Fundamenta Informaticae 80 (2007) 415–433
IOS Press

A Data Mining Formalization to Improve Hypergraph Minimal Transversal Computation

Céline Hébert[*†], **Alain Bretto**[*], **Bruno Crémilleux**[*]
Department of Informatics
University of Caen, France
celine.hebert@info.unicaen.fr
alain.bretto@info.unicaen.fr
bruno.cremilleux@info.unicaen.fr

Abstract. Finding hypergraph transversals is a major algorithmic issue which was shown having many connections with the data mining area. In this paper, by defining a new Galois connection, we show that this problem is closely related to the mining of the so-called *condensed representations* of frequent patterns. This data mining formalization enables us to benefit from efficient algorithms dedicated to the extraction of condensed representations. More precisely, we demonstrate how it is possible to use the levelwise framework to improve the hypergraph minimal transversal computation by exploiting an anti-monotone constraint to safely prune the search space. We propose a new algorithm MTMINER to extract minimal transversals and provide experiments showing that our method is efficient in practice.

Keywords: Data mining, Hypergraph minimal transversals, Levelwise framework, Condensed representations.

1. Introduction

Hypergraphs can be thought as a generalization of graphs where the edges, called hyperedges, connect more than two vertices. A *minimal transversal* is a set of vertices intersecting all the hyperedges which

Address for correspondence: GREYC, CNRS - UMR 6072, Campus Côte de Nacre, boulevard du Maréchal Juin, BP 5186, 14032 Caen Cedex
[*]Thank the anonymous referee for detailed comments and helpful suggestions.
[†]Thanks Khaled Elbassioni for his help in using the DUAL prototype.

is minimal with respect to the inclusion. Computing hypergraph minimal transversals is a major research question because having many applications in computer science [9]. However, this is a difficult task from the complexity point of view partly because the number of minimal transversals can be exponentially large in the size of the input hypergraph. The complexity of finding all the minimal transversals of a hypergraph still remains an open issue and it is not known yet if an output polynomial total time algorithm exists. To alleviate the writing, we generally omit the word minimal and we say hypergraph transversals for hypergraph minimal transversals in this paper.

The hypergraph transversal problem has many connections with data mining issues [13]. Mannila and Toivonen proved that there is a close relationship between the borders of theories (see Section 2.1 for a short definition) and the minimal transversals [19]. It has been shown that minimal transversals are linked to useful patterns called emerging patterns which highlight contrasts between classes [1]. Recently, minimal transversals have been efficiently used to relate different clustering results for visualizing transactional data for knowledge discovery [8].

In this paper, we revisit the search of minimal transversals by taking advantage of recent progress on pattern condensed representations [5]. Let us briefly introduce basic background on condensed representations. A pattern condensed representation gives a synthesis of large datasets and highlights the correlations embedded in the data. One advantage of this approach is to provide powerful safe pruning criteria during the mining and thus improve the efficiency of algorithms. Usually, pattern condensed representations address the minimal frequency constraint even if there are also condensed representations for frequency-based measures [22]. A pattern is *frequent* if it occurs in the dataset more than a user-specified threshold. Condensed representations of frequent patterns restrict the mining to specific patterns like the free (or key) patterns or the closed patterns [4, 20]. These patterns partition the search space in equivalence classes, the free patterns are their minimal elements (with respect to the pattern inclusion) and the closed patterns are their maximal elements. In practice, mining either all the free patterns or the closed patterns is enough to infer the frequency of any pattern. *Freeness* has one interesting property: its anti-monotonicity with respect to the pattern inclusion (a constraint q is *anti-monotone* if and only if for all patterns X and Y, $q(X)$ and $Y \subseteq X$ implies $q(Y)$). It gives a safe pruning criterion for levelwise search in the pattern lattice [19, 4].

Our key idea is to reuse the concept of condensed representation (more precisely the freeness) for minimal transversal computation. Our main contributions are twofold. First, thanks to the definition of a new Galois connection, we set the hypergraph transversal issue in the levelwise framework. From this formalization, we deduce an anti-monotone property which enables us to safely prune the search space to get minimal transversals. Second, we provide a new algorithm MTMINER that efficiently computes all the minimal transversals of a given hypergraph. Thus, we experimentally demonstrate that a data mining algorithm can solve instances better than the algorithm [11] with the best theoretical complexity for the hypergraph transversal problem.

The rest of the paper is organized as follows. Section 2 discusses related work, summarizes the main points of this work and gives preliminary definitions. Section 3 presents the core of our method to compute hypergraph minimal transversals. The definition of a hypergraph Galois connection and an extension-based characterization of minimal transversals are given. Section 4 details the pruning strategy based on an anti-monotone property to reduce the search space and make the minimal transversal computation efficient. Our algorithm MTMINER, based on the above mentioned Galois connection, is presented in Section 5. Section 6 gives an experimental evaluation of MTMINER. We end this paper by a discussion about further research in Section 7.

2. Context

2.1. Background

Considerable efforts have been devoted to the search of minimal transversals [2, 11, 1, 16]. In [11], Fredman and Khachiyan give an algorithm with a time complexity equal to $P(n) + s^{\log(s)}$ where n is the number of vertices of the input hypergraph, P is a polynomial and s is the combined size of the input and the output. It is, to the best of our knowledge, the algorithm having the lowest complexity for finding the minimal transversals of a given hypergraph. Many works [6, 23, 17] focus on the case when the hyperedge size is bounded by a constant. In this paper, we make no restriction on the input hypergraph. Due to the gap between theoretical complexity and practical use, other works also address implementation [3] and experimental evaluation [15] issues for hypergraph transversal algorithms.

Gunopulos et al. [12] point out the fact that it is possible to compute the hypergraph minimal transversals in a levelwise manner. However, no advantage is taken from condensed representations, no specific algorithm or implementation are provided and no experimental result is given. As mentioned in [1], being a minimal transversal is not an anti-monotone property. This is what prevents levelwise algorithms from being efficient. In this paper, we solve this problem by providing an anti-monotone constraint that can be used to efficiently mine the minimal transversals of a hypergraph.

In the field of knowledge discovery, Mannila and Toivonen [19] define a theory as the set of patterns satisfying a given property or constraint. The maximal elements of a theory and the minimal elements which do not belong to the theory constitute the positive and the negative border of this theory. Mannila and Toivonen also relate the borders of a theory to the minimal transversals in [19]. They prove that the negative border of a theory equals the minimal transversals of the hypergraph composed of the complements of the patterns of the positive border. By exploiting this property (as a stopping criterion), [12, 21] propose to compute the positive border of the frequent patterns by using minimal transversals in depth-first algorithms. We will see in Section 6 how our method can be useful for improving the computation of minimal transversals in such cases.

2.2. Preliminaries

2.2.1. Main ideas

Figure 1 outlines our approach. Considering that both databases and hypergraphs can be represented as boolean matrices, our proposal is to benefit from links between attribute patterns and sets of vertices to apply a data mining algorithm in order to efficiently compute hypergraph minimal transversals. More precisely, we exhibit a Galois connection on hypergraphs which allows to exploit an anti-monotone property. Afterwards, we reuse the principle of the algorithm described in [14] which is based on a levelwise search. We make use of efficient pruning criteria and we show that our algorithm is efficient on hypergraph hard instances. The originality of this work is to use a pattern extraction algorithm to compute transversals contrary to the approaches exposed in [12, 21] that aim at discovering interesting patterns by means of minimal transversals.

