techniques are able to extract patterns embedded in different subspaces within very high dimensional, massive datasets. FPM is very well suited to selecting or constructing good features in complex data and also for building global classification models of the datasets [26].

The specific tasks encompassed by FPM include the mining of increasingly complex and informative patterns, in complex structured and unstructured relational datasets, such as: Itemsets or co-occurrences [1] (transactional, unordered data), Sequences [2,24] (temporal or positional data, as in text mining, bioinformatics), Tree patterns [25,3] (XML/semistructured data), and Graph pat-

* This work was supported by NSF Grant EIA-0103708 under the KD-D program, NSF CAREER Award IIS-0092978, and DOE Early Career PI Award DE-FG02-02ER25538.

** The work was done while Paolo was at RPI.

J.-F. Boulicaut et al. (Eds.): Constraint-Based Mining, LNAI 3848, pp. 362–379, 2005.

terns [10,13,21,22] (complex relational data, bioinformatics). Figure 1 shows examples of these different types of patterns; in a generic sense a pattern denotes links/relationships between several objects of interest. The objects are denoted as nodes, and the links as edges. Patterns can have multiple labels, denoting various attributes, on both the nodes and edges.

The current practice in frequent pattern mining basically falls into the paradigm of incremental algorithm improvement and solutions to very specific problems. While there exist tools like MLC++ [12], which provides a collection of algorithms for classification, and Weka [20], which is a general purpose Java library of different data mining algorithms including itemset mining, these systems do not have an unifying theme or framework, there is little database support, and scalability to massive datasets is questionable. Moreover, these tools are not designed for handling complex pattern types like trees and graphs.

Our work seeks to address all of the above limitations. In this paper we describe Data Mining Template Library (DMTL), a generic collection of algorithms and persistent data structures, which follows a generic programming paradigm [4]. DMTL provides a systematic solution for the whole class of pattern mining tasks in massive, relational datasets. The main contributions of DMTL are as follows:

- The design and implementation of generic data structures and algorithms to handle various pattern types like itemsets, sequences, trees and graphs.
- Design and implementation of generic data mining algorithms for FPM, such as depth-first and breadth-first search.
- Persistent data structures for supporting efficient pattern frequency computations using a tightly coupled database (DBMS) approach.
- Native support for both a vertical and horizontal database formats for highly efficient mining.
- DMTL's support for pre-processing steps like data mapping and discretization of continuous attributes and creation of taxonomies. etc.

One of the main attractions of a generic paradigm is that the generic algorithms for mining are guaranteed to work for **any** pattern type. Each pattern has a list of properties it satisfies, and the generic algorithm can utilize these properties to speed up the mining. We conduct a detailed set of experiments to show the scalability and efficiency of DMTL for different pattern types like itemsets, sequences, trees and graphs. Our results indicate that DMTL is competitive with the special purpose algorithms designed for a particular pattern type, especially with increasing database sizes.

1.1 Related Work

Previous research in integrating mining and databases has mainly looked at SQL support. DMQL [8] is a mining query language to support common mining tasks. MSQL [9] is an extension of SQL to generate and selectively retrieve sets of rules from a large database. The MINE RULE SQL operator [15] and Query

flocks [18] extend the semantics of association rules, allowing more generalized queries to be performed. A comprehensive study of several architectural alternatives for database and mining integration were studied in [16]. This body of work is complementary to ours, since these SQL operators can be used as a front end to DMTL. Also, DMTL is optimized for the class of frequent patterns.

There has been limited work in integrating other mining tasks with databases. A middleware for classification was proposed in [5]; it decomposes and schedules classification primitives over a back-end SQL database. Two generic SQL operations called count-by-group (for class histograms) and compute-tuple-distances (for point distances) were identified in [6] for classification and clustering tasks, respectively.

2 Preliminaries

The problem of mining frequent patterns can be stated as follows: Let $\mathcal{N} = \{x_1, x_2, \ldots, x_{n_v}\}$ be a set of n_v distinct nodes or vertices. A pair of nodes (x_i, x_j) is called en edge. Let $\mathcal{L} = \{l_1, l_2, \ldots, l_{n_l}\}$, be a set of n_l distinct labels. Let $L_n : \mathcal{N} \rightarrow \mathcal{L}$, be a node labeling function that maps a node to its label $L_n(x_i) = l_j$, and let $L_e : \mathcal{N} \times cE \rightarrow \mathcal{L}$ be an edge labeling function, that maps an edge to its label $L_e(x_i, x_j) = l_k$.

A *pattern* P is simply a relation on $\mathcal{N}$, $P \subseteq \mathcal{N} \times \mathcal{N}$, that is $P = \{(x_i, x_j) \mid x_i, x_j \in \mathcal{N}\}$, such that P satisfies some user-specified conditions $\mathcal{C}$ (i.e., $\mathcal{C}(P)$ is true). It is also intuitive to represent a pattern P as a graph (P_V, P_E), with labeled vertex set $P_V \subset \mathcal{N}$ and labeled edge set $P_E = \{(x_i, x_j) \mid x_i, x_j \in P_V\}$. The number of nodes in a pattern P is called its *size*. A pattern of size k is called a k-pattern. In some applications P is a symmetric relation, i.e., $(x_i, x_j) = (x_j, x_i)$ (unordered edges), while in other applications P is anti-symmetric, i.e., $(x_i, x_j) \neq (x_j, x_i)$ (ordered edges). A path in P is a set of distinct nodes $\{x_{i_0}, x_{i_1}, x_{i_n}\}$, such that $(x_{i_j}, x_{i_{j+1}})$ in an edge in P_E for all $j = 0 \cdots n-1$. The number of edges gives the length of the path. If x_i and x_j are connected by a path of length n we denote it as $x_i <_n x_j$. Thus the edge (x_i, x_j) can also be written as $x_i <_0 x_j$.

Given two patterns P and Q, we say that P is a *subpattern* of Q (or Q is a super-pattern of P), denoted $P \preceq Q$ if and only if there exists a 1-1 mapping f from nodes in P to nodes in Q, such that for all $x_i, x_j \in P_V$: i) $L_n(x_i) = L_n(f(x_i))$, ii) $L_e(x_i, x_j) = L_e(f(x_i), f(X_j))$, and iii) $(x_i, x_j) \in P_V$ iff (if and only if) $(f(x_i), f(x_j)) \in Q_V$. In some cases we are interested in embedded subpatterns. P is an *embedded subpattern* of Q if: i) $L_n(x_i) = L_n(f(x_i))$, iii) $L_e(x_i, x_j) = L_e(f(x_i), f(X_j))$, and iii) $(x_i, x_j) \in P_V$ iff $f(x_i) <_l f(x_j)$, i.e., $f(x_i)$ is connected to $f(x_j)$ on some path. If $P \preceq Q$ we say that P is contained in Q or Q contains P.

A database $\mathcal{D}$ is just a collection (a multi-set) of patterns. A database pattern is also called an *object*. Let $\mathcal{O} = \{o_1, o_2, \ldots, o_{n_o}\}$, be a set of n_o distinct *object identifiers (oid)*. An object has a unique identifier, given by the function $O(d_i) = o_j$, where $d_i \in \mathcal{D}$ and $o_j \in \mathcal{O}$. The number of objects in $\mathcal{D}$ is given as $|\mathcal{D}|$.

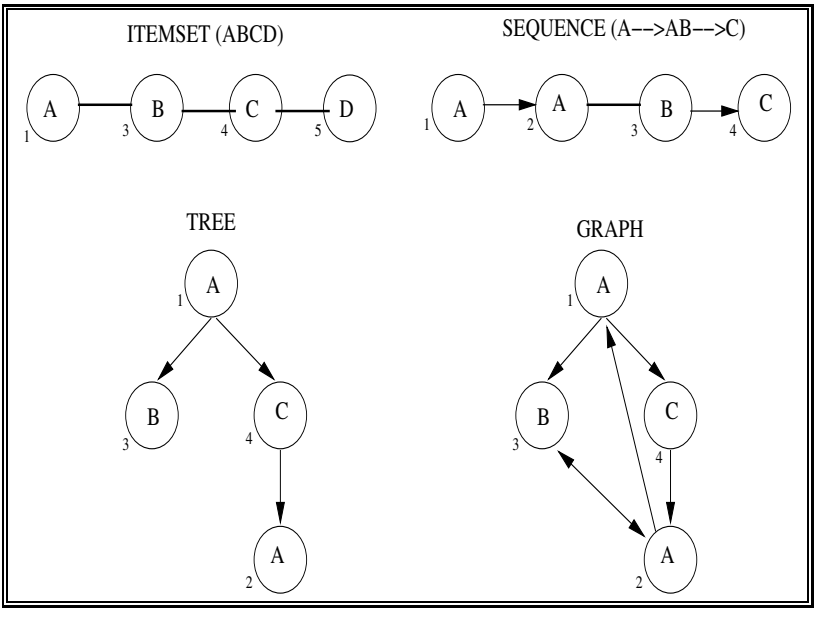


Fig. 1. FPM Instances

The *absolute support* of a pattern P in a database $\mathcal{D}$ is defined as the number of objects in $\mathcal{D}$ that contain P, given as $\pi^a(P, \mathcal{D}) = |\{P \preceq d \mid d \in \mathcal{D}\}|$. The *(relative) support* of P is given as $\pi(P, \mathcal{D}) = \frac{\pi^a(P,\mathcal{D})}{|D|}$. A pattern is *frequent* if its support is more than some user-specified minimum threshold, i.e., if $\pi(P, \mathcal{D}) \geq \pi^{\min}$. A frequent pattern is *maximal* if it is not a subpattern of any other frequent pattern. A frequent pattern is *closed* if it has no super-pattern with the same support. The frequent pattern mining problem is to enumerate all the patterns that satisfy the user-specified $\pi^{\min}$ frequency requirement (and any other user-specified conditions).

The main observation in FPM is that the sub-pattern relation $\preceq$ defines a partial order on the set of patterns. If $P \preceq Q$, we say that P is more general than Q, or Q is more specific than P. The second observation used is that if Q is a frequent pattern, then all sub-patterns $P \preceq Q$ are also frequent. The different FPM algorithms differ in the manner in with they search the pattern space.

2.1 FPM Instances

Some common types of patterns include itemsets, sequences, trees, and graphs, as shown in Figure 1. In fact, every pattern can be modeled as a graph; the nodes (x_i) are shown under each circle and the node labels ($L_n(x_i)$) are shown inside the circle, whereas edge labels have been omitted.

In an itemset [1] no two nodes have the same label. Let $V = \{x_1, x_2, \cdots x_k\}$ be a node set such that $L_n(x_i) \neq L_n(x_j)$ for all $x_i, x_j \in V$, and $L_n(x_i) < L_n(X_{i+1}$

for all $1 \leq i \leq k-1$. There are several possible formulation of the itemset pattern: i) *vertex-only*: An itemset pattern P is just a of vertices, i.e., $P_V = V$ and $P_E = \emptyset$, ii) *linear*: Figure 1 shows another formulation, where the itemset is defined as $P_V = V$, and $P_E = \{(x_i, x_{i+1}) | x_i, x_{i+1} \in P_V\}$, iii) *clique*: A third alternative is to represent itemset P as a clique, i.e., $P_V = V$ and $P_E = \{(x_i, x_j) \mid i < j \text{ and } x_i, x_j \in P_V\}$.

In sequence mining [2], a sequence is modeled as an ordered list of itemsets, and thus the different nodes in a sequence can have the same label. We can model a sequence pattern P as being made up of a sequence of n itemsets P^i, $i = 1, \cdots n$, using the linear formulation (as shown in Figure 1); note that using the vertex-only formulation is problematic, since it results in a disconnected pattern. Thus P has a vertex set made up of n disjoint subsets $P_V = \bigcup_{i=1}^{n} P_V^i$. The edge set P contains all the edges within P^i (consecutive and undirected), and P_E contains, a directed edge for every pair of consecutive itemsets, i.e., from the last node of P^i to the first node of P^{i+1}.

