Data mining, Hypergraph Transversals, and Machine Learning*
Extended abstract

Dimitrios Gunopulos
IBM Almaden RC K55/B1
650 Harry Rd.
San Jose CA 95120
USA
gunopulo@almaden.ibm.com

Roni Khardon†
Aiken Computation Laboratory
Harvard University
Cambridge, MA 02138
USA
roni@das.harvard.edu

Heikki Mannila
University of Helsinki
Department of Computer Science
P.O. Box 26, FIN-00014 Helsinki
Finland
Heikki.Mannila@cs.helsinki.fi

Hannu Toivonen
University of Helsinki
Department of Computer Science
P.O. Box 26, FIN-00014 Helsinki
Finland
Hannu.Toivonen@cs.helsinki.fi

Abstract

Several data mining problems can be formulated as problems of finding maximally specific sentences that are interesting in a database. We first show that this problem has a close relationship with the hypergraph transversal problem. We then analyze two algorithms that have been previously used in data mining, proving upper bounds on their complexity. The first algorithm is useful when the maximally specific interesting sentences are "small". We show that this algorithm can also be used to efficiently solve a special case of the hypergraph transversal problem, improving on previous results. The second algorithm utilizes a subroutine for hypergraph transversals, and is applicable in more general situations, with complexity close to a lower bound for the problem. We also relate these problems to the model of exact learning in computational learning theory, and use the correspondence to derive some corollaries.

1 Introduction

Data mining, or knowledge discovery in databases, aims at finding useful information from large masses of data; see [9] for a useful summary. The field of data mining has recently expanded considerably: both applications and methods have been developed at a fast pace. However, the theory of the area is still at its infancy: there is virtually no information about the complexity of data mining problems, and there even is no generally accepted formal framework for the area.

In this paper we attack the problem of obtaining useful general results about the complexity of data mining problems. We consider problems that can be formulated as finding maximal elements in a lattice of sentences or patterns. Examples of such problems include finding frequent sets (the essential stage in finding association rules) [1], finding episodes from sequences [21], and finding keys or inclusion dependencies from relation instances [17]. It has been previously observed [19] that the problem of enumerating maximal elements is intimately connected to the hypergraph transversal problem [8]. We utilize this fact in analyzing two algorithms that have been used in practical applications of data mining.

Several variations of the *levelwise* algorithm have been successfully applied to problems of data mining [2, 19, 20, 21]. This algorithm computes the maximal elements by walking up in the lattice of interesting sentences, one level at a time. We show that under some conditions this algorithm is indeed optimal thus explaining its empirical success and shedding some light on when and why it is useful. Furthermore, we show that this algorithm can be used to efficiently solve a special case of the hypergraph transversal problem, improving on previous theoretical results.

We also analyze the algorithm *Dualize and Advance*. This algorithm was motivated by the relation to hypergraph transversals, and was recently used in an empirical study of data mining problems [11]. We show that the algorithm has sub-exponential complexity, coming close to a lower bound for the problem, and that improving this bound hinges on the complexity of the hypergraph transversal problem (which is an open problem).

We also show a correspondence between the data mining problem and the model of exact learning in computational learning theory. While it seems intuitive that the problems

*Part of this research was done while the first two authors were visiting the university of Helsinki.

†Research supported by ONR grant N00014-95-1-0550, and ARO grant DAAL03-92-G-0115.

are similar, the formal relation was not clear. In particular, in the problem of mining association rules, weak predictors $A \Rightarrow B$ are found so that A and B are frequent in the database. In contrast, in machine learning the task is to find a single (and normally strong) predictor based on all attributes.

The data mining task considered in this paper directly corresponds to the subtask of finding frequent sets in the problem of mining association rules. We show that there is a direct correspondence between this abstracted data mining problem studied here and the problem of learning monotone functions with membership queries [3]. Thus we show that the learning problem is embedded in the problem of mining association rules. We use this relation to derive corollaries (mainly) for learning theory. Interestingly, an algorithm similar to the dualize and advance algorithm (though slightly more complicated) has been previously suggested in learning theory in an analysis of worst case complexity of learning [5], and can be used to yield a similar result.

To summarize, we analyze and clarify the properties of two data mining algorithms that have already proved useful in several applications. This is also used to show that a new subproblem of hypergraph transversals is solvable. Furthermore, we expose a relation to the problem of learning that proves useful in driving some corollaries of these results.

The rest of the paper is organized as follows. In Section 2 we present a simple framework for data mining that can be used to describe a variety of problems, and in Section 3 we look at the complexity of this task. Section 4 discusses the levelwise algorithm, and Section 5 discusses the dualize and advance algorithm. In Section 6 we study the connection to learning monotone functions.

2 A Framework for Data Mining

The model of knowledge discovery that we consider is the following. We are given a database $\mathbf{r}$, a language $\mathcal{L}$ for expressing properties or defining subgroups of the data, and an interestingness predicate q for evaluating whether a sentence $\varphi \in \mathcal{L}$ defines an interesting subclass of $\mathbf{r}$. The task is to find the theory of $\mathbf{r}$ with respect to $\mathcal{L}$ and q, i.e., the set $Th(\mathcal{L}, \mathbf{r}, q) = \{\varphi \in \mathcal{L} \mid q(\mathbf{r}, \varphi) \text{ is true}\}$.

Note that we are not specifying any satisfaction relation for the sentences of $\mathcal{L}$ in $\mathbf{r}$: this task is taken care of by the interestingness predicate q. For some applications, $q(\mathbf{r}, \varphi)$ could mean that φ is true or almost true in $\mathbf{r}$, or that φ defines (in some way) an interesting subgroup of $\mathbf{r}$. The approach has been used in various forms for example in [1, 2, 6, 7, 13, 14, 17, 21].

