

MISTA

Mining in Semi-Structured Data

Project Proposal

September 19, 2003

1 The Project

On July 17, 2003, the project has been granted by the Netherlands Organisation for Scientific Research (NWO), with project number 612.066.304.

1.1 Project Title

The title of the project is “Mining in Semi-Structured Data”.

1.2 Project Acronym

The project acronym is *Mista*.

1.3 Principal Investigator

The principal investigator, also contact address, is: dr. W.A. Kusters, Leiden Institute of Advanced Computer Science (LIACS), Universiteit Leiden, P.O. Box 9512, 2300 RA Leiden, The Netherlands; phone +31-71-5277059, email kusters@liacs.nl. Dr. A.J. Feelders is co-leader of the project.

2 Summary

Semi-structured data arises when the source or the environment does not impose a rigid structure on the data and when data is combined from several heterogeneous sources. Examples include the World Wide Web and bioinformatics databases; the situation also occurs in data warehousing. Unlike (nearly) unstructured raw data such as image and sound, semi-structured data has some structure: objects share (parts of) their structure.

Most data mining algorithms are not designed for semi-structured data and should at least be adapted in order to deal with such data. The objective of the project is to do research on foundations of efficient and effective mining algorithms with a focus on tree- and graph-structured data in combination with constraint-based mining and condensed representations.

3 Classification

The main classification of the project is Subdiscipline 2.2: Data mining and data warehousing. Relevant themes from NOAG-i 2001–2005 are: 5: Modeling, Simulation and Visualization (MSV), 6: Intelligent Systems (IS) and 7: Algorithms and Formal Methods (AFM).

4 Composition of the Research Team

The following persons participate in the project:

name	affiliation	specialization(s)
prof.dr. J.N. Kok	Leiden	coordination, data mining, artificial intelligence
dr. W.A. Kusters	Leiden	data mining, artificial intelligence
drs. S. Nijssen	Leiden	data mining, optimization
Ph.D. researcher (OiO)	Leiden	data mining
prof.dr. A.P.J.M. Siebes	Utrecht	data mining, databases, bioinformatics
dr. A.J. Feelders	Utrecht	data mining
Ph.D. researcher (OiO)	Utrecht	data mining

Leiden:

Leiden Institute of Advanced Computer Science (LIACS),
Universiteit Leiden, P.O. Box 9512, 2300 RA Leiden, The Netherlands;
<http://www.liacs.nl/>

Utrecht:

Institute of Information and Computing Sciences,
Universiteit Utrecht, P.O. Box 80.089, 3508 TB Utrecht, The Netherlands;
<http://www.cs.uu.nl/>

The formal Ph.D. advisor for the Leiden researcher is professor Kok, the formal Ph.D. advisor for the Utrecht researcher is professor Siebes.

5 Research School

The Leiden group participates in the Research School IPA, the Institute for Programming Research and Algorithmics (Instituut voor Programmatuurkunde en Algoritmiek), the Utrecht group participates in the Research School SIKS, the School for Information and Knowledge Systems (School voor Informatie- en KennisSystemen).

6 Description of Proposed Research

More and more data sets do not fit in the rigid relational model because the individual data items do not have the same structure completely. Rather, the data items share only partly the same structure. Such databases are called *semi-structured databases* (see [WL00]).

In a semi-structured database, there is no fixed database schema: conceptually the data is stored in a graph-like structure (like XML) which contains both information about the data as well as the data itself. Prime examples of semi-structured databases are XML databases and many of the bioinformatics databases. These bioinformatics databases differ from standard databases in the sense that