In tree mining [25,3], typically rooted, ordered and labeled trees are considered. Thus a tree pattern P consists of the vertex set $P_V = \{r, x_1, x_2, \cdots\}$, where r is a special node called root. A tree pattern must satisfy all tree properties, namely i) the root has no parent, i.e., $(x_i, r) \notin P_E$ for any $x_i \in P_V$, ii) the edges are directed, i.e., if $(x_i, x_j) \in P_E$, then $(x_j, x_i) \notin P_E$), iii) a node has only one parent, i.e., if $(x_i, x_j) \in P_E$, then $(x_k, x_j) \notin P_E$ for any $x_k \neq x_i$, iv) the tree is connected, i.e., for all $x_i \in P_V$, there exists a path from the root r to x_i, and v) tree has no cycles. Furthermore for ordered trees the order of a nodes' children matters. This means that there is an ordering of edges in P_E, such that (x_i, x_j) comes before (x_i, x_k) in P_E only if x_j is before x_k in the ordering of x_i's children.

Finally, by definition a pattern can model any general graph, as well as any special constraints that might appear in graph mining [10,13,21], such as connected graphs, or induced subgraphs. It is also possible to model other patterns such as DAGs (directed acyclic graphs).

3 DMTL: Data Structures and Algorithms

The C++ Standard Template Library (STL) provides efficient, generic implementations of widely used algorithms and data structures, which tremendously aid effective programming. Like STL, DMTL is a collection of generic data mining algorithms and data structures. In addition, DMTL provides persistent data and index structures for efficiently mining any type of pattern or model of interest. The user can mine custom pattern types, by simply defining the new pattern types, but there is no need to implement a new algorithm, since any generic DMTL algorithm can be used to mine them. Since the mined models and patterns are persistent and indexed, this means the mining can be done efficiently over massive databases, and mined results can be retrieved later from the persistent store.

Following the ideology of generic programming, DMTL provides a standardized, general, and efficient implementation of frequent pattern mining tasks by

isolating the concept of data structures or containers, as they are called in generic programming, from algorithms. DMTL provides container classes for representing different patterns (such as itemsets and sequences) and collection of patterns, containers for database objects (horizontal and vertical), and containers for temporary mining results. These container classes support persistency when required.

Generic algorithms, on the other hand are independent of the container and can be applied on any valid container. These include algorithms for performing intersections of the vertical lists [23,24,25] for itemsets or sequences or other patterns. Generic algorithms are also provided for mining itemsets and sequences [1,17,23,24], as well as for finding the maximal or closed patterns [7,27]. Finally DMTL provides support for the database management functionality, preprocessing support for mapping data in different formats to DMTL's native formats, as well as for data transformation (such as discretization of continuous values).

In this section we focus on the containers and algorithms for mining. In later sections we discuss the database support in DMTL as well as support for preprocessing and post-processing.

Figure 2 shows the different DMTL container classes for PMT and the relationship among them. At the lowest level set the different kinds of pattern-types one might be interested in mining (such as itemsets, sequences, and several variants). A pattern is uses the base pattern-type classes to provide a generic container. There are several pattern family types (such as pvector, plist, etc.)

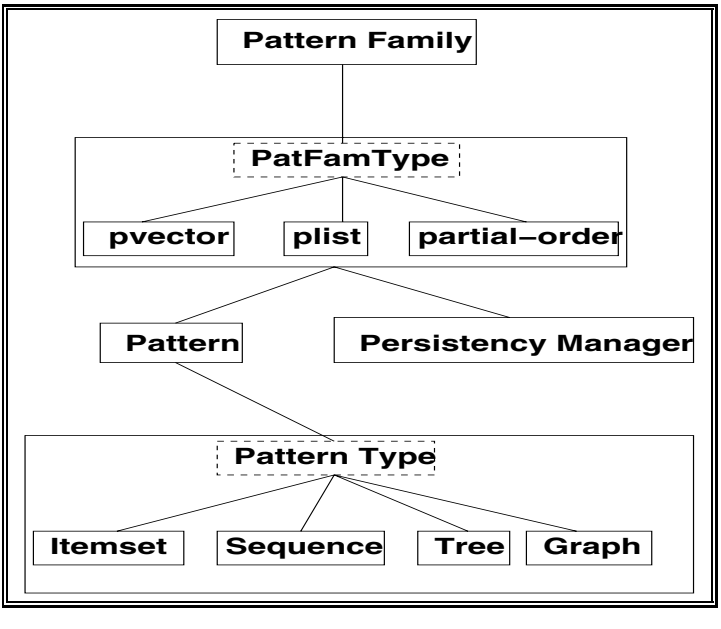


Fig. 2. DMTL Container Hierarchy

which together with a persistency manager class make up different pattern family classes. More details on each class appears below.

3.1 Pattern

In DMTL a pattern is a generic container, which can be instantiated as an itemset, sequence, tree or a graph, specified as `Pattern<class P>` by means of a template argument called pattern-type (`P`). A generic pattern is simply a pattern-type whose frequency we need to determine in a larger collection or database of patterns of the same type.

3.2 Pattern Type

This allows users to select the type of pattern they want to mine, and as long as certain operations are defined on the pattern-type all the generic algorithms provided by DMTL can be used. The main source of flexibility of PMT is that developers can easily define new types of patterns to suit their needs and once the operations are defined on them all the generic algorithms of DMTL can be used on the new pattern types. For example, an itemset can be defined as pattern-type `vector<int>`, denoting a set of items (`int` in this case), A sequence pattern-type can defined as `list<vector<int>>`, denoting an ordered list of itemsets. If we want to include a time field along with the different itemsets in a sequence, we can define a new sequence type as follows `list<pair<time, vector<int>>>`, i.e., a list of (`time,vector<int>`) pairs, where time is a user-defined type to note when each event occurs. All algorithms are guaranteed to work with **any** pattern type.

3.3 Pattern Family

In addition to the basic pattern classes, most pattern mining algorithms operate on a collection of patterns. The pattern family is a generic container `PatternFamily<class PatFamType>` to store groups of patterns, specified by the template parameter `PatFamType`. `PatFamType` represents some persistent class provided by DMTL, that provides seamless access to the members, whether they be in memory or on disk. All access to patterns of a family is through the iterator provided by the `PatFamType` class. `PatternFamily` provides generic operations to add and remove patterns to/from the family, to find the maximal or closed patterns in the family, as well as a `count()` function that finds the support of all patterns, in the database, using functions provided by the database class.

3.4 Pattern Family Type

DMTL provides several persistent classes to store groups of patterns. Each such class is templatized on the pattern-type (P) and a persistency manager class `PM`. An example is `pvector<class P, class PM>`, a persistent vector class. It has

the same semantics as a STL vector with added memory management and persistency. Thus a pattern family for itemsets can be defined in terms of the pvector class as follows: `PatternFamily<pvector<Itemset, PM>>`. Another class is `plist<P,PM>`. Instead of organizing the patterns in a linear structure like a vector or list, another persistent family type DMTL class, `partial-order<P,PM>`, organizes the patterns according to the sub-pattern/super-pattern relationship. While pvector and partial-order provide the same interface, certain operations will be more efficient in one class than the other. For example, inserts and deletions are cheaper for `plists`, while the maximality and closed testing functions will be cheaper for partial-orders, since the patterns are already organized according to sub/super-pattern relation.

3.5 Persistent Containers

An important aspect of DMTL is to provide a user-specified level of persistency for all DMTL classes. To support large-scale data mining, DMTL provides automatic support for out-of-core computations, i.e., memory buffer management, via the persistency manager class PM. The PatternFamilyType class uses the persistency manager (PM) to support the buffer management for patterns. The details of implementation are hidden from PatternFamily; all generic algorithms continue to work regardless of whether the family is (partially) in memory or on disk. The implementation of a persistent container (like pvector) is similar to the implementation of a volatile container (like STL vector); the difference being that instead of pointers one has to use offsets and instead of allocating memory using `new` one has to request it from the persistency manager class. More details on the persistency manager will be given later.

We saw above that PatternFamily uses the `count()` function to find the support of all patterns in the family, in the database; at the end of `count()` all patterns have their support field set to their frequency in the database. DMTL provides *native* support for both the horizontal and vertical database formats. The generic `count()` algorithm does not impose any restriction on the type of database used, i.e., whether it is horizontal or vertical. The `count()` function uses the interface provided by the DB class, passed as a parameter to `count()`, to get pattern supports. More details on the DB class and its functions will be given later.

3.6 Generic Mining Algorithms

The pattern mining task can be viewed as a search over the pattern space looking for those patterns that match the minimum support constraint. For instance in itemset mining, the search space is the set of all possible subsets of items. Various search strategies are possible leading to several popular variants of the mining algorithms. DMTL provides generic algorithms encapsulating these search strategies; by their definition these algorithms can work on any type of pattern: Itemset, Sequence, Tree or Graph. An example is the generic algorithm `DFS-Mine<class PatFamType> (PatternFamily<PatFamType> &pf,`

`DB &db, ...)` , which mines the frequent patterns using a depth-first search (DFS) [23,24]. The generic DFS mining algorithm takes in a pattern family and the database. The types of patterns and persistency manager are specified by the pattern family type. The DFS algorithm in turn relies on other generic subroutines for creating equivalence classes, for generating candidates, and for support counting. There is also a generic `BFS-Mine` that performs Breadth-First Search [1,17] over the pattern space.

4 DMTL: Persistency and Database Support

DMTL is the back-end server that actually provides the persistency, and indexing support for both the patterns and the database. DMTL supports DMTL by seamlessly providing support for memory management, data layout, high-performance I/O, as well as tight integration with database management systems (DBMS). It supports multiple back-end storage schemes including flat files, embedded databases, and relational or object-relational DBMS. DMTL also provides persistent pattern management facilities, i.e., mined patterns can themselves be stored in a pattern database for retrieval and interactive exploration.

DMTL provides native database support for both the horizontal [1] and vertical [23,24,25] data formats. In the horizontal approach, each object has an oid along with the itemset comprising the object. Thus object with $oid = 1$ is the set $\{A, C, T, W\}$. In contrast, the vertical format maintains for each label (and itemset) its oid list, a set of all oids where it occurs. For example, the label A appears in oids 1, 3, 4, and 5. Thus its oid list is given as 1345 (omitting set notation).

It is also worth noting that since in many cases the database contains the same kind of objects as the patterns to be extracted (i.e., the database can be viewed as a pattern family), the same database functionality used for horizontal format can be used for providing persistency for pattern families. It is relatively straightforward to store a horizontal format object, and by extension, a family of such patterns, in any object-relational database. Thus the persistency manager for pattern families can handle both the original database and the patterns that are generated while mining. DMTL provides the required buffer management so that the algorithms continue to work regardless of whether the database/patterns are in memory or on disk.

4.1 Vertical Attribute Tables

To provide native database support for objects in the vertical format, DMTL adopts a fine grained data model, where records are stored as *Vertical Attribute Tables* (VATs). Given a database of objects, where each object is characterized by a set of properties or attributes, a VAT is essentially the collection of objects that share the same values for the attributes. For example, for a relational table, `cars`, with the two attributes, `color` and `brand`, a VAT for the property `color=red` stores all the transaction identifiers of cars whose color is red. The main advantage of VATs is that they allow for optimizations of query intensive

applications like data mining where only a subset of the attributes need to be processed during each query. As was mentioned earlier these kinds of vertical representations have proved to be useful in many data mining tasks [23,24,25].

In DMTL there is typically one VAT per pattern. A VAT is an entity composed of a *body*, which contains the list of object identifiers in which a given pattern occurs. For storing database sequences a VAT needs, in addition, a `time` field for each occurrence of the pattern. For tree and graph patterns the body type is different. A VAT is defined as `VAT<class V>`, where `V` is a vat-type class.

Depending on the pattern type being mined the vat-type class may be different. For instance for itemset mining it suffices to keep only the object identifiers where a given itemset appears. In this case the vat-type is simply an `int` (assuming that oid is an integer). On the other hand for sequence mining one needs not only the oid, but also the time stamp for the last AV pair in the sequence. For sequences the vat-type is then `pair<int, time>`, i.e., a pair of an `int`, denoting the oid, a nd `time`, denoting the time-stamp. Different vat-types must also provide operations like equality testing (for itemsets and sequences), and less-than testing (for sequences; a oid-time pair is less then another if they have the same oid, and the first one happens before the second).

Given the generic setup of a VAT, DMTL defines a generic algorithm to join/intersect two VATs. For instance in vertical itemset mining, the support for an itemset is found by intersection the VATs of its lexicographic first two subsets. A generic intersection operation utilizes the equality operation defined on the vat-type to find the intersection of any two VATs. On the other hand in vertical sequence mining the support of a new candidate sequence is found by a temporal join on the VATs, which in turn uses the less-than operator defined by the vat-type. Since the itemset vat-type typically will not provide a less-than operator, if the DMTL developer tries to use temporal intersection on itemset vat-type it will generate a *compile time* error! This kind of concept-checking support provided by DMTL is extremely useful in catching library misuses at compile-time rather than at run-time.

