Efficient Data Mining Based on Formal Concept Analysis

Gerd Stumme

Institut für Angewandte Informatik und Formale Beschreibungsverfahren AIFB, Universität Karlsruhe, D–76128 Karlsruhe, Germany
www.aifb.uni-karlsruhe.de/WBS/gst; stumme@aifb.uni-karlsruhe.de

Abstract. Formal Concept Analysis is an unsupervised learning technique for conceptual clustering. We introduce the notion of *iceberg concept lattices* and show their use in Knowledge Discovery in Databases (KDD). Iceberg lattices are designed for analyzing very large databases. In particular they serve as a condensed representation of frequent patterns as known from association rule mining.
In order to show the interplay between Formal Concept Analysis and association rule mining, we discuss the algorithm TITANIC. We show that iceberg concept lattices are a starting point for computing condensed sets of association rules without loss of information, and are a visualization method for the resulting rules.

1 Introduction

Knowledge discovery in databases (KDD) is defined as the non-trivial extraction of valid, implicit, potentially useful and ultimately understandable information in large databases [17]. For several years, a wide range of applications in various domains have benefited from KDD techniques and many work has been conducted on this topic. The problem of mining frequent patterns arose first as a sub-problem of mining association rules [1], but it then turned out to be present in a variety of problems [18]: mining sequential patterns [3], episodes [26], association rules [2], correlations [10, 37], multi-dimensional patterns [21, 22], maximal patterns [8, 53, 23], closed patterns [47, 31–33]. Since the complexity of this problem is exponential in the size of the binary database input relation and since this relation has to be scanned several times during the process, efficient algorithms for mining frequent patterns are required.

The task of mining frequent patterns can be described as follows: Given a set G of objects, a set M of attributes (or items), a binary relation $I \subseteq G \times M$ (where $(g,m) \in I$ is read as "object g has attribute m"), and a threshold minsupp $\in [0,1]$, determine all subsets X of M (also called *patterns* here) where the *support* $\mathrm{supp}(X) := \frac{\mathrm{card}(X')}{\mathrm{card}(G)}$ (with $X' := \{g \in G \mid \forall m \in X\colon (g,m) \in I\}$) is above the threshold minsupp.

The set of these *frequent patterns* itself is usually not considered as a final result of the mining process, but rather an intermediate step. Its most prominent use are certainly association rules. The task of mining association rules is to determine all pairs $X \to Y$ of subsets of M such that $\mathrm{supp}(X \to Y) := \mathrm{supp}(X \cup Y)$ is above the threshold minsupp, and the *confidence* $\mathrm{conf}(X \to Y) := \frac{\mathrm{supp}(X \cup Y)}{\mathrm{supp}(X)}$ is above a given threshold $\mathrm{minconf} \in [0,1]$. Association rules are for instance used in warehouse basket analysis, where the warehouse management is interested in learning about products frequently bought together.

Since determining the frequent patterns is the computationally most expensive part, most research has focused on this aspect. Most algorithms follow the way of the well-known Apriori algorithm [2]. It is traversing iteratively the set of all patterns in a levelwise manner. During each iteration one level is considered: a subset of candidate patterns is created by joining the frequent patterns discovered during the previous iteration, the supports of all candidate patterns are counted, and the infrequent ones are discarded. A variety of modifications of this algorithm arose [11, 29, 34, 48] in order to improve different efficiency aspects. However, all of these algorithms have to determine the supports of *all* frequent patterns and of some infrequent ones in the database.

Other algorithms are based on the extraction of maximal frequent patterns, from which all supersets are infrequent and all subsets are frequent. They combine a levelwise bottom-up traversal with a top-down traversal in order to quickly find the maximal frequent patterns. Then, all frequent patterns are derived from these ones and one last database scan is carried on to count their support. The most prominent algorithm using this approach is Max-Miner [8]. Experimental results have shown that this approach is particularly efficient for extracting maximal frequent patterns, but when applied to extracting all frequent patterns, performances drastically decrease because of the cost of the last scan which requires roughly an inclusion test between each frequent pattern and each object of the database. As for the first approach, algorithms based on this approach have to extract the supports of *all* frequent patterns from the database.

While all techniques mentioned so far count the support of all frequent patterns, this is by no means necessary. In the next section, we will show that the knowledge of some supports is sufficient for deriving all other supports. This way, we are able to decrease computation time. An additional result is the visualization of representative frequent patterns in *iceberg concept lattices*, which is discussed in Section 3. In Section 4, we sketch the principle of one of the algorithms, called Titanic. Last but not least, iceberg concept lattices allow to drastically reduce the number of rules that are to be presented to the user, without any information loss. This is the topic of Section 5. The paper summarizes joint work with Lotfi Lakhal, Yves Bastide, Nicolas Pasquier, and Rafik Taouil as presented in [5, 6, 42, 43].