Obviously, if $\mathcal{L}$ is infinite and $q(\mathbf{r}, \varphi)$ is satisfied for infinitely many sentences, (an explicit representation of) $Th(\mathcal{L}, \mathbf{r}, q)$ cannot be computed. For the above formulation to make sense, the language $\mathcal{L}$ has to be defined carefully.

As already considered by Mitchell [23], we use a specialization/generalization relation between sentences. (See, e.g., [15] for an overview of approaches to related problems.) A *specialization relation* is a partial order $\preceq$ on the sentences in $\mathcal{L}$. We say that φ is more general than θ, if $\varphi \preceq \theta$; we also say that θ is more specific than φ. The relation $\preceq$ is a *monotone specialization relation* with respect to q if the quality predicate q is monotone with respect to $\preceq$, i.e., for all $\mathbf{r}$ and φ we have the following: if $q(\mathbf{r}, \varphi)$ and $\varphi' \preceq \varphi$, then $q(\mathbf{r}, \varphi')$. In other words, if a sentence φ is interesting according to the quality predicate q, then also all less special (i.e., more general) sentences $\varphi' \preceq \varphi$ are interesting. We write $\sigma \prec \tau$ if $\sigma \preceq \tau$ and not $\tau \preceq \sigma$.

Typically, the relation $\preceq$ is (a restriction of) the semantic implication relation: if $\sigma \preceq \tau$, then $\tau \models \sigma$, i.e., for all databases $\mathbf{r}$, if $\mathbf{r} \models \tau$, then $\mathbf{r} \models \sigma$. Note that if the interestingness predicate q is defined in terms of statistical significance or something similar, then the semantic implication relation is not a monotone specialization relation with respect to q: a more specific statement can be interesting, even when a general statement is not.

Given a specialization relation $\preceq$, the set $Th(\mathcal{L}, \mathbf{r}, q)$ can be represented by enumerating only its maximal elements, i.e., the set

$$\begin{aligned} MTh(\mathcal{L}, \mathbf{r}, q) \quad = \quad & \{\phi \in Th(\mathcal{L}, \mathbf{r}, q) \mid \\ & \text{for no } \theta \in Th(\mathcal{L}, \mathbf{r}, q) \ \ \phi \prec \theta\} \end{aligned}$$

Here again, one should be careful when working with infinite lattices. We assume throughout the paper that the maximal elements exist and are well defined, and similarly for the minimal elements outside the theory $Th(\mathcal{L}, \mathbf{r}, q)$. This definitely holds in finite lattices, and can be useful in more general cases as well.

In this paper we consider the following problem.

Problem 1 (MaxTh) *Given $\mathcal{L}$, $\mathbf{r}$, and q, what is the complexity of computing $MTh(\mathcal{L}, \mathbf{r}, q)$.*

It is easy to show [19] that finding frequent sets, episodes, keys, or inclusion dependencies are instances of the problem MaxTh. Especially for the problem of finding keys (or, more generally, functional dependencies) from relation instances the current framework has lots of connections to previous work. We describe here the problem of computing frequent sets, and use this example throughout the paper. Descriptions of the other mappings can be found in [19].

Association Rules and Frequent Sets: Given a 0/1 relation $\mathbf{r}$ with attributes R, an association rule is an expression $X \Rightarrow A$, where $X \subseteq R$ and $A \in R$. The intuitive meaning of such a rule is that if a row has 1 in all attributes of X then it tends also to have 1 in column A. Typically, in data mining, association rules are searched so that the set of rows having 1 in the attributes in $X \cup A$ is large enough; if we were to draw random rows from $\mathbf{r}$, it is required that such rows will be drawn with frequency at least σ, for some fixed σ. The actual frequency is called the *support* of the rule. The ratio of rows including 1 in $X \cup A$ to those including 1 in the set X is called the *confidence* of the rule.

Given the above description, a major sub-task that is usually solved first is that of computing frequent sets. Namely, given a 0/1 relation $\mathbf{r}$, compute all subsets Z such that the frequency of rows having 1 in all attributes of Z is larger than σ. Clearly, this is an instance of the problem discussed above; $\mathcal{L}$ is the set of subsets of R, and q corresponds to having frequency higher than σ. The set $Th(\mathcal{L}, \mathbf{r}, q)$ corresponds to the set of frequent sets, and similarly we can talk on maximal frequent sets.

Once the frequent sets are found the problem of computing association rules from them is straightforward. For each frequent set Z, and for each $A \in Z$ one can test the confidence of the rule $Z \setminus A \Rightarrow A$.

3 Complexity of Finding all Interesting Sentences

To study the complexity of the generation problem we introduce some notation and basic results that appeared previously in [19].

Consider a set S of sentences from $\mathcal{L}$ such that S is closed downwards under the relation $\preceq$, i.e., if $\theta \in S$ and $\varphi \preceq \theta$, then $\varphi \in S$. The *border* $Bd(S)$ of S consists of those sentences σ such that all generalizations of σ are in S and none of the specializations of σ is in S. Those sentences σ in $Bd(S)$ that are in S are called the *positive border*[1] $Bd^+(S)$, and those sentences σ in $Bd(S)$ that are not in S are the *negative border* $Bd^-(S)$. In other words,

$$Bd(S) = Bd^+(S) \cup Bd^-(S),$$

where

$$Bd^+(S) = \{\sigma \in S \mid \text{ for all } \gamma \text{ s.t. } \sigma \prec \gamma, \text{ we have } \gamma \notin S\}$$

and

$$Bd^-(S) = \{\sigma \in \mathcal{L} \setminus S \mid \text{ for all } \gamma \prec \sigma, \text{ we have } \gamma \in S\}.$$

The positive border of the theory is the set of its maximal elements, i.e., $MTh(\mathcal{L}, \mathbf{r}, q) = Bd^+(Th(\mathcal{L}, \mathbf{r}, q))$. Note that $Bd(S)$ can be small even for large S.