DMTL provides support for creating VATs during the mining process, i.e., during algorithms execution, as well as support for updating VATs (add and delete operations). In DMTL VATs can be either persistent or non-persistent. Finally DMTL uses indexes for a collection of VATs for efficient retrieval based on a given attribute-value, or a given pattern.

4.2 Storage and Persistency Manager

The database support for VATs and for the horizontal family of patterns is provided by DMTL in terms of the following classes, which are illustrated in Figure 3.

`Vat-type`**:** A class describing the vat-type that composes the body of a VAT, for instance `int` for itemsets and `pair<int,time>` for sequences.

`VAT<class V>`**:** The class that represents VATs. This class is composed of a collection of records of vat-type V.

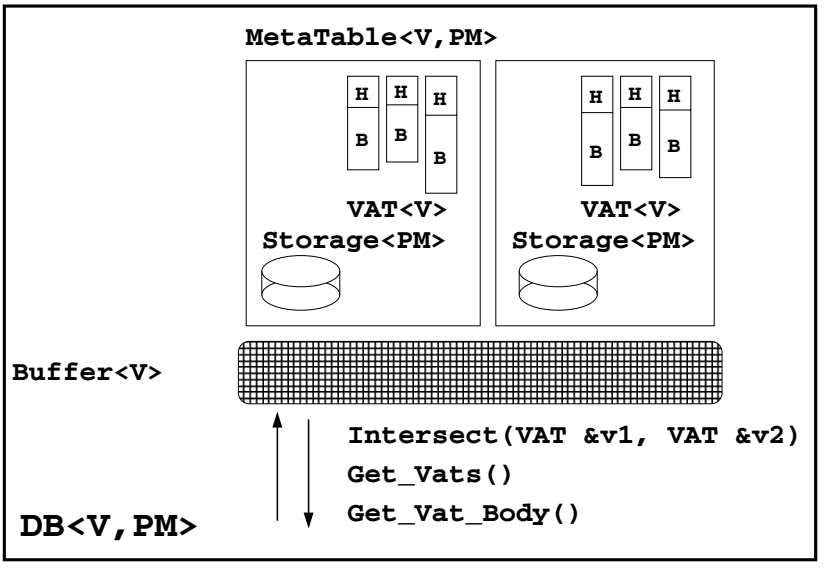


Fig. 3. DMTL: High level overview of the different classes used for Persistency

`Storage<class PM>`**:** The generic persistency-manager class that implements the physical persistency for VATs and other classes. The class PM provides the actual implementations of the generic operations required by Storage. For example, PM_metakit and PM_gigabase are two actual implementations of the `Storage` class in terms of different DBMS like Metakit [19], a persistent C++ library that natively supports the vertical format, and Gigabase [11], an object-relational database. Other implementations can easily be added as long as they provide the required functionality.

`MetaTable<class V, class PM>`**:** This class represents a collection of VATs. It stores a list of VAT pointers and the adequate data structures to handle efficient search for a specific VAT in the collection. It also provides physical storage for VATs. It is templatized on the vat-type V and on the `Storage` implementation PM.

`DB<class V, class PM>`**:** The database class which holds a collection of `Metatables`. This is the main user interface to VATs and constitutes the database class DB referred to in previous sections. It supports VAT operations such as intersection, as well as the operations for data import and export. The double template follows the same format as that of the `Metatable` class.

`Buffer<class V>`**:** A fixed-size main-memory buffer to which VATs are written and from which VATs are accessed, used for buffer management to provide seamless support for main-memory and out-of-core VATs (of type V).

A diagram of the class interaction is displayed in Figure 3. As previously stated, the `DB` class is the main DMTL interface to VATs and the persistency manager

for patterns. It has as data members an object of type `Buffer<V>` and a collection of `MetaTables<V,PM>`.

The `Buffer<V>` class is composed of a fixed size buffer which will contain as many VAT bodies. When a VAT body is requested from the `DB` class, the buffer is searched first. If the body is not already present there, it is retrieved from disk, by accessing the Metatable containing the requested VAT. If there is not enough space to store the new VAT in the buffer, the buffer manager will (transparently) replace an existing VAT with the new one. A similar interface is used to provide access to patterns in a persistent family or the horizontal database.

The `MetaTable` class stores all the pointers to the different `VAT` objects. It provides the mapping between the patterns, called header, and their VATs, called the body, via a hashed based indexing scheme. In the figure the H refers to a pattern and B its corresponding VAT. The `Storage` class provides for efficient lookup of a particular `VAT` object given the header.

4.3 VAT Persistency

VATs can be in one of three possible states of persistence:

- volatile: the VAT is fully loaded and available in main memory only.
- buffered: the VAT is handled as if it were in main memory, but it is actually kept on disk in an out-of-core fashion.
- persistent: the VAT is disk resident and can be retrieved after the program execution, i.e.: the VAT is inserted in the VATdatabase.

Volatile VATs are created and handled by directly accessing the `VAT` class members. Buffered VATs are managed from the `DB` class through `Buffer` functions. Buffered VATs must be inserted into the file associated with a `Metatable`, but when a buffered VATis no longer needed, its space on disk can be freed. A method for removing a VAT from disk is provided in the `DB` class. If such method is not called, then the VAT will be persistent, i.e., it will remain in the metatable and in the storage associated with it after execution.

4.4 Buffer Management

The `Buffer` class provides methods to access and to manage a fixed size buffer where the most recently used VATs/patterns are stored for fast retrieval. The idea behind the buffer management implemented in the `Buffer` class is illustrated in Figure 4.

A fixed size buffer is available as a linear block of memory of objects of type V. Records are inserted and retrieved from the buffer as linear chunks of memory. To start, the buffer is empty. When a new object is inserted, some data structures are initialized in order to keep track of where every object is placed so it can be accessed later. Objects are inserted one after the other in a round-robin fashion. When there is no more space left in the buffer, the least recently used (LRU) block (corresponding to one entire VAT body, or a pattern) is removed. While the current implementation provides a LRU buffering strategy, as part of future work we will consider more sophisticated buffer replacement strategies that closely tie with the mining.

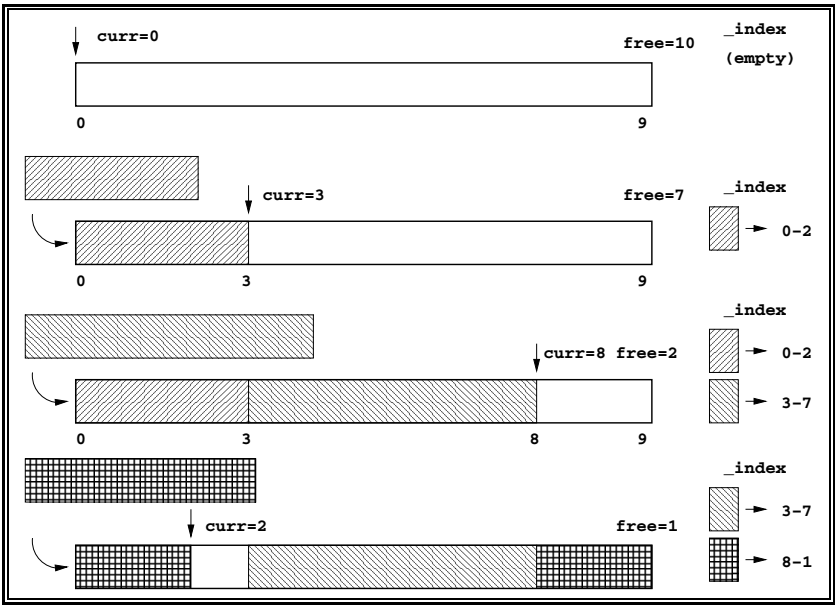


Fig. 4. Buffer Management: The buffer manager keeps track of the current position in the buffer (curr), the amount of free space (free), and an index to keep track of where an object is. As new objects are inserted these are updated and when an existing object is replaced when the buffer becomes full.

4.5 Storage

Physical storage of VATs and pattern families can be implemented using different storage systems, such as a DBMS or ad-hoc libraries. In order to abstract the details of the actual system used, all storage-related operations are provided in a generic class, `Storage`. Implementations of the `Storage` class for MetaKit [19] and Gigabase [11] backends are provided in the DMTL. Other implementations can easily be added as long as they provide the required functionality.

The `DB` class is a doubly templated class where both the vat-type and the storage implementation need to be specified. An example instantiation of a `DB` class for itemset patterns would therefore be `DB<int,PM_metakit>` or `DB<int, PM_gigabase>`.

5 DMTL: Pre-processing Support

DMTL provides support for dynamic mapping of data into VATs at run-time over the same base database using a *mapping* class, which transforms original attribute values into *mapped values* according to a set of user specified mapping directives, contained in a configuration file. For every input database there has to be an XML configuration file. For the definition of the syntax of such file, we

follow the approach presented in [14] by Talia *et al.*. The format of such file is the following.

```
<?xml version="1.0"?>
<!DOCTYPE Datasource SYSTEM "dmtl_config.dtd">
<Data model=relational source=ascii_file>
    <Access> [...] </Access>
    <Structure>
       <Format> [...] </Format>
       <Attributes> [...] </Attributes>
    </Structure>
</Data>
```

5.1 Attributes

Configuration used for mapping attribute values are contained in the `<Structure>` section. The `<Format>` section contains the characters used as record separator and field separator. An `<Attribute>` section must be present for each attribute (or column) in the input database. Such section might be something like: `<Attribute name="price" type="continuous" units="Euro" ignore="yes"> [ ... ] </Attribute>`. Possible attributes for the `<Attribute>` tag are: `name`: the name of the attribute, `type`: one of continuous, discrete, categorical, `units`: the unit of measure for values (currency, weight, etc.), `ignore`: should a VAT be created for this attribute or not.

5.2 Mapping

The mapping information is enclosed in the `<Mapping>` section. Mapping can be different for categorical, continuous or discrete fields. For continuous values we can specify a fixed step discretization within a range:

```
<Attribute name="price" type="continuous">
<Mapping min="1.0" max="5.0" step=".5">0
</Mapping> </Attribute>
```

In this case the field `price` will be mapped to `(max-min)/step = (5-1)/.5 = 10` values, labeled with integers starting from `0`. It is also possible to specify non-uniform discretizations, omitting the `step` attribute and explicitly specifying all the ranges and labels. For categorical values we can also specify a mapping, that allows for taxonomies or other groupings.