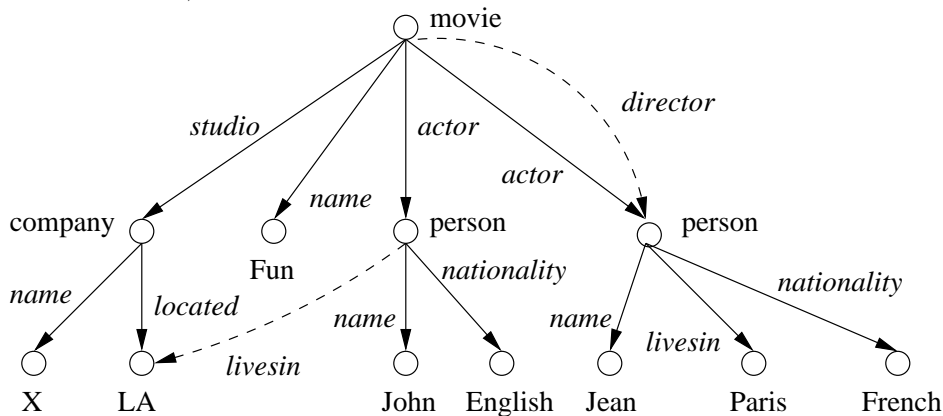
1. in a particular database one can find many different kinds of data (e.g., genome sequences, pointers to journal articles, web pages, biochemical data, physical data, information about mutation experiments, etc.);
2. the databases are often distributed and contain many links to other databases;
3. often, data is missing, and there is conflicting data.

Since in normal databases all data items share the same structure, the structure of the data items plays no role in standard data mining. For semi-structured databases, however, the structure of an individual data item encodes an important part of its semantics. Therefore, mining for patterns and models in the structure of the data, e.g., for frequent substructures, is an important

aspect of mining semi-structured databases. In that respect, semi-structured data mining is close to multi-relational data mining [KSM02] and Inductive Logic Programming: in both cases patterns have structure.

Intermezzo

In order to get some idea of semi-structured databases we examine a simplified movie database (the example is inspired by [WL00]). Suppose that a *Movie* object has a *Name*, *Persons* and a *Company*. Every *Person* has a (not necessarily unique) *Name*, usually a *Nationality* and a home *Town*. A *Company* has a *Name* and a home *Town*. All objects can have more data elements, such as pictures. Other databases contain extensive information on the *Persons*, and so on.



The tree above represents a sample movie. Note that it is also possible to have a graph-like structure (including the dotted arrows), where locations are shared between companies and actors, and an actor can appear as director too.

Now an interesting pattern may be the presence of movies with many local actors, or the search for the most often occurring actor. And how many companies from LA have many movies with French actors? At first sight this may only look like answering SQL-queries, but it rather requires the discovery of the interesting ones.

The aim of the project is to develop algorithms that discover rules or patterns from large collections of semi-structured data. We will focus on the discovery of patterns that represent interactions between entities, like frequent sub-trees and frequent sub-graphs. These can be seen as generalizations of the popular frequent item sets, which form the basis of association rules (see [AMS⁺96]). For example, for data stored in tree structured XML documents (widely occurring in practice), instead of discovering frequent item sets one would like to discover frequent (sub-)trees. Frequent trees give a feeling for the general information content in the database: it is a way of summarizing the data. This information can then be used to formulate queries, as a guideline for building indexes, as basis for structure based clustering (as an example for the “classical” case, see [KMO99]) and for discovering access patterns.

The project is divided into two parts: one which focuses on the algorithms for finding frequent trees and graphs, and one which aims at the employment of user-defined constraints and condensed representations.

1. Algorithms

Finding frequent sub-trees is not straightforward. First one has to decide on the type of trees and on a notion of inclusion on trees. Here are several possibilities that will heavily influence the algorithm. For example, are the elements in the tree ordered or not? Do we consider sets or multi-sets? When do we consider a tree to be a sub-tree of another tree?

In this context, it is also possible to consider graphs. The occurrence of cycles may complicate the matter further. (For example, in a movie database a director may serve as an actor

in his/her own movie.) There is an interesting link with the theory of parallel processes. Parallel processes are modeled by means of graphs and several notions of inclusion have been extensively studied in this theory, see, e.g., [vG90]. We intend to exploit this link.

Checking of the inclusion relation should be performed as efficiently as possible. The actual mining algorithm should be a balance between generation of candidate trees, pruning of the search space and checking of the inclusion of candidate trees on all elements of the database. Assumptions about the size of the database will also influence the algorithm. For example, algorithms simplify if one can assume that one can store inclusion information. Also, the fact that the whole database may or may not fit in main memory, necessitates different approaches.

