Reductions for Frequency-Based Data Mining Problems

Stefan Neumann*[1] and Pauli Miettinen[2]

[1]University of Vienna, Faculty of Computer Science, Vienna, Austria
stefan.neumann@univie.ac.at
[2]Max Planck Institute for Informatics, Saarland Informatics Campus, Germany
pauli.miettinen@mpi-inf.mpg.de

Abstract

Studying the computational complexity of problems is one of the – if not the – fundamental questions in computer science. Yet, surprisingly little is known about the computational complexity of many central problems in data mining. In this paper we study frequency-based problems and propose a new type of reduction that allows us to compare the complexities of the maximal frequent pattern mining problems in different domains (e.g. graphs or sequences). Our results extend those of Kimelfeld and Kolaitis [ACM TODS, 2014] to a broader range of data mining problems. Our results show that, by allowing constraints in the pattern space, the complexities of many maximal frequent pattern mining problems collapse. These problems include maximal frequent subgraphs in labelled graphs, maximal frequent itemsets, and maximal frequent subsequences with no repetitions. In addition to theoretical interest, our results might yield more efficient algorithms for the studied problems.

1 Introduction

Computational complexity is a fundamental concept in computer science, with the P vs. NP question being the most famous open problem in the field. Yet, outside some NP- and #P-hardness proofs, computational complexity of the central data mining problems is surprisingly little studied. This is perhaps even more true for the *frequency-based problems*, that is, for problems where the goal is to enumerate all sufficiently frequent patterns (that admit other possible constraints). Problems such as frequent itemset mining, frequent subgraph mining, and frequent subsequence mining all belong to this family of problems.

*The first author gratefully acknowledges the financial support from the Doctoral Programme "Vienna Graduate School on Computational Optimization" which is funded by the Austrian Science Fund (FWF, project no. W1260-N35).

Often the only computational complexity argument for these problems is the observation that the output can be exponentially large with respect to the input, and hence any algorithm might need exponential time to enumerate the results.

We argue that this view is too limited for two reasons. First, there are more fine-grained models of complexity than just the running time. In particular, for enumeration problems we can use the framework of Johnson et al. [18]: in short, instead of studying the total running time with respect to the input size, we can consider it as a function of the total size of input *and output*, or study the time it takes to create a *new* pattern when a set of patterns is already known (see Section 2.2 for more details). This framework allows us to argue about the time complexity of enumeration problems with potentially exponential output sizes. Another approach is the counting complexity framework of Valiant [30] (see Section 2.3).

The second reason why we argue that the "output is exponential" is a too limited view for the computational complexity is that a significant question in computational complexity is the relationships between the problems, that is, questions like "can we solve problem X efficiently if we can solve problem Y efficiently?" The main tool for answering these kinds of questions are *reductions* between problems. In this work, we introduce a new type of reduction between frequency-based problems called *maximality-preserving reduction* (see Section 4). Our reduction maps the maximal patterns of one problem to the maximal patterns of the other problem, thus allowing us to study questions like "can we find the maximal frequent subgraphs on labelled graphs using maximal frequent itemset mining algorithms?" Surprisingly, the answer to this question turns out to be positive, although it requires that we consider specially constrained maximal frequent pattern mining problems; we call the general class of such problems *feasible frequency-based problems* (see Section 5).

Our Contributions. We study a number of maximal pattern mining problems, including maximal subgraph mining in labelled graphs (and in more restricted structures), maximal frequent itemset mining, and maximal subsequence mining with no repetitions (see Section 2.4 for definitions of all of these problems). We summarize our results in Figure 1: the arrows show which problem can be reduced to which other problem either using non-constraining reductions (black and red lines), or with possible constraints on the feasible solutions (dashed lines). As can be seen in Figure 1, all problems can be reduced to each other (potentially with constraints). Given that the constrained reductions are transitive (Lemma 10), we can state our main result:

Theorem 1 (Informal)**.** *Maximal subgraph mining in labelled graphs (and in more restricted structures), maximal frequent itemset mining, and maximal subsequence mining with no repetitions are equally hard problems when we are allowed to constrain the pattern space.*