Above we assumed that the set S is closed downwards. We generalize the notation for sets S that are not closed downwards by simply defining that $Bd(S) = Bd(S')$ where S' is the downward closure of S. The generalization is similar for negative and positive borders.

Some straightforward lower bounds for the problem of finding all frequent sets are given in [2, 20]. Now we consider the problem of lower bounds in a more realistic model of computation.

The main effort in finding interesting sets is in the step where the interestingness of subgroups are evaluated against the database. Thus we consider the following model of computation. Assume the only way of getting information from the database is by asking questions of the form

Is-interesting Is the sentence φ interesting, i.e., does $q(\mathbf{r}, \varphi)$ hold?

Theorem 2 [19] *Any algorithm for computing $Th(\mathcal{L}, \mathbf{r}, q)$ that accesses the data using only Is-interesting queries must use at least $|Bd(Th(\mathcal{L}, \mathbf{r}, q))|$ queries.*

This result, simple as it seems, gives as a corollary a result about finding functional dependencies that in the more specific setting is not easy to find; cf. [17, 19]. Similarly, the corresponding verification problem requires at least this number of queries.

Problem 3 (Verification) *Given $\mathcal{L}$, $\mathbf{r}$, q, and a set $S \subseteq \mathcal{L}$. Verify that $S = MTh(\mathcal{L}, \mathbf{r}, q)$.*

Corollary 4 [19] *Given $\mathcal{L}$, $\mathbf{r}$, q, and a set $S \subseteq \mathcal{L}$, determining whether $S = MTh(\mathcal{L}, \mathbf{r}, q)$ requires in the worst case at least $|Bd(S)|$ evaluations of the predicate q, and it can be solved using exactly this number of evaluations of q.*

We now show that the verification problem is closely related to computing hypergraph transversals. A collection $\mathcal{H}$ of subsets of R is a (*simple*) *hypergraph*, if no element of $\mathcal{H}$ is empty and if $X, Y \in \mathcal{H}$ and $X \subseteq Y$ imply $X = Y$. The elements of $\mathcal{H}$ are called the *edges* of the hypergraph, and the elements of R are the *vertices* of the hypergraph. Given a simple hypergraph $\mathcal{H}$ on R, a *transversal* T of $\mathcal{H}$ is a subset of R intersecting all the edges of $\mathcal{H}$, that is, $T \cap E \neq \emptyset$ for all $E \in \mathcal{H}$.

[1] I.e., the positive border corresponds to the set "S" of [23].

Transversals are also called *hitting sets*. Here we consider *minimal transversals*: a transversal T of $\mathcal{H}$ is minimal if no $T' \subset T$ is a transversal. The collection of minimal transversals of $\mathcal{H}$ is denoted by $\mathrm{Tr}(\mathcal{H})$. It is a hypergraph on R.

Problem 5 (HTR) *Given a hypergraph $\mathcal{H}$, construct $Tr(\mathcal{H})$.*

For more information on hypergraphs see [4]. The computational problem of computing transversals appears in various branches of computer science; a comprehensive study of this problem is given by [8]. The HTR problem also appears in several forms in databases. In particular, the problem of translating between a set of functional dependencies and their corresponding Armstrong relation [16, 17] is at least as hard as this problem and equivalent to it in special cases [8]. Further discussion of these issues is given by [12, 18].

Notice that in general the output for this problem may be exponentially larger than its input, and thus the question is whether it can be solved in time polynomial in both its input size and output size. We say that an algorithm is *output* $T()$ *time* algorithm for the problem if it runs in time $T(I, O)$ where I is the input size, and O is the corresponding output size. A more strict condition, that we will use here, requires that the output transversals be enumerated, and that the time to compute the i'th transversal will be measured against the input size and i. That is, an algorithm solves the problem in *incremental* $T(I, i)$ *time* if the i'th transversal is computed in time $T(I, i)$. For further discussion and other variations see [8].

The exact complexity of the HTR problem is yet unknown. A sub-exponential solution for the problem has been recently discovered [10], and several special cases can be solved in polynomial time [8, 22]. We improve on one of these results here.

Now we return to the verification problem. Given $S \subseteq \mathcal{L}$, we have to determine whether $S = MTh(\mathcal{L}, \mathbf{r}, q)$ holds using as few evaluations of the interestingness predicate as possible.

Definition 6 (Representing as Sets) *Let $\mathcal{L}$ be the language, $\preceq$ a specialization relation, and R a set; denote by $\mathcal{P}(R)$ the powerset of R. A function $f : \mathcal{L} \to \mathcal{P}(R)$ is a* representation of $\mathcal{L}$ (and $\preceq$) as sets, *if f is one-to-one and surjective, f and its inverse are computable, and for all θ and φ we have $\theta \preceq \varphi$ if and only if $f(\theta) \subseteq f(\varphi)$.*

Thus, representing as sets requires that the structure imposed on $\mathcal{L}$ by $\preceq$ is isomorphic to a subset lattice. In particular, the lattice must be finite, and its size must be a power of 2. Note that frequent sets, functional dependencies with a fixed right-hand sides, and inclusion dependencies are easily representable as sets; the same holds for monotone Boolean functions. However, the language of [21] used for discovering episodes in sequences does not satisfy this condition.