6 Experiments

DMTL is implemented using C++ Standard Template Library [4]. We present some experimental results on the time taken by DMTL to load databases and to

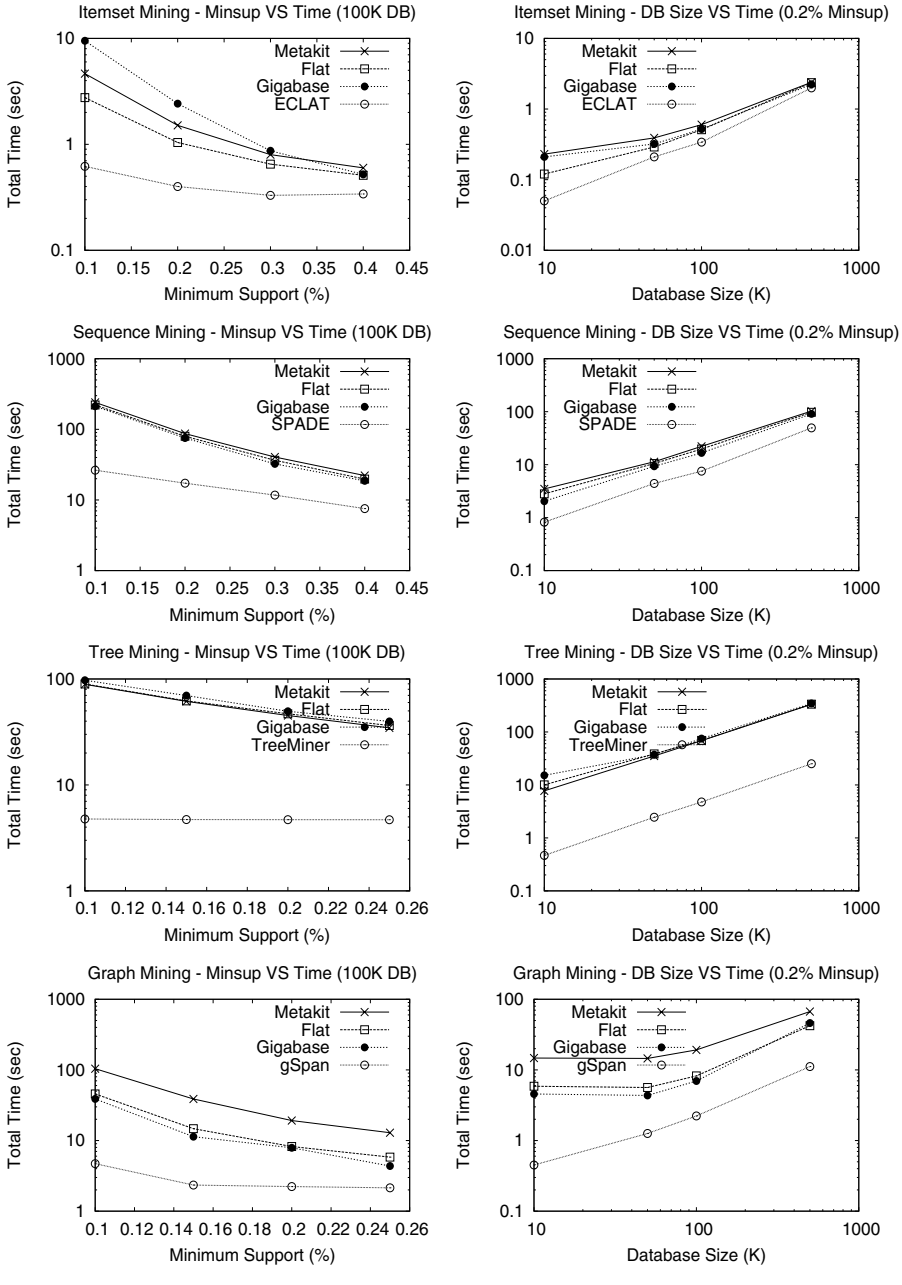


Fig. 5. Itemset, Sequence, Tree and Graph Mining: Effect of Minimum Support and Database Size

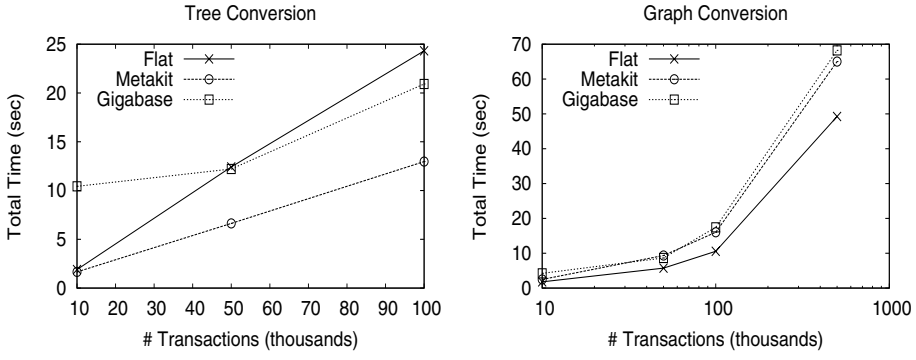


Fig. 6. Database Conversion and Loading Times

perform different types of pattern mining on them. We used the IBM synthetic database generator [1] for itemset and sequence mining, the tree generator from [25] for tree mining and the graph generator by [13], with sizes ranging from $10k$ to $1000k$ (or 1 million). The experiment were run on a Pentium4 2.8Ghz Processor with 6GB of memory, running Linux.

Figure 5 shows the DMTL mining time versus the specialized algorithms for itemset mining (ECLAT [23]), sequences (SPADE [24]), trees (TreeMiner [25]) and graphs (gSpan [21]). For the DMTL algorithms, we show the time with a flat-file (Flat) persistency manager/database, with the metakit backend (Metakit) and the gigabase backend (Gigabase). The left hand column shows the effect of minimum support on the mining time for the various patterns. We find that for all pattern types DMTL is within a factor of 10 of the specialized algorithms even as we decrease the minimum support on a database with 100K records. The column on the right hand size shows the effect of increasing database sizes on these algorithms. We find that as the number of objects increase the gap between DMTL algorithms and the specialized ones starts to decrease. We expect that as we increase the number of records, the specialized algorithms will break down, while DMTL will continue to run since it explicitly manages the memory buffers. Comparing the three backend implementations, we find that the flatfile approach has a slight edge, but the object-oriented gigabase database backend is almost as fast. On the other hand the embedded database Metakit is generally slower.

Figure 6 shows the time taken to convert the input data into VATs. The times are shown for the three different backends (flat, metakit and gigabase) for upto 1 million objects. We find that these three approaches are roughly the same, with the maximum difference being a factor of 2.

7 Conclusions

In this paper we describe the design and implementation of the DMTL prototype for an important subset of FPM tasks, namely mining frequent itemsets,

sequences, trees, and graphs. Following the ideology of generic programming, DMTL provides a standardized, general, and efficient implementation of frequent pattern mining tasks by isolating the concept of data structures or containers, from algorithms. DMTL provides container classes for representing different patterns, collection of patterns, and containers for database objects (horizontal and vertical). Generic algorithms, on the other hand are independent of the container and can be applied on any valid pattern. These include algorithms for performing intersections of the VATs, or for mining.

The generic paradigm of DMTL is a first-of-its-kind in data mining, and we plan to use insights gained to extend DMTL to other common mining tasks like classification, clustering, deviation detection, and so on. Eventually, DMTL will house the tightly-integrated and optimized primitive, generic operations, which serve as the building blocks of more complex mining algorithms. The primitive operations will serve all steps of the mining process, i.e., pre-processing of data, mining algorithms, and post-processing of patterns/models. Finally, we plan to release DMTL as part of open-source, and the feedback we receive will help drive more useful enhancements. We also hope that DMTL will provide a common platform for developing new algorithms, and that it will foster comparison among the multitude of existing algorithms.

References

1. R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. Inkeri Verkamo. Fast discovery of association rules. In U. Fayyad and et al, editors, *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI Press, Menlo Park, CA, 1996.
2. R. Agrawal and R. Srikant. Mining sequential patterns. In *11th Intl. Conf. on Data Engg.*, 1995.
3. T. Asai, K. Abe, S. Kawasoe, H. Arimura, H. Satamoto, and S. Arikawa. Efficient substructure discovery from large semi-structured data. In *2nd SIAM Int'l Conference on Data Mining*, April 2002.
4. M. H. Austern. *Generic Programming and the STL.* Addison Wesley Longman, Inc., 1999.
5. S. Chaudhri, U. Fayyad, and J. Bernhardt. Scalable classification over SQL databases. In *15th IEEE Intl. Conf. on Data Engineering*, March 1999.
6. A. Freitas and S. Lavington. *Mining very large databases with parallel processing.* Kluwer Academic Pub., Boston, MA, 1998.
7. K. Gouda and M. J. Zaki. Efficiently mining maximal frequent itemsets. In *1st IEEE Int'l Conf. on Data Mining*, November 2001.
8. J. Han, Y. Fu, W. Wang, K. Koperski, and O. Zaiane. DMQL: A data mining query language for relational databases. In *1st ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, June 1996.
9. T. Imielinski and A. Virmani. MSQL: A query language for database mining. *Data Mining and Knowledge Discovery: An International Journal*, 3:373–408, 1999.
10. A. Inokuchi, T. Washio, and H. Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In *4th European Conference on Principles of Knowledge Discovery and Data Mining*, September 2000.

11. Konstantin Knizhnik. Gigabase, object-relational database management system. http://sourceforge.net/projects/gigabase.
12. R. Kohavi, D. Sommerfield, and J. Dougherty. Data mining using mlc++, a machine learning library in c++. *International Journal of Artificial Intelligence Tools*, 6(4):537–566, 1997.
13. M. Kuramochi and G. Karypis. Frequent subgraph discovery. In *1st IEEE Int'l Conf. on Data Mining*, November 2001.
14. C. Mastroianni, D. Talia, and P. Trunfio. Managing heterogeneous resources in data mining applications on grids using xml-based metadata. In *Proceedings of The 12th Heterogeneous Computing Workshop*, 2002.
15. R. Meo, G. Psaila, and S. Ceri. A new SQL-like operator for mining association rules. In *22nd Intl. Conf. Very Large Databases*, 1996.
16. S. Sarawagi, S. Thomas, and R. Agrawal. Integrating association rule mining with databases: alternatives and implications. In *ACM SIGMOD Intl. Conf. Management of Data*, June 1998.
17. R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *5th Intl. Conf. Extending Database Technology*, March 1996.
18. D. Tsur, J.D. Ullman, S. Abitboul, C. Clifton, R. Motwani, and S. Nestorov. Query flocks: A generalization of association rule mining. In *ACM SIGMOD Intl. Conf. Management of Data*, June 1998.
19. Jean-Claude Wippler. Metakit. http://www.equi4.com/metakit/.
20. I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann Publishers, 1999.
21. X. Yan and J. Han. gspan: Graph-based substructure pattern mining. In *IEEE Int'l Conf. on Data Mining*, 2002.
22. X. Yan and J. Han. Closegraph: Mining closed frequent graph patterns. In *ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, August 2003.
23. M. J. Zaki. Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, 12(3):372-390, May-June 2000.
24. M. J. Zaki. SPADE: An efficient algorithm for mining frequent sequences. *Machine Learning Journal*, 42(1/2):31–60, Jan/Feb 2001.
25. M. J. Zaki. Efficiently mining frequent trees in a forest. In *8th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, July 2002.
26. M. J. Zaki and C.C. Aggarwal. Xrules: An effective structural classifier for xml data. In *9th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, August 2003.
27. M. J. Zaki and C.-J. Hsiao. CHARM: An efficient algorithm for closed itemset mining. In *2nd SIAM International Conference on Data Mining*, April 2002.

Inductive Querying for Discovering Subgroups and Clusters*

Albrecht Zimmermann and Luc De Raedt

Chair of Machine Learning, Institute of Computer Science,
Albert-Ludwigs-University, Freiburg, Georges-Köhler-Allee 079,
79110 Freiburg, Germany
{azimmerm, deraedt}@informatik.uni-freiburg.de

Abstract. We introduce the problem of **cluster-grouping** and show that it integrates several important data mining tasks, i.e. subgroup discovery, mining correlated patterns and aspects from clustering. The problem of cluster-grouping can be regarded as a new type of inductive optimization query that asks for the k best patterns according to a convex criterion. The algorithm *CG* for solving cluster-grouping problems is presented and the underlying mechanisms are discussed. The approach is experimentally evaluated on a number of real-life data sets. The results indicate that the algorithm improves upon the subgroup discovery algorithm *CN2-WRAcc* and is competitive with the clustering algorithm *CobWeb*.

Keywords: clustering, subgroup discovery, correlated pattern mining, inductive querying, constraint-based mining.

1 Introduction

Many problems and settings are described in the machine learning and data mining literature. The techniques that will be addressed in this paper include: subgroup discovery [1,2], clustering [3,4], and correlated pattern mining [5].

In subgroup discovery, the goal is to find groups (often in the form of conjunctive rules $c_1 \wedge ... \wedge c_n \rightsquigarrow a$) that are statistically over- or under-represented w.r.t. a particular target attribute a and thus, since they show unexpected behaviour, are considered interesting. For instance, the group *smoker* is an interesting subgroup w.r.t. *cancer*, as smokers have a higher probability of having cancer. Correlated pattern mining [5,6] can be viewed as an extension of subgroup discovery where one is looking for *interesting* rules (w.r.t. statistical criteria such as χ^2) but with no fixed target attribute. Subgroup discovery is thus closely related to rule learning, while correlated pattern mining is more similar to association rule discovery. In clustering [3,4], the goal is to compute interesting groups, and in *conceptual* clustering, it is furthermore desired to obtain symbolic descriptions of the clusters.

* A 3-page abstract of this paper appeared as Albrecht Zimmermann, Luc De Raedt: *Cluster-Grouping: From Subgroup Discovery to Clustering.* ECML 2004: 575–577.

J.-F. Boulicaut et al. (Eds.): Constraint-Based Mining, LNAI 3848, pp. 380–399, 2005.