In some sense, our results unify all existing hardness results for frequency-based problems by putting them into a general framework using maximality-preserving reductions. These reductions preserve all interesting theoretical

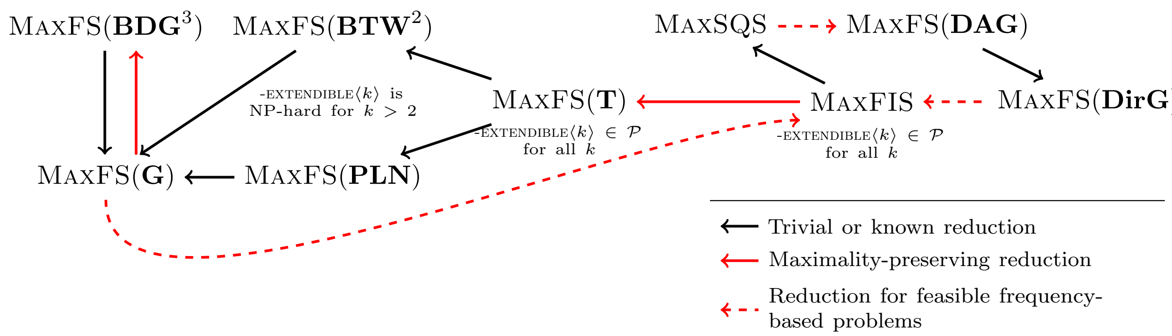

Figure 1: The hierarchy of maximal frequency-based problems with the results from this paper. Arrows point from the "easier" to the "harder" problem. See Section 2.4 for the abbreviated problem names used in the picture. Maximality-preserving reductions are defined in Section 4 and feasible frequency-based problems are defined in Section 5.

aspects like NP- or #P-hardness, but are also restricted enough to maintain the special properties of the transactions.

In fact, from a practical point of view, our reductions show that if we have an algorithm that can effectively find, say, the maximal frequent itemsets that admit the constraints from the reductions, we can use that algorithm to solve maximal frequent subgraph mining and maximal frequent subsequence mining problems efficiently. Luckily, as we will see in Section 6, the constrained maximal patterns are indeed easy to mine in practice. Alternatively, the reductions can be used to guide how ideas from algorithms for one set of problems can be transferred to algorithms for the other set of problems (e.g. from frequent subsequence mining to frequent subgraph mining or vice versa).

This paper is an extended version of our conference paper [23]. It contains all omitted proofs, and experimental evaluation.

Outline of the Paper. We will cover the basic definitions and frameworks used in this paper in Section 2, where we will also formally define the problems we are working with. Section 3 presents related work and existing hardness results for the problems we consider. We introduce the (unconstrained) maximality-preserving reductions in Section 4. In particular, the reductions corresponding to the solid red lines in Figure 1 are presented in Section 4.2. The feasible frequency-based problems, and the corresponding constrained reductions (dashed red lines in Figure 1) and related results are presented in Section 5. In Section 6 we show that our reductions can be used in practice and yield efficient algorithms.

2 Preliminaries

In this section we quickly cover the basic definitions of frequency-based problems, enumeration problems, and counting complexity. In addition, we present the definitions of the problems we consider in the paper.

2.1 Frequency-based Problems

A *frequency-based* problem $\mathcal{P}$ consists of[1]:

- A set of labels $\mathcal{L}$; for example, $\mathcal{L} = \{1, \ldots, n\}$.
- A set *transactions*($\mathcal{P}$) consisting of possible transactions over the labels $\mathcal{L}$.
- A set *patterns*($\mathcal{P}$) $\subseteq$ *transactions*($\mathcal{P}$) of possible patterns over the labels $\mathcal{L}$.
- A partial order $\sqsubseteq$ over *transactions*($\mathcal{P}$).

