Frequent Hypergraph Mining*

Tamás Horváth[1], Björn Bringmann[2], and Luc De Raedt[2]

[1] Fraunhofer IAIS, Sankt Augustin, Germany
tamas.horvath@iais.fraunhofer.de
[2] Dept. of Computer Science, Katholieke Universiteit Leuven, Belgium
{bjoern.bringmann,luc.deraedt}@cs.kuleuven.be

Abstract. The class of frequent hypergraph mining problems is introduced which includes the frequent graph mining problem class and contains also the frequent itemset mining problem. We study the computational properties of different problems belonging to this class. In particular, besides negative results, we present practically relevant problems that can be solved in incremental-polynomial time. Some of our practical algorithms are obtained by reductions to frequent graph mining and itemset mining problems. Our experimental results in the domain of citation analysis show the potential of the framework on problems that have no natural representation as an ordinary graph.

1 Introduction

The field of data mining has studied increasingly expressive representations in the past few years. Whereas the original formulation of frequent pattern mining still employed itemsets [1], researchers have soon studied more expressive representations such as sequences and episodes (e.g., [16]), trees (e.g., [4,21]), and more recently, graph mining has become an important focus of research (e.g., [13,18,19]). These developments have been motivated and accompanied by new and challenging application areas. Indeed, itemsets apply to basket-analysis, sequences and episodes to alarm monitoring, trees to document mining, and graph mining to applications in computational chemistry.

In this paper, we introduce the next natural step in this evolution: the mining of *labeled hypergraphs*. In a similar way that tree mining generalizes sequence mining, and graph mining generalizes tree mining, hypergraph mining is a natural generalization of frequent pattern mining in *undirected* graphs. The presented framework is especially applicable to problem domains which do not have a natural representation as ordinary graphs. One such application is used in the experimental section of this paper. It is concerned with citation analysis, more specifically, with analyzing bibliographies of a set of papers. The bibliography of a paper can be viewed as a hypergraph, in which each author corresponds to a vertex and each paper to the hyperedge containing all authors of the paper.

* An early version of this paper appeared in T. Gärtner, G.C. Garriga, and T. Meinl (Eds.), *Proc. of the International Workshop on Mining and Learning with Graphs*, pages 25–36, ECML/PKDD'06 workshop proceedings, Berlin, Germany, 2006.

By mining for frequent subhypergraphs in the bibliographies of a set of papers (e.g. past KDD conference papers), one should be able to discover common citation patterns in a particular domain (such as SIGKDD). These patterns might then be employed in a recommender system that assists scientists while making bibliographies. A similar approach in a basket-analysis context allows one to represent the transactions over a specific period of time of *one family* as a hypergraph, where the products correspond to the vertices and the transactions to the hyperedges. Mining such data could provide insight into the overall purchasing behavior of families.

The main contribution of this paper is the introduction of a general framework of mining frequent hypergraphs. The framework can be specialized in a number of different ways, according to the notion of the generalization relation employed as well as the class of hypergraphs considered. We consider different problems where the generalization relation is defined by *subhypergraph isomorphism*, study their computational properties, and present positive and negative results. More specifically, we show that there is no output-polynomial time algorithm for mining frequent connected subhypergraphs, and even in the case of strong *structural* assumptions on the hyperedges, this problem cannot be solved efficiently by the usual level-wise frequent pattern mining approach (see, e.g., [15]). On the other hand, by restricting the functions labeling the vertices, we achieve positive results. Some of the results are obtained by employing reductions from frequent hypergraph mining problems to ordinary graph mining and itemset mining problems. We also present experiments in the above sketched citation analysis domain which indicate that these reductions can effectively be applied in practice. Essentially, we gathered the bibliographies of 5 SIGKDD, 30 SIGMOD, and 30 SIGGRAPH conferences and searched for frequent hypergraphs in each conference.

The rest of the paper is organized as follows: in Section 2, we introduce the necessary notions concerning hypergraphs and in Section 3, we define the problem class of frequent hypergraph mining. In Section 4, we study the frequent subhypergraph mining problem. In Section 5, we present some experiments using the citation analysis problem, and finally, in Section 6, we conclude and list some problems for future work. Due to space limitations, some of the proofs are only sketched in this version.

2 Notions and Notations

We recall some basic notions and notations related to graphs and hypergraphs (see, e.g., [2,6]). For a set S and non-negative integer k, $[S]^k$ denotes the family of all k-subsets of S, i.e., $[S]^k = \{S' \subseteq S : |S'| = k\}$.

Graphs and Hypergraphs. An *(undirected) graph* G consists of a finite set V of *vertices* and a family $\mathcal{E} \subseteq [V]^2$ of *edges*. G is *bipartite* if G has a vertex 2-coloring, i.e., if V admits a partition into V_1 and V_2 such that $E \notin [V_1]^2 \cup [V_2]^2$ for every $E \in \mathcal{E}$. A *hypergraph* H is a pair $(V, \mathcal{E})$, where V is a finite set and $\mathcal{E}$ is a family of nonempty subsets of V such that $\bigcup_{E \in \mathcal{E}} E = V$. The elements of V and

$\mathcal{E}$ are called *vertices* and *edges* (or *hyperedges*), respectively. H is *r-uniform* for some integer $r > 0$ if $\mathcal{E} \subseteq [V]^r$. The *rank* of H, denoted $r(H)$, is the cardinality of its largest hyperedge, i.e., $r(H) = \max_{E \in \mathcal{E}} |E|$, and the *size* of H, denoted $\text{size}(H)$, is the number of hyperedges of H, i.e., $\text{size}(H) = |\mathcal{E}|$.

By definition, a hyperedge may contain one or more vertices. Note that ordinary undirected graphs without isolated vertices form a special case of hypergraphs, i.e., the class of 2-uniform hypergraphs. We note that every hypergraph $H = (V, \mathcal{E})$ can be represented by a *bipartite incidence* graph $B(H) = (V \cup \mathcal{E}, \mathcal{E}')$, where $\mathcal{E}' = \{\{v, E\} : v \in V, E \in \mathcal{E}, \text{ and } v \in E\}$.