2 Mining Frequent Patterns with Formal Concept Analysis

Consider two patterns X and Y such that both describe exactly the same set of objects, i. e., $X' = Y'$. So if we know the support of one of them, we do not need to count the support of the other one in the database. In fact, we can introduce an equivalence relation θ on the powerset $\mathfrak{P}(M)$ of M by $X\theta Y \iff X' = Y'$. If we knew the relation from the beginning, it would be sufficient to count the support of one pattern of each class only — all other supports can then be derived.

Of course one does not know θ in advance, but one can determine it along the computation. It turns out that one usually has to count the support of more than one pattern of each class, but normally not of all of them. The percentage of patterns to be considered depends on how correlated the data are: The more correlated the data are, the fewer counts have to be performed.

This observation was independently made by three research groups around 1997/98, inspired by the theory of Formal Concept Analysis: L. Lakhal and his database group in Clermont–Ferrand, M. Zaki in Troy, NY, and the author in Darmstadt. The first algorithm based on this idea was Close [31], followed by A-Close [32], ChARM [55], PASCAL [6], Closet [33], and TITANIC [41,42], each having its own way to exploit the equivalence relation which is hidden in the data. In Section 4, we will sketch the TITANIC algorithm as an example.

All these algorithms make use of the theory of *Formal Concept Analysis (FCA).* Introduced in the early 1980ies as a formalization of the concept of 'concept' [51], FCA has over the years grown to a powerful theory for data analysis, information retrieval, and knowledge discovery [45]. In Artificial Intelligence (AI), FCA is used as a knowledge representation mechanism [46] and as conceptual clustering method [38,12,27]. In database theory, FCA has been extensively used for class hierarchy design and management [28,52,14,50,36,16]. Its usefulness for the analysis of data stored in relational databases has been demonstrated with the commercially used management system TOSCANA for Conceptual Information Systems [49].

FCA has been applied in a wide range of domains, including medicine, psychology, social sciences, linguistics, information sciences, machine and civil engineering etc. (cf. [45]). Over all, FCA has been used in more than 200 projects, both on the scientific and the commercial level. For instance, FCA has been applied for analyzing data of children with diabetes [35], for developing qualitative theories in music esthetics [25], for managing emails [13], for database marketing [19], and for an IT security management system [9].

FCA formalizes a concept of 'concept' as established in the international standard ISO 704: a concept is considered as a unit of thought constituted of two parts: its extension and its intension [51,15]. This understanding of 'concept' is first mentioned explicitly in the Logic of Port Royal [4]. To allow a formal description of extensions and intensions, FCA starts with the same type of data as association rule mining: a *(formal) context* $\mathbb{K} := (G, M, I)$ consists of a set G of objects [German: Gegenstände], a set M of attributes [Merkmale], and a binary relation $I \subseteq G \times M$. As above, we define, for $A \subseteq G$,

$$A' := \{m \in M \mid \forall g \in A\colon (g, m) \in I\} \;;$$

and for $B \subseteq M$, we define dually

$$B' := \{g \in G \mid \forall m \in B\colon (g,m) \in I\} \ .$$

Now, a *formal concept* is a pair (A,B) with $A \subseteq G$, $B \subseteq M$, $A' = B$ and $B' = A$. A is called *extent* and B is called *intent* of the concept. The set $\underline{\mathfrak{B}}(\mathbb{K})$ of all concepts of a formal context $\mathbb{K}$ together with the partial order $(A_1,B_1) \leq (A_2,B_2) :\Leftrightarrow A_1 \subseteq A_2$ (which is equivalent to $B_1 \supseteq B_2$) is called *concept lattice* of $\mathbb{K}$.

It turns out that each concept intent (here also called *closed pattern*) is exactly the largest pattern of the equivalence class of θ it belongs to. For any pattern $X \subseteq M$, the concept intent of its equivalence class is the set X''. The concept intents can hence be considered as 'normal forms' of the (frequent) patterns. In particular, the concept lattice contains all information to derive the support of all (frequent) patterns.

3 Iceberg Concept Lattices

While it is not really informative to study the set of all frequent patterns, the situation changes when we consider the closed patterns among them only. The concepts they belong to are called *frequent concepts*, and the set of all frequent concepts is called *iceberg concept lattice* of the context $\mathbb{K}$ for the threshold minsupp. We illustrate this by a small example. Figure 1 shows the iceberg concept lattice of the Mushroom database from the *UCI KDD Archive* [7] for a minimum support of 85 %.

The Mushroom database consists of 8,416 objects (mushrooms) and 22 (nominally valued) attributes. We obtain a formal context by creating one (Boolean) attribute for each of the 80 possible values of the 22 database attributes. The resulting formal context has thus 8,416 objects and 80 attributes. For a minimum support of 85 %, this dataset has 16 frequent patterns, namely all 2^4 possible combinations of the attributes 'veil type: partial', 'veil color: white', 'gill attachment: free', and 'ring number: one'. Only seven of them are closed. The seven frequent concepts are shown in Figure 1.