Given S, we can compute $Bd^+(S)$ without looking at the data $\mathbf{r}$ at all: simply find the most special sentences in S. The negative border $Bd^-(S)$ is also determined by S, but finding the most general sentences in $\mathcal{L} \setminus S$ can be difficult. We now show how minimal transversals can be used in the task. Assume that (f, R) represents $\mathcal{L}$ as sets, and consider the hypergraph $\mathcal{H}(S)$ on R containing as edges the complements of sets $f(\varphi)$ for $\varphi \in Bd^+(S)$: $\mathcal{H}(S) = \{R \setminus f(\varphi) \mid \varphi \in Bd^+(S)\}$. Then $\mathrm{Tr}(\mathcal{H}(S))$ is a hypergraph on R, and hence we can apply f^{-1} to it: $f^{-1}(\mathrm{Tr}(\mathcal{H}(S))) = \{f^{-1}(H) \mid H \in \mathrm{Tr}(\mathcal{H}(S))\}$. We have the following.

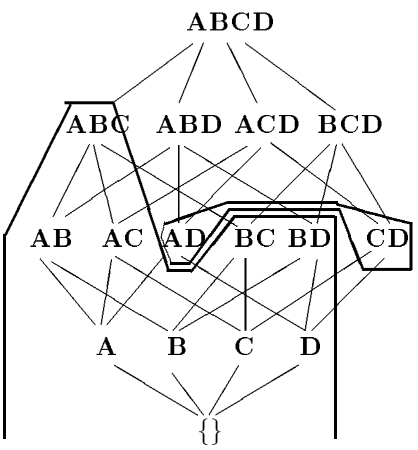

Figure 1: The interesting sentences and negative border in the problem of computing frequent sets.

Theorem 7 [19] $f^{-1}(Tr(\mathcal{H}(S))) = Bd^-(S)$.

Example 8 *Consider the problem of computing frequent sets, where* $R = \{A, B, C, D\}$*, and let* $S = \{ABC, BD\}$*, where we use a shorthand notation for sets, e.g., we represent* $\{A, B, C\}$ *by* ABC*. Then the downward closure of* S *is equal to* $\{ABC, AB, AC, BC, BD, A, B, C, D\}$*, and* S *includes the maximal elements. The negative border (that can be found by drawing the corresponding lattice, see Figure 1) is* $Bd^-(S) = \{AD, CD\}$*.*

For this problem we already have $\mathcal{L}$ *represented as sets and thus use the identity mapping* $f(X) = X$*, thus* $\mathcal{H}(S) = \{D, AC\}$*. It is easy to see that* $Tr(\{D, AC\}) = \{AD, CD\}$*, and thus* f^{-1} *indeed yields the correct answer.*

The requirement for representing as sets is quite strong. It is however necessary. In particular the mapping f must be surjective, that is, cover all of $P(R)$. Otherwise, after computing the transversal, a set may not have an inverse mapping to be applied in the last transformation in the theorem. This is indeed the case in the episodes of [21].

As we have seen, for languages representable as sets the notions of negative border and the minimal transversals give the same results. In the next sections we make use of this result to study the complexity of the data mining problem, computing hypergraph transversals, and also exact learning of monotone functions.

4 The Levelwise Algorithm

Several variants of the levelwise algorithm have been used before [2, 20, 21]. The algorithm solves the problem MaxTh by simply finding all interesting statements, i.e., the whole theory $Th(\mathcal{L}, \mathbf{r}, q)$ going bottom up. The method is as follows:

Algorithm 9 *The* levelwise algorithm *for finding all interesting statements.*
Input: *A database* $\mathbf{r}$*, a language* $\mathcal{L}$ *with specialization relation* $\preceq$*, and a quality predicate* q*.*
Output: *The set* $Th(\mathcal{L}, \mathbf{r}, q)$*.*
Method:

1. $C_1 := \{\varphi \in \mathcal{L} \mid$ *there is no* φ' *in* $\mathcal{L}$ *such that* $\varphi' \prec \varphi\}$;
2. $i := 1$;
3. **while** $C_i \neq \emptyset$ **do**
4. $\quad L_i := \{\varphi \in C_i \mid q(\mathbf{r}, \varphi)\}$;
5. $\quad C_{i+1} := \{\varphi \in \mathcal{L} \mid$ *for all* $\varphi' \prec \varphi$ *we have* $\varphi' \in \bigcup_{j \leq i} L_j\} \ \setminus \ \bigcup_{j \leq i} C_j$;
6. $\quad i := i + 1$;
7. **od**;
8. *output* $\bigcup_{j<i} L_j$;

The algorithm works iteratively, alternating between *candidate generation* and *evaluation* phases. First, in the generation phase of an iteration i, a collection C_i of new *candidate* sentences is generated, using the information available from more general sentences. Then the quality predicate is computed for these candidate sentences. The collection L_i will consist of the interesting sentences in C_i. In the next iteration $i+1$, candidate sentences in C_{i+1} are generated using the information about the interesting sentences in $\bigcup L_j$. Note that using the notion of border, Step 5 of the algorithm can be written as $C_{i+1} := Bd^-(\bigcup_{j \leq i} L_j) \setminus \bigcup_{j \leq i} C_j$.

The algorithm aims at minimizing the amount of database processing, i.e., the number of evaluations of q (Step 4). Note that the computation to determine the candidate collection does not involve the database (Step 5). For example, in computations of frequent sets Step 5 used only a negligible amount of time [2].

Clearly, by definition, the algorithm finds the maximal interesting sentences. Moreover, we show that under certain conditions the algorithm does not take too much time. The following theorem is immediate.

Theorem 10 *The levelwise algorithm computes the set of interesting sentences correctly, and it evaluates the predicate* q

$$|Th(\mathcal{L}, \mathbf{r}, q) \cup Bd^-(Th(\mathcal{L}, \mathbf{r}, q))|$$

times.