Labeled Hypergraphs. A *labeled hypergraph* is a triple $H = (V, \mathcal{E}, \lambda)$, where $(V, \mathcal{E})$ is a hypergraph, and $\lambda : V \to \mathbb{N}$ is a *labeling function*[1]. Unless otherwise stated, by hypergraphs (resp. graphs) we always mean labeled hypergraphs (resp. labeled graphs), and denote the set of vertices, the set of edges, and the labeling function of a hypergraph (resp. graph) H by V_H, $\mathcal{E}_H$, and λ_H, respectively. The set of all hypergraphs is denoted by $\mathcal{H}$ and $\mathcal{H}_r$ denotes the set of all r-uniform hypergraphs. For a hypergraph $H \in \mathcal{H}$ and subset $V' \subseteq V_H$, we denote the *multiset*[2] $\{\lambda_H(v) : v \in V'\}$ by $\lambda^H(V')$. A *path* connecting the vertices $u, v \in V_H$ is a sequence $E_1, \ldots, E_k$ of edges of H such that $u \in E_1$, $v \in E_k$, and $E_i \cap E_{i+1} \neq \emptyset$ for every $i = 1, \ldots, k-1$. A hypergraph is *connected* if there is a path between any pair of its vertices. The set of connected hypergraphs is denoted by $\mathcal{H}^c$. Clearly, $\mathcal{H}^c \subset \mathcal{H}$.

Injective Hypergraphs. Depending on the labeling functions, in this paper we will consider two special classes of hypergraphs. A hypergraph $H \in \mathcal{H}$ is *node injective* if λ_H is injective, and it is *edge injective* whenever $\lambda^H(E) = \lambda^H(E')$ if and only if $E = E'$ for every $E, E' \in \mathcal{E}_H$. The sets of node and edge injective hypergraphs will be denoted by $\mathcal{H}^{\text{ni}}$ and $\mathcal{H}^{\text{ei}}$, respectively. Clearly, $\mathcal{H}^{\text{ni}} \subseteq \mathcal{H}^{\text{ei}} \subseteq \mathcal{H}$.

Hypergraph Isomorphism. Let $H_1, H_2 \in \mathcal{H}$ be hypergraphs. H_1 and H_2 are called *isomorphic*, denoted by $H_1 \simeq H_2$, if there is a bijection $\varphi : V_{H_1} \to V_{H_2}$ such that

– φ preserves the labels, i.e., $\lambda_{H_1}(v) = \lambda_{H_2}(\varphi(v))$ for every $v \in V_{H_1}$, and
– φ preserves the hyperedges in both directions, i.e., for every $E \subseteq V_{H_1}$ it holds that $E \in \mathcal{E}_{H_1}$ if and only if $\{\varphi(v) : v \in E\} \in \mathcal{E}_{H_2}$.

Throughout this paper, two hypergraphs H_1 and H_2 are considered to be the same if $H_1 \simeq H_2$.

[1] We will only consider labeling functions defined on the vertex set because any hypergraph $H = (V, \mathcal{E}, \lambda)$ with $\lambda : V \cup \mathcal{E} \to \mathbb{N}$ satisfying $\lambda(v) \neq \lambda(E)$ for every $v \in V$ and $E \in \mathcal{E}$ can be transformed into a hypergraph $H' = (V', \mathcal{E}', \lambda')$ with $V' = V \cup \{v_E : E \in \mathcal{E}\}$, $\mathcal{E}' = \{E \cup \{v_E\} : E \in \mathcal{E}\}$, and with $\lambda' : V' \to \mathbb{N}$ mapping every new vertex $v_E \in V' \setminus V$ to $\lambda(E)$ and every $v \in V$ to $\lambda(v)$.

[2] A *multiset* M is a pair (S, f), where S is a set and f defines the multiplicity of the elements of S in M, i.e., f is a function mapping S to the cardinal numbers greater than 0.

Subhypergraphs. A *subhypergraph* of a hypergraph $H \in \mathcal{H}$ is a hypergraph $H' \in \mathcal{H}$ satisfying $V_{H'} \subseteq V_H$, $\mathcal{E}_{H'} \subseteq \mathcal{E}_H$, and $\lambda_{H'}(v) = \lambda_H(v)$ for every $v \in V_{H'}$.

3 Frequent Hypergraph Mining

Many problems in data mining can be viewed as a special case of the problem of enumerating the elements of a *quasiordered* set[3], which satisfy some monotone property (see, e.g., [3,12]). In this section, we define a new class of subproblems of this enumeration problem, the class of *frequent hypergraph mining problems.* In the next section, we then discuss the computational aspects of some problems belonging to this class. We start with the definition of a more general problem class.

The Frequent Pattern Mining Problem Class ($\mathcal{C}_{\mathrm{FPM}}$): Each problem belonging to this class is given by a *fixed* triple $(\mathcal{L}_D, \mathcal{L}_P, \preccurlyeq)$, where $\mathcal{L}_D$ is a *transaction language*, $\mathcal{L}_P$ is a *pattern language*, and $\preccurlyeq$, called the *generalization relation*, is a quasiorder on $\mathcal{L}_D \cup \mathcal{L}_P$. For such a triple, the $(\mathcal{L}_D, \mathcal{L}_P, \preccurlyeq)$-Frequent-Pattern-Mining problem is defined as follows: *Given* a finite set $\mathcal{D} \subseteq \mathcal{L}_D$ of *transactions* and an integer $t > 0$, called *frequency threshold*, *compute* the set $\mathcal{F}_{(\mathcal{L}_D, \mathcal{L}_P, \preccurlyeq)}(\mathcal{D}, t)$ of *frequent patterns* defined by

$$\mathcal{F}_{(\mathcal{L}_D, \mathcal{L}_P, \preccurlyeq)}(\mathcal{D}, t) = \{\varphi \in \mathcal{L}_P : |\{\tau \in \mathcal{D} : \varphi \preccurlyeq \tau\}| \geq t\} \ .$$

The transitivity of $\preccurlyeq$ implies that frequency is a monotone property, i.e., for every $\varphi, \theta \in \mathcal{L}_P$ it holds that $\theta \in \mathcal{F}_{(\mathcal{L}_D, \mathcal{L}_P, \preccurlyeq)}(\mathcal{D}, t)$ whenever $\varphi \in \mathcal{F}_{(\mathcal{L}_D, \mathcal{L}_P, \preccurlyeq)}(\mathcal{D}, t)$ and $\theta \preccurlyeq \varphi$.