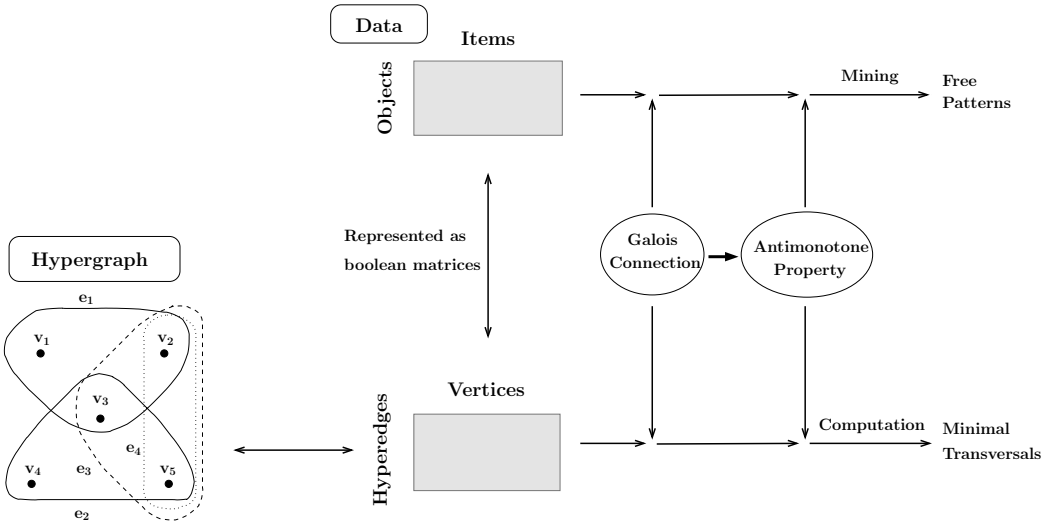


Figure 1. Overview of our approach.

2.2.2. Definitions

This section gives basic definitions for the three fields approached in this article: databases, hypergraphs and Galois connections.

Databases A database $\mathcal{D}$ is a relation $\mathcal{R}$ between a set $\mathcal{A}$ of *attributes* and a set $\mathcal{O}$ of *objects*: for $a \in \mathcal{A}, o \in \mathcal{O},\ a\,\mathcal{R}\,o$ if and only if the object o contains the attribute a. Table 1 provides an example of a database with 5 attributes and 6 objects. A *pattern* is a subset of $\mathcal{A}$. The frequency of a pattern X is the number of objects in $\mathcal{D}$ containing X; it is denoted by $\mathcal{F}(X)$. A pattern X is γ-*frequent* if its frequency is greater than or equal to γ. For instance, the pattern $\{a_1, a_3\}$ is 2-frequent because it appears in two objects (o_1 and o_6). X is a *free* pattern if for each $X_1 \subset X,\ \mathcal{F}(X) < \mathcal{F}(X_1)$. In Table 1, $\{a_1, a_3\}$ is not free since $\mathcal{F}(\{a_1\}) = 2 = \mathcal{F}(\{a_1, a_3\})$. On the contrary, the pattern $\{a_3, a_5\}$ is free because $\mathcal{F}(\{a_3\}) = 4 > \mathcal{F}(\{a_3, a_5\}) = 2$ and $\mathcal{F}(\{a_5\}) = 3 > \mathcal{F}(\{a_3, a_5\}) = 2$.

		Attributes				
		a_1	a_2	a_3	a_4	a_5
Objects	o_1	1	1	1	0	0
	o_2	0	0	1	1	1
	o_3	0	1	1	0	1
	o_4	0	1	0	0	1
	o_5	0	1	0	1	0
	o_6	1	0	1	0	0

Table 1. An example of a database $\mathcal{D}$.

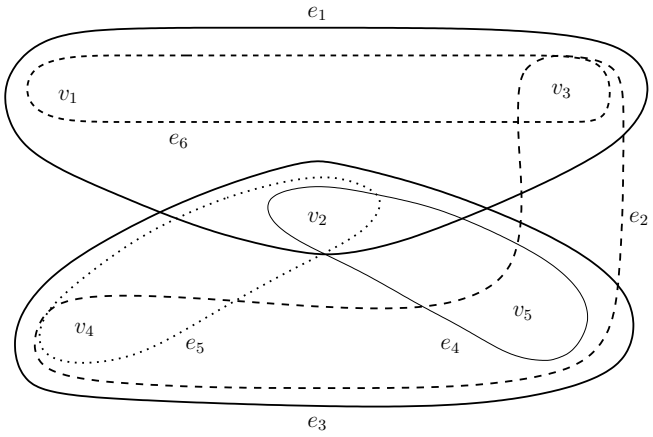
Hypergraphs A *hypergraph* $\mathcal{H}$ is a pair $(\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = \{v_1, v_2, ..., v_n\}$ is a finite set of vertices and $\mathcal{E} = (e_i)_{i \in \{1,2,...,m\}}$ a set of nonempty subsets of $\mathcal{V}$ called hyperedges such that

$$\bigcup_{1 \leq i \leq m} e_i = \mathcal{V}.$$

An example of a hypergraph both drawn and represented by its adjacency matrix is given in Figure 2. A subset T of $\mathcal{V}$ is a *transversal* of $\mathcal{H}$ if T intersects each hyperedge of $\mathcal{E}$. We denote by $Tr(\mathcal{H})$ the set of all transversals of $\mathcal{H}$:

$$Tr(\mathcal{H}) = \{T \subseteq \mathcal{V} | \text{ for all } i \in \{1, 2, ..., m\},\ e_i \cap T \neq \emptyset\}.$$

For instance, $\{v_2, v_3, v_5\}$ is a transversal of the hypergraph represented in Figure 2. A subset T of $\mathcal{V}$ is a *minimal transversal* of $\mathcal{H}$ if $T \in Tr(\mathcal{H})$ and if no proper subset of T is in $Tr(\mathcal{H})$. The set of all minimal transversals of $\mathcal{H}$ is denoted by $MinTr(\mathcal{H})$. $(\mathcal{V}, MinTr(\mathcal{H}))$ is a hypergraph (see [2, p. 43]) called the *transversal hypergraph* of $\mathcal{H}$. The maximal size of a minimal transversal of $\mathcal{H}$ is denoted by $t(\mathcal{H})$, i.e., $t(\mathcal{H}) = \max_{T \in MinTr(\mathcal{H})} |T|$. In Figure 2, $\{v_2, v_3, v_5\}$ is not a minimal transversal since $\{v_2, v_3\}$ is a transversal too. $\{v_2, v_3\}$ is a minimal transversal because neither $\{v_2\}$ nor $\{v_3\}$ is a transversal.



Vertices

Hyperedges

	v_1	v_2	v_3	v_4	v_5
e_1	1	1	1	0	0
e_2	0	0	1	1	1
e_3	0	1	1	0	1
e_4	0	1	0	0	1
e_5	0	1	0	1	0
e_6	1	0	1	0	0

Figure 2. An example of a hypergraph $\mathcal{H}$.

