Frequent Hypergraph Mining

Tamás Horváth[1,2], Björn Bringmann[3], and Luc De Raedt[3]

[1] Department of Computer Science III, University of Bonn, Germany
[2] Fraunhofer IAIS, Sankt Augustin, Germany
[3] Institute for Computer Science, Machine Learning Lab, University of Freiburg, Germany

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., [11]), trees (e.g., [4]), and more recently, graph mining has become an important focus of research (e.g., [12, 13]). 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 graph mining. 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. 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 type 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 the frequent hypergraph mining problem even in the case of strong *structural* assumptions on the hyperedges. On the other hand, by restricting the functions labeling the vertices, we get 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 present also 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, proofs are only sketched or even omitted in this short version.

2 Notions and Notations

We recall some basic notions and notations related to graphs and hypergraphs (see, e.g., [2, 5] for detailed introductions into these fields). For a set S and non-negative integer k, $[S]^k$ denotes the family of 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 set $\mathcal{E} \subseteq [V]^2$ 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 and the *size* of H, denoted size(H), is the number of hyperedges of H.

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 λ, called *labeling function*, is a function mapping V to $\mathbb{N}$.[4] Unless

[4] 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 $E \in \mathcal{E}$

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*[5] $\{\lambda_H(v) : v \in V'\}$ 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 $H_1 \simeq H_2$.

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[6], which satisfy some monotone property (see, e.g., [3, 9]). 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 ($\mathcal{C}_{\text{FPM}}$): 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 quasi-order 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

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

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 $\lambda(v)$.

[5] 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.

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

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 and graphs, respectively.

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 satisfying $\mathcal{L}_D, \mathcal{L}_P \subseteq \mathcal{H}$.

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 satisfying $\mathcal{L}_D, \mathcal{L}_P \subseteq \mathcal{H}_2$ (i.e., they are sets of labeled graphs).

Clearly, $\mathcal{C}_{\mathrm{FGM}} \subsetneq \mathcal{C}_{\mathrm{FHM}} \subsetneq \mathcal{C}_{\mathrm{FPM}}$. Furthermore, 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.

To sketch the relation among frequent pattern mining problems, we need the notion of *polynomial reduction*. More precisely, let $P_1 = (\mathcal{L}_{D,1}, \mathcal{L}_{P,1}, \preccurlyeq_1)$ and $P_2 = (\mathcal{L}_{D,2}, \mathcal{L}_{P,2}, \preccurlyeq_2)$ be frequent pattern mining problems, and $I_1 = (\mathcal{D}_1, t_1)$ and $I_2 = (\mathcal{D}_2, t_2)$ be instances of P_1 and P_2, respectively. Then I_1 is *polynomially equivalent* to I_2 if there is a function $f : \mathcal{L}_{P,1} \to \mathcal{L}_{P,2}$ such that

(i) f is a bijection between $\mathcal{F}_{(\mathcal{L}_{D,1},\mathcal{L}_{P,1},\preccurlyeq_1)}(\mathcal{D}_1, t_1)$ and $\mathcal{F}_{(\mathcal{L}_{D,2},\mathcal{L}_{P,2},\preccurlyeq_2)}(\mathcal{D}_2, t_2)$ and
(ii) the inverse of f on $\mathcal{F}_{(\mathcal{L}_{D,2},\mathcal{L}_{P,2},\preccurlyeq_2)}(\mathcal{D}_2, t_2)$ can be computed in polynomial time.

The definition implies that $\varphi \in \mathcal{L}_{P,1}$ is frequent for I_1 if and only if $f(\varphi) \in \mathcal{L}_{P,2}$ is frequent for I_2. Using the notion of polynomial equivalence, we say that P_1 is *polynomially reducible* to P_2 if there is a function g from the set of instances of P_1 to the set of instances of P_2 such that

(i) I is polynomially equivalent to $g(I)$ for every $I \in P_1$ and
(ii) g can be computed in polynomial time.

Thus, if P_1 is polynomial-time reducible to P_2 then any enumeration algorithm solving P_2 can be used to solve P_1.