Example 11 *Consider, again, the problem of computing frequent sets where* $R = \{A, B, C, D\}$ *and* $MaxTh = \{ABC, BD\}$*, i.e., the situation of Figure 1. The levelwise algorithm works its way up from the bottom. It starts by evaluating the singletons* A, B, C*, and* D*; all of these are frequent. In the second iteration* C_2 *contains pairs of attributes such that both attributes are frequent, in this case all attribute pairs. Of them,* AB, AC, BC*, and* BD *are frequent.* C_3 *then contains such sets of size three whose all subsets are frequent, i.e., the set* ABC*, which is actually frequent. Notice that the negative border corresponds exactly to the sets that have been found not interesting along the way, that is the sets* AD *and* CD*.*

In order to further analyze the complexity we use the following notation. Denote by $\mathbf{rank}(\phi)$, the rank of a sentence ϕ, defined as follows. If for no $\theta \in \mathcal{L}$ we have $\theta \prec \phi$ then $\mathbf{rank}(\phi) = 0$. Otherwise, $\mathbf{rank}(\phi) = 1 + \max\{\mathbf{rank}(\theta) \mid \theta \prec \phi\}$. Denote by $dc(k)$ the maximal size of the downward closure of any sentence ϕ of rank $\leq k$. Also, by $\mathbf{width}(\mathcal{L}, \preceq)$ denote the maximal number of immediate successors on $\mathcal{L}$ and $\preceq$. That is,

$$\mathbf{width}(\mathcal{L}, \preceq) = \max_{\theta} |\{\phi \mid \theta \prec \phi \text{ and for no } \psi, \theta \prec \psi \prec \phi\}|.$$

Theorem 12 *Let* k *be the maximal rank over all interesting sentences in the problem* $(\mathcal{L}, \mathbf{r}, q)$*. The levelwise algorithm* *computes the set of interesting sentences correctly, and the number of queries it makes is bounded by*

$$dc(k)\,\mathbf{width}(\mathcal{L},\preceq)\,|MTh(\mathcal{L},\mathbf{r},q)|.$$

Proof. The number of sentences below any maximal element is bounded by $dc(k)$, and thus the number of elements not rejected from C_i at all stages together is bounded by $dc(k)|MTh(\mathcal{L},\mathbf{r},q)|$. Each of these sentences might create at most $\mathbf{width}(\mathcal{L},\preceq)$ new sentences for consideration in C_{i+1} that may be rejected (i.e. they are in $Bd^-(Th(\mathcal{L},\mathbf{r},q))$). □

This result holds for any $(\mathcal{L},\mathbf{r},q)$. For problems representable as sets one can derive more explicit bounds. In particular, in the problem of frequent sets the rank corresponds to the size of the set, the width is the number of attributes, and $dc(k)=2^k$. A standard assumption in practical applications is that the size of frequent sets is bounded. In these cases the levelwise algorithm is indeed efficient:

Corollary 13 *Let k be the size of the largest frequent set, and n the number of attributes. The levelwise algorithm computes the set of frequent sets correctly, and the number of queries it makes is bounded by $2^k n|MTh(\mathcal{L},\mathbf{r},q)|$.*

As a further corollary of the above we get that if the size of frequent sets is not too large then the size of $Bd^-(Th(\mathcal{L},\mathbf{r},q))$ is not prohibitive and thus the problem is feasible.

Corollary 14 *Let k be the size of the largest frequent set, and n the number of attributes.*
(i) For any k the size of sets in $Bd^-(Th(\mathcal{L},\mathbf{r},q))$ is bounded by $k+1$.
(ii) For $k=O(\log n)$, the size of $Bd^-(Th(\mathcal{L},\mathbf{r},q))$ is bounded by $O(n^{O(1)}|MTh(\mathcal{L},\mathbf{r},q)|)$.

We thus get an application for hypergraph transversals:

Corollary 15 *For $k=O(\log n)$, the problem of computing hypergraph transversals, where the edges of the input graph are all of size at least $n-k$, is solvable in input polynomial time by the levelwise algorithm.*
Proof. If the edge size is at least $n-k$, then the maximal sets that are not transversals are of size at most k. Set non-transversals to be "interesting" and use the algorithm. We get that the negative border is the required transversal hypergraph. □

This improves on previous result by [8] (Theorem 5.4) that show that this is possible for constant k (and uses a brute force enumeration algorithm using property (i) above). Notice that the levelwise algorithm does not use the structure original hypergraph directly. All it does is to test whether certain subsets are transversals of it or not.

5 The Dualize and Advance Algorithm

In this section we present and analyze an algorithm for computing all maximal interesting sentences that uses the connection between transversals and the computation of maximal elements. A variant of this *Dualize and Advance* algorithm has recently been used by [11], but no complexity analysis was provided there. This algorithm is far more applicable than the levelwise method, as this does not investigate all interesting statements, but rather jumps more or less directly to maximal ones. Thus it can be used even in the cases where not all interesting sentences are small. While the algorithm can be phrased for any $(\mathcal{L},\preceq)$, our analysis only holds for problems representable as sets, and we thus describe it in the restricted setting. To further simplify notation, we describe the algorithm in terms of finding interesting subsets of attributes (as is the case in computing frequent sets). The general case follows by using the representation as sets.