Let $V \subseteq \mathcal{V}$ be a subset of vertices and $E \subseteq \mathcal{E}$ be a subset of hyperedges. We define the set of hyperedges which contain at least one vertex of V:

$$\mathcal{E}(V) = \{e \in \mathcal{E} | \ \exists v \in V \text{ with } v \in e\}$$

and the set of vertices which appear at least in one hyperedge of E:

$$\mathcal{V}(E) = \{v \in \mathcal{V} | \ \exists e \in E \text{ with } v \in e\}.$$

In Figure 2, $\mathcal{E}(\{v_2, v_5\}) = \{e_1, e_2, e_3, e_4, e_5\}$ and $\mathcal{V}(\{e_4, e_5\}) = \{v_2, v_4, v_5\}$.

It should be underlined that both datasets and hypergraphs can be represented as boolean matrices.

Galois connections Let $(A, \leq_1)$ and $(B, \leq_2)$ be two partially ordered sets. In the rest of the paper, we will denote the poset $(A, \leq_1)$ by A if there is no ambiguity about the partial order $\leq_1$. A *Galois connection* between $(A, \leq_1)$ and $(B, \leq_2)$ consists of two antitone functions $f : A \rightarrow B$ and $g : B \rightarrow A$ such that $f \circ g$ and $g \circ f$ are extensive, i.e., for all $y \in B,\ y \leq_2 f \circ g(y)$ and for all $x \in A,\ x \leq_1 g \circ f(x)$. f is the *intension* and g the *extension*.

When used for pattern extraction tasks, the sets $2^{\mathcal{A}}$ and $2^{\mathcal{O}}$ are ordered by inclusion and the Galois connection $(f_{\mathcal{D}}, g_{\mathcal{D}})$ is defined as follows: for all $O \subseteq \mathcal{O},\ f_{\mathcal{D}}(O) = \{a \in \mathcal{A} |$ for all $o \in O, a\mathcal{R}o\}$ and for all $A \subseteq \mathcal{A},\ g_{\mathcal{D}}(A) = \{o \in \mathcal{O} |$ for all $a \in A, a\mathcal{R}o\}$. In Table 1, the extension of $\{a_1, a_3\}$ is $\{o_1, o_6\}$; the intension of $\{o_3, o_4, o_5\}$ is $\{a_2\}$.

Property 1 enables us to compute the extension when joining several subsets of B and is given as an exercise in [7, p. 45].

Property 1. Let (f, g) be a Galois connection between the posets $(2^A, \leq_1)$ and $(2^B, \leq_2)$. Then for any family $\{V_j \subset B | j \in \mathcal{J}\}$:

$$g(\bigcup_{j \in \mathcal{J}} V_j) = \bigcap_{j \in \mathcal{J}} g(V_j).$$

For instance in Table 1, thanks to Property 1, we have $g_{\mathcal{D}}(\{a_1, a_2, a_3\}) = g_{\mathcal{D}}(\{a_1, a_3\}) \cap g_{\mathcal{D}}(\{a_2\})$ and thus $g_{\mathcal{D}}(\{a_1, a_2, a_3\}) = \{o_1\}$ since $g_{\mathcal{D}}(\{a_2\}) = \{o_1, o_3, o_4, o_5\}$.

3. Extension-based approach for computing hypergraph transversals

This section presents the core of our approach to compute minimal transversals. We start by defining a hypergraph Galois connection, then we specify the key points to build our algorithm: the link between hypergraphs and pattern extraction, and a characterization of the minimal transversals according to the previously mentioned Galois connection.

3.1. Hypergraph Galois operators

We define a new Galois connection $(f_{\mathcal{H}}, g_{\mathcal{H}})$ associated to a hypergraph $\mathcal{H}$. We will see in Section 4 that such a connection is required to apply both an anti-monotone constraint and the levelwise framework to the minimal transversal computation. In the rest of the paper, the sets $2^{\mathcal{V}}$ and $2^{\mathcal{E}}$ are ordered by inclusion $\subseteq$.

Definition 3.1. The Galois operators $f_{\mathcal{H}}$ and $g_{\mathcal{H}}$ associated to the hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ are:

$$\text{for all } E \in 2^{\mathcal{E}},\ f_{\mathcal{H}}(E) = \mathcal{V} \backslash \mathcal{V}(E)$$

$$\text{for all } V \in 2^{\mathcal{V}},\ g_{\mathcal{H}}(V) = \mathcal{E} \backslash \mathcal{E}(V)$$

For a set of hyperedges E, $f_{\mathcal{H}}(E)$ contains all the vertices that do not appear in any hyperedge of E. For a set of vertices V, $g_{\mathcal{H}}(V)$ corresponds to the hyperedges that do not contain any vertex of V. Basically, this is equivalent to saying that V is a transversal of the partial hypergraph defined by the hyperedges

Database with the connection $(f_{\mathcal{D}}, g_{\mathcal{D}})$	Any Galois connection (f, g)	Hypergraph with the connection $(f_{\mathcal{H}}, g_{\mathcal{H}})$
$(2^{\mathcal{A}}, \subseteq)$ the power set of the attributes	the poset $(B, \geq_2)$	$(2^{\mathcal{V}}, \subseteq)$ the power set of the vertices
$(2^{\mathcal{O}}, \subseteq)$ the power set of the objects	the poset $(A, \geq_1)$	$(2^{\mathcal{E}}, \subseteq)$ the power set of the hyperedges
$g_{\mathcal{D}}$: the objects containing a given attribute pattern	g: the extension	$g_{\mathcal{H}}$: the hyperedges not covered by a given vertex pattern
the attribute patterns appearing in the same objects	equivalence class	the vertex patterns covering the same hyperedges
the missing patterns	the sets with an empty extension	the transversals of $\mathcal{H}$
the free patterns	the minimal elements of the equivalence classes	the minimal transversals of a partial hypergraph
the negative border of the present patterns	the minimal elements of the equivalence class with an empty extension	the minimal transversals of $\mathcal{H}$

Table 2. Correspondence between databases, Galois connections and hypergraphs.

$\mathcal{E} \backslash g_{\mathcal{H}}(V) = \mathcal{E}(V)$. In Figure 2, the extension of $\{v_1, v_3\}$ is $\{e_4, e_5\}$ because neither v_1 nor v_3 intersects e_4 or e_5; the intension of $\{e_1, e_3\}$ is $\{v_4\}$ since neither e_1 nor e_3 contains v_4.

Lemma 3.1. The pair $(f_{\mathcal{H}}, g_{\mathcal{H}})$ defines a Galois connection between the posets $(2^{\mathcal{E}}, \subseteq)$ and $(2^{\mathcal{V}}, \subseteq)$.

Proof:
Let V_1 and V_2 be in $2^{\mathcal{V}}$, E_1 and E_2 be in $2^{\mathcal{E}}$ such that $V_1 \subseteq V_2$ and $E_1 \subseteq E_2$.

It is obvious that the hyperedges in $g_{\mathcal{H}}(V_2)$ do not contain any vertex of V_1 so $g_{\mathcal{H}}(V_2) \subseteq g_{\mathcal{H}}(V_1)$ and $g_{\mathcal{H}}$ is antitone. Similarly, we obtain that $f_{\mathcal{H}}$ is antitone.

By definition, the vertices in V_1 do not appear in the hyperedges of $g_{\mathcal{H}}(V_1)$ thus $V_1 \subseteq f_{\mathcal{H}} \circ g_{\mathcal{H}}(V_1)$ and $f_{\mathcal{H}} \circ g_{\mathcal{H}}$ is extensive. Similarly, we prove that $g_{\mathcal{H}} \circ f_{\mathcal{H}}$ is extensive too. □

Specifying the appropriate Galois connection $(f_{\mathcal{H}}, g_{\mathcal{H}})$ for the hypergraph case enables us to make an analogy between databases and hypergraphs, thus to adapt a pattern extraction algorithm for minimal transversal computation. The next section details this relation.