Although these three techniques are in the literature perceived as being quite different, it turns out that they share a number of characteristics. The key contribution of this paper is the introduction of the *cluster-grouping* problem that subsumes subgroup discovery [1,2], clustering [3], and correlated pattern mining [5]. Cluster-grouping is concerned with finding rules $b_1 \wedge ... \wedge b_k \rightsquigarrow h_1 \vee ... \vee h_n$ that score best w.r.t. an interestingness function σ and a data set $\mathcal{E}$. Cluster grouping rules state that for examples covered by the condition part $b_1 \wedge ... \wedge b_n$ it is possible to reliably predict the attributes h_i in the conclusion part. To address the cluster-grouping problem, we develop the branch-and-bound algorithm *CG* that utilizes the convexity of different evaluation functions. It significantly extends the correlated pattern mining framework by Morishita *et al.* [5], in that it generalizes Morishita *et al.*'s approach to an arbitrary dimension. *CG* is experimentally validated against the heuristic *CN2-WRAcc* algorithm [1] for subgroup discovery and the *CobWeb* algorithm [3] for conceptual clustering. The experiments show that *CG* typically finds better (even optimal) subgroups while exploring less candidates than *CN2-WRAcc*, and that *CG* finds cluster definitions in the form of conjunctions, which are competitive in terms of *Category Utility* [7] with the clusters found by *CobWeb*.

The cluster grouping problem can be viewed as a novel type of inductive query [8,9], in which one is interested in the k best rules with regard to the interestingness function σ. This type of query thus looks for the k optimal rules, which explains why we employ a branch-and-bound algorithm. This is an inductive optimization query which differs from the typical type of constraints used, in which one can decide whether a pattern satisfies the constraint independently of the other patterns in the space. Still, cluster-grouping queries are both declarative and powerful as they can be used to find optimal solutions for a wide range of mining tasks, i.e. subgroup discovery, correlated pattern mining as well as clustering.

We proceed as follows. In the next section we introduce the necessary notations used throughout the paper, and define the problem. In section 3, we explain how cluster-grouping unifies the data mining tasks mentioned. In the fourth section, we discuss the upper bound computation which is necessary for efficiently solving the problem and in section 5 present the algorithm developed. We evaluate our approach in section 6 and finally, we touch upon related work, formulate our conclusions and discuss future research directions in sections 7 and 8.

2 Preliminaries

We consider the problem of cluster-grouping as one of learning interesting, i.e. strongly correlating, rules on a given data set.

2.1 Rules

Let $\mathcal{A} = \{A_1, ..., A_d\}$ be a set of ordered attributes and $\mathcal{V}[A] = \{V_1, ..., V_p\}$ the domain of A. An *instance* e is then a tuple $\langle v_1, ..., v_d \rangle$ with $v_i \in \mathcal{V}[A_i]$. A multiset $\mathcal{E} = \{e_1, ..., e_n\}$ is called a *data set*.

Definition 1 ***(Literal).*** *A* ***literal*** l *is an attribute-value-pair* $A = v$ *with* $v \in \mathcal{V}[A]$*. An instance* $\langle v_1, ..., v_d \rangle$ *is* ***covered*** *by a literal* l *of the form* $A_i = v$ *iff* $v_i = v$*.*

Definition 2 ***(Rule).*** *A* ***rule*** r *is of the form* $b \Rightarrow h$ *with* $b = l_1 \wedge ... \wedge l_i$ *the* ***rule body****, and* $h = l'_1 \vee ... \vee l'_d$ *the* ***rule head****. An instance* e *is covered by* b *iff it is covered by all its literals and it is covered by the entire rule* r *iff it is covered by at least one literal in* h *as well.*

2.2 Interestingness Measures

As mentioned above we consider interesting rules to be rules that have a strong correlation between their body and their head. It is necessary to define the notion of *support*, when working with correlation measures:

Table 1. Contingency table for $b \Rightarrow h_1$

	h_1	$\neg h_1$	
b	$sup(b \Rightarrow h_1) = y_1$	$sup(b \Rightarrow \neg h_1) = x - y_1$	$sup(b) = x$
$\neg b$	$sup(\neg b \Rightarrow h) = m_1 - y_1$	$sup(\neg b \Rightarrow \neg h_1) = n - m_1 - (x - y_1)$	$sup(\neg b) = n - x$
	$sup(h_1) = m_1$	$sup(\neg h_1) = n - m_1$	n

Definition 3 ***(Support).*** *For a literal* l*, we define*

$$sup(l) = |\{e \mid e \text{ is covered by } l\}|$$

the ***support*** *of* l*. Similarly, the support of a rule body* b *is defined as*

$$sup(b) = |\{e \mid e \text{ is covered by } b\}|$$

and of a rule with a single consequent

$$sup(b \Rightarrow h) = |\{e \mid e \text{ is covered by } b \wedge e \text{ is covered by } h\}|$$

For the remainder of this paper, we will use the following notation to refer to occurrence counts of rules:

Definition 4 ***(Occurrence Counts).*** *For a given rule* $b \Rightarrow h_1 \vee ... \vee h_d$ *and a given data set* $\mathcal{E}$ *we define:*

$$\boldsymbol{n} = |\mathcal{E}|,\ \boldsymbol{m}_i = sup(h_i),\ \boldsymbol{x} = sup(b),\ \boldsymbol{y}_i = sup(b \Rightarrow h_i)$$

To facilitate the use of correlation measures, occurrence counts are often organized in contingency tables. A contingency table for a rule body having two values (i.e. *true* and *false*) and a single binary-valued target literal is shown in Table 1. Note that the sum of the cells in a row (column) is equal to the margins of the table, i.e. the rightmost (down-most) entry in a row (column). Correlationmeasures compare for a given cell the product of the corresponding margins

to the cell count, thus comparing *expected* to *observed* frequency, and score the difference. Comparing the expected and observed frequency for the upper left cell would e.g. in the χ^2-measure take the form $\frac{(y_1 - m_1 x/n)^2}{m_1 x/n}$.

Example 1. *Consider as an example a database consisting of 50 instances* ($\boldsymbol{n}$)*, for half of which* "heavy" *is* true ($\boldsymbol{m}_1$)*. Assume furthermore that* "strong eater" *occurs with support 10* ($\boldsymbol{x}$) *in the database. If eight of the ten instances for which* "strong eater" *is* true *also have* "heavy" =true ($\mathbf{y}_1$)*, then the* χ^2 *measure would give the deviation of expected from observed frequency for the upper left cell a score of* 1.8*.*

Table 2. Pseudo-Contingency table for $b \Rightarrow h_1 \vee h_2$

	h_1	$\neg h_1$	h_2	$\neg h_2$	
b	$sup(b \Rightarrow h_1)$	$sup(b \Rightarrow \neg h_1)$	$sup(b \Rightarrow h_2)$	$sup(b \Rightarrow \neg h_2)$	$sup(b) = x$
$\neg b$	$sup(\neg b \Rightarrow h_1)$	$sup(\neg b \Rightarrow \neg h_1)$	$sup(\neg b \Rightarrow h_2)$	$sup(\neg b \Rightarrow \neg h_2)$	$sup(\neg b) = n - x$
	$sup(h_1) = m_1$	$sup(\neg h_1) = n - m_1$	$sup(h_2) = m_2$	$sup(\neg h_2) = n - m_2$	n

Normally, increasing the number of involved literals leads to an increase of dimension of the contingency table to capture all dependencies among the literals. Since we are not interested in the dependencies between the h_i, we use tables such as the one in Table 2. We call this kind of table a *pseudo*-contingency table. The main difference w.r.t. a regular high-dimensional contingency table, a so-called multi-way table, is that the margin of a row is not equal to the sum of row-cells anymore. Calculation of a correlation measure still consists of comparing the product of the margins to the cell count.

Definition 5 ***(Stamp Point).*** *For a given data set* $\mathcal{E}$ *every rule* r *of the form* $b \Rightarrow h_1 \vee ... \vee h_d$ *induces a tuple* $\langle x, y_1, ..., y_d \rangle$ *of variables introduced in definition 4. This tuple is called the* ***stamp point*** *of* r*, denoted* sp(r)*, cf. [5].*

Consider now an interestingness measure such as χ^2, *Category Utility*, *Information Gain*, or *Weighted Relative Accuracy* defined on a pseudo-contingency table. Since $\boldsymbol{n}$ and the $\boldsymbol{m}_i$ are constant for a given data set, a given interestingness measure $\sigma(r)$ can be defined as a function of $d+1$ variables

$$\sigma : \mathbb{N}^{d+1} \mapsto \mathbb{R},$$

mapping the stamp point *sp(r)* to a real number.

Example 2. *This means that* sp("strong eater"⇒"heavy")=$\langle 10, 8 \rangle$*, a deviation that is quantified by* $\chi^2(\langle 10, 8 \rangle) = 4.5$*. Lets also consider* "sportive" *as interesting. Under the assumption that among all instances that are matched by* "strong eater" =true*, three also have* "sportive"=true*, we arrive at the stamp point* sp("strong eater"⇒"heavy"∨"sportive") = $\langle 10, 8, 3 \rangle$*, and if furthermore* "sportive" *'s frequency on the entire dataset is 30,* $\chi^2(\langle 10, 8, 3 \rangle) = 10.397$*.*

Now we are able to define the cluster-grouping problem:

Definition 6 ***(Cluster-Grouping Problem).***
Given:

- *A set of literals* $\mathcal{L}$
- *A data set* $\mathcal{E}$
- *An interestingness measure* σ
- *A set of target literals* $\mathcal{T}$

Find:
The set of k *rules expressible in* $\mathcal{L}$ *having the highest value of* σ *on* $\mathcal{E}$ *w.r.t. the given target literals* $\mathcal{T}$*.*

3 Unifying Several Data Mining Tasks

Let us now formulate the three data mining tasks mentioned before, i.e. correlated pattern mining, subgroup discovery and conceptual clustering, in the framework introduced above.

3.1 Correlated Pattern Mining

The correlated pattern mining task was introduced by Morishita and Sese [5] in the context of itemset mining. Their main motivation lies in the fact that association rules with very high confidence might still carry no information. Therefore, using a correlation measure to evaluate the quality of rules found will result in more interesting relationships. The task of correlated pattern mining can be formulated as a cluster-grouping problem with the following characteristics:

Let $\mathcal{I} = \{I_1, ..., I_z\}, \forall I \in \mathcal{I} : \mathcal{V}[I] = \{false, true\}$ the set of items,

- $\mathcal{L} = \{I = true \mid I \in \mathcal{I}\}$
- $\mathcal{E}$ a transaction data base
- σ is a correlation measure such as χ^2
- $\mathcal{T}$ a single literal $l \in \mathcal{L}$

While Morishita and Sese restrict their approach to the classical itemset setting, the use of a correlation measure would also allow the mining of negative relationships, i.e. the expansion of $\mathcal{L}$ to include literals of the form $i = false, i \in \mathcal{I}$, would allow one to find relationships in which the absence of an item correlates with the presence of another.

3.2 Subgroup Discovery

Lavrač *et al.* [1] argue convincingly that a rule learning algorithm such as *CN2* [10], used together with a metric measuring positive correlation such as *Weighted Relative Accuracy* (*WRAcc*) [11], can be employed to find subgroups, i.e., subsets

of a data set that show unexpected behaviour with regard to a specific target attribute. The subgroup discovery problem can be formulated as a cluster-grouping problem in the following way:

Let $A_t \in \mathcal{A}$ be the target attribute we are interested in.

- $\mathcal{L} = \{A = v \mid A \in \mathcal{A} \setminus \{A_t\}, v \in \mathcal{V}[A]\}$
- $\mathcal{E}$ a data set
- σ is *WRAcc*
- $\mathcal{T} = \{A_t = v \mid v \in \mathcal{V}[A_t]\}$

Lavrač *et al.* employ beam search to find subgroups. Depending on the beam size, this approach may lead to suboptimal rules being found. Additionally, if there are several *best* subgroups the wrong beam size might cause the exclusion of some of them.