We now define two subclasses of $\mathcal{C}_{\mathrm{FPM}}$ by restricting the transaction and pattern languages to hypergraphs ($\mathcal{C}_{\mathrm{FHM}}$) and graphs ($\mathcal{C}_{\mathrm{FGM}}$).

The Frequent Hypergraph Mining Problem Class ($\mathcal{C}_{\mathrm{FHM}}$): It consists of the set of $(\mathcal{L}_D, \mathcal{L}_P, \preccurlyeq)$-Frequent-Pattern-Mining problems, where $\mathcal{L}_D, \mathcal{L}_P \subseteq \mathcal{H}$ (i.e., $\mathcal{L}_D$ and $\mathcal{L}_P$ are sets of labeled hypergraphs).

The Frequent Graph Mining Problem Class ($\mathcal{C}_{\mathrm{FGM}}$): It is the set of $(\mathcal{L}_D, \mathcal{L}_P, \preccurlyeq)$-Frequent-Pattern-Mining problems, where $\mathcal{L}_D, \mathcal{L}_P \subseteq \mathcal{H}_2$ (i.e., $\mathcal{L}_D$ and $\mathcal{L}_P$ are sets of labeled graphs).

Clearly, $\mathcal{C}_{\mathrm{FGM}} \subsetneq \mathcal{C}_{\mathrm{FHM}} \subsetneq \mathcal{C}_{\mathrm{FPM}}$. It also holds that the *frequent itemset mining problem* [1] belongs to $\mathcal{C}_{\mathrm{FPM}}$; for this problem we have $\mathcal{L}_D = \mathcal{L}_P = \{X \subset \mathbb{N} : |X| < \infty\}$ and $\preccurlyeq$ is the subset relation. In fact, the frequent itemset mining problem is contained by $\mathcal{C}_{\mathrm{FHM}}$. Indeed, this problem can be considered as the $(\mathcal{H}_1^{\mathrm{ni}}, \mathcal{H}_1^{\mathrm{ni}}, \preccurlyeq)$-Frequent-Hypergraph-Mining problem, where $\preccurlyeq$ is the subhypergraph relation and the transaction and pattern languages are the set of 1-uniform node injective hypergraphs.

[3] A binary relation is a *quasiorder* (or *preorder*), if it is reflexive and transitive.

The *parameter* of a $(\mathcal{L}_D, \mathcal{L}_P, \preccurlyeq)$-Frequent-Hypergraph-Mining problem formulated above is the *size* of $\mathcal{D}$ defined by

$$\text{size}(\mathcal{D}) = \max\left\{\sum_{H\in\mathcal{D}} \text{size}(H), \max_{H\in\mathcal{D}} r(H)\right\} .$$

Note that the size of the output, i.e. the set to be enumerated, can be exponential in the size of the input. Because in such cases, it is impossible to compute them in time polynomial only in the size of the input, we investigate whether the enumeration problems can be solved in *incremental polynomial time* or at least in *output-polynomial time* (or *polynomial total time*) (see, e.g., [9,14]). In the first, more restrictive case, the algorithm is required to list the first N elements of the output in time polynomial in the *combined size* of the input and the set of these N elements for every integer $N > 0$. In the second, more liberal case, the algorithm has to solve the problem in time polynomial in the combined size of the input and the *entire* set to be enumerated. Note that the class of output-polynomial time algorithms entails the class of incremental polynomial time algorithms.

To sketch the relation among frequent pattern mining problems, we need the notion of *polynomial reduction.* More precisely, we say that the frequent pattern mining problem $P_1 = (\mathcal{L}_{D,1}, \mathcal{L}_{P,1}, \preccurlyeq_1)$ is *polynomially reducible* to the frequent pattern mining problem $P_2 = (\mathcal{L}_{D,2}, \mathcal{L}_{P,2}, \preccurlyeq_2)$ if there exist functions

$$g : 2^{\mathcal{L}_{D,1}} \times \mathbb{N} \to 2^{\mathcal{L}_{D,2}} \times \mathbb{N} \;\text{ and }\; f : \mathcal{L}_{P,2} \to \mathcal{L}_{P,1}$$

satisfying the following properties:

(i) $|\mathcal{F}_{(\mathcal{L}_{D,1},\mathcal{L}_{P,1},\preccurlyeq_1)}(I)| = |\mathcal{F}_{(\mathcal{L}_{D,2},\mathcal{L}_{P,2},\preccurlyeq_2)}(g(I))|$ for every $I \in 2^{\mathcal{L}_{D,1}} \times \mathbb{N}$,

(ii) f is injective, and

(iii) g and f can be computed in polynomial time.

That is, a pattern $\varphi \in \mathcal{L}_{P,1}$ is frequent for I if and only if it is the image (under f) of a pattern $\varphi' \in \mathcal{L}_{P,2}$ which is frequent for $g(I)$. Thus, if P_1 is polynomial-time reducible to P_2 then any enumeration algorithm solving P_2 in incremental polynomial (resp. output-polynomial) time can be used to solve P_1 in incremental polynomial (resp. output-polynomial) time.

Clearly, several frequent hypergraph mining problems, even frequent graph mining problems, cannot be solved in output-polynomial time (unless P = NP). In Proposition 1 below we present a simple example of such a hard problem.

Proposition 1. *Let $\mathcal{L}_D \subseteq \mathcal{H}_2$ and let $\mathcal{L}_P \subseteq \mathcal{H}_2$ be the set of complete graphs such that every vertex of every graph in $\mathcal{L}_D \cup \mathcal{L}_P$ is labeled by the same symbol, say* 1*, and let $\preccurlyeq$ be the homomorphism $\preccurlyeq_h$ between labeled graphs*[4]*. Then, unless $P = NP$, the $(\mathcal{L}_D, \mathcal{L}_P, \preccurlyeq_h)$-*Frequent-Graph-Mining *problem cannot be solved in output polynomial time.*

[4] A *homomorphism* from a hypergraph $H_1 \in \mathcal{H}$ to a hypergraph $H_2 \in \mathcal{H}$, denoted $H_1 \preccurlyeq_h H_2$, is a function $\varphi : V_{H_1} \to V_{H_2}$ preserving the labels and edges.