3.2. Extension and hypergraph transversals

This section characterizes the minimal transversals of a hypergraph thanks to the extension defined in Section 3.1 and links hypergraphs to databases. Table 2 establishes a correspondence between terms in Galois connections, databases and hypergraphs. To better understand the analogy with databases, we will refer to the subsets of $\mathcal{V}$ as *vertex patterns*.

It should be noticed that the extension $g_{\mathcal{H}}$ induces an equivalence relation on $2^{\mathcal{V}}$: for all $V, V' \subset \mathcal{V}$, $V \sim V' \Leftrightarrow g_{\mathcal{H}}(V) = g_{\mathcal{H}}(V')$. This relation enables us to define extension equivalence classes in the partially ordered set $2^{\mathcal{V}}$ (see Definition 3.2). Defined in pattern extraction, such equivalence classes are called frequency classes because the patterns belonging to the same equivalence class have the same frequency.

Definition 3.2. The equivalence class of a vertex pattern V is denoted by $\mathcal{R}_{g_{\mathcal{H}}}(V)$ and is defined as follows:

$$\mathcal{R}_{g_{\mathcal{H}}}(V) = \{V' \in \mathcal{V} | g_{\mathcal{H}}(V') = g_{\mathcal{H}}(V)\}.$$

In the hypergraph from Figure 2, $g_{\mathcal{H}}(\{v_1, v_5\})$ equals $\{e_5\}$ and $\mathcal{R}_{g_{\mathcal{H}}}(\{v_1, v_5\})$ equals $\{\{v_1, v_5\}, \{v_3, v_5\}, \{v_1, v_3, v_5\}\}$.

Transversals characterization Lemma 3.2 proves that the transversals of $\mathcal{H}$ correspond to the vertex patterns V such that $|g_{\mathcal{H}}(V)| = 0$.

Lemma 3.2. The vertex pattern V is a transversal of the hypergraph $\mathcal{H}$ if and only if $|g_{\mathcal{H}}(V)| = 0$.

$|g_{\mathcal{H}}(V)| = 0$ means that V has an empty extension and thus, intersects each hyperedge of the hypergraph $\mathcal{H}$. In our example, $|g_{\mathcal{H}}(\{v_2, v_3, v_5\})|$ equals zero and it was underlined in Section 2.2.2 that $\{v_2, v_3, v_5\}$ is a transversal of $\mathcal{H}$. Note that in data mining, the attribute patterns having an empty extension correspond to patterns that do not appear in any object of $\mathcal{D}$. Such patterns do not belong to the database and they are called *missing patterns* in Table 2. Lemma 3.2 leads to Corollary 3.1 that identifies all the transversals as the elements of the equivalence class containing the vertex pattern $\{v_1, v_2, ..., v_n\}$:

Corollary 3.1. $\mathcal{R}_{g_{\mathcal{H}}}(\{v_1, v_2, ..., v_n\})$ equals $Tr(\mathcal{H})$ and will be denoted by $\mathcal{R}^{\emptyset}_{g_{\mathcal{H}}}$.

Proof:
$\{v_1, v_2, ..., v_n\}$ covers all the hyperedges in $\mathcal{E}$ and its equivalence class corresponds to the vertex patterns having an empty extension. From Lemma 3.2, we have equality with $Tr(\mathcal{H})$. □

In Figure 2, $\mathcal{R}_{g_{\mathcal{H}}}(\{v_1, v_2, v_3, v_4, v_5\})$ is equal to:

$\{\{v_2, v_3\}, \{v_2, v_4\}, \{v_1, v_2, v_3\}, \{v_1, v_2, v_4\}, \{v_1, v_4, v_5\}, \{v_2, v_3, v_4\}, \{v_2, v_3, v_5\}, \{v_2, v_4, v_5\},$
$\{v_3, v_4, v_5\}, \{v_1, v_2, v_3, v_4\}, \{v_1, v_2, v_3, v_5\}, \{v_1, v_2, v_4, v_5\}, \{v_1, v_3, v_4, v_5\}, \{v_2, v_3, v_4, v_5\},$
$\{v_1, v_2, v_3, v_4, v_5\}\}$

and gives all the transversals of $\mathcal{H}$.

Minimal elements characterization Let us define the minimal elements for any extension equivalence class:

Definition 3.3. A vertex pattern V having no proper subset V' such that V' belongs to $\mathcal{R}_{g_{\mathcal{H}}}(V)$ is a *minimal generator*.

For instance, the vertex pattern $\{v_3, v_4\}$ is a minimal generator in Figure 2. Actually, no proper subset of $\{v_3, v_4\}$ belongs to $\mathcal{R}_{g_\mathcal{H}}(\{v_3, v_4\})$ as $g_\mathcal{H}(\{v_3, v_4\}) = \{e_4\}$ differs from $g_\mathcal{H}(\{v_3\}) = \{e_4, e_5\}$ and from $g_\mathcal{H}(\{v_4\}) = \{e_1, e_3, e_4, e_6\}$ (see Figure 3). The minimal generators can be characterized according to the gap between equivalence classes (see Lemma 3.3): if a vertex pattern V and one of its subsets have the same extension, V is not a minimal generator.

Lemma 3.3. The vertex pattern V is a minimal generator if and only if for all $v \in V$, $|g_\mathcal{H}(V)| < |g_\mathcal{H}(V \backslash \{v\})|$.

Proof:
V is a minimal generator if and only if:
for all $V' \subset V$, $V' \notin \mathcal{R}_{g_\mathcal{H}}(V) \Leftrightarrow$ for all $V' \subset V$, $g_\mathcal{H}(V') \neq g_\mathcal{H}(V)$
$\Leftrightarrow$ for all $V' \subset V$, $|g_\mathcal{H}(V')| \geq |g_\mathcal{H}(V)|$
$\Leftrightarrow$ for all $v \in V$, $|g_\mathcal{H}(V \backslash \{v\})| \geq |g_\mathcal{H}(V)|$ □

A minimal generator V is a minimal transversal of the partial hypergraph defined by $\mathcal{E}(V)$. We will see in Section 4 how the minimality can be used to reduce the search space and speed up a levelwise algorithm. Similarly, in the data mining area, the *free* patterns are defined as the minimal elements of the frequency equivalence classes (see [4, 20]).

Figure 3 shows a sample of the vertex patterns and their extension in Figure 2. As $\{v_3\}$ and $\{v_1, v_3\}$ have the same extension $\{e_4, e_5\}$, they belong to the same equivalence class. $\mathcal{R}^{\emptyset}_{g_\mathcal{H}}$ contains at least the patterns $\{v_2, v_3\}$ and $\{v_1, v_2, v_3\}$. $\{v_2, v_3\}$ is a minimal generator because neither $\{v_2\}$ nor $\{v_3\}$ belongs to $\mathcal{R}^{\emptyset}_{g_\mathcal{H}}$.