Given a frequency-based problem $\mathcal{P}$, a *database* $D_{\mathcal{P}}$ is a finite multiset of elements from *transactions*($\mathcal{P}$). For a database $D_{\mathcal{P}}$ and a *support threshold* τ, a pattern $p \in$ *patterns*($\mathcal{P}$) is called τ*-frequent* if

$$\mathrm{supp}(p, D_{\mathcal{P}}) := |\{t \in D_{\mathcal{P}} : p \sqsubseteq t\}| \geq \tau.$$

In other words, a pattern p is frequent if it appears in at least τ transactions of the database. When τ is clear from the context, we will call p only *frequent*. A pattern $p \in$ *patterns*($\mathcal{P}$) is a *maximal frequent* pattern if p is frequent and all patterns $q \in$ *patterns*($\mathcal{P}$) with $p \sqsubset q$ are not frequent. Given a database $D_{\mathcal{P}}$, we denote the set of all maximal frequent patterns by $\textsc{Max}(D_{\mathcal{P}}, \tau)$, i.e., $\textsc{Max}(D_{\mathcal{P}}, \tau) = \{p \in \textit{patterns}(\mathcal{P}) : p \text{ is a maximal } \tau\text{-frequent pattern in } D_{\mathcal{P}}\}$.

When the parameter τ is not part of the input but fixed to some integer, we write $\mathcal{P}^\tau$ to denote the resulting problem.

2.2 Enumeration Problems

An *enumeration relation* $\mathcal{R}$ is a set of strings $\mathcal{R} = \{(x, y)\} \subset \{0,1\}^* \times \{0,1\}^*$ such that

$$\mathcal{R}(x) := \{y \in \{0,1\}^* : (x, y) \in \mathcal{R}\}$$

is finite for every x. A string $y \in \mathcal{R}(x)$ is called a *witness* for x. We call $\mathcal{R}$ an NP*-relation* if (1) there exists a polynomial p such that $|y| \leq p(|x|)$ for all $(x, y) \in \mathcal{R}$, and (2) there exists a polynomial-time algorithm deciding if $(x, y) \in \mathcal{R}$ for any given pair (x, y).

Following [21], we define the following problems for an enumeration relation $\mathcal{R}$:

[1] A similar definition was given in Gunopulos et al. [12].

- $\mathcal{R}$-ENUMERATE: The input is a string x. The task is to output the set $\mathcal{R}(x)$ without repetitions.
- $\mathcal{R}$-EXTEND: The input is a string x and a set $Y \subseteq \mathcal{R}(x)$. The task is to compute a string y such that $y \in \mathcal{R}(x) \setminus Y$ or to output that no such element exists.
- $\mathcal{R}$-EXTENDIBLE: The input is a string x and a set $Y \subseteq \mathcal{R}(x)$. The task is to decide whether $\mathcal{R}(x) \setminus Y \neq \emptyset$.
- $\mathcal{R}$-EXTENDIBLE$\langle k \rangle$: The input is a string x and a set $Y \subseteq \mathcal{R}(x)$ with the restriction that $|Y| < k$. The task is to decide whether $\mathcal{R}(x) \setminus Y \neq \emptyset$.

The problem $\mathcal{R}$-EXTEND can be viewed as the decision version of $\mathcal{R}$-EXTEND. Note that by repeatedly running an algorithm for $\mathcal{R}$-EXTEND, one can solve $\mathcal{R}$-ENUMERATE. Further observe that any algorithm solving $\mathcal{R}$-EXTEND can be used to solve $\mathcal{R}$-EXTENDIBLE.

Enumeration Complexity. Johnson et al. [18] introduced different notions for the complexity of enumeration problems. Let $\mathcal{R}$ be an enumeration relation. An algorithm solving $\mathcal{R}$-ENUMERATE is called an *enumeration algorithm*.

For enumeration problems it might be the case that the output $\mathcal{R}(x)$ is exponentially larger than the input x. Due to this, measuring the running time of an enumeration algorithm only as a function of $|x|$ can be too restrictive; instead, one can include the size of $\mathcal{R}(x)$ in the complexity analysis. Then the running time of an algorithm is measured as function of $|x| + |\mathcal{R}(x)|$. This consideration gives rise to the following definitions:

- An enumeration algorithm runs in *total polynomial time* if its running time is polynomial in $|x| + |\mathcal{R}(x)|$.
- An enumeration algorithm has *polynomial delay* if the time spent between outputting two consecutive witnesses of $\mathcal{R}(x)$ is always polynomial in $|x|$.
- An enumeration algorithm runs in *incremental polynomial time* if on input x and after outputting a set $Y \subseteq \mathcal{R}(x)$ it takes time polynomial in $|x|+|Y|$ to produce the next witness from $\mathcal{R}(x) \setminus Y$.