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

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

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., [10]). 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. 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 properly entails the class of incremental polynomial time algorithms.

To close this section, we note that several frequent hypergraph mining problems, even frequent graph mining problems, cannot be solved in output-polynomial time. In Theorem 1 below we present such a hard problem.

Theorem 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*[7]*. Then, unless P = NP, the $(\mathcal{L}_D, \mathcal{L}_P, \preccurlyeq_h)$-*FREQUENT-GRAPH-MINING *problem cannot be solved in output polynomial time.*

Proof (sketch). 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 $C \in \mathcal{F}_{(\mathcal{L}_D, \mathcal{L}_P, \preccurlyeq_h)}(\mathcal{D}, t)$ with k vertices. Since $|\mathcal{F}_{(\mathcal{L}_D, \mathcal{L}_P, \preccurlyeq_h)}(\mathcal{D}, t)| \leq |V|$, the $(\mathcal{L}_D, \mathcal{L}_P, \preccurlyeq_h)$-FREQUENT-GRAPH-MINING problem cannot be computed in output polynomial time (unless P = NP, of course), as otherwise the NP-complete maximum clique problem [8] could be decided in polynomial time by computing the largest pattern in $|\mathcal{F}_{(\mathcal{L}_D, \mathcal{L}_P, \preccurlyeq_h)}(\mathcal{D}, t)|$.

4 Frequent Subhypergraph Mining

By Theorem 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, of course). 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 $\preccurlyeq_i$, we consider the following two problems of $\mathcal{C}_{\mathrm{FHM}}$:

(i) The $(\mathcal{H}, \mathcal{H}, \preccurlyeq_i)$-FREQUENT-HYPERGRAPH-MINING problem called the *frequent subhypergraph mining problem* and
(ii) the $(\mathcal{H}, \mathcal{H}^c, \preccurlyeq_i)$-FREQUENT-HYPERGRAPH-MINING problem called the *frequent connected subhypergraph mining problem*.

[7] 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.

Algorithm 1 FREQUENT SUBHYPERGRAPH MINING

Require: instance $(\mathcal{D}, t)$
Ensure: $\mathcal{F}_{(\mathcal{H},\mathcal{H},\preccurlyeq_i)}(\mathcal{D}, t)$

1: $\mathcal{F} := \emptyset$
2: $\mathcal{B}_\mathcal{D} := \{LB(H) : H \in \mathcal{D}\}$
3: Compute a next t-frequent 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

4.1 A Naïve Algorithm

Using the bipartite graph representation of hypergraphs, in this section we present a naïve algorithm solving the frequent subhypergraph mining problem. For an instance $(\mathcal{D}, t)$ of this 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 $H \in \mathcal{H}$, 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 $v \in V_{LB(H)} = V_H \cup \mathcal{E}_H$,

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

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 $\mathcal{F}_{(\mathcal{H},\mathcal{H},\preccurlyeq_i)}(\mathcal{D}, t)$ of t-frequent subhypergraphs for the instance $(\mathcal{D}, t)$ can be computed by Algorithm 1.

Due to space limitations, we omit a detailed analysis of Algorithm 1 that does not work in output polynomial time in the worst case. We note, however, that even this naïve algorithm proved to be effective in time on the citation analysis domain (cf. Section 5).

4.2 Negative Results

In this section we investigate further problems obtained by *structural* restrictions hoping to identify tractable fragments of the frequent subhypergraph mining problem. Unfortunately, we have not been able to find any interesting positive result in this way. This is because, as indicate the negative results of this section, the problem remains hard even for very restricted hypergraph classes. Without proof, in Theorem 2 below we state that even for 2-uniform hypergraphs (i.e., ordinary graphs), the frequent connected subhypergraph mining problem is intractable in output-polynomial time. Since this is one

of the most frequently considered frequent graph mining problems, the negative result below may be of interest in itself.

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

As a further restriction, we consider the frequent subhypergraph mining problem restricted to acyclic hypergraphs [7] 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 [14]. 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 [7]: 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.