In the diagram, each node stands for formal concept. The intent of each concept (i. e., each frequent closed pattern) consists of the attributes labeled at or above the concept. The number shows its support. One can clearly see that all mushrooms in the database have the attribute 'veil type: partial'. Furthermore the diagram tells us that the three next-frequent attributes are: 'veil color: white' (with 97.62 % support), 'gill attachment: free' (97.43 %), and 'ring number: one' (92.30 %). There is no other attribute having a support higher than 85 %. But even the combination of all these four concepts is frequent (with respect to our threshold of 85 %): 89.92 % of all mushrooms in our database have one ring, a white partial veil, and free gills. This concept with a quite complex description contains more objects than the concept described by the fifth-most attribute, which has a support below our threshold of 85 %, since it is not displayed in the diagram.

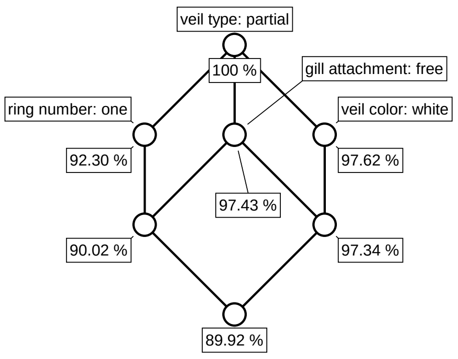

Fig. 1. Iceberg concept lattice of the mushroom database with minsupp = 85 %

In the diagram, we can detect the implication

$$\{\text{ring number: one, veil color: white}\} \Rightarrow \{\text{gill attachment: free}\} \ .$$

It is indicated by the fact that there is no concept having 'ring number: one' and 'veil color: white' (and 'veil type: partial') in its intent, but not 'gill attachment: free'. This implication has a support of 89.92 % and is globally valid in the database (i. e., it has a confidence of 100 %).

If we want to see more details, we have to decrease the minimum support. Figure 2 shows the MUSHROOM iceberg concept lattice for a minimum support of 70 %. Its 12 concepts represent all information about the 32 frequent patterns for this threshold. One observes that, of course, its top-most part is just the iceberg lattice for minsupp = 85 %. Additionally, we obtain five new concepts, having the possible combinations of the next-frequent attribute 'gill spacing: close' (having support 81.08 %) with the previous four attributes. The fact that the combination {veil type: partial, gill attachment: free, gill spacing: close} is not realized as a concept intent indicates another implication:

$$\{\text{gill attachment: free, gill spacing: close}\} \Rightarrow \{\text{veil color: white}\} \qquad (*)$$

This implication has 78.52 % support (the support of the most general concept having all three attributes in its intent) and — being an implication — 100 % confidence.

By further decreasing the minimum support, we discover more and more details. Figure 3 shows the MUSHROOMS iceberg concept lattice for a minimum support of 55 %. It shows four more partial copies of the 85 % iceberg lattice, and three new, single concepts.

The Mushrooms example shows that iceberg concept lattices are suitable especially for strongly correlated data. In Table 1, the size of the iceberg concept lattice (i. e., the number of all frequent closed patterns) is compared with the number of all frequent patterns. It shows for instance, that, for the minimum

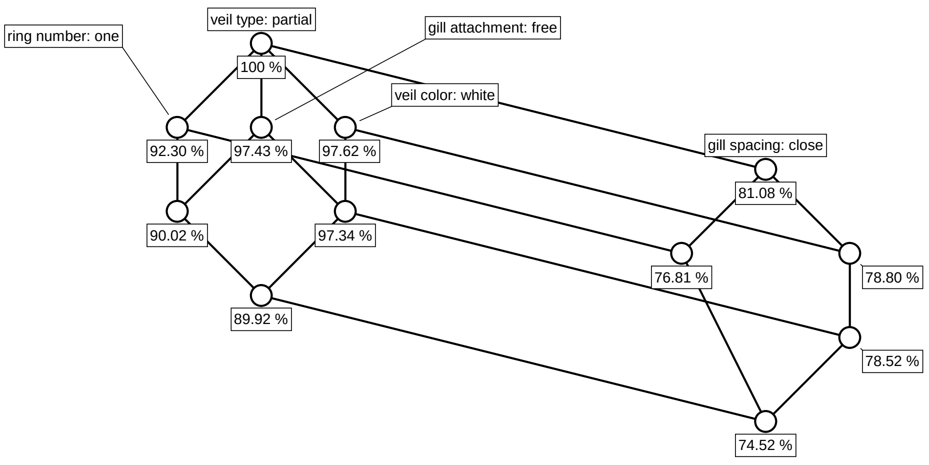

Fig. 2. Iceberg concept lattice of the mushroom database with minsupp = 70 %

Table 1. Number of frequent closed itemsets and frequent itemsets for the Mushrooms example

minsupp	# frequent closed itemsets	# frequent itemsets
85 %	7	16
70 %	12	32
55 %	32	116
0 %	32.086	2^{80}

support of 55 %, only 32 frequent closed itemsets are needed to provide all information about the support of all 116 frequent itemsets one obtains for the same threshold.