Algorithm 16 *The* Dualize and Advance *algorithm for finding all interesting statements.*
Input: *A database $\mathbf{r}$, a language $\mathcal{L}$ with specialization relation $\preceq$, and a quality predicate q.*
Output: *The set $MTh(\mathcal{L},\mathbf{r},q)$.*
Method:

1. $C_1 := \emptyset$;
2. $i := 1$;
3. $\overline{D_i} := \{$*complements of sets in* $C_i\}$;
4. *Use a subroutine to enumerate the minimal transversals of* $\overline{D_i}$;
5. *For each transversal X enumerated:*
6 *if X is interesting then declare X a counterexample and quit loop;*
7. *otherwise continue enumeration;*
8. *If all transversals were non-interesting, output C_i and exit;*
9. *Find a maximal superset Y of X that is interesting (using a greedy procedure that adds one attribute at a time);*
10. $C_{i+1} = C_i \cup \{Y\}$;
11. $i = i+1$;
12. *Go To 3;*

The algorithm works iteratively, finding a new maximal interesting set in each iteration. The procedure for finding new maximal interesting sets utilizes the idea of transversals. Once a number of interesting sentences has been found, the negative border of these sentences is computed using a transversal computation. Each of the elements of this negative border is checked to see whether it is interesting. If the set of interesting sentences found so far is complete, then all these sets are not interesting. If the set is not complete then at least one of the elements of the computed negative border is interesting. This interesting set can be extended to a new maximal interesting set by a series of queries, in a greedy manner.

Example 17 *Consider again the problem of computing frequent sets described in Figure 1.*

The dualize and advance algorithm starts with $C_1=\emptyset$, and $\overline{D_1}=\{ABCD\}$. The transversals are $Tr(\overline{D_1})=\{A,B,C,D\}$. Assume that in Step 6 the transversal A is tested first. Then A is found interesting and the algorithm continues in Step 9 to find a maximal element Y. This can be done by adding one attribute at a time, and testing whether q holds, and yields $Y=ABC$. In the next iteration $C_2=\{ABC\}$, $\overline{D_2}=\{D\}$, and $Tr(\overline{D_2})=\{D\}$. In Step 6 D is found to be interesting, and in Step 9 the algorithm finds that $Y=BD$ is maximal interesting. We therefore have $C_3=\{ABC,BD\}$, $\overline{D_3}=\{D,AC\}$, and $Tr(\overline{D_3})=\{AD,AC\}$. All the elements of the transversal are not interesting and therefore the algorithm stops. The set C_3 is exactly MTh and $Tr(\overline{D_3})$ is $Bd^-(MTh)$.

In order to establish correctness we start with a simple lemma:

Lemma 18 *For any iteration i of the algorithm, if $C_i \neq MTh(\mathcal{L}, \mathbf{r}, q)$ then at least one of the elements of $Tr(\overline{D_i})$ is interesting.*

Proof. First note that the elements of C_i are verified by the algorithm to be maximal interesting. We therefore have $C_i \subseteq MTh(\mathcal{L}, \mathbf{r}, q)$. Now if there is a set $c \in MTh(\mathcal{L}, \mathbf{r}, q) \setminus C_i$, then (since $\preceq$ is monotone) there is a minimal interesting set not in C_i, that is there is an interesting set in $Bd^-(C_i)$. (Just walk down in the lattice to find such a set).

As we saw earlier sets of attributes are already represented as sets and the identity mapping $f(X) = X$ is used. We thus get from Theorem 7 that $\mathrm{Tr}(\overline{D_i}) = Bd^-(S)$ and one of these elements is interesting. □

The question is how many sets X should be enumerated before finding such a counterexample on the negative border. The following example shows that there are cases where the size of $MTh(\mathcal{L}, \mathbf{r}, q)$ and its negative border are small, but in an intermediate step the size of the negative border of C_i may be large.

Example 19 [16] *Consider the case where $MTh = MTh(\mathcal{L}, \mathbf{r}, q)$ includes all sets of size $n-2$, and $Bd^-(MTh)$ thus includes all sets of size $n-1$. Further consider the case where C_i is such that $\overline{D_i} = \{\{x_{2i-1}, x_{2i}\} \mid 1 \leq i \leq n/2\}$. Then the size of $Tr(\overline{D_i})$ is $2^{n/2}$ while $Bd^-(MTh)$ is small.*

Nevertheless, as the following lemma shows, the number of sets that have to be enumerated is not too large.

Lemma 20 *For any iteration i of the algorithm, if $C_i \neq MTh(\mathcal{L}, \mathbf{r}, q)$ then the number of sets enumerated before a counterexample set X is found is bounded by $|Bd^-(MTh(\mathcal{L}, \mathbf{r}, q))|$.*

Proof. Denote $Bd^- = Bd^-(MTh(\mathcal{L}, \mathbf{r}, q))$. We show that each set X enumerated either exactly matches an element of Bd^- exactly, or is interesting. In other words the set X cannot be both not interesting, and a strict superset of an element in Bd^-. It follows that at most $|Bd^-|$ elements need to be enumerated.

To prove the claim notice that every set that is deemed interesting by the C_i is indeed interesting. If X is both not interesting, and a strict superset of an element Z in Bd^-, then Z, which is not interesting, is claimed interesting by C_i (since $X \in Bd^-(C_i)$ is minimal, and $Z \subset X$); a contradiction. □

We need the following notation. Let $\Sigma \subseteq \mathcal{L}$, by $\mathbf{rank}(\Sigma)$ we denote the maximal rank over all sentences in Σ.

$$\mathbf{rank}(\Sigma) = \max_{\theta \in \Sigma} \quad \mathbf{rank}(\theta)$$

Theorem 21 *If there is an incremental T(I,i) time algorithm for computing hypergraph transversals then $MTh = MTh(\mathcal{L}, \mathbf{r}, q)$ can be computed in time polynomial in $|MTh|$ and $T(|MTh|, |Bd^-(MTh)|)$, while using at most $|MTh| \cdot (|Bd^-(MTh)| + \mathbf{rank}(MTh)\mathbf{width}(\mathcal{L}, \preceq))$ queries.*