Even for connected subhypergraphs of 3-uniform β-acyclic hypergraphs, we have a negative result. Let $\mathcal{B}_3$ denote the set of 3-uniform β-acyclic hypergraphs.

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). Using a polynomial reduction from the *subforest isomorphism* problem[8], one can show that deciding subhypergraph isomorphism between 3-uniform, connected, β-acyclic hypergraphs is NP-hard. This implies the statement.

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., [9]).

4.3 Tractable Cases

In contrast to the approach discussed in Section 4.2 above, in this section we consider further special cases of the frequent subhypergraph mining problem that are 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

[8] 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 [8].

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 (cf. also Section 5): *Given* a set $\mathcal{D}$ of articles and a frequency threshold $t > 0$, *print* 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 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. 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 $\mathcal{C}_{\text{FHM}}$ (cf. 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 $H_1, H_2 \in \mathcal{H}^{\text{ni}}$, $H_1 \preccurlyeq_i H_2$ 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.*

Due to space limitation, we omit the proof of the theorem which is based on 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.*

Algorithm 2 MINING EDGE INJECTIVE HYPERGRAPHS

Require: finite set $\mathcal{D} \subseteq \mathcal{H}^{\text{ei}}$ and integer $t > 0$
Ensure: $\mathcal{F}_{(\mathcal{H}^{\text{ei}}, \mathcal{H}^{\text{ei}}, \preccurlyeq_i)}(\mathcal{D}, t)$

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** $\exists H' \in \mathcal{H}$ 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

Edge Injective Hypergraphs We now generalize the previous positive result to *edge injective hypergraphs*. Since a hypergraph now may contain two vertices with the same label, a family of multisets of labels doesn't define a hypergraph uniquely. Hence, a reduction to frequent itemset mining is not applicable to this case.

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. We first note that subhypergraph isomorphism between edge injective hypergraphs can be decided in polynomial time. To compute the set of t-frequent hypergraphs, we use an Apriori-like 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. Each set in C_k consists of k multisets. For a set X' in C_k (see the

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

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 edges of H' is equal to X'. Since edges are invectively 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 yes, 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 evaluate our methods on the citation analysis problem discussed earlier. Three bibliographic datasets, the KDD, SIGMOD, and SIGGRAPH, were constructed from the ACM Digital Library[9]. They correspond to the set of all reference lists of papers found in the proceedings of the respective conferences. The characteristics of the datasets are listed in Table 1.

Each paper was represented as a hypergraph, as described in Section 4.3. The resulting hypergraphs are *node injective*, and in almost all cases, also disconnected. Because most existing graph miners only consider *connected* graphs, we added one special hyperedge to each paper, which connects all authors cited in that paper.

We performed experiments with the naïve algorithm based on reduction to frequent bipartite graph mining (cf. Section 4.1), as well as with the reduction to frequent itemset mining (cf. Section 4.3). All experiments were run on a workstation, running Suse Linux 9.2, 3.2 GHz, 2GB of RAM. As graph miner, we employed Siegfried Nijssens' GASTON [12] and as itemset miner, Bart Goethals' implementation[10] of Apriori. Because we did not employ a specialized hypergraph or graph miner, the data had to be pre- and post-processed. We used several `perl`-scripts to realize this. The pre- and post-processing steps run in time linear in the number of hypergraphs.

5.1 Experimental Results

The number of frequent patterns for different frequency thresholds for the three datasets is given in Figure 1. The runtime of the itemset miner was always below 0.1 seconds. Different from that, the naïve algorithm required much higher runtimes; 833.5, 16.8,

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

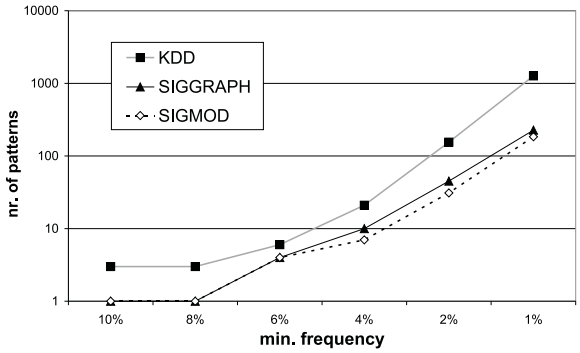

Fig. 1. Number of frequent patterns for different frequency thresholds.

and 9.9 seconds for KDD, SIGGRAPH, and SIGMOD, respectively. These higher runtimes are essentially due to the problem that only a fraction of the frequent bipartite graphs correspond in fact to subhypergraphs.