We note that $\mathcal{R}$-ENUMERATE is in incremental polynomial time if and only if $\mathcal{R}$-EXTEND is in polynomial time. Additionally, observe that a polynomial total time algorithm can be used to decide if $\mathcal{R}(x) \neq \emptyset$.

Relationship to Frequency-Based Problems. We note that frequency-based problems are special cases of enumeration problems. Let $\mathcal{P}$ be a frequency-based problem. We define the enumeration relation $\mathcal{R}$ corresponding to $\mathcal{P}$ by setting

$$\mathcal{R} = \{(x, y) : x = (D_{\mathcal{P}}, \tau), y \in \text{MAX}(D_{\mathcal{P}}, \tau)\},$$

i.e., $\mathcal{R}$ consists of all possible databases $D_{\mathcal{P}}$, support thresholds τ and all maximal frequent patterns y for the tuples $(D_{\mathcal{P}}, \tau)$.

Observe that $\mathcal{R}(x) = \mathcal{R}(D_{\mathcal{P}}, \tau) = \textsc{Max}(D_{\mathcal{P}}, \tau)$ and, hence, the problem $\mathcal{R}$-ENUMERATE is exactly the same problem as outputting all maximal frequent patterns in $\textsc{Max}(D_{\mathcal{P}}, \tau)$. The problem $\mathcal{R}$-EXTEND is to output a maximal frequent pattern in $\textsc{Max}(D_{\mathcal{P}}, \tau) \setminus Y$ for a given set of maximal patterns Y. The problems $\mathcal{R}$-EXTENDIBLE and $\mathcal{R}$-EXTENDIBLE$\langle k \rangle$ are the corresponding decision versions of the problems.

Since $\mathcal{R}$ and $\mathcal{P}$ yield the same enumeration problems, we will also write $\mathcal{P}$-ENUMERATE, $\mathcal{P}$-EXTENDIBLE, $\mathcal{P}$-EXTEND, $\mathcal{P}$-EXTENDIBLE$\langle k \rangle$. Often we will write $\mathcal{P}$ to denote the problem $\mathcal{P}$-ENUMERATE.

2.3 Counting Complexity

For a given enumeration relation $\mathcal{R}$, the function $\#\mathcal{R} : \{0,1\}^* \to \mathbb{N}$ returns the number of witnesses for a given string, i.e., $\#\mathcal{R}(x) = |\mathcal{R}(x)|$ for $x \in \{0,1\}^*$. The complexity class #P (pronounced "sharp P") contains all functions $\#\mathcal{R}$ for which $\mathcal{R}$ is an NP-relation; it was introduced by Valiant [30]. A function $F : \{0,1\}^* \to \mathbb{N}$ is #P-*hard* if there exists a Turing reduction from every function in #P to F.

For two NP-relations $\mathcal{R}, \mathcal{Q} : \{0,1\}^* \to \mathbb{N}$, a *parsimonious reduction from $\#\mathcal{R}$ to $\#\mathcal{Q}$* is a polynomial-time computable function $f : \{0,1\}^* \to \{0,1\}^*$ such that $\#\mathcal{R}(x) = \#\mathcal{Q}(f(x))$ for all $x \in \{0,1\}^*$. Note that a parsimonious reduction from a #P-hard problem $\mathcal{R}$ to a problem $\mathcal{Q}$ implies that $\mathcal{Q}$ is #P-hard.

An example for a #P-hard problem is counting *the number* of satisfying assignments of a SAT formula. Note that such an algorithm can decide if the given formula is satisfiable or not (by checking if the number of satisfying assignments is larger than 0). Hence, #P is a superset of NP.

In fact, Toda and Ogiwara [28] showed that all problems in the polynomial-time hierarchy can be solved in polynomial-time when one has access to an oracle for a #P-hard function.

Observe that an algorithm solving $\mathcal{R}$-ENUMERATE can solve $\#\mathcal{R}$ by counting the number of witnesses in its output.