4 Computing the Iceberg Concept Lattice with Titanic

For illustrating the principles underlying the algorithms for mining frequent (closed) patterns using FCA, we sketch one representative called TITANIC. For a more detailed discussion of the algorithm, we refer to [42].

TITANIC is counting the support of so-called key patterns (and of some candidates for key patterns) only: A *key pattern* (or *minimal generator*) is every minimal pattern in an equivalence class of θ. TITANIC makes use of the fact that the set of all key patterns has the same property as the set of all frequent patterns: it is an order ideal in the powerset of M. This means that each subset of a key pattern is a key pattern, and no superset of a non-key pattern is a key pattern. Thus we can reuse the pruning approach of Apriori for computing the supports of all frequent key patterns. Once we have computed them, we have computed the support of at least one pattern in each equivalence class of θ,

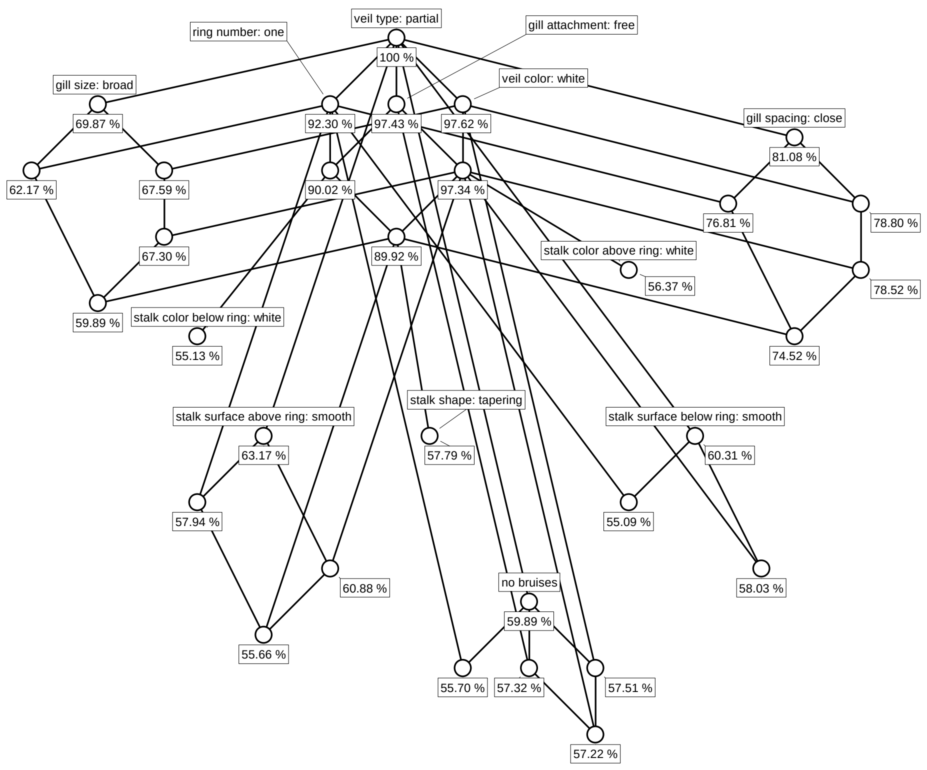

Fig. 3. Iceberg concept lattice of the mushroom database with minsupp = 55 %

and we know the relation θ completely. Hence we can deduce the support of all frequent patterns without accessing the database any more.

Figure 4 shows the principle of TITANIC. Its basic idea is as the original Apriori algorithm: At the ith iteration, we consider only patterns with cardinality i (called *i–patterns* for short), starting with $i = 1$ (step 1). In step 2, the support of all candidates is counted. For $i = 1$, the candidates are all 1–patterns, later they are all i–patterns which are potential key patterns.

Once we know the support of all i–candidates, we have enough information to compute for all $(i-1)$–key patterns their closure, i. e., the concept intent of their equivalence class. This is done in step 3, using the equation $X'' = X \cup \{x \in M \setminus X \mid \text{supp}(X) = \text{supp}(X \cup \{x\})$.

In step 4, all patterns which are either not frequent or non-key are pruned. For the latter we use a characterization of key patterns saying that a pattern is a key pattern iff its support is different from the support of all its immediate subsets. In strongly correlated data, this additional condition helps pruning a significant number of patterns.