Proof. Let $G = (V, \mathcal{E})$ be an unlabeled graph and let G' be the labeled graph obtained from G by assigning 1 to each vertex of G. Then, for $\mathcal{D} = \{G'\}$ and $t = 1$, we have that G has a clique of size k if and only if there is a pattern $C \in \mathcal{F}_{(\mathcal{L}_D,\mathcal{L}_P,\preccurlyeq_h)}(\mathcal{D}, t)$ with k vertices. Since $|\mathcal{F}_{(\mathcal{L}_D,\mathcal{L}_P,\preccurlyeq_h)}(\mathcal{D}, t)| \leq |V|$, the size of the output is bounded by a polynomial of the input parameters. But this implies that the $(\mathcal{L}_D, \mathcal{L}_P, \preccurlyeq_h)$-FREQUENT-GRAPH-MINING problem cannot be computed in output polynomial time (unless P = NP), as otherwise the maximum clique problem[5] could be decided in polynomial time by computing first $\mathcal{F}_{(\mathcal{L}_D,\mathcal{L}_P,\preccurlyeq_h)}(\mathcal{D}, t)$ and then the pattern of maximum size from $\mathcal{F}_{(\mathcal{L}_D,\mathcal{L}_P,\preccurlyeq_h)}(\mathcal{D}, t)$. □

4 Frequent Subhypergraph Mining

By Proposition 1, the class $\mathcal{C}_{\mathrm{FGM}}$, and thus the more general class $\mathcal{C}_{\mathrm{FHM}}$ as well, contains problems that cannot be solved in output polynomial time (unless P = NP). This negative result raises the challenge of identifying practically relevant and tractable problems belonging to $\mathcal{C}_{\mathrm{FHM}}$. In this section, we take a first step towards this direction by considering the problem of frequent hypergraph mining w.r.t. *subhypergraph isomorphism*. This problem, called *frequent subhypergraph mining*, is a natural problem of the frequent hypergraph mining problem class $\mathcal{C}_{\mathrm{FHM}}$ and can be applied to many practical problems. In Section 5, we will employ this setting to tackle the citation analysis problem sketched in the introduction.

We start with the definition of the generalization relation used in this section. Let $H_1, H_2 \in \mathcal{H}$. H_1 can be embedded into H_2 by *subhypergraph isomorphism*, denoted by $H_1 \preccurlyeq_i H_2$, if H_2 has a subhypergraph isomorphic to H_1. Note that $\preccurlyeq_i$ generalizes the notion of subgraph isomorphism between ordinary labeled graphs to hypergraphs. Since $\preccurlyeq_i$ is a partial order on $\mathcal{H}$, it is a generalization relation on every subset of $\mathcal{H}$. Using $\preccurlyeq_i$, in this section we consider the

$$(\mathcal{H}, \mathcal{H}^c, \preccurlyeq_i)\text{-FREQUENT-HYPERGRAPH-MINING}$$

problem of $\mathcal{C}_{\mathrm{FHM}}$ and will refer to this problem as the *frequent subhypergraph mining problem.*

Although the pattern language in the frequent subhypergraph mining problem is restricted to *connected* hypergraphs, any enumeration algorithm for this problem can in fact be used to enumerate frequent, not necessarily connected, subhypergraphs as well. Indeed, for a set $\mathcal{D} \subseteq \mathcal{H}$ of hypergraphs, one can consider the set of connected hypergraphs obtained from $\mathcal{D}$ by the following transformation: Let $\ell \in \mathbb{N}$ be a label not used in any of the hypergraphs in $\mathcal{D}$. For every $H \in \mathcal{D}$, introduce a new vertex v, label it by ℓ, and add v to each edge of H. Clearly, any subhypergraph of the obtained hypergraph is connected and uniquely represents a (not necessarily connected) subhypergraph of H.

[5] *Given* an unlabeled graph $G = (V, \mathcal{E})$, *find* a clique of maximum size in G. This problem is NP-complete [11]. A clique of G is a subset $V' \subseteq V$ such that each pair of vertices in V' are joined by and edge in $\mathcal{E}$.

4.1 Negative Results

In this section we show that the frequent subhypergraph mining problem cannot be solved in output-polynomial time. This follows directly from Theorem 2 below which states that even for ordinary graphs (i.e., 2-uniform hypergraphs), the frequent subhypergraph mining problem is intractable in output-polynomial time. Since this is one of the most frequently considered frequent graph mining problems, this negative result may be of interest in itself.

Theorem 2. *If P $\neq$ NP, there is no output-polynomial time algorithm solving the frequent subhypergraph mining problem even in the case of* 2*-uniform hypergraphs (i.e., ordinary graphs).*

Proof. We show that if such an algorithm would exist then the NP-complete Hamiltonian path problem could be decided in polynomial time. Let $G = (V, \mathcal{E})$ be an ordinary undirected graph with n vertices. Let $H_G, H \in \mathcal{H}_2$ be labeled graphs such that

- $V_{H_G} = V$, $\mathcal{E}_{H_G} = \mathcal{E}$,
- $|V_H| = n$, and $\mathcal{E}_H$ consists of $n - 1$ (hyper)edges that form a simple path (i.e., each vertex of H occurs exactly once in the path), and
- each vertex in H_G and H has the same label.

H will be used to make sure the size of the output is small. One can easily check that G has a Hamiltonian path if and only if there is a path of length $n-1$ in $F = \mathcal{F}_{(\mathcal{H},\mathcal{H}^c,\preccurlyeq_i)}(\{H_G, H\}, 2)$. Since $|F| \leq n$, this can be decided in time polynomial in n if F can be enumerated in output-polynomial time. $\square$

As another restriction, we consider the frequent subhypergraph mining problem restricted to acyclic hypergraphs [10] because several NP-hard problems on hypergraphs become polynomial for acyclic hypergraphs. A hypergraph $H \in \mathcal{H}$ is *α-acyclic* if one can remove all of its vertices and edges by deleting repeatedly either an edge that is empty or is contained by another edge, or a vertex contained by at most one edge [20]. Note that α-acyclicity is not a hereditary property, that is, α-acyclic hypergraphs may have subhypergraphs that are not α-acyclic. Consider for example the hypergraph $H \in \mathcal{H}$ such that $\mathcal{E}_H = \{\{a, b\}, \{b, c\}, \{a, c\}, \{a, b, c\}\}$. While H is α-acyclic, its subhypergraph obtained by removing the edge $\{a, b, c\}$ is not α-acyclic. To overcome this anomaly, the following proper subclass of α-acyclic hypergraphs is introduced in [10]: An α-acyclic hypergraph is *β-acyclic*, if each of its subhypergraphs is also α-acyclic. Note that forests are 2-uniform β-acyclic hypergraphs.