3.3 Conceptual Clustering

The goal in conceptual clustering is to group instances in a data set into groups that exhibit high intra-cluster similarity and high inter-cluster dissimilarity. Instances are considered similar if they agree on the values of many attributes. One measure for judging the quality of a set of clusters is *Category Utility*. The task of conceptual clustering - with binary attributes only - can be formulated as a cluster-grouping task with the following characteristics:

Let $\mathcal{A} = \{A_1, ..., A_d\}, \forall A \in \mathcal{A} : \mathcal{V}[A] = \{false, true\}$ be a set of attributes:

- $\mathcal{L} = \{A = v \mid A \in \mathcal{A}, v \in \mathcal{V}[A]\}$
- $\mathcal{E}$ a data set
- σ is *Category Utility*
- $\mathcal{T}$ all literals of the form $A = true, A \in \mathcal{A}$

The well-known clustering algorithm *CobWeb* [3] aims at finding clusters with high *Category Utility*. The drawbacks of *CobWeb* lie in the fact that it processes instances iteratively, possibly leading to a sub-optimal solution, and that the direct assignment of instances to clusters might lead to clusters that cannot be described using a conjunction of literals. Instead, *CobWeb* characterizes clusters by conditional probabilities on attribute-value pairs. For high-dimensional data this representation is probably not very human-readable; additionally, it makes assignments of future instances to clusters somewhat more difficult.

3.4 Contribution

The key contribution of this paper is 1) the introduction of the cluster-grouping problem that makes abstraction of these three problem settings and 2) the formulation of an algorithm that allows one to tackle the cluster-grouping problem.

The algorithm will always output the solution or solutions achieving the highest value of σ. Usually, this can only be guaranteed by considering all possible rules which is often computationally not feasible. This problem is solved using pruning techniques that are derived from the observation that many correlation measures (such as the ones mentioned above) are in fact convex functions.

4 Upper Bound on Convex Correlation Measures

While the cluster-grouping task can be tackled using a heuristic algorithm, such an approach would not be guaranteed to find the optimal rules while an exhaustive technique is not feasible for higher-dimensional data (and therefore a large set of literals). Based on the convexity of correlation measures it is however possible to calculate an upper bound on the future value of σ for specializations of a given rule. This upper bound is used to prune away parts of the search space known not to produce interesting solutions and focus the search on promising parts of the search space.

4.1 Convexity

Convexity is formally defined as follows.

Definition 7 ***(Convexity).*** *A function $f : D \mapsto \mathbb{R}$ is convex iff D is a convex set and $\forall x_1, x_2 \in D, \lambda \in [0,1] : f(\lambda x_1 + (1-\lambda)x_2) \geq \lambda f(x_1) + (1-\lambda)f(x_2)$.*

It can be proven that χ^2, *WRAcc*, and *Category Utility* as well as other correlation measures fulfill the second criterion.

For the definition to hold it is also necessary for the domain of the function to form a convex set. A set $S \subset \mathbb{N}^n$ is convex if there are no points a and b in S such that there is a point on the line between a and b that does not belong to S. We will argue below following paragraph why this holds for the domain of the correlation functions mentioned above.

Let r denote a rule of the form $b \Rightarrow h_1 \vee ... \vee h_d$, b' a specialization, that is an extension with at least one literal, of b, r' of the form $b' \Rightarrow h_1 \vee ... \vee h_d$, and $sp(r), sp(r')$ the corresponding stamp points. Define $S_{act} = \{sp(r')\}$ as the set of *actual* stamp points of all rules r' whose rule body is a specialization of b. This set is unknown until all specializations of b have been created and evaluated on the data set. However, instead of computing S_{act}, one can try to approximate it by the set of *possible* stamp points $S_{poss} \supseteq S_{act}$. The approximation is possible because the stamp point $sp(r) = \langle x, y_1, ..., y_d \rangle$ constrains all $sp(r')_{poss} = \langle x', y'_1, ..., y'_d \rangle \in S_{poss}$ in the following way:

1. $x \geq x' \geq 0$
2. $\forall i : x \geq y_i \geq 0$
3. $\forall i : y_i \geq y'_i \geq 0$
4. $\forall i : x' - y'_i \geq x - y_i$

Each of these inequalities defines one or more convex sets in $d+1$-dimensional space. Since S_{poss} is the intersection of all these sets it is a convex set itself. S_{poss} is the domain of σ given b. Convex functions take their extreme values at the points forming the convex hull of their domain D [12]. So by evaluating σ on the points forming the convex hull of S_{poss}, it is possible to obtain an upper bound for the value that σ can take on *any* point $sp(r') \in S_{poss}$. Since S_{act} is a subset of S_{poss}, this upper bound is also an upper bound on the value of σ on any point of S_{act}.

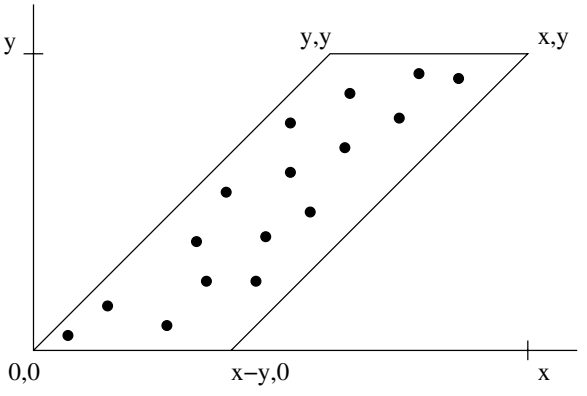


Fig. 1. Actual stamp points $sp(r')$ and convex hull of S_{poss} for $\langle x, y\rangle$(taken from [5])

For the two-dimensional case, the convex hull of S_{poss} is the parallelogram defined by the vertices $\langle x, y\rangle$, $\langle y, y\rangle$, $\langle x-y, 0\rangle$, $\langle 0, 0\rangle$ as shown in Figure 1. These vertices are derived by computing the four points $\langle x_{max}, y_{max}\rangle$, $\langle x_{min}, y_{max}\rangle$, $\langle x_{max}, y_{min}\rangle$, $\langle x_{min}, y_{min}\rangle$, while taking into account the four inequalities above.

Example 3. *Continuing our example, if the body of the rule is extended, e.g. with* "strong smoker"*, at most 10 instances are covered by the new rule body. Among those, maximally 8 will have* "heavy"=True*. Should* "strong smoker" *and* "heavy" *correlate negatively, the presence of* "strong smoker" *would reduce the number of instances in which* "heavy" *was present, which also means that the total number of instances covered is reduced. Any future stamp point will therefore be inside the parallelogram defined by* $\langle 0, 0\rangle, \langle 8, 8\rangle, \langle 2, 0\rangle, \langle 10, 8\rangle$.

For computing the upper bound on σ, the points $\langle x, y\rangle$ and $\langle 0, 0\rangle$ do not have to be considered. The first point describes a specialization with the same coverage as the current rule. Since such a specialization is expected to generalize less well to as yet unseen examples, we give preference to the rule already found. The second denotes a rule that does not cover any examples, which renders it useless. Therefore, $ub_\sigma(b) = \max\{\sigma(y, y), \sigma(x - y, 0)\}$.

Example 4. *Continuing our example from above, this means that for* σ *being* χ^2*,* $ub_{\chi^2}(r) = \max\{9.52, 2.08\}$*, given* $x = 10$*,* $y_1 = 8$*. Since* 9.52 *is larger than*

$\chi^2(x, y_1) = 4.5$ *there might be a specialization of* T *that discriminates better than* r *itself and therefore exploring this search path is worthwhile.*

4.2 Extension to Higher Dimensions

The approach described above was introduced by Morishita and Sese [5]. Their work is however restricted to a *single binary* target literal. To calculate an upper bound for the type of rules given in Definition 2, an extension to higher dimensions is necessary. Even though this extension is not too difficult, two problems arise:

Firstly, the number of convex hull points grows exponentially with the dimensionality of the data since all combinations of minimum and maximum values for x and the y_i have to be considered, meaning that $|\{min, max\}|^d = 2^d$ points would have to be evaluated. Secondly, it is insufficient to simply consider y_i as maximum value of y_i' and 0 as minimum. Instead, the dependencies between different y_i have to be considered.

Both of these problems can be somewhat lessened by enumerating the convex hull points in a different manner. This approach was first applied by Morishita and Sese [13] to the problem of clustering numerical values into clusters describable by binary attributes. To make the enumeration technique more understandable, consider *Category Utility* as an example. *Category Utility* for two clusters is defined as:

$$CU(C_1, C_2) = \frac{1}{2} \sum_{C \in \{C_1, C_2\}} P(C) \sum_{A \in \mathcal{A}} \sum_{v \in \mathcal{V}[A]} P(A = v \mid C)^2 - P(A = v)^2$$

Now consider the case that membership in the first of the two clusters is based on whether an instance is covered by a rule body b_r (for a given set of binary target attributes $\mathcal{A}_t$). If the instance is not covered by b_r, it is assigned to the second cluster. In that case, *Category Utility* can be re-written as:

$$CU(b_r, \mathcal{A}_t) = \frac{1}{2} \left(\sum_{b \in \{b_r, \neg b_r\}} P(b) \sum_{A \in \mathcal{A}_t} \sum_{v \in \{\text{true, false}\}} \frac{P(b \Rightarrow A = v)^2}{P(b)^2} - P(A = v)^2 \right)$$

This formula can be simplified by pushing the $\frac{1}{2}P(b)$ into the sum:

$$CU(b_r, \mathcal{A}_t) = \sum_{A \in \mathcal{A}_t} \sum_{v \in \{\text{true, false}\}} \sum_{b \in \{b_r, \neg b_r\}} \left(\frac{1}{2} P(b) \left(\frac{P(b \Rightarrow A = v)^2}{P(b)^2} - P(A = v)^2 \right) \right),$$

so that the total *Category Utility* can be expressed as the sum of several partial *Category Utilities*:

$$CU(b_r, \mathcal{A}_t) = \sum_{A \in \mathcal{A}_t} CU(b_r, \{A\})$$

Note that while *Category Utility* is used in this example the same property holds for correlation measures such as χ^2, *WRAcc*, *Information Gain* and others if the relations between the target attributes are ignored as in the cluster-grouping setting.

The set $\mathcal{A}_t$ corresponds to the tuple $\langle y_1, \ldots, y_d \rangle$, with every single $A_i \in \mathcal{A}_t$ corresponding to a y_i. Formulated in the *stamp point* notation, the above equation thus becomes:

$$CU(x, y_1, ..., y_d) = \sum_{i=1}^{d} CU(x, y_i),$$

and the maximization of CU the maximization of the sum of the partial *Category Utilities*:

$$\max_{\langle x', y'_1, \ldots, y'_d \rangle} CU(x', y'_1, \ldots, y'_d) = \max_{\langle x', y'_1, \ldots, y'_d \rangle} \sum_{i=1}^{d} CU(x', y'_i).$$

This alone does not make the maximization process easier since all 2^d points on the convex hull would still have to be evaluated. The number of computations would decrease if the maximization could happen for each term of the sum independently of the others. If this would be done for arbitrary x', the maximal values for different terms could induce conflicting values for x', resulting in an inconsistent stamp point. But for fixed x' each of the d terms can be maximized separately and if x' takes all values in $[0, x]$, we are not losing any solutions:

$$\max_{\langle x', y'_1, \ldots, y'_d \rangle} \sum_{i=1}^{d} CU(x', y'_i) = \max_{0 \leq x' \leq x} \left\{ \sum_{i=1}^{d} \max_{y'_i} CU(x', y'_i) \right\}$$

For each of the y' this maximum is independent of the other y'_i. The minimum and maximum values y^i_{min}, y^i_{max} are determined by the values x' and y_i. Thus, maximizing the partial *Category Utility* turns into calculating it for the minimum and maximum value and keeping the larger one:

$$\max_{0 \leq x' \leq x} \left\{ \sum_{i=1}^{d} \max_{y'_i} CU(x', y'_i) \right\} = \max_{0 \leq x' \leq x} \left\{ \sum_{i=1}^{d} \max_{y'_i \in \{y^i_{min}, y^i_{max}\}} CU(x', y'_i) \right\}$$

Therefore for a given x', $2d$ calculations have to be performed. Since $x' = 0$ and $x' = x$ can be ignored, following the argument in the last paragraph of section 4.1, a total of $2d(x - 2)$ calculations have to be performed to compute the upper bound on the values of σ for a given b_r.