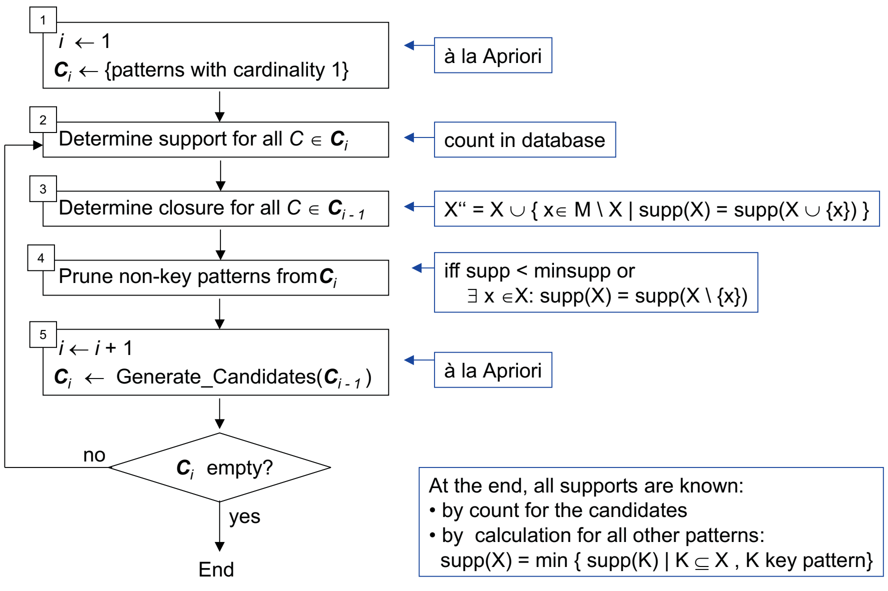
1
$i \leftarrow 1$
$C_i \leftarrow$ {patterns with cardinality 1}
à la Apriori
2
Determine support for all $C \in C_i$
count in database
3
Determine closure for all $C \in C_{i-1}$
X" = X ∪ { x∈ M \ X | supp(X) = supp(X ∪ {x}) }
4
Prune non-key patterns from C_i
iff supp < minsupp or
∃ x ∈X: supp(X) = supp(X \ {x})
5
$i \leftarrow i + 1$
$C_i \leftarrow$ Generate_Candidates(C_{i-1})
à la Apriori
no
C_i empty?
yes
End
At the end, all supports are known:
• by count for the candidates
• by calculation for all other patterns:
supp(X) = min { supp(K) | K ⊆ X , K key pattern}

Fig. 4. The Titanic algorithm

At the end of each iteration, the candidates for the next iteration are generated in step 5. The generation procedure is basically the same as for Apriori: An $(i{+}1)$–pattern is a candidate iff all its i–subpatterns are key patterns. As long as new candidates are generated, the next iteration starts. Otherwise the algorithm terminates.

It is important to note that — especially in strongly correlated data — the number of frequent key patterns is small compared to the number of all frequent patterns. Even more important, the cardinality of the largest frequent key pattern is normally smaller than the one of the largest frequent pattern. This means that the algorithm has to perform fewer iterations, and thus fewer scans of the database. This is especially important when the database is too large for main memory, as each disk access significantly increases computation time. A theoretical and experimental analysis of this behavior is given in [42], further experimental results are provided in [6].

5 Bases of Association Rules

One problem in mining association rules is the large number of rules which are usually returned. But in fact not all rules are necessary to present the infor-

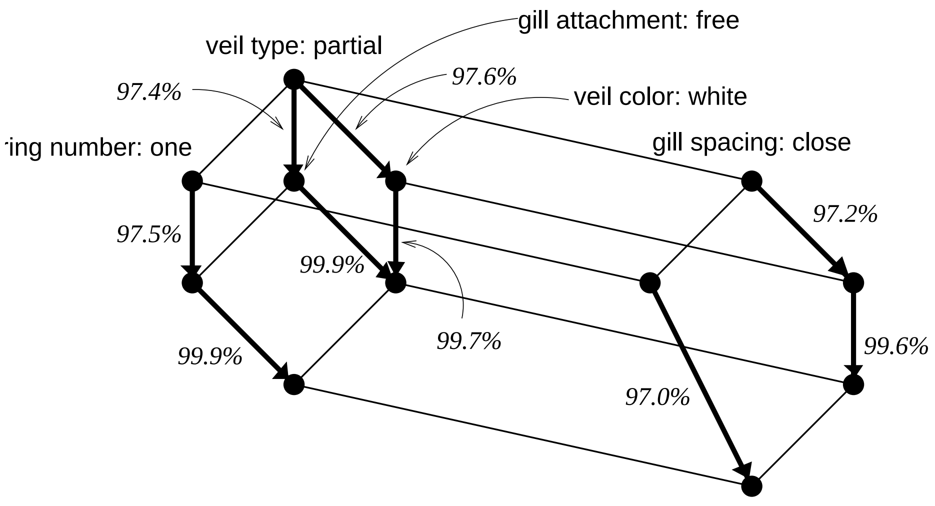

Fig. 5. Visualization of the Luxenburger basis for minsupp = 70 % and minconf= 95 %

mation. Similar to the representation of all frequent patterns by the frequent closed patterns, one can represent all valid association rules by certain subsets, so-called *bases*. In [5], [56], and [43], different bases for association rules are introduced. The computation of the bases does not require all frequent patterns, but only the closed ones.