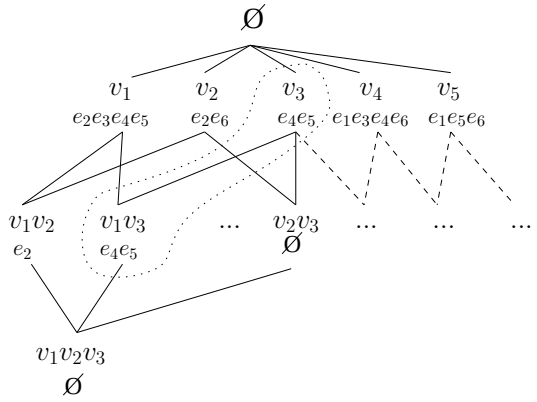


Figure 3. Sample of vertex patterns.

Minimal transversals From Corollary 3.1, we deduce that $MinTr(\mathcal{H})$ is equivalent to the minimal elements of $\mathcal{R}^{\emptyset}_{g_\mathcal{H}}$. The previous observations lead to Theorem 3.1 which gives an extension-based characterization of $MinTr(\mathcal{H})$:

Theorem 3.1. Let $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ be a hypergraph with the hypergraph Galois connection $(f_\mathcal{H}, g_\mathcal{H})$. $V \subseteq \mathcal{V}$ is in $MinTr(\mathcal{H})$ if and only if:

1. $|g_{\mathcal{H}}(V)| = 0$;
2. for all $v \in V$, $|g_{\mathcal{H}}(V)| < |g_{\mathcal{H}}(V \backslash \{v\})|$, i.e., V is a minimal generator.

Proof:
We mentioned in the previous paragraph that a minimal transversal is a minimal generator of $\mathcal{R}^{\emptyset}_{g_{\mathcal{H}}}$. Then, the combination of Lemma 3.2 and Lemma 3.3 immediately proves the result. □

Theorem 3.1 enables us to enumerate all the minimal transversals of the hypergraph given in Figure 2:

$$MinTr(\mathcal{H}) = \{\{v_2, v_3\}, \{v_2, v_4\}, \{v_1, v_2, v_5\}, \{v_1, v_4, v_5\}, \{v_3, v_4, v_5\}\}.$$

The equivalence given in Theorem 3.1 ensures the correctness and the completeness of the algorithm MTMINER described in Section 5.

4. Pruning strategy

In this section, we briefly describe the levelwise framework and present two *pruning properties* to improve the extraction of the minimal transversals. These properties naturally ensue the characterization of minimal transversals given in the previous section.

4.1. Significance of pruning in levelwise algorithms

We recall the principle to mine the patterns satisfying a constraint q under the levelwise framework. This approach applies a breadth-first search, starting from the shortest patterns (i.e., the patterns composed of one vertex) to the longest patterns statisfying the constraint q. The key idea is to combine this approach with an anti-monotone constraint with respect to the pattern inclusion. A constraint q is *anti-monotone* if and only if for all patterns X and Y, $q(X)$ and $Y \subseteq X$ implies $q(Y)$. Thus, anti-monotonicity results in an essential pruning criterion under the levelwise framework: if a pattern does not satisfy q, the same holds for every superset.

4.2. Pruning Criteria

Theorem 3.1 (Section 3.2) identifies the minimal transversals of a hypergraph as the minimal generators having an empty extension. This characterization leads to the two following pruning criteria. The first one comes from the anti-monotonicity of the minimality in the equivalence classes defined in Section 3.2. The second one is used as a stopping criterion.

Anti-Monotonicity of minimality in equivalence classes Property 2 establishes that being a minimal generator (see Section 3.2) is anti-monotone.

Property 2. The minimality in extension equivalence classes is an anti-monotone property.

Proof:
Assume that V is a minimal generator and there is v in V such that $V\backslash\{v\}$ is not. Then there is v' in $V\backslash\{v\}$ satisfying $g_{\mathcal{H}}(V\backslash\{v,v'\}) = g_{\mathcal{H}}(V\backslash\{v\})$ (Lemma 3.3). We have the following equalities:
$g_{\mathcal{H}}(V) = g_{\mathcal{H}}(V\backslash\{v\}) \cap g_{\mathcal{H}}(\{v\})$ (Property 1)
$= g_{\mathcal{H}}(V\backslash\{v,v'\}) \cap g_{\mathcal{H}}(\{v\})$
$= g_{\mathcal{H}}(V\backslash\{v'\})$ (Property 1)
Since there is v' in V such that $V\backslash\{v'\}$ and V have the same extension, V is not a minimal generator (Lemma 3.3) and this is in contradiction with our hypothesis. □

First pruning property As observed in Section 4.1, Property 2 allows to state an important pruning criterion: if a vertex pattern is not a minimal generator, then none of its supersets is a minimal generator.

Pruning Criterion 1. Let V be a vertex pattern. If there is $v \in V$ such that $|g_{\mathcal{H}}(V)| \geq |g_{\mathcal{H}}(V\backslash\{v\})|$, then no superset of V is a minimal transversal of $\mathcal{H}$.

Proof:
Suppose there is $v \in V$ such that $|g_{\mathcal{H}}(V)| \geq |g_{\mathcal{H}}(V\backslash\{v\})|$ (which is equivalent to $|g_{\mathcal{H}}(V)| = |g_{\mathcal{H}}(V\backslash\{v\})|$) and consider $W \in 2^{\mathcal{V}}$ such that $V \subseteq W$. By using Property 1, we have the following equalities:
$g_{\mathcal{H}}(W\backslash\{v\}) = g_{\mathcal{H}}(V\backslash\{v\}) \cap g_{\mathcal{H}}(W\backslash V)$
$= g_{\mathcal{H}}(V) \cap g_{\mathcal{H}}(W\backslash V)$
$= g_{\mathcal{H}}(W)$
We conclude thanks to Theorem 3.1 that W is not a minimal transversal. □

Pruning Criterion 1 is a powerful tool to avoid testing uninteresting vertex patterns: a levelwise algorithm can prune the search space from V. In Figure 2, $\{v_1, v_3\}$ is not a minimal generator since $\{v_1, v_3\}$ and $\{v_3\}$ have the same extension $\{e_4, e_5\}$ (see Figure 3). Consequently, all the supersets of $\{v_1, v_3\}$ are pruned thanks to Pruning Criterion 1.

Notice that Property 2 does not depend on the Galois connection. In fact, Pruning Criterion 1 remains completely true and can be used for any Galois connection. This fact explains why a similar criterion exists for the freeness constraint.

Second pruning property Obviously, if a vertex pattern V is a minimal transversal then its supersets are transversals but they cannot be minimal generators. This observation gives the second pruning property. When computing minimal transversals in a levelwise manner, once a minimal transversal is encountered, its supersets will not be tested when computing later on.

5. A levelwise algorithm for minimal transversals

In this section, we present our algorithm MTMINER (MT for Minimal Transversals Miner) to compute the minimal transversals of a hypergraph. MTMINER follows the principle of levelwise algorithms. It is based on the extension of the Galois connection defined in Section 3.2 and optimized thanks to the pruning properties exposed in Section 4. We prove that it is correct and complete and we finally give its theoretical complexity.