Let $\mathcal{B}_3$ denote the set of 3-uniform β-acyclic hypergraphs. For connected subhypergraphs of 3-uniform β-acyclic hypergraphs, the following negative result holds:

Proposition 3. *Given a finite set $\mathcal{D} \subseteq \mathcal{B}_3$ and integer $t > 0$, deciding whether $H \in \mathcal{F}_{(\mathcal{B}_3,\mathcal{H}^c,\preccurlyeq_i)}(\mathcal{D}, t)$ is NP-hard.*

Proof (sketch). We use a polynomial reduction from the *subforest isomorphism* problem[6]. Let F be a forest and T be a tree and consider the hypergraph F' (resp. T') obtained from F (resp. T) by introducing a new vertex labeled by a symbol $\ell \in \mathbb{N}$ used neither in F nor in T, and by adding the new vertex to each edge of F (resp. T). Clearly, $F', T' \in \mathcal{B}_3$ and there is a subgraph isomorphism from F to T if and only if $F' \preccurlyeq_i T'$, from which the statement follows. □

Proposition 3 above indicates that for the frequent subhypergraph mining problem, the usual frequent pattern mining approaches (such as the level-wise one) will not work in incremental polynomial time (unless $P = NP$) because they repeatedly test whether candidate patterns satisfy the frequency threshold (see, e.g., [12,15]).

4.2 A Naïve Algorithm

Despite the negative worst-case result stated in the previous section, in this section we present a naïve algorithm for the frequent subhypergraph mining problem. The algorithm is based on a reduction to mining frequent subgraphs from labeled bipartite graphs.[7]

More precisely, for an instance $(\mathcal{D}, t)$ of the frequent subhypergraph mining problem, let $n \in \mathbb{N}$ be an upper bound on the labels occurring in the hypergraphs of $\mathcal{D}$ and let μ be an injection assigning an integer greater than n to every finite multiset of $\mathbb{N}$. For a hypergraph $H \in \mathcal{D}$, let $LB(H) \in \mathcal{H}_2$ be the (labeled) *bipartite* graph such that

(i) $(V_{LB(H)}, \mathcal{E}_{LB(H)})$ is the *unlabeled* bipartite incidence graph of the unlabeled hypergraph $(V_H, \mathcal{E}_H)$, and
(ii) for every $v \in V_{LB(H)} = V_H \cup \mathcal{E}_H$,

$$\lambda_{LB(H)}(v) = \begin{cases} \lambda_H(v) & \text{if } v \in V_H \\ \mu(\lambda^H(v)) & \text{otherwise (i.e., } v \in \mathcal{E}_H) \end{cases} .$$

Clearly, a subgraph G of $LB(H)$ represents a subhypergraph of H if and only if each vertex of G corresponding to a hyperedge $E \in \mathcal{E}_H$ is connected with exactly $|E|$ vertices in G. Using the above transformation and considerations, the set $\mathcal{F}_{(\mathcal{H}, \mathcal{H}^c, \preccurlyeq_i)}(\mathcal{D}, t)$ of t-frequent subhypergraphs for the instance $(\mathcal{D}, t)$ can be computed by Algorithm 1.

Although by Theorem 2, Algorithm 1 does not work in output-polynomial time in the worst case, using a state-of-the-art frequent graph mining algorithm it proved to be effective in time on the citation analysis domain (see Section 5).

[6] Given a forest F and a tree T, decide whether T has a subgraph isomorphic to F. This problem is known to be NP-complete [11].

[7] Another naïve approach could be the following algorithm: Select a hyperedge E of a frequent pattern, attach a hyperedge E' to E, and compute the support count of the obtained pattern. Beside the intractability of deciding subhypergraph isomorphism, the number of possible attachments of E' to E can be exponential in their cardinality.

Algorithm 1. Frequent Subhypergraph Mining

```
Require: an instance $(\mathcal{D}, t) \in 2^{\mathcal{H}} \times \mathbb{N}$
Ensure: $\mathcal{F}_{(\mathcal{H},\mathcal{H}^c,\preccurlyeq_i)}(\mathcal{D}, t)$
1: $\mathcal{F} := \emptyset$
2: $\mathcal{B}_{\mathcal{D}} := \{LB(H) : H \in \mathcal{D}\}$
3: Compute a next $t$-frequent connected bipartite subgraph $B$ of the set $\mathcal{B}_{\mathcal{D}}$ if it exists;
   otherwise return $\mathcal{F}$
4: if $B$ corresponds to some hypergraph $H_B$ then
     $\mathcal{F} := \mathcal{F} \cup \{H_B\}$
5: goto 3
```

4.3 Tractable Cases

In this section we present positive results for two special cases of the frequent subhypergraph mining problem obtained by making assumptions on the *labeling functions* of the transaction hypergraphs. We first consider the problem for *node injective* hypergraphs, i.e., where the labeling functions are injective. We show that for this case, the frequent subhypergraph mining problem is polynomially reducible to the frequent itemset mining problem and hence, it can be solved in incremental-polynomial time [1]. We then generalize this positive result to edge injective hypergraphs, i.e., to hypergraphs not containing two different hyperedges that are mapped to the same multiset by the labeling function. Although node injective hypergraphs are a special case of edge injective hypergraphs, we discuss the two cases separately because node injective hypergraphs can be used to model many practical problems and they permit a simplified algorithmic approach.