Despite the difference between the runtimes of the two approaches, our experiments clearly indicate that both reductions can be quite effective in practice.

6 Conclusion and Further Research

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. Central to our results is the use of reductions from hypergraph mining problems to itemset and graph mining problems. These reductions are not only of theoretical interest, but allow us to set up a number of frequent hypergraph mining experiments even though we have not implemented a frequent hypergraph mining system. To our knowledge, the use of such reductions in data mining is new. Indeed, it is much more common in data mining to spend a lot of time and effort to develop new systems, even though they are sometimes only variants of existing ones. In our first step of studying some problems of $\mathcal{C}_{\mathrm{FHM}}$, we deliberately did not follow this common methodology, because there are many problems of $\mathcal{C}_{\mathrm{FHM}}$ that are interesting (which implies the need for implementing many variants and optimizations), and also, because we wanted to see how far the reductions would bring us. The experiments clearly indicate that - at least for the citation analysis problems studied - 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 [6], 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.

Acknowledgments

The authors thank Mario Boley and Stefan Wrobel for useful comments. Tamás Horváth was partially supported by the DFG project (WR 40/2-2) *Hybride Methoden und Systemarchitekturen für heterogene Informationsräume*.

References

1. R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast Discovery of Association Rules. In *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI/MIT Press, Cambridge, MA, 1996.
2. C. Berge. *Hypergraphs*. North Holland Mathematical Library, Vol. 445. Elsevier, Amsterdam, 1989.
3. E. Boros, K. Elbassioni, V. Gurvich, L. Khachiyan, and K. Makino. Dual Bounded Hypergraphs: A Survey. In *Proc. of the 2nd SIAM Conference on Data Mining*, pages 87–98, 2002.
4. Y. Chi, R. R. Muntz, S. Nijssen, and J. N. Kok. Frequent Subtree Mining - An Overview. *Fundamenta Informaticae*, 66(1-2):161–198, 2005.
5. R. Diestel. *Graph Theory*. Springer, New York, 2nd edition, 2000.
6. S. Dzeroski and N. Lavrac, editors. *Relational Data Mining*. Springer, New York, 2002.
7. R. Fagin. Degrees of Acyclicity for Hypergraphs and Relational Database Schemes. *Journal of the ACM*, 30(3):514–550, 1983.
8. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to NP-Completeness*. Freeman, San Francisco, CA, 1979.
9. D. Gunopulos, R. Khardon, H. Mannila, S. Saluja, H. Toivonen, and R. S. Sharm. Discovering All Most Specific Sentences. *ACM Trans. on Database Systems*, 28(2):140–174, 2003.
10. D. S. Johnson, M. Yannakakis, and C. H. Papadimitriou. On Generating All Maximal Independent Sets. *Information Processing Letters*, 27(3):119–123, 1988.
11. H. Mannila, H. Toivonen, and A. I. Verkamo. Discovery of Frequent Episodes in Event Sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289, 1997.
12. S. Nijssen and J. N. Kok. A Quickstart in Frequent Structure Mining Can Make a Difference. In *Proc. of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 647–652. ACM Press, New York, NY, 2004.
13. X. Yan and J. Han. Gspan: Graph-based Substructure Pattern Mining. In *Proc. of the 2002 IEEE International Conference on Data Mining*, pages 721–724. IEEE Computer Society, 2002.
14. C. T. Yu and M. Z. Ozsoyoglu. An Algorithm for Tree-Query Membership of a Distributed Query. In *Proc. of Computer Software and Applications Conference*, pages 306–312. IEEE Computer Society, 1979.