Proof. The bound follows by using the dualize and Advance algorithm. By Lemma 18, each iteration finds a new maximal set, and therefore the algorithm is correct, and the number of iterations is $|MTh|$. By Lemma 20, in each iteration the algorithm runs a transversals subroutine enumerating at most $|Bd^-(MTh)|$ sets, and asking the same number of queries before finding a counter example. The extension of the counter example X into a maximal set Y requires at most $\mathbf{rank}(MTh)$ stages each with at most $\mathbf{width}(\mathcal{L}, \preceq)$ queries. □

Thus we see that the connection to hypergraph transversals holds not only for the verification problem but also for the generation problem. We note that for the case of functional dependencies with fixed right hand side, and for keys, even simpler algorithms can be used [16, 12]. In this case one can access the database and directly compute $Bd^-(MTh)$ (according to the appropriate representation as sets, this corresponds to the so called agree sets of the relation). Then a single run of an HTR subroutine suffices. The current result holds even if the access to the database is restricted to "Is-interesting" queries.

Recently, Fredman and Khachiyan [10] presented an incremental algorithm for the HTR problem with time complexity $T(I, i) = (I+i)^{O(\log(I+i))}$. We can therefore conclude the following:

Corollary 22 *For any problem representable as sets, $MTh(\mathcal{L}, \mathbf{r}, q)$ can be computed in time $t(|MTh| + |Bd^-(MTh)|)$, where $t(n) = n^{O(\log n)}$, while using at most $|MTh| \cdot (|Bd^-(MTh)| + \mathbf{rank}(MTh)\mathbf{width}(\mathcal{L}, \preceq))$ queries.*

6 Relation to Learning Theory

We now show that the problems discussed above are very closely related to problems in learning theory. The simple scenario discussed in learning theory is as follows: a Boolean function $f : \{0,1\}^n \to \{0,1\}$ is fixed by some adversary (modeling a concept in the world). A learner is given access to some oracle giving it partial information on the function f. The task of the learner is to find a representation for a Boolean function that is identical (or approximates) f. In particular we will consider the model of exact learning with membership queries [3].

A membership query oracle $MQ(f)$ allows the learner to ask for the value of f on a certain point. That is, given $x \in \{0,1\}^n$, $MQ(f)$ returns the value $f(x)$. The learning algorithm is given access to $MQ(f)$, and the algorithm is required to produce an expression that computes f exactly.

Definition 23 *An algorithm is an exact learning algorithm with time complexity $T()$, query complexity $Q()$, and representation class $\mathcal{H}$, for a class of functions $\mathcal{F}$, if for all $f \in \mathcal{F}$, when given access to $MQ(f)$, the algorithm runs in time $T()$, asks $MQ()$ on at most $Q()$ points and then outputs a representation $h \in \mathcal{H}$ for a Boolean function such that h is equivalent to f.*

In the above definition we omitted the parameters of the functions $T()$ and $Q()$. Normally the algorithm is allowed time polynomial in the number of variables n, and the size of representing f in some representation language.

In particular we next consider the problem of learning *monotone* functions with membership queries. A function f is monotone if $f(x) = 1$, and $y \geq x$ implies $f(y) = 1$, where $\leq$ is the normal partial order on $\{0,1\}^n$. We also consider the standard CNF and DNF representations for such functions. A term is a conjunction of literals, e.g. $x_1 x_2$ is a term. A DNF expression is a disjunction of terms, e.g. $x_1 x_2 \vee x_2 x_3$ is a DNF expression. Similarly a CNF expression is a conjunction of disjunctions, e.g. $(x_1 \vee x_2)(x_2 \vee x_3)$ is a CNF expression. It is well known that monotone functions have unique minimum size representations in both DNF and CNF, that include all minimal terms or clauses respectively of the function. (A minimal term, called a prime implicant, is a term that implies f and such that every subset of it does not imply f.)

In the scenario that follows the learning algorithm is allowed time relative to the number of attributes n, and the sum of sizes of its DNF and CNF representations. That is, we consider $T(m)$, and $Q(m)$ where $m = n + |DNF(f)| + |CNF(f)|$.

The correspondence between learning monotone functions and computing interesting sets is thus straightforward. The elements of $\{0,1\}^n$ correspond to subsets of the variables so that a value 1 implies that the corresponding attribute is in the set. The value of the function on an assignment corresponds to the negation of the interestingness relation q. Since q is monotone, the function is monotone. Membership queries now naturally correspond to Is-interesting queries. We therefore get:

Theorem 24 *The problem of computing interesting sentences for problems representable as sets is equivalent to the problem of learning monotone functions with membership queries, with representation class CNF (or DNF).*

Example 25 *The problem of computing frequent sets described in Figure 1 is mapped to the problem of learning the function f whose DNF representation is $f = AD \vee CD$ and whose CNF representation is $f = (A \vee C)(D)$. The terms of the DNF correspond to the elements of Bd^-, and the clauses of the CNF are the complements of the sets in MTh.*

As an immediate corollary of the results in Section 4 We get:

Corollary 26 *The levelwise algorithm can be used to learn the class of monotone CNF expressions where each clause has at least $n-k$ attributes and $k = O(\log n)$, in polynomial time, and with a polynomial number of membership queries.*

As a corollary of Theorem 2 we get a lower bound:

Corollary 27 *Any algorithm that learns monotone functions with membership queries must use at least $|DNF(f)| + |CNF(f)|$ queries.*

While the bound is not surprising, it explains the lower bound given by Angluin [3]. It is shown there that an algorithm may need to take time exponential in the DNF size when not allowed CNF size as a parameter. Indeed the CNF size of the function used to show the lower bound is exponential. (The lower bound in [3] is, however, more complex since the learner has access to several additional oracles.) On the other hand, by using Theorem 21 we see that with membership queries alone one can come close to this lower bound.