5.1. Outline

Figure 4 schematizes the search space. For the hypergraph given in Figure 2, Figure 3 is a piece of the search space depicted in Figure 4. The vertex patterns are represented in a lattice and divided into two groups: those having an empty extension and the others (i.e., the non transversals). MTMINER starts by covering the minimal generators which are not transversals of the input hypergraph $\mathcal{H}$. Remind that the search space is reduced thanks to Pruning Criterion 1 since it is sufficient to consider the minimal generators instead of all the non transversals. Figure 4 illustrates that the minimal transversals stand on the negative border of the non transversals. The close relationship between negative borders of theories and minimal transversals was pointed out in [13]. When a vertex pattern satisfies Theorem 3.1, it is a minimal transversal and the computation stops because the second pruning property holds.

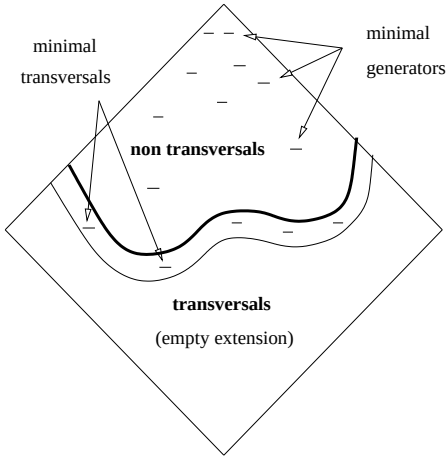


Figure 4. Search space when computing hypergraph transversals.

5.2. Algorithm MTMINER

This section details the algorithm MTMINER which computes all the minimal transversals for the input hypergraph $\mathcal{H}$. A vertex pattern which is not a minimal transversal and which is not removed because of non minimality (Pruning Criterion 1) is a *generator*. Only one scan of the hypergraph is needed since it is possible to compute the extension by intersecting the generators extensions (see Property 1).

Algorithm	MTMINER
Input:	Hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$
Output:	Minimal transversals of $\mathcal{H}$

1. *// initialization*
   $Trav := \{\{v\} \in \mathcal{V} \mid |g_{\mathcal{H}}(\{v\})| = 0\}$
   $\mathcal{G}en_1 := \{\{v\} \in \mathcal{V} \mid |\mathcal{E}| > |g_{\mathcal{H}}(\{v\})| > 0\}$
   $k := 1$
2. **while** $\mathcal{G}en_k \neq \emptyset$ **do**

3. **for each** $(V \cup \{v_1\}, V \cup \{v_2\}) \in \mathcal{G}en_k \times \mathcal{G}en_k$ **do**
   // candidate generation ($k+1$ vertices)
   $W := V \cup \{v_1\} \cup \{v_2\}$
   // extension computation by using Property 1
   $g_{\mathcal{H}}(W) := g_{\mathcal{H}}(V \cup \{v_1\}) \cap g_{\mathcal{H}}(V \cup \{v_2\})$

4. *// verification and pruning*
   $i := 1$

5. *// Pruning Criterion 1*
   while $i \leq k+1$ **and** $W \backslash \{v_i\} \in \mathcal{G}en_k$ **and** $|g_{\mathcal{H}}(W)| < |g_{\mathcal{H}}(W \backslash \{v_i\})|$ **do**
   $i := i + 1$
   od

6. **if** $i = k+2$ **then**
   // Second pruning property
   if $|g_{\mathcal{H}}(W)| = 0$ **then** $Trav = Trav \cup \{W\}$
   else $\mathcal{G}en_{k+1} := \mathcal{G}en_{k+1} \cup \{W\}$
   od
   $k := k + 1$
   od

7. **return** $Trav$

$\mathcal{G}en_k$ is the set of generators with k vertices. The minimal transversals are stored in $Trav$.

In Step 1, $Trav$ is initialized with all the vertices having an empty extension and $\mathcal{G}en_1$ is initialized with the minimal generators with an nonempty extension (Lemma 3.3).

The main loop begins in Step 2: it stops when there is no generator left at level k. At level $k+1$, candidates are generated by joining two vertex patterns having $k-1$ vertices in common. The extension of a candidate W is computed by intersecting its generators extensions (Property 1). Lemma 3.3 is used to test whether the candidate W is a minimal generator. If W is not a minimal generator, this candidate and all its supersets are deleted by applying Pruning Criterion 1. Step 6 tests whether W is a transversal of $\mathcal{H}$ with Lemma 3.2. If W is minimal but is not a transversal, it is added to $\mathcal{G}en_{k+1}$. When W is a minimal generator and a transversal of $\mathcal{H}$, Theorem 3.1 is used and W is added to $Trav$.

Theorem 5.1 proves that MTMINER is correct and complete.

Theorem 5.1. The algorithm MTMINER extracts all the minimal transversals from the input hypergraph $\mathcal{H}$.

Proof:
Let us prove that a set of vertices W in $Trav$ is a minimal transversal of $\mathcal{H}$. We test in Step 5 if $|g(W)| < |g(W \backslash \{v\})|$ for all v in W, which ensures that W is a minimal element of $\mathcal{R}_{g_{\mathcal{H}}}(W)$ (Lemma 3.3). Step 6 establishes that W is a transversal (cf. Lemma 3.2).

[Completeness] The algorithm MTMINER covers the whole vertices search space thanks to the principle of the levelwise algorithms. The accuracy of the used pruning criteria (Criterion 1 and second pruning property) entails the completeness of MTMINER. □

5.3. Complexity

For each minimal transversal T, MTMINER explores at most $2^{|T|}$ vertex patterns. Consequently, the maximal number of operations equals:

$$\sum_{T \in MinTr(\mathcal{H})} 2^{|T|}.$$

This upper bound is not reached in most of the cases because at level k in the lattice $2^{\mathcal{V}}$, for two transversals T_1 and T_2, the two sublattices often have a nonempty intersection. The vertex patterns of this intersection are not verified twice.

Because $|T| \leq t(\mathcal{H})$, we have Theorem 5.2:

Theorem 5.2. The algorithm MTMINER computes $MinTr(\mathcal{H})$ with a time complexity equal to:

$$\mathcal{O}(2^{t(\mathcal{H})} \times |MinTr(\mathcal{H})|).$$

This complexity depends on $t(\mathcal{H})$ and $|MinTr(\mathcal{H})|$ which is the size of the output. The complexity of the best known algorithm that computes all minimal transversals of a hypergraph, is given in the introduction and depends on the size of both the input hypergraph and the output. Since we do not know any relation between $t(\mathcal{H})$ and the size of the input hypergraph, we cannot compare these algorithms in a theoretical manner. Thus, an experimental comparison is proposed in Section 6. We will also see that $t(\mathcal{H})$ remains very small in practice when the hyperedges of the input hypergraph contain a lot of vertices.

6. Experimental evaluation

We compare MTMINER with two prototypes: DUAL and THG. DUAL is described in [3] and is based on the algorithm [11] with the best theoretical complexity addressing the hypergraph transversal problem. THG is an improvement of Berge's algorithm [2, p. 52], it was proven to be efficient in practice [16]. The implementations were downloaded at url `http://rutcor.rutgers.edu/~boros/IDM/DualizationCode.html` and `http://lca.ceid.upatras.gr/~estavrop/transversal/`. MTMINER is available at url `http://users.info.unicaen.fr/~chebert/mtminer.html`. The first experiment (Section 6.1) consists in comparing MTMINER to DUAL and THG on random hypergraphs. The second experiment (Section 6.2) shows how MTMINER enables us to compute the negative border of a set of frequent patterns, whereas the other prototypes fail. All the tests were performed on a 2.2 GHz Pentium IV processor with Linux operating system using 3 GB of RAM memory.