2.4 Problems Considered in This Paper

All problems considered in this paper are frequency-based problems. For the sake of brevity, we only define $\mathcal{L}$, *transactions*$(\cdot)$, *patterns*$(\cdot)$, and $\sqsubseteq$ for each problem (see, e.g., [1] for more thorough definitions).

The *maximal frequent itemset mining* problem, denoted as MaxFIS, is as follows: We have n labels $\mathcal{L} = \{1, \dots, n\}$; *transactions*(MaxFIS) and *patterns*(MaxFIS) are given by $2^{\mathcal{L}}$; $\sqsubseteq$ is the standard subset relationship $\subseteq$.

The *maximal frequent subsequence mining* problem, denoted as MaxSQS, is as follows: $\mathcal{L} = \{1, \dots, n\}$ is the set of labels. A *sequence* $S = \langle S_1, \dots, S_m \rangle$ of length m consists of m events S_i with $S_i \in \mathcal{L}$; we require that *each label appears at most once per sequence*. The sets *transactions*(MaxSQS) and *patterns*(MaxSQS) are the sets consisting of all sequences of arbitrary lengths. For two sequences $S = \langle S_1, \dots S_r \rangle$ and $T = \langle T_1, \dots, T_k \rangle$, we have $T \sqsubseteq S$ if

$k \leq r$ and there exist indices $1 \leq i_1 \leq \cdots \leq i_k \leq r$ such that $T_j = S_{i_j}$ for each $j = 1, \ldots, k$.

Let $\mathcal{G}$ be a class of *vertex-labelled* graphs, which contain each label at most once. The *maximal frequent subgraph mining* problem, MaxFS($\mathcal{G}$), is as follows: We have n labels $\mathcal{L} = \{1, \ldots, n\}$; *transactions*(MaxFS($\mathcal{G}$)) and *patterns*(MaxFS($\mathcal{G}$)) are given by all labelled graphs in $\mathcal{G}$ with labels from $\mathcal{L}$; $\sqsubseteq$ is the standard subgraph relationship for labelled graphs (i.e., we consider arbitrary subgraphs, not necessarily induced subgraphs).

In the remainder of the paper, we will consider the following graph classes, all of which are labelled and connected:

- **T** — undirected trees,
- **BDG**^{b} — undirected graphs of bounded degree at most b,
- **BTW**^{w} — undirected graphs of bounded treewidth at most w,
- **PLN** — undirected planar graphs,
- **G** — general undirected graphs,
- **DAG** — directed acyclic graphs,
- **DirG** — directed graphs.

Throughout the paper we will only consider labelled graphs *in which each label appears at most once.* In this restricted setting, the subgraph isomorphism problem can be solved in polynomial-time. This a necessary condition for our reductions to work since Kimelfeld and Kolaitis [20, Prop. 3.4] showed that for certain *unlabelled* graph classes $\mathcal{G}$, MaxFS($\mathcal{G}$) is not an NP-relation.

3 Related Work

Counting Complexity. The study of counting problems was initiated when Valiant [30] introduced #P. Provan and Ball [26] showed #P-hardness for many graph problems such as counting the number of maximal independent sets in bipartite graphs. Later, more #P-hardness results were obtained for even more restricted graph classes [17, 29].

Johnson et al. [18] introduced the notions of polynomial total time, polynomial delay, and incremental polynomial time to obtain a better understanding of the computational complexity of enumeration problems.

Computational Complexity of Data Mining Problems. Gunopulos et al. [12] introduced a general class of problems similar to frequency-based problems. For this class of problems, they proved #P-hardness for mining frequent sets, and provided an algorithm to mine maximal frequent sets.

Yang [32] proved #P-hardness for determining the number of *maximal* frequent itemsets and other problems.

Theorem 2 (Yang [32]). *The following problems are* #P*-complete:* MaxFIS*,* MaxFS(**T**)*,* MaxFS(**G**)*,* MaxSQS.

Boros et al. [7] showed that given a set of maximal frequent itemsets Y, it is NP-complete to decide whether there exists another maximal frequent itemset that is not contained in Y.