2. Constraints and condensed representations

In practice, frequent patterns suffer from an embarrassment of richness: one often gets too many results. If all patterns would be interesting, the fact that one gets many would be a — perhaps unfortunate — fact of life. However, many of them convey little or no useful information. Hiding or even eliminating such results is clearly beneficial to the user. We consider two lines of attack:

- (a) *Constraint-based mining*: devise algorithms that employ user-defined constraints in order to reduce the number of uninteresting results.
- (b) *Condensed representations*: develop more compact representations of the frequent patterns instead of their simple enumeration, and develop algorithms that mine directly for those condensed representations.

Fortunately constraint-based mining and mining for condensed representations are orthogonal approaches. In other words, the two of them can be combined.

6.1 Algorithms

The Past

Let us first mention some of the existing techniques. Association rules are used in all sorts of business situations, and there is also a rich theory behind (see, e.g., [Ada01]). There exist several algorithms that generate association rules, such as the famous APRIORI algorithm from [AMS⁺96]. Much attention has been given to fast implementations, see [dGKPP02].

Informally, association rules can be seen as a kind of if-then rules: if a person buys a newspaper, he or she also buys chocolate. The twist association rules bring to classical if-then rules is a *conditional probability*. If a person buys a newspaper, there is a probability that he or she will also buy chocolate. This conditional probability is known as *confidence* in the literature.

Another measure that is generally connected with association rules is that of *support*: the fraction of customers for whom the rule holds, or rather the relative number of customers buying all items occurring in the rule (the so-called underlying itemset). If there is just one customer that buys newspapers and he or she also happens to buy chocolate, the association rule is not very interesting.

Next to being an interestingness measure, the support of a rule also plays a key role in the standard algorithms for association rule discovery. Given thresholds *minsup* and *minconf* for the support and confidence, these algorithms compute all association rules whose support and confidence exceed these thresholds. For these rules the underlying itemset is called frequent.

Originally, association rules were introduced in a binary, non-temporal setting. For example, one considered a collection of transactions at the check-out of a store only recording whether a certain item was bought or not. Later, many extensions have been introduced, e.g., sequences (time), hierarchical clusters of items, and item-counts.

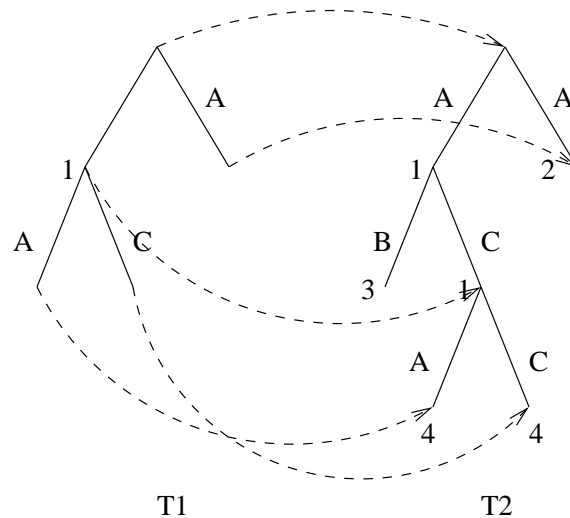


Figure 1: Two example trees

The Future

A very interesting new direction of research is to examine semi-structured data. As an example, consider the two trees in Figure 1. Now, assume the database contains a large number of labeled trees like $T2$. What kind of patterns can one be searching for? One possible pattern is given in the example tree $T1$: a tree pattern in which nodes and edges may or may not be labeled. In order to define frequency of this pattern in the set of trees, one has to define in which case a pattern is covered by a tree in the database. There are many choices for this covering relation. As is illustrated, one could define the relation such that $T1$ is included into $T2$; in this case every unlabeled edge in the pattern can correspond to a path in a transaction. In another possible definition of the relation this may not be allowed. Which choice is interesting, is likely to depend on the application.

In this small example (see also the Intermezzo on p. 3), one can already see that there are many possible choices. Among others, one could make different choices for:

- Representation of the data: are edges labeled, nodes labeled, or is there no labeling? Is the data a graph, an unordered tree, or an ordered tree?
- Representation of patterns.
- Counting frequency: do we count multiple occurrences of a pattern tree (graph) in the same database tree (graph)?
- Inclusion: can two edges in a pattern be matched with the same edge in the data? Can one edge in a pattern be mapped to a path in the data? Should the root of a pattern tree be matched to the root of a data tree?