6.1. Evaluation on random hypergraphs

In this experiment, the comparison is made on randomly generated hypergraphs based on the Erdős-Rényi model [10]. We use a parameter p which corresponds to the proportion of 1 in the incidence

matrix of the input hypergraph. The higher p, the larger the hyperedges are and the denser the incidence matrix is. In data mining, it means that the data are highly correlated.

The first issue consists in studying the computation time according to p. We know that the performances of MTMINER are closely linked to $t(\mathcal{H})$ (see Section 5.3) and the performances of DUAL to the size of the output $|MinTr(\mathcal{H})|$. Although we do not have any evidence of it, we expect that the cardinality of the largest minimal transversal $t(\mathcal{H})$ is high when the incidence matrix of the hypergraph is sparse. Consequently, sparse hypergraphs represent the most difficult case for MTMINER. Performances (run-times in seconds) in Table 3 confirm this statement. Note that DUAL outperforms THG for $p = 0.1$.

p	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
DUAL	326.70	fail	fail	fail	fail	fail	fail	fail	59.29
THG	9.56	117.15	1,015.26	7,272.22	fail	fail	fail	fail	7,308.28
MTMINER	0.25	4.14	48.72	530.02	fail	fail	fail	fail	fail
$t(\mathcal{H})$	3	5	7	8	?	?	?	?	41
$\|MinTr(\mathcal{H})\|$	26,939	339,372	2,634,205	16,237,137	?	?	?	?	4,396

Table 3. Run-time performances with $|\mathcal{V}| = 50$, $|\mathcal{E}| = 1,000$ and p between 0.9 and 0.1.

On the contrary, on dense hypergraphs (when p ranges from 0.9 to 0.6), MTMINER clearly outperforms THG. The prototype DUAL fails except for $p = 0.9$. All the prototypes fail when p is varying between 0.5 and 0.2. For such values of p, we believe that the number of minimal transversals is high and they tend to be very large.

The second issue is to show the efficiency of MTMINER with respect to DUAL and THG according to the number of hyperedges. Table 4 points out the run-time benefit brought by MTMINER (we fix $p = 0.8$). For instance, when $|\mathcal{E}| = 20,000$, THG needs about 30 hours to extract the $7,628,650$ minimal transversals, while MTMINER needs 169 seconds (DUAL fails).

$\|\mathcal{E}\|$	200	400	600	800	1,000	2,000
DUAL	297.80	1,042.72	1,865.88	2,681.69	4,143.26	17,854.75
THG	4.11	15.82	40.01	67.18	120.03	672.07
MTMINER	0.52	1.17	2.11	2.75	4.17	10.67
$\|\mathcal{E}\|$	3,000	5,000	7,000	10,000	20,000	
DUAL	fail	fail	fail	fail	fail	
THG	1,871.67	4,540.11	10,400.55	26,324.78	106,623.39	
MTMINER	16.67	38.28	57.94	88.64	168.72	

Table 4. Run-time performances with $|\mathcal{V}| = 50$, $p = 0.8$ and $|\mathcal{E}|$ between 200 and $20,000$.

6.2. Computing the negative border of a theory

In Section 2.1, we briefly described the problem of finding the negative border of a set of frequent patterns by using hypergraph minimal transversals [12, 21]. More precisely, the depth-first algorithm for finding frequent patterns presented in [12] includes a step consisting in computing the minimal transversals of the complements of the patterns belonging to the positive border. We want to test the behavior of DUAL, THG and MTMINER for this step.

We conducted experiments on benchmarks coming from the UCI repository, a summary and an access to the benchmarks are provided at the url `http://www.ics.uci.edu/~mlearn/MLSummary.html`. We used three benchmarks : `MUSHROOM` which is a $8,124 \times 120$ data, `LETTER-RECOGNITION` a $20,000 \times 74$ data and `PUMSB` a $49,046 \times 7,118$. We first compute the positive border of frequent patterns. Then we determine the complements of the sets of the positive border. At last, we apply the three prototypes on the complements of the patterns of the positive border. The run-times in seconds for the three benchmarks are given in Table 5, Table 6 and Table 7. In the tables, we also provide the following parameters: the frequency threshold, the number of hyperedges $|\mathcal{E}|$ and the density of the input hypergraph $\mathcal{H}$, the maximal size of a minimal transversal $t(\mathcal{H})$ and the size of the output $|MinTr(\mathcal{H})|$.

On the benchmark `MUSHROOM`, MTMINER clearly outperforms DUAL and THG. We think that the density of the input hypergraphs is the major reason why MTMINER is so efficient. Since the patterns in the positive border do not contain a lot of attributes, their complements are very large and the input hypergraphs are very dense. For `LETTER-RECOGNITION`, DUAL fails whatever the frequency threshold. MTMINER spends one second to mine almost 80,000 minimal transversals while THG needs almost half an hour. Let us note that mining frequent patterns in `PUMSB` requires a high frequency threshold. For this benchmark, MTMINER is the only prototype which succeeds in computing the minimal transversals.

frequency	800	600	400	200	100	50	30	10	1
DUAL	53.52	82.09	278.17	840.89	2,248.50	5,647.58	12,059.95	35,612.74	3,477.25
THG	0.60	1.52	5.10	29.90	117.87	404.11	1,128.39	3,161.14	103.30
MTMINER	0.27	0.58	1.55	4.48	13.48	30.49	48.21	94.98	85.89
density	0.731	0.736	0.741	0.753	0.763	0.771	0.778	0.782	0.773
$\lvert\mathcal{E}\rvert$	573	918	1,477	3,111	5,776	9,857	15,232	30,809	8,124
$t(\mathcal{H})$	6	6	7	7	8	9	9	10	7
$\lvert MinTr(\mathcal{H})\rvert$	6,244	8,235	16,375	31,331	51,678	77,990	100,573	118,234	22,294

Table 5. Run-time performances for the benchmark `MUSHROOM`.

frequency	5,000	3000	1,000	800	600	400	300	200	100
DUAL	fail	fail	fail	fail	fail	fail	fail	fail	fail
THG	0.01	0.21	39.54	106.96	371.01	1,750.11	5,096.74	13,891.36	77,468.80
MTMINER	0	0	0.13	0.21	0.42	1.09	1.89	4.48	15.36
density	0.972	0.962	0.94	0.937	0.932	0.925	0.921	0.9147	0.905
$\lvert\mathcal{E}\rvert$	79	347	5,579	8,979	15,779	33,015	52,554	96,355	228,278
$t(\mathcal{H})$	3	4	7	7	8	9	9	11	11
$\lvert MinTr(\mathcal{H})\rvert$	524	1,851	16,961	25,298	43,302	79,479	121,307	207,246	453,280

Table 6. Run-time performances for the benchmark `LETTER-RECOGNITION`.

Finally, these experiments highlight the twofold advantages of MTMINER: the minimal transversal extraction becomes feasible for dense hypergraphs and when other algorithms do not fail, MTMINER is much faster.

frequency	48,000	45,000	40,000	35,000
DUAL	fail	fail	fail	fail
THG	fail	fail	fail	fail
MTMINER	0.01	0.28	4.34	24.78
density	0.9996	0.9993	0.9989	0.9985
$\|\mathcal{E}\|$	3	144	2,341	10,417
$t(\mathcal{H})$	1	5	9	13
$\|MinTr(\mathcal{H})\|$	7,120	7,483	14,085	41,020

Table 7. Run-time performances for the benchmark PUMSB.