5 The *CG*-Algorithm

In this section we present the algorithm *CG* for cluster-grouping. We also explain the upper bound calculation in more depth.

```
CG
E - data set
σ - correlation measure

P := {⊤}, τ := −∞, S := ∅
while P ≠ ∅
  b_mp := arg max_{ub(b)} {b ∈ P}
  C := ρ(b_mp)
  ∀c_i ∈ C
     compute sp(c_i), calculate σ(c_i)
     ub_σ(c_i) := UpperBound(sp(c_i))
     τ := max{τ, σ(c_i)}
  S := {s ∈ S | σ(s) = τ} ∪ {c ∈ C | σ(c) = τ}
  P := {p ∈ P | ub_σ(p) ≥ τ} ∪ {c ∈ C | ub_σ(c) ≥ τ}
return S
```

Fig. 2. The *CG* algorithm

The cluster-grouping algorithm *CG* (listed in Figure 2) is essentially a branch-and-bound algorithm. Starting from the most general rule body (denoted by $\top$), in each iteration the rule body $b_{mp} \in P$ with the highest upper bound is specialized. We use an optimal refinement operator ρ:

Definition 8 ***(Optimal Refinement Operator).*** *Let $\mathcal{L}$ be a set of literals, $\prec$ a total order on $\mathcal{L}$, $\tau \in \mathbb{R}$.*

$$\rho(b) = \{b \wedge l_i \mid l_i \in \mathcal{L}, ub_\sigma(l_i) \geq \tau, \forall l \in r : l \prec l_i\}$$

is an ***optimal refinement operator****.*

The operator ensures that each rule body will be created and evaluated at most once during a run of the algorithm. Since only literals are added whose upper bound exceeds the threshold, the resulting specializations have a chance of exceeding or matching the current threshold. The created specializations are then evaluated on the data set and the σ-scores and upper bounds are calculated. All specializations whose upper bound is not above the threshold τ are pruned. If possible, the threshold is raised. For the case shown in the Figure 2, the best score seen so far is used as threshold, but the algorithm can be trivially modified so that k best rules are found. Specializations whose score matches the current threshold are added to the set of solutions S. The set of promising rule bodies P is pruned using the threshold and all specializations whose upper bound exceeds τ are added.

5.1 Upper Bound Computation

As the upper bound is so essential for the mining process, we explain the algorithm for computing it in greater detail.

UpperBound
σ - correlation measure
$\langle x, y_1, ..., y_d \rangle$ - stamp point

$x' = 1$
$ub_\sigma = 0$
while $x' \leq x - 1$
  $\forall i \in \{1, ..., d\}$
    $y^i_{max} = \min\{x', y_i\}, y^i_{min} = \max\{0, y_i - (x - x')\}$
    $add_i = \max\{\sigma(x', y^i_{max}), \sigma(x', y^i_{min})\}$
  $ub_\sigma = \max\{ub_\sigma, \sum_{i=1}^{d} add_i\}$
  $x'++$
return ub_σ

Fig. 3. Algorithm for calculating the upper bound on σ

It works as follows. The loop runs from 1 through $x - 1$ (since $x' = 0$ and $x' = x$ do not have to be considered). For each possible x' value the corresponding maximum and minimum values y^i_{max}, y^i_{min} of a variable y'_i, given y_i and x', are calculated. Maximizing the correlation measure for fixed x' amounts to maximizing the terms $\sigma(x', y'_i)$. This is achieved by evaluating the measure for each maximum and minimum value y^i_{max}, y^i_{min} and choosing the larger value of those two to be added to the σ value for the current x'. The largest of these $x - 2$ correlation values is taken as the upper bound on σ.

To calculate the maximum value of σ for a given x', it is necessary to derive the minimum and maximum value for each y'_i while considering the inequalities in section 4.1. The details of deriving those extreme values are described in the following paragraphs.

Minimum-Maximum-Derivation. From inequality 2 it follows that y'_i cannot be greater than x' and inequality 3 states that $y'_i \leq y_i$. Therefore y^i_{max} can be calculated as $y^i_{max} = \min\{x', y_i\}$.

Since the y_i are occurrence counts in a data set, they cannot become negative. Inequality 4 finally has to be understood in the following way: since every instance that is counted for a y_i is also counted for x, a decrease in y_i has to be accompanied by a decrease in x. This means that y^i_{min} cannot always be set to zero. If the difference between x' and x is less than y_i, y_i must not be decreased by more than that difference. Therefore $y^i_{min} = \max\{0, y_i - (x - x')\}$.

Example 5. *Lets consider two values for x' during the loop. For $x' = 9$, $y'_1 \leq \min\{8, 9\}$, and since only one instance less would be covered, even if* "heavy" =True *held for this instance, the occurrence of* "heavy" *cannot sink below 7. Thus $7 \leq y'_1 \leq 8$. By similar reasoning one arrives at $2 \leq y'_2 \leq 3$. For $x' = 4$ the situation is somewhat different. Since x was reduced by 6 and each of the instances not covered anymore might have included* "heavy" =True *and* "sportive" =True*, the minimum values for y'_1 and y'_2 are* $\max\{0, 8 - 6 = 2\}$

and $\max\{0, 3-6=-3\}$ *respectively. Similarly, their value is bounded by the respective values of the* y_i *and* $x'=4$*. This leads to* $2 \leq y'_1 \leq 4$ *and* $0 \leq y'_2 \leq 3$*.*

6 Experimental Evaluation

Let us now investigate the applicability and performance of the *CG* algorithm on the three tasks addressed in this paper: correlated pattern mining, subgroup discovery and conceptual clustering.

The data sets employed in the experiments were selected from the *UCI Machine Learning Repository* [14]. Since the approach is limited to data sets described by binary attributes, continuous attributes were discretized and nominal attributes binarized. Two options for discretization were employed. In the naïve version, the mean value of an attribute is computed and taken as a threshold. For some data sets this leads to unbalanced attributes. For these sets, we also split the attribute values into two equal frequency bins. They are denoted with a trailing "-equal" in the name

6.1 Correlated Pattern Mining

Since the rules mined by Morishita and Sese are special cases of clustering grouping rules and the pruning technique is based on the same principles, it follows that the *CG* algorithm is applicable to correlated pattern mining and also that it will produce the same solution as Morishita and Sese's approach. We therefore repeat no experiments on this task.

6.2 Subgroup Discovery

To evaluate *CG* on subgroup discovery, we compare it with *CN2-WRAcc* [1] in two settings. The first setting corresponds to the inner loop of the *CN2-WRAcc* algorithm that employs beam search to find the best rule; the second one to the full *CN2-WRAcc* implementation that incorporates the covering algorithm.

***CN2-WRAcc* Beam Search.** As argued before, this setting corresponds to cluster-grouping. Whereas *CG* guarantees to find the optimal rules, the *CN2-WRAcc* beam search is heuristic and offers no guarantees of optimality. Therefore, in a first set of experiments concerning subgroup discovery, we investigate (1) to what extent *CN2-WRAcc* beam search finds optimal rules and also (2) how the *CG* algorithm compares with beam search in terms of efficiency. As the criterion for the optimality (1), we computed all optimal rules using *CG*, and verified whether *CN2-WRAcc* finds *any* optimal rule (*OF* in the tables) and *all* rules with optimal values (*AF*). With regard to (2), we compared the number of candidate rules that were evaluated by the two algorithms. Because the performance of beam search depends heavily on the beam-size, we experimented with different beam sizes.

Table 3. Number of Candidate Rules evaluated and Optimality of Subgroups found by Beam Search

Data Set	$BS1_{Avg}$	$BS5_{Avg}$	$BS10_{Avg}$	CG_{Min}	CG_{Max}	CG_{Avg}	AF	OF
Car	85	278	483	21	102	45	Yes	Yes
Zoo	2133	9173	17630	147	5775	1525	No*	Yes
Nursery	141	498	970	27	274	85	Yes	Yes
Breast-W	529	1882	3539	91	99	95	Yes	Yes
Voting Record	301	1139	2162	33	36	35	Yes	Yes
Mushroom	1806	7674	15006	172	219	196	No**	No**
Soybean	2036	9076	17712	1943557 (13116^{+})	2097405 (468869^{+})	2007660 (284680^{+})	No*	No*

*Not for all classes, for some not for all beam sizes
**Not for all beam sizes
$^{+}$Break value $f = 5$

The first column of Table 3, $BS1_{Avg}$, denotes the number of candidate patterns evaluated during a search with beam size 1 by *CN2*, $BS5_{Avg}$ and $BS10_{Avg}$ for beam size 5 and 10 accordingly. The columns CG_{Min} and CG_{Max} denote the minimum and maximum number of candidates generated by *CG*. For *CG* those numbers vary with the class considered, whereas for *CN2* these numbers are quite close to one another for the different classes. This also explains why for *CN2* we report on the average values. In addition, the average number of candidates CG_{Avg} is reported to show the general behavior of the algorithm.

In most cases there is a substantial reduction in the number of candidate rules considered during the search process even when the *CG*-algorithm is compared to the greedy (i.e. beam size 1) approach. The difference is even more pronounced for wider beams. For those cases where more candidate rules are considered by *CG*, the beam search algorithm often either fails to uncover all interesting rules or finds suboptimal ones. This also happens in cases in which the beam size approach evaluates more rules than *CG*.

The experiments also show that increasing the beam size does not necessarily lead to better results. For the first class considered in the *soybean* data set, beam sizes 10 and 15 induce incomplete rule sets while beam sizes 4 and 5 generate the correct results, showing the difficulty of guessing the right beam size. The most likely explanation for this effect is probably that locally good solutions are kept, that are excluded by narrower beams. If those solutions have a large number of refinements they replace promising rules that would have been kept for specialization in runs with smaller beam sizes. It might be interesting to further investigate this phenomenon. The circumstances under which not (all) optimal rules are found are denoted by the asterisks.

In some cases the *CG* algorithm investigates an excessive amount of candidate rules. The effect occurs especially for rather small values of σ and is due to the upper bound being overly optimistic. We therefore built a variant that is no longer guaranteed to find the optimal rules: whenever the pruning threshold is raised the number of candidate rules considered so far is memorized. Once

f times that number of candidates has been visited without improvement of the pruning threshold the search is terminated. The cutoff value f has to be determined by the user. This version of the algorithm behaves like a local search approach, assuming that failure to find a higher value in the (rather extensive) neighborhood of the currently best solution means that a global maximum has been found. This variant is computationally less expensive than *CG*, while it still finds all optimal rules on the studied data sets even for small value of f ($f = 5$).

***CN2-WRAcc* Sequential Covering.** In the second experiment, we employ the sequential covering algorithm as a wrapper around *CG* and the *CN2-WRAcc* beam search procedure. This corresponds to the original setting addressed by *CN2-WRAcc*. The results are shown below (Table 4).

To consider the most favourable efficiency results for *CN2-WRAcc*, we report only the minimum and maximum number of candidate rules considered using beam-size 1. Wider beams will generate more candidate patterns. Averaging these numbers is not really sensible since the impact of the number of iterations per class is obviously strong. These results are contrasted with those for *CG*. The last two columns convey the same information as in Table 3. To allow for a fair comparison we considered larger beam sizes for *CN2-WRAcc* when deciding whether the algorithm was capable of finding (all) optimal rules. The results for sequential covering are in line with and confirm those for the underlying components.

Table 4. Number of Candidate Rule evaluated and Optimality of Subgroups found by the Sequential Covering Approach

Data Set	$BS1_{Min}$	$BS1_{Max}$	CG_{Min}	CG_{Max}	CG_{Avg}	AF	OF
Car	76	651	21	203	101	Yes	Yes
Zoo	2109	2153	147	5775	1525	No^*	Yes
Nursery	129	292	27	274	95	Yes	Yes
Breast-W	3703	6349	739	1382	1061	Yes	Yes
Voting Record	903	903	2235	34659 (4480^+)	18447 (3358^+)	Yes	Yes
Mushroom	3558	8376	805	5638	3222	No^{**}	No^{**}

*Not for all classes, for some not for all beam sizes
**Not for all beam sizes
+Break value $f = 5$

6.3 Clustering

In the third experiment, we evaluate the performance of a hierarchical clustering algorithm based on *CG* with *Category Utility*. *CG* is used to find the best rule w.r.t. to *Category Utility* on the original data set. The rule found is used to split the data set into two subsets. *CG* is then called recursively on these subsets.

The algorithm continues until the best *Category Utility* on a subset falls below a user-specified criterion. In this way, a kind of decision tree is induced that can be used to assign future instances to the clusters. We compare our clustering approach to the well-known clustering algorithm *CobWeb*. We employed the Weka [15] implementation of *CobWeb* in our experiments. For each data set with c classes, the two clustering algorithms were run until they constructed c clusters. This was enforced by setting the minimal threshold for further refining clusters to an adequate value.[1] This was done to provide a fair basis for comparison. Furthermore, because *CobWeb* is incremental and sensitive to the order in which the examples are presented, the *CobWeb* results were averaged over 10 randomized orders.