Theorem 3 (Boros et al. [7]). MaxFIS-extendible *and* MaxFIS-extend *are* NP*-complete.*

Kimelfeld and Kolaitis [20] proved structural results on mining frequent subgraphs of certain graph classes. Their results allow to distinguish the computational complexities of MaxFS(**T**) and MaxFS($\mathcal{G}$) where $\mathcal{G}$ is either **G**, **PLN**, $\mathbf{BDG}^b$ with $b > 2$, or $\mathbf{BTW}^w$ with $w > 1$. This is also depicted in Figure 1.

Theorem 4 (Kimelfeld and Kolaitis [21]). *For every fixed k, the problem* MaxFS(**T**)-extendible$\langle k \rangle$ *can be solved in polynomial time.*

For every fixed τ, the problem MaxFS$^\tau(\mathcal{G})$-enumerate *can be solved in polynomial time for any class of graphs $\mathcal{G}$ from Section 2.4.*

The following problems are NP*-complete:*

- MaxFS($\mathcal{G}$)-extendible *for* $\mathcal{G} \in \{\mathbf{G}, \mathbf{PLN}, \mathbf{BDG}^b, \mathbf{BTW}^w\}$ *with* $w \geq 1$ *and* $b \geq 3$.
- MaxFS($\mathcal{G}$)-extendible$\langle k \rangle$ *for* $\mathcal{G} \in \{\mathbf{G}, \mathbf{PLN}, \mathbf{BDG}^b, \mathbf{BTW}^w\}$ *with* $w > 1$ *and* $b > 2$ *and for every* $k > 2$.

In the journal version [21] of their paper [20], Kimelfeld and Kolaitis give computational hardness results for subgraph mining problems in which the set *patterns*() is more restricted than *transactions*(). For example, they consider the computational complexity of mining maximal subtrees from planar graphs. They also consider mining unlabelled maximal subgraphs.

Mining Maximal Frequent Patterns. Many algorithms were proposed to mine maximal frequent patterns from different types of data such as itemsets [8, 16, 19], subsequences [3], trees [31, 33], and general graphs [22]. However, the main focus of those papers was not to investigate the computational complexity of these problems. See (for example) the book by Aggarwal [1] for many more references to algorithms for efficiently computing maximal frequent patterns.

Constraint-based Pattern Mining. Many algorithms were proposed to mine frequent patterns with constraints on the structure of the patterns [4–6, 9, 10, 24, 25]. Due to lack of space we cannot review all of them, but refer to Han et al. [15] for references to many papers on constrained pattern mining. Greco et al. [11] presented techniques for mining taxonomies of process models which can also be viewed as constraint-based pattern mining. The work on constraint programming for itemset mining by Raedt et al. [27] and follow-up work (e.g. [14]) can also be used to mine itemsets or other frequency-based problems with constraints.

4 Maximality-Preserving Reductions

In this section, we introduce maximality-preserving reductions and state some of their properties in Section 4.1. In Section 4.2, we prove reductions between the problems MaxFIS, MaxSQS, and MaxFS($\mathcal{G}$) for $\mathcal{G} \in \{\mathbf{T}, \mathbf{BDG}^3, \mathbf{G}\}$. Combining our reductions with the statements from Section 3, we arrive at the following theorem.

Theorem 5. *Our reductions imply the following hardness results:*

1. *For any fixed k,* MaxFIS-extendible$\langle k \rangle$ *can be solved in polynomial time.*
2. *For any fixed τ,* MaxFIS$^\tau$-enumerate *can be solved in polynomial time.*
3. *The problems* MaxFS($\mathbf{G}$) *and* MaxFS($\mathbf{BDG}^3$) *exhibit exactly the same hardness w.r.t. the notions of Sections 2.2 and 2.3. More concretely, let $\mathcal{P}$ be* MaxFS($\mathbf{G}$) *or* MaxFS($\mathbf{BDG}^3$)*. Then the following statements are true:*
   - $\mathcal{P}$-enumerate *is* #P*-hard.*
   - $\mathcal{P}$-extendible *is* NP*-hard.*
   - *For $k > 2$, the problem* $\mathcal{P}$-extendible$\langle k \rangle$ *is* NP*-hard.*
   - *For fixed τ, the problem* $\mathcal{P}^\tau$-enumerate *is solvable in polynomial time.*

The proof of the theorem follows from our reductions later in this section and the theorems from Section 3.