Each of these choices results in a different algorithm. While some choices are really interesting for some application, others may not: for example, in an HTML file (which may be considered as a tree), the same tag may occur at any level; depending on the data mining goal, patterns which are counted at each depth separately, may or may not be interesting. Our first task (**Task 1**) is to analyze these questions.

The search for mining algorithms of this kind has only started recently. A proof of concept was already delivered in 1996 by an Inductive Logic Programming (ILP) algorithm, see [DT01] for a more recent overview. More efficient algorithms were developed in recent years for some special

cases. We developed a more efficient ILP algorithm for some datasets in [NK01]. As another example, in [AAK⁺02], techniques from [Bay98] are extended to labeled ordered trees.

Although each of these references proposes a feasible algorithm for a particular setup, there are still many interesting research questions, including:

- For which algorithmic choices are good applications available?
- What is the trade-off between applicability and computability?
- Are there efficient mining algorithms for setups that have not been investigated yet?

This is the second topic of investigation, referred to as **Task 2**.

The third line of approach originates from techniques from relational databases, that already inspired efficient mining algorithms for unordered, labeled trees using the Object Exchange Model (OEM). Such trees could also be extracted from many multi-relational databases. During the development of an algorithm for this setup, several kinds of questions arise, creating the third research topic (**Task 3**):

- In order for the algorithm to be efficient, how does one make sure that a pattern is only discovered once? For sets, this question has been solved (see [dGKPP02] for fast implementations), but can this algorithm be applied to unordered trees? We showed in [NK02] how this can be done by joining frequent item sets recursively in new sets. For graphs, this issue is even more important, as graph isomorphism is believed to be NP complete.
- How does one count the patterns efficiently? For many inclusion relations for trees, algorithms have already been developed. These inclusion algorithms can be used in mining algorithms, but there is much room for optimization: many patterns are very similar, such that there may be possibilities for evaluating them together.
- How can the generation of patterns interact with the counting of patterns? For example, for ordered trees a depth-first candidate generation algorithm was developed that evaluates trees while they are being built. Similar approaches may also be usable while mining for ordered trees.

In our current proposal we like to address the tasks mentioned above. We would like to answer the proposed questions in such a way that it can lead to practical applications.

6.2 Constraints and Condensed Representations

In many cases, a substantial part of the patterns generated by frequent pattern mining algorithms is of little or no interest to the user (see also [dGKW01]). To attack this problem we want to find a middle road between the two extremes of pure hypothesis testing (“classical” data analysis) and pure search. In *constraint-based* mining [PH02, PF02], mining is performed under the guidance of various kinds of constraints provided by the user. In many practical applications it turns out that users have a very clear idea of what they consider to be interesting, and this knowledge can be exploited in the mining process. This leads to both more effective and more efficient algorithms. More *effective* in the sense that a larger part of the generated patterns is interesting for the user, and more *efficient* since the constraints may be used to reduce the search space.

The use of constraints to prune the search space in algorithms for finding frequent *itemsets* has been studied extensively in the literature (see for example [PH00, LNHP99, PHL01, GWL00, PH02, BAG00, JB02a]). An important question from the efficiency viewpoint is how far the constraints can be “pushed into” the mining process. This viewpoint has led to a classification of constraints into five categories: antimonotone, monotone, succinct, convertible and inconvertible.

For example, a constraint is called *anti-monotone* if whenever an itemset violates the constraint, so does any of its supersets. The APRIORI property itself is an example of an anti-monotone constraint: whenever an itemset is not frequent, none of its supersets can be. In frequent itemset mining, interesting constraints can be defined by associating an attribute, e.g., price, with each

item. In that case the constraint $\max(\text{price}, \text{itemset}) \leq 100$ is an example of an anti-monotone constraint. The constraint $\text{avg}(\text{price}, \text{itemset}) \geq 25$ is not anti-monotone since if we add an expensive item to an itemset that violates the constraint, the extended itemset may actually satisfy it. If however the items are always added to an itemset in descending order of price, then it *does* become anti-monotone. Hence the constraint is called *convertible*.