Here we will only show by an example (taken from [43]), how these bases look like. We have already discussed how implications (i. e., association rules with 100 % confidence) can be read from the line diagram. The Luxenburger basis for approximate association rules (i. e., association rules with less than 100 % confidence) can also be visualized directly in the line diagram of an iceberg concept lattice. It makes use of results of [24] and contains only those rules $B_1 \to B_2$ where B_1 and B_2 are frequent concept intents and where the concept (B_1', B_1) is an immediate subconcept of (B_2', B_2). Hence there corresponds to each approximate rule in the Luxenburger base exactly one edge in the line diagram. Figure 5 visualizes all rules in the Luxenburger basis for minsupp = 70 % and minconf = 95 %. For instance, the rightmost arrow stands for the association rule {veil color: white, gill spacing: close} $\to$ {gill attachment: free}, which holds with a confidence of 99.6 %. Its support is the support of the concept the arrow is pointing to: 78.52 %, as shown in Figure 2. Edges without label indicate that the confidence of the rule is below the minimum confidence threshold. The visualization technique is described in more detail in [43]. In comparison with other visualization techniques for association rules (as for instance implemented in the IBM Intelligent Miner), the visualization of the Luxenburger basis within the iceberg concept lattice benefits of the smaller number of rules to be represented (without loss of information!), and of the presence of a 'reading direction' provided by the concept hierarchy.

6 Conclusion

We have shown that results of Formal Concept Analysis increase on one hand the performance of data mining algorithms, and improve on the other hand the visualization of the results. There remains still a huge potential for further exploitation of FCA for data mining and knowledge discovery.

References

1. R. Agrawal, T. Imielinski, A. Swami. Mining association rules between sets of items in large databases. *Proc. SIGMOD Conf.*, 1993, 207–216
2. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. *Proc. VLDB Conf.*, 1994, 478–499 (Expanded version in IBM Report RJ9839)
3. R. Agrawal and R. Srikant. Mining sequential patterns. In *Proc. of the 11th Int'l Conf. on Data Engineering (ICDE)*, pages 3–14, Mar. 1995.
4. A. Arnauld, P. Nicole: *La logique ou l'art de penser — contenant, outre les règles communes, plusieurs observations nouvelles, propres à former le jugement.* Ch. Saveux, Paris 1668
5. Y. Bastide, N. Pasquier, R. Taouil, G. Stumme, L. Lakhal: Mining Minimal Non-Redundant Association Rules Using Frequent Closed Itemsets. In: J. Lloyd, V. Dahl, U. Furbach, M. Kerber, K.–K. Lau, C. Palamidessi, L. M. Pereira, Y. Sagiv, P. J. Stuckey (eds.): *Computational Logic — CL.* Proc. 1st Intl. Conf. on CL (6th Intl. Conf. on Database Systems). LNAI **1861**, Springer, Heidelberg 2000, 972–986
6. Y. Bastide, R. Taouil, N. Pasquier, G. Stumme, L. Lakhal: Mining Frequent Patterns with Counting Inference. *SIGKDD Explorations* **2**(2), Special Issue on Scalable Algorithms, 2000, 71–80
7. S. D. Bay. The UCI KDD Archive [http://kdd.ics.uci.edu]. Irvine, CA: University of California, Department of Information and Computer Science.
8. R. J. Bayardo: Efficiently Mining Long Patterns from Databases. *Proc. SIGMOD '98*, 1998, 85–93
9. K. Becker, G. Stumme, R. Wille, U. Wille, M. Zickwolff: Conceptual Information Systems Discussed Through an IT-Security Tool. In: R. Dieng, O. Corby (eds.): *Knowledge Engineering and Knowledge Management. Methods, Models, and Tools.* Proc. EKAW '00. LNAI **1937**, Springer, Heidelberg 2000, 352–365
10. S. Brin, R. Motwani, and C. Silverstein. Beyond market baskets: Generalizing association rules to correlation. In *Proc. ACM SIGMOD Int'l Conf. on Management of Data*, pages 265–276, May 1997.
11. S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In *Proc. ACM SIGMOD Int'l Conf. on Management of Data*, pages 255–264, May 1997.
12. C. Carpineto, G. Romano: GALOIS: An Order-Theoretic Approach to Conceptual Clustering. *Machine Learning.* Proc. ICML 1993, Morgan Kaufmann Prublishers 1993, 33–40
13. R. Cole, G. Stumme: CEM – A Conceptual Email Manager. In: B. Ganter, G. W. Mineau (eds.): *Conceptual Structures: Logical, Linguistic, and Computational Issues.* Proc. ICCS '00. LNAI **1867**, Springer, Heidelberg 2000, 438–452