4.1 Definition and Properties

We formally define maximality-preserving reductions to make explicit which properties are required by reductions in order to be useful for understanding the complexity of frequency-based problems w.r.t. to the notions of Sections 2.2 and 2.3.

Definition 1. Let $\mathcal{P}$ and $\mathcal{Q}$ be two frequency-based problems, let $D_\mathcal{P}$ be a database for $\mathcal{P}$, and let τ be a support threshold. A *maximality-preserving reduction* from $\mathcal{P}$ to $\mathcal{Q}$ defines an instance $(D_\mathcal{Q}, \tau)$ using a polynomial-time computable injective function $f\colon \mathit{transactions}(\mathcal{P}) \to \mathit{transactions}(\mathcal{Q})$ with the following properties:

1. $f(\mathit{patterns}(\mathcal{P})) \subseteq \mathit{patterns}(\mathcal{Q})$.
2. For all $p, p' \in \mathit{transactions}(\mathcal{P})$, $p \sqsubseteq_\mathcal{P} p'$ if and only if $f(p) \sqsubseteq_\mathcal{Q} f(p')$.
3. The inverse $f^{-1}\colon \mathit{transactions}(\mathcal{Q}) \to \mathit{transactions}(\mathcal{P})$ of f can be computed in polynomial time.

4. $p \in \textsc{Max}(D_{\mathcal{P}}, \tau)$ if and only if $f(p) \in \textsc{Max}(D_{\mathcal{Q}}, \tau)$, where $D_{\mathcal{Q}} = f(D_{\mathcal{P}}) = \{f(t) : t \in D_{\mathcal{P}}\}$. Additionally, for all $q \in \textsc{Max}(D_{\mathcal{Q}}, \tau)$ the preimage $f^{-1}(q)$ exists.

Intuitively, the properties can be interpreted as follows: Property 1 asserts that f maps valid patterns from *patterns*($\mathcal{P}$) to valid patterns in *patterns*($\mathcal{Q}$); this condition is necessary if *patterns*($\mathcal{Q}$) $\subsetneq$ *transactions*($\mathcal{Q}$). Property 2 asserts that f maintains subset properties. Property 3 will be necessary to recover patterns in $\mathcal{P}$ from those found in $\mathcal{Q}$. Property 4 requires that the maximal frequent patterns in $D_{\mathcal{P}}$ are the same as those in $D_{\mathcal{Q}}$ under the mapping f; here, the database $D_{\mathcal{Q}}$ is given by applying the function f to each transaction in $D_{\mathcal{P}}$.

Properties. Observe that Property 4 implies that there exists a bijective relationship between the maximal frequent patterns in $D_{\mathcal{P}}$ and in $D_{\mathcal{Q}}$. Hence, we have $|\textsc{Max}(D_{\mathcal{P}}, \tau)| = |\textsc{Max}(D_{\mathcal{Q}}, \tau)|$. This shows that maximality-preserving reductions are special cases of parsimonious reductions and that they preserve #P-hardness.

In fact, maximality-preserving reductions are slightly stronger than parsimonious reductions. They do not only preserve the number of maximal frequent patterns in both databases, but they enable us to recover the maximal frequent patterns in $D_{\mathcal{P}}$ from those in $D_{\mathcal{Q}}$: By injectivity of f and due to Property 4, we can reconstruct $\textsc{Max}(D_{\mathcal{P}}, \tau)$ in polynomial time from $\textsc{Max}(D_{\mathcal{Q}}, \tau)$. Hence, maximality-preserving reductions can be used to argue about the complexity of extendibility problems as discussed in Section 2.2.

Further, note that by choice of $D_{\mathcal{Q}}$ in Property 4, $D_{\mathcal{Q}}$ has the same number of transactions as $D_{\mathcal{P}}$, and that no dependency within different transactions is created by the mapping f. Additionally, by Property 2, the support of a pattern p in $D_{\mathcal{P}}$ is a lower bound on the support of $f(p)$ in $D_{\mathcal{Q}}$ (since for each transaction $t \in D_{\mathcal{P}}$ with $p \sqsubseteq t$, $f(p) \sqsubseteq f(t)$).