Corollary 28 *If there is an incremental T(I,i) time algorithm for computing hypergraph transversals then there is a learning algorithm for monotone functions with membership queries, that produces both a DNF and a CNF representation for the function. The number of MQ queries is bounded by $|CNF(f)| \cdot (|DNF(f)| + n^2)$. The running time of the algorithm is polynomial in n and $T(|CNF(f)|, |DNF(f)|)$.*

We note that this corollary can be derived from a more general construction in [5] (Theorem 18) that studies the complexity of learning, and uses NP-Oracles. For monotone functions, the NP-Oracle used there, can be replaced with a procedure for HTR, yielding a similar result. This has been previously observed [24]. The algorithm that results is similar to the dualize and advance algorithm but is slightly more complicated.

Here again using the result of Fredman and Khachiyan [10] we can derive a sub-exponential learning algorithm for this problem.

Corollary 29 *There is a learning algorithm for monotone functions with membership queries. The running time of the algorithm is bounded by $T(m)$, where $m = |DNF(f)| + |CNF(f)|$, and $T(m) = m^{O(\log m)}$. The number of MQ queries is bounded by $|CNF(f)| \cdot (|DNF(f)| + n^2)$.*

The relation to hypergraph transversals can also be used in the other direction. However, since the learning problem is not defined as an incremental task, the next corollary only yields an output $T()$ time algorithm.

Corollary 30 *If there is a learning algorithm for monotone functions, that produces a DNF representation, and whose running time and number of queries is bounded by $T(m)$, where m is as above, then there is an output $T()$ time algorithm for the hypergraph transversal problem.*

Acknowledgments

We wish to thank Eyal Kushilevitz for pointing out the work in [5], and Christino Tamon for useful discussions.

References

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD'93)*, pages 207 – 216, May 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*, pages 307 – 328. AAAI Press, Menlo Park, CA, 1996.

[3] D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319 – 342, Apr. 1988.

[4] C. Berge. *Hypergraphs. Combinatorics of Finite Sets.* North-Holland Publishing Company, Amsterdam, 3rd edition, 1973.

[5] N. H. Bshouty, R. Cleve, R. Gavalda, S. Kannan, and C. Tamon. Oracles and queries that are sufficient for exact learning. *Journal of Computer and System Sciences*, 52:421 – 433, 1996.

[6] L. De Raedt and M. Bruynooghe. A theory of clausal discovery. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI–93)*, pages 1058 – 1053, Chambéry, France, 1993. Morgan Kaufmann.

[7] L. De Raedt and S. Džeroski. First-order jk-clausal theories are PAC-learnable. *Artificial Intelligence*, 70:375 – 392, 1994.

[8] T. Eiter and G. Gottlob. Identifying the minimal transversals of a hypergraph and related problems. *SIAM Journal on Computing*, 24(6):1278 – 1304, Dec. 1995.

[9] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery: An overview. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 1 – 34. AAAI Press, Menlo Park, CA, 1996.

[10] M. Fredman and L. Khachiyan. On the complexity of dualization of monotone disjunctive normal forms. *Journal of Algorithms*, 21:618 – 628, 1996.

[11] D. Gunopulos, H. Mannila, and S. Saluja. Discovering all most specific sentences by randomized algorithms. In *Proceedings of the International Conference on Database Theory (ICDT'97)*, pages 215 – 229, Delphi, Greece, Jan. 1997.

[12] R. Khardon. Translating between Horn representations and their characteristic models. *Journal of AI Research*, 3:349 – 372, 1995.

[13] J.-U. Kietz and S. Wrobel. Controlling the complexity of learning in logic through syntactic and task-oriented models. In S. Muggleton, editor, *Inductive Logic Programming*, pages 335 – 359. Academic Press, London, 1992.

[14] W. Kloesgen. Efficient discovery of interesting statements in databases. *Journal of Intelligent Information Systems*, 4(1):53 – 69, 1995.

[15] P. Langley. *Elements of Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1995.

[16] H. Mannila and K.-J. Räihä. Design by example: An application of Armstrong relations. *Journal of Computer and System Sciences*, 33(2):126 – 141, 1986.

[17] H. Mannila and K.-J. Räihä. *Design of Relational Databases*. Addison-Wesley Publishing Company, Wokingham, UK, 1992.

[18] H. Mannila and K.-J. Räihä. Algorithms for inferring functional dependencies. *Data & Knowledge Engineering*, 12(1):83 – 99, Feb. 1994.

[19] H. Mannila and H. Toivonen. On an algorithm for finding all interesting sentences. In *Cybernetics and Systems, Volume II, The Thirteenth European Meeting on Cybernetics and Systems Research*, pages 973 – 978, Vienna, Austria, Apr. 1996. Extended version available as: Levelwise search and borders of theories in knowledge discovery, Report C-1997-8, University of Helsinki, Department of Computer Science.

[20] H. Mannila, H. Toivonen, and A. I. Verkamo. Efficient algorithms for discovering association rules. In *Knowledge Discovery in Databases, Papers from the 1994 AAAI Workshop (KDD'94)*, pages 181 – 192, Seattle, Washington, July 1994.

[21] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovering frequent episodes in sequences. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD'95)*, pages 210 – 215, Montreal, Canada, Aug. 1995.

[22] N. Misra and L. Pitt. On bounded-degree hypergraph transversals. Manuscript., 1995.

[23] T. M. Mitchell. Generalization as search. *Artificial Intelligence*, 18:203 – 226, 1982.

[24] C. Tamon. Private communication. 1997.