14. H. Dicky, C. Dony, M. Huchard, T Libourel: On automatic class insertion with overloading. *OOPSLA 1996*, 251–267
15. B. Ganter, R. Wille: *Formal Concept Analysis: Mathematical Foundations.* Springer, Heidelberg 1999
16. R. Godin, H. Mili, G. Mineau, R. Missaoui, A. Arfi, T. Chau: Design of class hierarchies based on concept (Galois) lattices. *TAPOS* **4**(2), 1998, 117–134
17. J. Han and M. Kamber. *Data Mining: Concepts and Techniques.* Morgan Kaufmann, Sept. 2000.
18. J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proc. ACM SIGMOD Int'l Conf. on Management of Data*, pages 1–12, May 2000.
19. J. Hereth, G. Stumme, U. Wille, R. Wille: Conceptual Knowledge Discovery and Data Analysis. In: B. Ganter, G. Mineau (eds.): *Conceptual Structures: Logical, Linguistic, and Computational Structures.* Proc. ICCS 2000. LNAI **1867**, Springer, Heidelberg 2000, 421–437
20. Y. Huhtala, J. Kärkkäinen, P. Porkka, H. Toivonen: TANE: an efficient algorithm for discovering functional and approximate dependencies. *The Computer Journal* **42**(2), 1999, 100–111
21. M. Kamber, J. Han, and Y. Chiang. Metarule-guided mining of multi-dimensional association rules using data cubes. In *Proc. of the 3rd KDD Int'l Conf.*, Aug. 1997.
22. B. Lent, A. Swami, and J. Widom. Clustering association rules. In *Proc. of the 13th Int'l Conf. on Data Engineering (ICDE)*, pages 220–231, Mar. 1997.
23. D. Lin and Z. M. Kedem. Pincer-Search: A new algorithm for discovering the maximum frequent set. In *Proc. of the 6th Int'l Conf.on Extending Database Technology (EDBT)*, pages 105–119, Mar. 1998.
24. M. Luxenburger: Implications partielles dans un contexte. *Mathématiques, Informatique et Sciences Humaines* **29**(113), 1991, 35–55
25. K. Mackensen, U. Wille: Qualitative Text Analysis Supported by Conceptual Data Systems. *Quality and Quantity: Internatinal Journal of Methodology* **2**(33), 1999, 135–156
26. H. Mannila, H. Toivonen, and A. I. Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289, Sept. 1997.
27. G. Mineau, G., R. Godin: Automatic Structuring of Knowledge Bases by Conceptual Clustering. *IEEE Transactions on Knowledge and Data Engineering* **7**(5), 1995, 824–829
28. M. Missikoff, M. Scholl: An algorithm for insertion into a lattice: application to type classification. *Proc. 3rd Intl. Conf. FODO 1989.* LNCS **367**, Springer, Heidelberg 1989, 64–82
29. J. S. Park, M. S. Chen, and P. S. Yu. An efficient hash based algorithm for mining association rules. In *Proc. ACM SIGMOD Int'l Conf. on Management of Data*, pages 175–186, May 1995.
30. N. Pasquier, Y. Bastide, R. Taouil, L. Lakhal: Pruning Closed Itemset Lattices for Association Rules. *14ièmes Journées Bases de Données Avancées (BDA'98)*, Hammamet, Tunisia, 26–30 October 1998
31. N. Pasquier, Y. Bastide, R. Taouil, L. Lakhal: Efficient mining of association rules using closed itemset lattices. *Journal of Information Systems*, **24**(1), 1999, 25–46
32. N. Pasquier, Y. Bastide, R. Taouil, L. Lakhal: Discovering frequent closed itemsets for association rules. *Proc. ICDT '99.* LNCS **1540**. Springer, Heidelberg 1999, 398–416