We intend to define interesting classes of constraints for richer data structures than sets of items, namely trees and graphs. Referring to the movie database example of the Intermezzo on p. 3, we could for example specify that we are only interested in movies with at least three French actors. The classification of such a constraint of course critically depends on the definition of tree inclusion. For at least one reasonable definition this is an example of a monotone constraint: if a tree satisfies the constraint, then any of its “supertrees” will also.

Not every constraint can be pushed deep into the mining process of every frequent pattern mining algorithm. For example, a convertible constraint that is neither monotonic, nor antimonotonic, nor succinct cannot be pushed deep into the APRIORI mining algorithm [PH02]. As it turns out, pattern-growth methods [HP00, dGKPP02] are particularly suited to push the constraints deep into the mining process.

To extend the ideas of constraint-based mining to cover frequent patterns in semi-structured data, we consider the following issues:

- The definition of classes of useful constraints on the structures (trees and graphs) to be mined from semi-structured data.
- The development of a categorization of those constraints similar to what has been done for mining frequent itemsets (antimonotone, monotone, etc.).
- The development of algorithms that can use such constraints to increase the efficiency of the search process. The class of pattern-growth algorithms [HP00, PH02, dGKPP02] appears to be a promising candidate, but other possibilities will be considered as well.

A complementary method of reducing the large amount of frequent patterns is to use so-called *condensed representations* [Bay98, JB02b]. Two important examples of condensed representations for frequent itemsets are *maximal* frequent itemsets and *closed* frequent itemsets. Maximal frequent itemsets are frequent itemsets for which none of their supersets is also frequent. Clearly, each frequent itemset is a subset of a maximal frequent itemset. MAXMINER (see [Bay98]) is an algorithm that directly mines these maximal frequent itemsets from the database. Closed frequent itemsets are itemsets that completely characterize their associated set of transactions. That is, a frequent itemset X is closed if it contains *all* items that occur in every transaction in which X is bought. The closed frequent itemsets form a *basis* for all frequent sets; see [PBTL99] for more details as well as an algorithm for their discovery. The underlying idea of both concepts is that the set of all maximal/closed frequent itemsets represents all frequent itemsets but is far smaller.

Our aim is to develop condensed representations for patterns, i.e., trees and graphs, that are mined from semi-structured data, and to develop algorithms for their discovery. Furthermore, it is interesting to investigate the *combination* of the two approaches discussed above, i.e., employing user-defined constraints and the use of condensed representations. This leads to constraint-based mining of maximal patterns and closed patterns for trees and graphs.

7 Work Programme

The two researchers are supposed to deliver a Ph.D. thesis within 4 years. During the course of the project, results will be published by means of the usual conferences. The first 3 months are reserved for literature study.

The rest of the first year of the first Ph.D. student is devoted to the development of ideas about mining for frequent trees. In particular, the different choices (tree-inclusion, ...) should be listed, and the corresponding algorithms should be devised (dealing with Task 1 and Task 2 from Section 6.1). In the second year attention is given to implementation and experimentation, also

addressing Task 3. As benchmarks, a bioinformatics database, a supermarket dataset and a movie dataset are used. During this year it is decided where to proceed in the third year, taking into account the input from the second Ph.D. student, or more general the results of the project so far, and other new developments. In principle the purpose is to attack more general graph-like structures.

For the second Ph.D. student, the rest of the first year is devoted to the definition of classes of useful constraints on trees as well as the development of condensed representations. In the second year the research will be directed toward the development of algorithms that are able to employ those constraints and condensed representations. This will be done in close cooperation with the other Ph.D. student, building on the algorithms developed for mining frequent trees whenever possible. The Ph.D. student will make a start with implementation and experimentation, using a bioinformatics database, a supermarket dataset and a movie dataset as benchmarks. This will be continued in the third year.

For both Ph.D. students, the fourth and final year will be devoted to the production of their theses.