Node Injective Hypergraphs. As mentioned above, many practical data mining problems can be modeled by node injective hypergraphs, i.e., by hypergraphs from $\mathcal{H}^{\text{ni}}$. Such applications include problem domains consisting of a finite set of objects (vertices) with a unique identifier. For node injective hypergraphs, we consider the $(\mathcal{H}^{\text{ni}}, \mathcal{H}^{\text{ni}}, \preccurlyeq_i)$-Frequent-Hypergraph-Mining problem which is a special case of the frequent subhypergraph mining problem.

As an example of a practical application of this problem, we consider the *citation analysis* task mentioned in the introduction (see also Section 5): *Given* a set $\mathcal{D}$ of articles and a frequency threshold $t > 0$, *compute* each family $\mathcal{F}$ of groups of authors satisfying the following property: there exists a subset $\mathcal{D}' \subseteq \mathcal{D}$ of articles of cardinality at least t such that for every group $F \in \mathcal{F}$ of authors and for every article $D \in \mathcal{D}'$ it holds that D cites some article written by (exactly) the authors belonging to F. In this enumeration problem, we can assign a unique non-negative integer to each author, whose papers are cited by at least one article in $\mathcal{D}$.

We can use the *node injective hypergraph representation* of a paper's bibliography defined as follows. For each author cited in the bibliography, introduce a vertex and label it by the integer assigned to the author. Furthermore, for each

1. R. Agrawal, R. Srikant. *Publication 1.*
2. H. Mannila, H. Toivonen. *Publication 2.*
3. J. Quinlan. *Publication 3.*
4. H. Toivonen, R. Srikant, R. Agrawal. *Publication 4.*

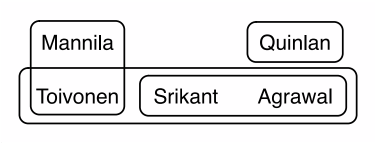

Fig. 1. An example reference-list and the according hypergraph

cited work add a hyperedge E to the set of hyperedges, where E consists of the vertices representing the cited work's authors. Clearly, the hypergraph obtained in this way is always node injective, similar to the example in Figure 1.[8] Our database $\mathcal{D}$ is a set of such node injective hypergraphs.

Theorem 4 below states that for node injective hypergraphs, the frequent subhypergraph mining problem is polynomially reducible to the frequent itemset mining problem. We recall that the frequent itemset mining problem can be considered as a problem belonging to the class $\mathcal{C}_{\mathrm{FHM}}$ (see Section 3). Notice that in the theorem below, subhypergraphs may be non-connected. The theorem is based on the fact that for every node injective hypergraphs $H_1, H_2 \in \mathcal{H}^{\mathrm{ni}}$, $H_1 \preccurlyeq_i H_2$ if and only if for every $E_1 \in \mathcal{E}_{H_1}$, there is a hyperedge $E_2 \in \mathcal{E}_{H_2}$ such that $\lambda^{H_1}(E_1) = \lambda^{H_2}(E_2)$, i.e.,

$$H_1 \preccurlyeq_i H_2 \iff \{\lambda^{H_1}(E) : E \in \mathcal{E}_{H_1}\} \subseteq \{\lambda^{H_2}(E) : E \in \mathcal{E}_{H_2}\} \ .$$

Note that the above equivalence implies that $\preccurlyeq_i$ can be decided efficiently for node injective hypergraphs.

Theorem 4. *The frequent subhypergraph mining problem for node injective hypergraphs is polynomially reducible to the frequent itemset mining problem.*

Proof (sketch). The proof follows by considering the set of vertex labels of a hyperedge as an item for every hyperedge occurring in the transaction hypergraphs. □

Combining the above theorem with the results of [1], we have the following result on listing frequent subhypergraphs for node injective hypergraphs.

Corollary 5. *The frequent subhypergraph mining problem for node injective hypergraphs can be solved in incremental polynomial time.*

Edge Injective Hypergraphs. In Theorem 6 below we generalize the previous positive result to *edge injective hypergraphs*. Since edge injective hypergraphs may contain different vertices with the same label, they are not determined by a family of multisets of vertex labels (in contrast to the previous case). Hence, a polynomial reduction to frequent itemset mining is not applicable to this case.

[8] To facilitate better comprehensibility the artificial node connecting all hyperedges is omitted in this example.

Algorithm 2. Mining Edge Injective Hypergraphs

Require: an instance $(\mathcal{D}, t) \in 2^{\mathcal{H}^{\mathrm{ei}}} \times \mathbb{N}$
Ensure: $\mathcal{F}_{(\mathcal{H}^{\mathrm{ei}}, \mathcal{H}^{\mathrm{ei}}, \preccurlyeq_i)}(\mathcal{D}, t)$

```
1:  $X := \bigcup_{H \in \mathcal{D}} \{\lambda^H(E) : E \in \mathcal{E}_H\}$
2:  $F := \emptyset$
3:  $k := 0$
4:  while $k = 0 \vee L_k \neq \emptyset$ do
5:    $k := k + 1$
6:    $C_k := \begin{cases} X & \text{if } k = 1 \\ \{Y_1 \cup Y_2 \in [X]^k : Y_1, Y_2 \in L_{k-1}\} & \text{otherwise} \end{cases}$
7:    $L_k := \emptyset$
8:    forall $X' \in C_k$ do
9:      $Q := \emptyset$
10:     forall $H \in \mathcal{D}$ do
11:       if $H$ has a subhypergraph $H'$ s.t. $X' = \{\lambda^{H'}(E) : E \in \mathcal{E}_{H'}\}$ then
12:         if $\exists (H'', f) \in Q$ s.t. $H'' \simeq H'$ then
13:           change $(H'', f)$ in $Q$ to $(H'', f + 1)$
14:         else $Q := Q \cup \{(H', 1)\}$
15:     endfor
16:     flag := TRUE
17:     forall $(H, f) \in Q$ s.t. $f \geq t$ do
18:       $F := F \cup \{H\}$
19:       if flag then
20:         $L_k := L_k \cup \{X'\}$
21:         flag := FALSE
22:       endif
23:     endfor
24:   endfor
25: endwhile
26: return $F$
```

Theorem 6. *The frequent subhypergraph mining problem for edge injective hypergraphs can be solved in incremental polynomial time.*

Proof (sketch). Due to space limitations, we only sketch the proof. Without proof we first note that subhypergraph isomorphism between edge injective hypergraphs can be decided in polynomial time. To compute the set of frequent hypergraphs, we use a level-wise algorithm given in Algorithm 2.