33. J. Pei, J. Han, R. Mao: CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets. *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery 2000*, 21–30
34. A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databases. In *Proc. of the 21th Int'l Conf. on Very Large Data Bases (VLDB)*, pages 432–444, Sept. 1995.
35. P. Scheich, M. Skorsky, F. Vogt, C. Wachter, R. Wille: Conceptual Data Systems. In: O. Opitz, B. Lausen, R. Klar (eds.): *Information and Classification.* Springer, Berlin-Heidelberg 1993, 72–84
36. I. Schmitt, G. Saake: Merging inheritance hierarchies for database integration. *Proc. 3rd IFCIS Intl. Conf. on Cooperative Information Systems*, New York City, Nework, USA, August 20-22, 1998, 122–131
37. C. Silverstein, S. Brin, and R. Motwani. Beyond market baskets: Generalizing association rules to dependence rules. *Data Mining and Knowledge Discovery*, 2(1), Jan. 1998.
38. S. Strahringer, R. Wille: Conceptual clustering via convex-ordinal structures. In: O. Opitz, B. Lausen, R. Klar (eds.): *Information and Classification.* Springer, Berlin-Heidelberg 1993, 85–98
39. G. Stumme: *Conceptual Knowledge Discovery with Frequent Concept Lattices.* FB4-Preprint **2043**, TU Darmstadt 1999
40. G. Stumme, R. Taouil, Y. Bastide, L. Lakhal: Conceptual Clustering with Iceberg Concept Lattices. *Proc. GI–Fachgruppentreffen Maschinelles Lernen '01.* Universität Dortmund **763**, Oktober 2001
41. G. Stumme, R. Taouil, Y. Bastide, N. Pasquier, L. Lakhal: Fast computation of concept lattices using data mining techniques. *Proc. 7th Intl. Workshop on Knowledge Representation Meets Databases*, Berlin, 21–22. August 2000. CEUR-Workshop Proceeding. http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/
42. G. Stumme, R. Taouil, Y. Bastide, N. Pasqier, L. Lakhal: Computing Iceberg Concept Lattices with Titanic. *J. on Knowledge and Data Engineering* **42**(2), 2002, 189–222
43. G. Stumme, R. Taouil, Y. Bastide, N. Pasquier, L. Lakhal: Intelligent Structuring and Reducing of Association Rules with Formal Concept Analysis. In: F. Baader. G. Brewker, T. Eiter (eds.): *KI 2001: Advances in Artificial Intelligence.* Proc. KI 2001. LNAI **2174**, Springer, Heidelberg 2001, 335–350
44. G. Stumme, R. Wille, U. Wille: Conceptual Knowledge Discovery in Databases Using Formal Concept Analysis Methods. In: J. M. Żytkow, M. Quafofou (eds.): *Principles of Data Mining and Knowledge Discovery.* Proc. 2nd European Symposium on PKDD '98, LNAI **1510**, Springer, Heidelberg 1998, 450–458
45. G. Stumme, R. Wille (eds.): *Begriffliche Wissensverarbeitung – Methoden und Anwendungen.* Springer, Heidelberg 2000
46. G. Stumme: Formal Concept Analysis on its Way from Mathematics to Computer Science. *Proc. 10th Intl. Conf. on Conceptual Structures (ICCS 2002).* LNCS, Springer, Heidelberg 2002
47. R. Taouil, N. Pasquier, Y. Bastide, L. Lakhal: Mining Bases for Association Rules Using Closed Sets. *Proc. 16th Intl. Conf. ICDE 2000*, San Diego, CA, US, February 2000, 307
48. H. Toivonen. Sampling large databases for association rules. In *Proc. of the 22nd Int'l Conf. on Very Large Data Bases (VLDB)*, pages 134–145, Sept. 1996.
49. F. Vogt, R. Wille: TOSCANA – A graphical tool for analyzing and exploring data. LNCS **894**, Springer, Heidelberg 1995, 226–233

50. K. Waiyamai, R. Taouil, L. Lakhal: Towards an object database approach for managing concept lattices. *Proc. 16th Intl. Conf. on Conceptual Modeling*, LNCS **1331**, Springer, Heidelberg 1997, 299–312
51. R. Wille: Restructuring lattice theory: an approach based on hierarchies of concepts. In: I. Rival (ed.). *Ordered sets.* Reidel, Dordrecht–Boston 1982, 445–470
52. A. Yahia, L. Lakhal, J. P. Bordat, R. Cicchetti: iO2: An algorithmic method for building inheritance graphs in object database design. *Proc. 15th Intl. Conf. on Conceptual Modeling.* LNCS **1157**, Springer, Heidelberg 1996, 422–437
53. M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New algorithms for fast discovery of association rules. In *Proc. of the 3rd Int'l Conf. on Knowledge Discovery in Databases (KDD)*, pages 283–286, Aug. 1997.
54. M. J. Zaki, M. Ogihara: Theoretical Foundations of Association Rules, *3rd SIGMOD'98 Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD)*, Seattle, WA, June 1998, 7:1–7:8
55. M. J. Zaki, C.–J. Hsiao: ChARM: An efficient algorithm for closed association rule mining. Technical Report 99–10, Computer Science Dept., Rensselaer Polytechnic Institute, October 1999
56. M. J. Zaki: Generating non-redundant association rules. *Proc. KDD 2000.* 34–43