The work programme is summarized in the following table:

year	first Ph.D. student	second Ph.D. student
1; 3 months	literature	literature
1; 9 months	mining trees; algorithms	constraints; condensations
2	implementation and experimentation	algorithms
3	generalization (graphs)	implementation and experimentation
4	thesis	thesis

The researchers are expected to design and develop prototypes, that can be implemented and tested on synthetic and real life databases: the different approaches need to be verified for their practical implications. To this end we have contacts with companies such as insurance firms and large retailers, as well as a close cooperation with the department of Biology of Utrecht University and the Netherlands Institute of Developmental Biology (Hubrecht Laboratory). We plan some Master's projects to accompany the current research. In such a project the Master's student should implement and validate a technique on a real life database of one of the above mentioned organisations.

Both researchers are supposed to take part in the educational programmes of their respective Research Schools. The Large Distributed Databases group in Utrecht participates in the Research School SIKS (School for Information and Knowledge Systems), the Leiden Algorithms and Programm methodology group in the Research School IPA (Institute for Programming Research and Algorithmics). Both SIKS and IPA offer a wide range of basic and advanced courses for Ph.D. students. The supervisors will make a selection of relevant courses from the course programmes, together with the researchers. Whenever appropriate, they can attend activities organized by the other Research School; they should plan to visit these activities together. Furthermore there is the possibility of supervised self-study, participation in relevant courses from our Master programmes, and visits to European summer schools.

8 Literature

The five main contributions of the research team (in connection with the current project) are: [KSM02], [NK02], [CFS01], [dGKW01] and [dGKPP02].

References

- [AAK⁺02] T. Asai, K. Abe, S. Kawasoe, H. Arimura, H. Sakamoto, and S. Arikawa. Efficient substructure discovery from large semi-structured data. In R. Grossman, J. Han, V. Kumar, H. Mannila, and R. Motwani, editors, *Proceedings of the Second SIAM International Conference on Data Mining (SDM2002)*, pages 158–174, 2002.
- [Ada01] J.-M. Adamo. *Data Mining for Association Rules and Sequential Patterns*. Springer-Verlag, 2001.
- [AMS⁺96] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A.I. Verkamo. Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI/MIT Press, 1996.
- [BAG00] R.J. Bayardo, R. Agrawal, and D. Gunopulos. Constraint-based rule mining in large, dense databases. *Data Mining and Knowledge Discovery*, 4:217–240, 2000.
- [Bay98] R.J. Bayardo Jr. Efficiently mining long patterns from databases. In L.M. Haas and A. Tiwary, editors, *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 85–93. ACM Press, 1998.
- [CFS01] R. Castelo, A.J. Feelders, and A. Siebes. MAMBO: Discovering association rules based on conditional independencies. In F. Hoffmann, D.J. Hand, N. Adams, D. Fisher, and G. Guimaraes, editors, *Proceedings of the Fourth International Conference on Intelligent Data Analysis (IDA2001)*, volume 2189 of *Lecture Notes in Computer Science*, pages 289–298. Springer-Verlag, 2001.
- [dGKPP02] J.M. de Graaf, W.A. Kusters, W. Pijls, and V. Popova. A theoretical and practical comparison of depth first and FP-growth implementations of Apriori. In H. Blockeel and M. Denecker, editors, *Proceedings of the Fourteenth Belgium-Netherlands Artificial Intelligence Conference (BNAIC 2002)*, pages 115–122, 2002.
- [dGKW01] J.M. de Graaf, W.A. Kusters, and J.J.W. Witteman. Interesting fuzzy association rules in quantitative databases. In L. De Raedt and A. Siebes, editors, *Proceedings of the Fifth European Conference on Principles of Data Mining and Knowledge Discovery (PKDD 2001)*, volume 2168 of *Lecture Notes in Artificial Intelligence*, pages 140–151. Springer-Verlag, 2001.
- [DT01] L. Dehaspe and H.T.T. Toivonen. Discovery of relational association rules. In N. Lavrac and S. Dzeroski, editors, *Relational Data Mining*, pages 189–212. Springer-Verlag, 2001.
- [GWL00] G. Grahne, X. Wang, and L.V.S. Lakshmanan. Efficient mining of constrained correlated sets. In *Proceedings of the 16th International Conference on Data Engineering (ICDE'00)*, pages 512–521, 2000.
- [HP00] J. Han and J. Pei. Mining frequent patterns by pattern-growth: Methodology and implications. *SIGKDD Explorations*, 2:14–20, 2000.
- [JB02a] B. Jeudy and J-F. Boulicaut. Constraint-based discovery and inductive queries: application to association rule mining. In *Proceedings of the European Science Foundation workshop on pattern detection and discovery in data mining*, volume 2447 of *Lecture Notes in Artificial Intelligence*, pages 110–124. Springer-Verlag, 2002.
- [JB02b] B. Jeudy and J-F. Boulicaut. Using condensed representations for interactive association rule mining. In T. Elomaa, H. Mannila, and H. Toivonen, editors, *Proceedings of the Sixth European Conference on Principles of Data Mining and Knowledge Discovery (PKDD 2002)*, volume 2431 of *Lecture Notes in Artificial Intelligence*, pages 225–236. Springer-Verlag, 2002.