7. Conclusion and discussion

In this paper, we linked the hypergraph transversal problem to the pattern extraction domain, enabling to exploit an anti-monotone property and to benefit from efficient pruning methods. Due to the large number of interesting patterns and the need of completeness, data mining algorithms are designed to face very large outputs. As the number of minimal transversals is often sizable (see Section 6), it is not surprising that a pattern dedicated algorithm can help in solving this problem. Thus, we proposed a new algorithm based on the above mentioned techniques for computing hypergraph minimal transversals. We proved that our algorithm becomes an order of magnitude faster than other algorithms and enables us to efficiently compute minimal transversals especially in dense hypergraphs. As said in Section 2.1, many works address the hypergraph transversal problem when the hyperedge size is bounded. On the contrary, our approach furnishes a complementary means of computing hypergraph transversals when the hyperedge size is high.

This work could be extended in many directions. From a practical point of view, it seems that to use only one algorithm is not sufficient to efficiently compute the minimal transversals for any input hypergraph. Sparse hypergraphs are likely to contain few and long minimal transversals and this is suitable for approaches like DUAL's one. In dense hypergraphs, these approaches fail because the number of minimal transversals is often large, whereas levelwise methods like MTMINER succeed. This is the reason why it would be interesting to better characterize the cases where each prototype is efficient.

In Section 3.2, we showed that MTMINER covers all the present patterns in a database to compute the minimal transversals. In [18], the average number of frequent patterns (for a given frequency) is studied. We think that this work could be applied to provide:

- the average number of minimal transversals : the minimal transversals of a hypergraph can be regarded as patterns having a frequency equal to zero (also called missing patterns in Table 2), the average number of minimal transversal could be deduced from this observation;

- the cardinality of the largest minimal transversal(s) $t(\mathcal{H})$.

As these parameters are involved in the theoretical complexity of MTMINER and DUAL, their estimation would allow to compare them in a theoretical manner.

References

[1] Bailey, J., Manoukian, T., Ramamohanarao, K.: A Fast Algorithm for Computing Hypergraph Transversals and its Application in Mining Emerging Patterns, *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM'03)*, IEEE Computer Society, Los Alamitos, CA, USA, 2003.

[2] Berge, C.: *Hypergraphs: Combinatorics of Finite Sets*, North-Holland, Amsterdam, 1989.

[3] Boros, E., Elbassioni, K., Gurvich, V., Khachiyan, L.: An Efficient Implementation of a Quasi-Polynomial Algorithm for Generating Hypergraph Transversals, *Proceedings of the 11th Annual European Symposium on Algorithms (ESA'03)*, LNCS 2832, Springer, Budapest, Hungary, 2003.

[4] Boulicaut, J. F., Bykowski, A., Rigotti, C.: Free-sets: a Condensed Representation of Boolean Data for the Approximation of Frequency Queries, *Data Mining and Knowledge Discovery journal*, **7**(1), 2003, 5–22.

[5] Calders, T., Goethals, B.: Minimal k-Free Representations of Frequent Sets, *Proceedings of the 7th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'03)*, LNAI 2838, Springer, Dubvronik, Croatia, 2003.

[6] Damaschke, P.: Parameterized Enumeration, Transversals, and Imperfect Phylogeny Reconstruction, *Proceedings of the 1st International Workshop on Parameterized and Exact Computation*, LNCS 3162, Springer, Bergen, Norway, 2004.

[7] Denecke, K., Wismath, S. L.: *Universal Algebra and Applications in Theoretical Computer Science*, Chapman and Hall/CRC, 2002.

[8] Durand, N., Crémilleux, B., Suzuki, E.: Visualizing Transactional Data with Multiple Clusterings for Knowledge Discovery, *Proceedings of the 16th International Symposium on Methodologies for Intelligent Systems (ISMIS 2006)*, LNAI 4203, Springer, Bari, Italy, 2006.

[9] Eiter, T., Gottlob, G.: Hypergraph Transversal Computation and Related Problems in Logic and AI, *Proceedings of the 8th European Conference on Logic in Artificial Intelligence (JELIA'02)*, LNCS 2424, Springer, Cosenza, Italy, 2002.

[10] Erdős, P., Rényi, A.: On Random Graphs, *Publicationes Mathematicae*, **6**, 1959, 290–297.

[11] Fredman, M. L., Khachiyan, L.: On the Complexity of Dualization of Monotone Disjunctive Normal Forms, *Journal of Algorithms*, **21**(3), 1996, 618–216.

[12] Gunopulos, D., Khardon, R., Mannila, H., Saluja, S., Toivonen, H., Sharm, R. S.: Discovering all most specific sentences, *ACM Transactional Database System*, **28**(2), 2003, 140–174.

[13] Gunopulos, D., Khardon, R., Mannila, H., Toivonen, H.: Data mining, Hypergraph Transversals, and Machine Learning, *Proceedings of the 16th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'97)*, ACM Press, Tucson, Arizona, 1997.

[14] Hébert, C., Crémilleux, B.: Mining Frequent δ-Free Patterns in Large Databases, *Proceedings of the 8th International Conference on Discovery Science (DS'05)*, LNAI 3735, Springer-Verlag, Singapore, 2005.

[15] Kavvadias, D. J., Stavropoulos, E. C.: Evaluation of an Algorithm for the Transversal Hypergraph Problem, *Proceedings of the 3rd Workshop on Algorithm Engineering (WAE'99)*, LNCS 1668, Springer, London, UK, 1999.

[16] Kavvadias, D. J., Stavropoulos, E. C.: An Efficient Algorithm for the Transversal Hypergraph Generation, *Journal of Graph Algorithms and Applications*, **9**(2), 2005, 239–264.

[17] Khachiyan, L., Boros, E., Elbassioni, K. M., Gurvich, V.: A New Algorithm for the Hypergraph Transversal Problem, *Proceedings of the 11th International Computing and Combinatorics Conference (COCOON'05)*, LNCS 3595, Springer, Kunming, China, 2005.

[18] Lhote, L., Rioult, F., Soulet, A.: Average Number of Frequent (Closed) Patterns in Bernoulli and Markovian Databases, *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM'05)*, Houston, Texas, 2005.

[19] Mannila, H., Toivonen, H.: Levelwise Search and Borders of Theories in Knowledge Discovery, *Data Mining and Knowledge Discovery journal*, **1**(3), 1997, 241–258.

[20] Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Discovering Frequent Closed Itemsets for Association Rules, *Proceedings of 7th International Conference on Database Theory (ICDT'99)*, LNAI 1331, Springer Verlag, Jerusalem, Israel, 1999.

[21] Satoh, K., Uno, T.: Enumerating Maximal Frequent Sets Using Irredundant Dualization, *Proceedings of the 6th International Conference on Discovery Science (DS'03)*, LNCS 2843, Springer, Sapporo, Japan, 2003.

[22] Soulet, A., Crémilleux, B., Rioult, F.: Condensed Representation of EPs and Patterns Quantified by Frequency-based Measures, *Proceedings of the 4th International Workshop on Knowledge Discovery in Inductive Databases (KDID'05)*, LNCS 3377, Springer, Porto, Portugal, 2005.

[23] Wahlström, M.: Exact Algorithms for Finding Minimum Transversals in Rank-3 Hypergraphs, *Journal of Algorithms*, **51**(2), 2004, 107–121.