In line 1 of the algorithm, X is initialized as the set of multisets corresponding to the edges in the transaction hypergraphs. In C_k (line 6), we compute a family of candidate sets of multisets; the elements of C_k consist of k multisets corresponding to the vertex labels of k hyperedges. For every X' in C_k (see the loop starting at line 8), we check for every $H \in \mathcal{D}$ whether H has a subhypergraph H' such that the set of multisets defined by the vertex labels of the edges of H' is equal to X'. Since edges are injectively labeled, H' must contain exactly k hyperedges. If H has such a subhypergraph H' then we check whether we have already found another hypergraph in the database which has a subhypergraph isomorphic to H'. If so, we increment the counter of this subhypergraph (line 13); otherwise we add H' with frequency 1 to the set Q (line 14). In the loop (17–23) we update the set of frequent hypergraphs and L_k. One can show that this algorithm works in incremental polynomial time. □

5 Experimental Evaluation

In this section, we empirically evaluate (i) the naïve algorithm (see Sect. 4.2) and (ii) the method based on the reduction of the node injective case to frequent itemset mining (see Sect. 4.3) on the citation analysis problem discussed earlier.

5.1 Bibliographic Datasets

Three different bibliographic data sets were constructed from the ACM Digital library[9]: KDD, SIGMOD, and SIGGRAPH. They correspond to the set of all reference lists of papers found in the proceedings of the respective conferences. The characteristics of the data-sets are listed in Table 1.

Table 1. Datasets used. We list the total number of papers in the proceedings and the number of authors occurring in the reference lists of the corresponding papers.

dataset	years	papers	authors
KDD	99-04	499	6966
SIGMOD	74-04	1404	11984
SIGGRAPH	74-04	1519	13192

A simple parser was used to extract the authors and cited papers occurring in the reference lists. Each paper was then represented as a hypergraph, as already discussed above. The resulting hypergraphs are *node injective* and disconnected in almost all cases. Most existing graph miners only consider *connected* graphs. Hence, we added one special hyperedge to each paper, which connects all authors cited in that paper such that the naïve algorithm could be employed.

All experiments were run on a workstation, running Suse Linux 9.2, 3.2 GHz, 2GB of RAM. As graph miner for the naïve algorithm, we employed Siegfried Nijssen's implementation of GASTON [18] and as item-set miner for the reduction method, Bart Goethals' implementation of Apriori[10]. Since we did not employ a specialized hypergraph or graph miner, the data had to be pre- and post-processed. The pre- and post-processing steps run in time linear in the number of hypergraphs.

[9] *http://www.acm.org/*
[10] *http://www.cs.helsinki.fi/u/goethals/software/*

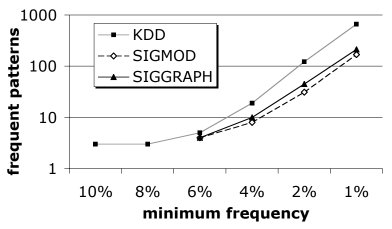

Fig. 2. Frequent patterns in the *node* injective setting

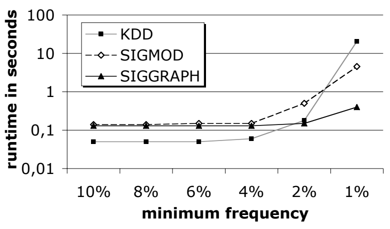

Fig. 3. Runtimes for the naïve approach in the *node* injective setting

5.2 Experimental Results

As mentioned above, we performed experiments employing reductions to frequent bipartite graph mining and to frequent item-set mining. The empirical results are given in Figures 2 and 3.

The runtime of the method based on the reduction to frequent item-set mining was always below 0.01 seconds. Different from that, the naïve approach based on the reduction to frequent bipartite graph mining shows much higher runtimes. In detail, the 1% settings required 20.36, 4.55, and 0.4 seconds for KDD, SIGMOD, and SIGGRAPH respectively. The higher runtimes for the naïve approach are essentially due to the problem that only a fraction of the frequent bipartite graphs are in fact subhypergraphs (see line 4 of Algorithm 1). As usual for frequent pattern mining techniques, the runtime and size of the frequent pattern space increase exponentially with a decreasing level of minimum support (see Figures 2 and 3). One of the frequent subhypergraphs in the KDD dataset was $\{\{Agrawal, Srikant\}, \{Agrawal, Swami, Imielinski\}\}$, while in the SIGGRAPH dataset the hypergraph $\{\{Sproull, Newmann\}\}$ was very frequent. The experimental results reveal that the reduction approach can be successfully employed in practice. As expected due to the theoretical results, the reduction to item-set mining for injective subhypergraphs is much less computationally expensive. All experiments w.r.t. node injective subhypergraphs were finished in less than a fraction of a second, which – in our opinion – indicates that it is not worth-while to implement a special purpose data mining system for this task.

6 Conclusion and Further Research

In this paper the problem class $\mathcal{C}_{\mathrm{FHM}}$ of frequent hypergraph mining was introduced. It forms a natural extension of traditional frequent itemset and graph mining. Several problems of $\mathcal{C}_{\mathrm{FHM}}$ were studied and positive and negative complexity results were obtained.

In our first step of studying some problems of $\mathcal{C}_{\mathrm{FHM}}$ we deliberately did not develop and implement a special hypergraph mining algorithm, because there are many problems of $\mathcal{C}_{\mathrm{FHM}}$ that are interesting (which implies the need for implementing many variants and optimizations). Instead, some of our theoretical and practical results have been obtained by reductions to frequent graph mining and itemset mining problems. The experiments clearly indicate that – at least for the citation analysis problems studied – these reductions can be quite effective in practice. In addition, these experiments provide evidence that frequent hypergraph mining is indeed a useful generalization of frequent itemset and graph mining and is likely to yield many interesting applications.

Finally we list some open questions.