- [KMO99] W.A. Kosters, E. Marchiori, and A. Oerlemans. Mining clusters with association rules. In D.J. Hand, J.N. Kok, and M.R. Berthold, editors, *Proceedings of the Third Symposium on Intelligent Data Analysis (IDA-99)*, volume 1642 of *Lecture Notes in Computer Science*, pages 39–50. Springer-Verlag, 1999.
- [KSM02] A.J. Knobbe, A. Siebes, and B. Marseille. Involving aggregate functions in multi-relational search. In T. Elomaa, H. Mannila, and H. Toivonen, editors, *Proceedings of the Sixth European Conference on Principles of Data Mining and Knowledge Discovery (PKDD 2002)*, volume 2431 of *Lecture Notes in Artificial Intelligence*, pages 287–298. Springer-Verlag, 2002.
- [LNHP99] L. Lakshmanan, R. Ng, J. Han, and A. Pang. Optimization of constrained frequent set queries with 2-variable constraints. In *Proceedings of the 1999 ACM-SIGMOD International Conference on Management of Data (SIGMOD’99)*, pages 157–168, 1999.
- [NK01] S. Nijssen and J.N. Kok. Faster association rules for multiple relations. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI’01)*, pages 891–896, 2001.
- [NK02] S. Nijssen and J.N. Kok. Tree sets: Towards a set-oriented view on multi-relational data mining. In H. Blockeel and M. Denecker, editors, *Proceedings of the Fourteenth Belgium-Netherlands Artificial Intelligence Conference (BNAIC 2002)*, pages 219–226, 2002.
- [PBTL99] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In *Proceedings of the 7th International Conference on Database Theory (ICDT)*, pages 398–416, 1999.
- [PF02] R. Potharst and A. Feelders. Classification trees for problems with monotonicity constraints. *SIGKDD Explorations*, 4:1–10, 2002.
- [PH00] J. Pei and J. Han. Can we push more constraints into frequent pattern mining? In *Proceedings of the 2000 ACM SIGKDD International Conference on Knowledge Discovery in Databases (KDD’00)*, pages 350–354, Aug. 2000.
- [PH02] J. Pei and J. Han. Constrained frequent pattern mining: A pattern-growth view. *SIGKDD Explorations*, 4:31–39, 2002.
- [PHL01] J. Pei, J. Han, and L.V.S. Lakshmanan. Mining frequent itemsets with convertible constraints. In *Proceedings of the 17th International Conference on Data Engineering (ICDE’01)*, pages 433–442, 2001.
- [vG90] R.J. van Glabbeek. The linear time-branching time spectrum. In J.C.M. Baeten and J.W. Klop, editors, *Proceedings of CONCUR ’90. Theories of Concurrency: Unification and Extension*, number 458 in *Lecture Notes in Computer Science*, pages 278–297. Springer-Verlag, 1990.
- [WL00] K. Wang and H. Liu. Discovering structural association of semistructured data. *IEEE Transactions on Knowledge and Data Engineering*, 12:353–371, 2000.

9 Requested Budget

The requested budget is as follows. We apply for two Ph.D. researchers (OiO’s), both for a period of 4 years. The total costs amount:

$$2 \times (154,898 + 4,538) = 318,872 \text{ euro.}$$

The hosting universities supply hardware and software.