The results are shown in Table 5. The evaluation criteria employed to compare the performance of both algorithms are:

- *Rand Index* [16] for comparing the agreement of the clusterings found,
- *Category Utility* of the discovered solutions.

The *Rand* index is the fraction of pairwise grouping decisions on which the two clusterings agree. Let $\mathcal{E} = \{e_1, ..., e_n\}$ be a data set and $\mathcal{C}_1, \mathcal{C}_2$ two clusterings of $\mathcal{E}$. For each pair of instances e_i, e_j, $\mathcal{C}_l$ either assigns them to the same cluster or to different clusters. Let *pos* be the number of decision where e_i, e_j are in the same cluster in both clusterings and *neg* the number of decisions where they belong to different clusters in both $\mathcal{C}_l$. The *Rand Index* is defined as:

$$Rand(\mathcal{C}_1, \mathcal{C}_2) = \frac{pos + neg}{n * (n-1)/2}$$

Table 5. *Category Utility* and *Rand Index* for clusterings found

Data set	***CU CG***	***CU CobWeb***	***Rand Index***
Breast-W	0.62	0.6496 ± 0.0001	0.91675 ± 0.006
Breast-W-equal	1.088	$1.147 \pm 1.95 * 10^{-5}$	0.93093 ± 0.0025
Credit-A	0.379	0.374 ± 0.0178	0.95283 ± 0.148
Credit-A-equal	0.6241	0.6243 ± 0.00067	0.9959 ± 0.0034
Glass	0.301	0.291 ± 0.0125	0.90391 ± 0.081
Hepatitis	0.446	0.459 ± 0.0142	0.83667 ± 0.0534
Iris	0.5369	0.5321 ± 0.0083	0.91952 ± 0.0799
Sick	0.2132	0.2077 ± 0.0171	0.98196 ± 0.0567
Voting Record	1.362	1.468 ± 0.0001	0.85753 ± 0.0043
Zoo (6 clusters)	0.6398	0.6349 ± 0.005	0.994093 ± 0.0043
Zoo (5 clusters)	0.7187	0.7196 ± 0.004	0.964357 ± 0.0056

As can be seen in Table 5, the results of *CG* are comparable with those of *CobWeb*. Indeed, the *Category Utilities* of the obtained clusterings are very

[1] Due to the different orderings of the examples, *CobWeb* sometimes finds 5 clusters and sometimes 6 in the zoo data set. Therefore, both 5 cluster and 6 cluster experiments were performed using *CG* as well.

similar. Sometimes, *CG* is slightly better, sometimes *CobWeb* is. Furthermore, the *Rand Index* indicates that the agreement between the clusters found is high, more than 90% in all but two cases.

On the other hand, *CG* computes definitions of the clusters, whereas *CobWeb* does not. Therefore, as a final means of comparison, we employed the rule learner PART [17], available within Weka, on the clusters produced by *CobWeb*. This procedure allows one to obtain conjunctive rules describing the clusters computed by *CobWeb*, which can then be compared to the conjunctive descriptions generated by the *CG*.

Table 6. Rules learned on *CobWeb* clusters using PART

Data set	*CobWeb*/*CG*-Rules	Assignment Accuracy
Breast-W	Superset	97.8%
Breast-W-equal	Superset/Specializations	97.8%
Credit-A	Same	100%
Credit-A-equal	Same/Superset	99.7%
Glass	Superset/Specializations	98.6%
Hepatitis	Superset/Specializations	94.2%
Iris	Same/Different	99.3%/100%
Sick	Same/Superset	100%
Voting Record	Superset	96.3
Zoo (6 clusters)	Same	97%
Zoo (5 clusters)	Same/Generalizations	98%

Table 6 shows the results of this evaluation. For each data set the relation between rules learned on the *CobWeb* clusters and the rules computed by *CG* is listed. Additionally, the right column, labeled **Assignment Accuracy**, shows the fraction of instances that those rules would be assign to their correct clusters. This is used as a measure for how well *CobWeb*'s clusters can be described by conjunctive rules.

There are varying relations between *CG*'s and *CobWeb*'s rules. *Superset* means that the *CobWeb* rules form a superset of the *CG* rules, *Specializations* that they are specializations (*Generalizations* analogous), *Same* that the same set of rules was found. Several entries for one data set denote that several *CobWeb* solutions (due to different ordering of instances) induced different rule sets and therefore different relations.

A special case is the iris data set. On some *CobWeb* solutions PART induced rules that were completely different from *CG* rules. Those had no misassignments. The rules learned using PART provide further evidence that the results obtained by *CobWeb* and *CG* are closely related. As *CobWeb* does not enforce conjunctive descriptions of the clusters *during* its search process, it is more flexible in assembling the clusters. However, because of the involvement of all attributes in the representation the results are also less interpretable. Because the resulting clusters are so close, *CG* seems preferable as it also produces symbolic descriptions.

7 Related Work

The *CG* algorithm is a substantial extension of the Morishita and Sese algorithm for correlated pattern mining. They have also developed a clustering algorithm based on their technique [13], the main difference being that their goal is to cluster numerical values (gene expression levels) using interclass variance. They also aim at describing the clusters found by conjunctions of binary attributes to make the resulting clusters more human-understandable.

The cluster-grouping problem is also related to feature selection in conceptual clustering and semi-flexible prediction [18,19]. Talavera's [18] motivation for feature selection in conceptual clustering is somewhat related to our motivation insofar as he is aiming for better comprehensibility, exclusion of irrelevant features and more efficient clustering processes (both when creating and using the clusters). There is also some similarity in where in the algorithm the feature selection happens, since it is redone for each node in the hierarchical clustering tree. This is called *local* or *dynamic* selection. The main differences are two-fold: firstly, Talavera's work still retains *CobWeb*'s representation and only achieves better comprehensibility by reducing the number of considered attributes. Secondly, in his approach each attribute is scored *before* the actual clustering step, whereas *CG* performs feature selection as part of the clustering process itself.

Cardie [19] defines semi-flexible prediction as learning to predict a set of features known *a priori* as opposed to inflexible prediction (classification) and flexible prediction (clustering). Her approach involves automated feature selection for each attribute to be predicted separately. These features are then used in subsequent independent prediction of the attributes. In contrast, we attempt to predict a disjunction of attributes from a shared set of antecedents instead.

Finally, cluster-grouping is in many aspects related to the confirmatory induction setting in the *Tertius* system by Flach *et al.* [20]. As in *CG*, rules with disjunctive rule heads are found. It is interesting to note in this context that the rule head is treated as a single target while *CG* treats each literal separately. Flach's work abstracts from the general correlation setting in which correlation is symmetric and instead focuses on the number of counter-instances to a given rule, thus considering only directed associations. Using an *optimistic* estimate (an upper bound) they prune non-promising candidates and find and rank optimal rules. Focusing on counter instances only allows more flexibility regarding the rule head, i.e. the set of literals need not be fixed.

8 Conclusion and Future Work

We have introduced the problem of cluster-grouping and argued that it unifies several popular data mining tasks. We developed the algorithm *CG*, a branch-and-bound algorithm that relies on convex correlation functions to find optimal solutions. The cluster-grouping framework lends itself in a natural way for inductive querying. However, rather than imposing normal constraints, cluster-grouping queries are a form of optimization query, in that they look for the k best patterns.

We have shown that our approach is an extension of Morishita's and Sese's work in that it allows one to deal with several target attributes. We have provided experimental evidence that the *CG* algorithm is well-suited for subgroup discovery in that it offers significant advantages when compared to the *CN2-WRAcc* approach, albeit sometimes at the cost of efficiency. *CG* is also competitive with one of the best-known clustering algorithms, *CobWeb*, while creating better interpretable solutions.

Further research will proceed in several directions. First, as can be seen in the experiments, the effectiveness of the pruning step depends strongly on the tightness of the upper bound calculated. Therefore, it is desirable to tighten future support estimates and therefore attainable values of σ. Second, since *Information Gain* is also convex, the technique should in principle be usable in the formation of multi-variate decision trees [21]. Third, probably necessary for usage with decision trees and an interesting extension in itself is the question of how to extend the *CG* to handle target attributes having more than two values. Fourth, since *Foil-Gain* is similar to *Information Gain* and also (under certain assumptions) a convex function, extension of the *CG* algorithm to first-order logic should be possible. Finally, in the context of inductive querying, some new challenges are raised by optimization queries. Most notably, the question arises as to whether we can integrate inductive optimization queries with the more traditional monotonic and anti-monotonic constraints that have been employed within the inductive querying literature.

Acknowledgments

We sincerely thank Shinichi Morishita and Jun Sese for helpful discussion and comments on our work. We also thank our fellow researchers for constructive comments and helpful suggestions.

This work was partly supported by the EU IST project cInQ (consortium on discovering knowledge with Inductive Queries), contract no. IST-2000-26469.

References

1. Todorovski, L., Flach, P.A., Lavrač, N.: Predictive performance of weighted relative accuracy. In Zighed, D.A., Komorowski, H., Zytkow, J.M., eds.: PKDD 2000, Lyon, France, Springer (2000) 255–264
2. Klösgen, W.: Efficient discovery of interesting statements in databases. Journal of Intelligent Information Systems **4** (1995) 53–69
3. Fisher, D.H.: Knowledge acquisition via incremental conceptual clustering. Machine Learning **2** (1987) 139–172
4. Cheeseman, P., Kelly, J., Self, M., Stutz, J., Taylor, W., Freeman, D.: Autoclass: A bayesian classification system. In Laird, J.E., ed.: ICML 1988, Ann Arbor, Michigan, USA, Morgan Kaufmann (1988) 54–64
5. Morishita, S., Sese, J.: Traversing itemset lattices with statistical metric pruning. In: Proceedings of the Nineteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, Dallas, Texas, USA, ACM (2000) 226–236

6. Bay, S.D., Pazzani, M.J.: Detecting group differences: Mining constrast sets. Data Mining and Knowledge Discovery **5** (2001) 213–246
7. Gluck, M.A., Corter, J.E.: Information, uncertainty, and the utility of categories. In: Proceedings of the 7th Annual Conference of the Cognitive Science Society, Irvine, California, USA, Lawrence Erlbaum Associate (1985) 283–287
8. Imielinski, T., Mannila, H.: A database perspective on knowledge discovery. Commun. ACM **39** (1996) 58–64
9. Raedt, L.D.: A perspective on inductive databases. SIGKDD Explorations **4** (2002) 69–77
10. Clark, P., Niblett, T.: The CN2 induction algorithm. Machine Learning **3** (1989) 261–283
11. Lavrač, N., Flach, P.A., Zupan, B.: Rule evaluation measures: A unifying view. In Džeroski, S., Flach, P.A., eds.: ILP 1999, Bled, Slovenia, Springer (1999) 174–185
12. Horst, R., Tuy, H.: Global Optimization - Deterministic Approaches. Springer (1996)
13. Sese, J., Morishita, S.: Itemset classified clustering. In Boulicaut, J.F., Esposito, F., Giannotti, F., Pedreschi, D., eds.: PKDD 2004, Pisa, Italy, Springer (2004) 398–409
14. Blake, C., Merz, C.: UCI repository of machine learning databases (1998)
15. Frank, E., Witten, I.H.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann (1999)
16. Rand, W.M.: Objective criteria for evaluation of clustering methods. Journal of the American Statistical Association **66** (1971) 846–850
17. Frank, E., Witten, I.H.: Generating accurate rule sets without global optimization. In Shavlik, J.W., ed.: ICML 1998, Madison, Wisconsin, USA, Morgan Kaufmann (1998) 144–151
18. Talavera, L.: Dynamic feature selection in incremental hierarchical clustering. In de Mántaras, R.L., Plaza, E., eds.: ECML 2000, Barcelone, Catalonia, Spain, Springer (2000) 392–403
19. Cardie, C.: Using decision trees to improve case-based learning. In: ICML 1993, Amherst, Massachusetts, USA, Morgan Kaufmann (1993) 25–32
20. Flach, P.A., Lachiche, N.: Confirmation-guided discovery of first-order rules with Tertius. Machine Learning **42** (2001) 61–95
21. Murthy, S.K.: On Growing Better Decision Trees from Data. PhD thesis, John Hopkins University, Baltimore, Maryland, USA (1997)

Author Index