(i) One of the challenges is to identify further problems of $\mathcal{C}_{\mathrm{FHM}}$ that are enumerable in incremental or at least in output-polynomial time.
(ii) Besides subhypergraph isomorphism, it would be interesting to investigate frequent hypergraph mining problems, where the generalization relation is defined by (constrained) *homomorphisms*.
(iii) Since many problems of $\mathcal{C}_{\mathrm{FHM}}$ can be reduced to frequent graph mining in bipartite graphs, it would be interesting to develop frequent graph mining algorithms specific to bipartite graphs.
(iv) The work on frequent hypergraph mining can be related to multi-relational data mining [7], where each instance consists of multiple tuples over multiple tables in a relational database. Multi-relational data mining techniques have been applied to graph mining problems. Hence, the question arises if they are also applicable to hypergraph mining, and vice versa.
(v) In a similar way that frequent hypergraph mining generalizes frequent graph mining in *undirected* graphs, frequent pattern mining in *relational structures* (see, e.g., [8]) can be considered as a generalization of frequent graph mining in *directed* graphs. Similarly to the problem class $\mathcal{C}_{\mathrm{FHM}}$, the *Frequent Relational Structure Mining Problem Class* ($\mathcal{C}_{\mathrm{FRSM}}$) can be defined as the set of $(\mathcal{L}_D, \mathcal{L}_P, \preccurlyeq)$-FREQUENT-PATTERN-MINING problems, where $\mathcal{L}_D$ and $\mathcal{L}_P$ are classes of relational τ-structures over some vocabulary τ. Thus, the $(\mathcal{L}_D, \mathcal{L}_P, \preccurlyeq)$-FREQUENT-RELATIONAL-STRUCTURE-MINING problem can be defined as follows: *Given* a finite set $\mathcal{D} \subseteq \mathcal{L}_D$ of relational τ-structures and an integer $t > 0$, *compute* the set of relational τ-structures from $\mathcal{L}_P$ that generalize at least t structures of $\mathcal{D}$ with respect to $\preccurlyeq$. To the best of our knowledge, there are only a few results towards this direction. In particular, related problems have been considered only for the generalization relations *relational homomorphism* [5] and *relational substructure isomorphism* [17]. The challenge is to identify tractable problems of $\mathcal{C}_{\mathrm{FRSM}}$.

Acknowledgments

The authors thank Mario Boley, Thomas Gärtner, Stefan Wrobel, and the anonymous reviewers for useful comments. Tamás Horváth was partially supported by the DFG project (WR 40/2-2) *Hybride Methoden und Systemarchitekturen für heterogene Informationsräume*. Björn Bringmann and Luc De Raedt were partially supported by the IQ project (EU grant IST-FET FP6-516169).

References

1. Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., Verkamo, A.I.: Fast discovery of association rules. In: Advances in Knowledge Discovery and Data Mining, pp. 307–328. AAAI/MIT Press (1996)
2. Berge, C.: Hypergraphs. North Holland Mathematical Library, vol. 445. Elsevier, Amsterdam (1989)
3. Boros, E., Elbassioni, K., Gurvich, V., Khachiyan, L., Makino, K.: Dual bounded hypergraphs: A survey. In: Proc. of the 2nd SIAM Conference on Data Mining, pp. 87–98 (2002)
4. Chi, Y., Nijssen, S., Muntz, R.R., Kok, J.N.: Frequent subtree mining–an overview. Fundamenta Informaticae 66, 161–198 (2005)
5. Dehaspe, L., Toivonen, H.: Discovery of frequent datalog patterns. Data Mining and Knowledge Discovery 3, 7–36 (1999)
6. Diestel, R.: Graph theory, 3rd edn. Springer, Heidelberg (2005)
7. Džeroski, S., Lavrač, N. (eds.): Relational Data Mining. Springer, New York (2002)
8. Ebbinghaus, H.-D., Flum, J.: Finite Model Theory, 2nd edn. Springer, Heidelberg (1999)
9. Eiter, T., Gottlob, G.: Identifying the minimal transversals of a hypergraph and related problems. SIAM Journal on Computing 24(6), 1278–1304 (1995)
10. Fagin, R.: Degrees of acyclicity for hypergraphs and relational database schemes. Journal of the ACM 30(3), 514–550 (1983)
11. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to NP-Completeness. Freeman, San Francisco, CA (1979)
12. Gunopulos, D., Khardon, R., Mannila, H., Saluja, S., Toivonen, H., Sharm, R.S.: Discovering all most specific sentences. ACM Transactions on Database Systems 28(2), 140–174 (2003)
13. Horváth, T., Ramon, J., Wrobel, S.: Frequent subgraph mining in outerplanar graphs. In: Proc. of the 12th ACM SIGKDD International Conference on Knowledge discovery and Data Mining, pp. 197–206. ACM Press, New York (2006)
14. Johnson, D.S., Yannakakis, M., Papadimitriou, C.H.: On generating all maximal independent sets. Information Processing Letters 27(3), 119–123 (1988)
15. Mannila, H., Toivonen, H.: Levelwise search and borders of theories in knowledge discovery. Data Mining and Knowledge Discovery 1(3), 241–258 (1997)
16. Mannila, H., Toivonen, H., Verkamo, A.I.: Discovery of frequent episodes in event sequences. Data Mining and Knowledge Discovery 1(3), 259–289 (1997)
17. Nijssen, S., Kok, J.N.: Efficient frequent query discovery in farmer. In: Proc. of the Seventeenth International Joint Conference on Artificial Intelligence, pp. 891–896. Morgan Kaufmann, San Francisco (2001)
18. Nijssen, S., Kok, J.N.: A quickstart in frequent structure mining can make a difference. In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, Washington, USA, August 22-25, 2004, pp. 647–652. ACM Press, New York (2004)

19. Yan, X., Han, J.: gspan: Graph-based substructure pattern mining. In: Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002), Maebashi City, Japan, 9-12 December 2002, pp. 721–724. IEEE Computer Society Press, Los Alamitos (2002)
20. Yu, C.T., Ozsoyoglu, M.Z.: An algorithm for tree-query membership of a distributed query. In: Proceedings of Computer Software and Applications Conference, pp. 306–312. IEEE Computer Society Press, Los Alamitos (1979)
21. Zaki, M.J.: Efficiently mining frequent trees in a forest. In: Proceedings of the 8th ACM SIGKDD International Conference, pp. 71–80. ACM Press, New York (2002)