However, although the number of transactions and *maximal* frequent patterns in both databases remains the same, the number of *frequent* patterns in $D_{\mathcal{Q}}$ might be exponentially larger than the number of frequent patterns in $D_{\mathcal{P}}$. For example, this is the case in the reduction in Lemma 8.

4.2 Reductions

In this section, we present three maximality-preserving reductions. Reductions similar to ones in Lemmas 6 and 7 were already presented by Yang [32], Kimelfeld and Kolaitis [21] and other authors. We only prove Property 4 of maximality-preserving reductions. The proofs of Properties 1–3 are straight-forward and follow from the definitions of the mapping f.

Reduction from MaxFIS to MaxFS(T). We show how to mine maximal itemsets by mining maximal subtrees.

Lemma 6. *There exists a maximality-preserving reduction from* MaxFIS *to* MaxFS(**T**).

Proof. Consider MaxFIS with labels $\mathcal{L} = \{1, \ldots, n\}$. We construct trees over labels from the alphabet $\mathcal{L}' = \{r, 1, \ldots, n\}$, where r is the label of the root nodes in the trees. For simplicity, we do not distinguish between vertices and their labels.

Construction of f. An itemset $\{i_1, \ldots, i_k\} \in$ *transactions*(MaxFIS) is mapped to a tree of depth 1 with root r and children $i_1, \ldots, i_k$, i.e., the tree has an edge (r, i_j) for all $j = 1, \ldots, k$.

Maximality-preserving. Observe that there exists a bijection between itemsets $I \subseteq \mathcal{L}$ and trees $f(I)$. Further note that for two itemsets I and J, $I \subseteq J$ if and only if $f(I) \subseteq f(J)$. It follows that an itemset I and a tree $f(I)$ must have the same supports in D_{MaxFIS} and in $D_{\text{MaxFS}(\mathbf{T})}$, respectively. The maximality then follows from the subset-property we observed. □

From MaxFIS to MaxSQS. We show how to mine maximal itemsets by mining maximal subsequences.

Lemma 7. *There exists a maximality-preserving reduction from* MaxFIS *to* MaxSQS*.*

Proof. Construction of f. Consider MaxFIS with labels $\mathcal{L} = \{1, \ldots, n\}$ and assume the labels are ordered w.r.t. to some arbitrary, but fixed, order $\prec$. Let $I = \{i_1, \ldots, i_m\} \subseteq \mathcal{L}$ be any itemset with m items. Assume w.l.o.g. that the items in I are ordered w.r.t. the fixed order, i.e., $i_j \prec i_{j+1}$. Then I is mapped to the sequence $\langle i_1, \ldots, i_m \rangle$ of length m.

Maximality-preserving. Observe that there exists a bijection between itemsets $I \subseteq \mathcal{L}$ and sequences $f(I)$ (under the fixed order). Further observe that for two itemsets I and J, $I \subseteq J$ if and only if $f(I) \sqsubseteq f(J)$. It follows that an itemset I and a sequence $f(I)$ must have the same supports in D_{MaxFIS} and in D_{MaxSQS}, respectively. The maximality then follows from the subset-property we observed. □

From MaxFS($\mathbf{G}$) to MaxFS($\mathbf{BDG}^3$). We show that mining maximal frequent subgraphs in graphs with degrees bounded by 3 can be used to mine maximal frequent subgraphs in general undirected graphs. Note that this is the tightest result we could hope for, since graphs with degree bounded by 2 are simply cycles or line graphs.

Lemma 8. *There exists a maximality-preserving reduction from* MaxFS($\mathbf{G}$) *to* MaxFS($\mathbf{BDG}^3$)*.*

Proof. Construction of f. Let $G = (V, E)$ be a graph with unbounded degree of the vertices over labels $\mathcal{L} = \{1, \ldots, n\}$. Denote the label of a vertex $v \in V$ by *label*(v). We construct a graph $G' = (V', E')$ with bounded degree 3 over the set of labels $\mathcal{L}' = \{1, \ldots, n\}^2$.

Intuitively, the construction of f is picked such that every original vertex $v \in V$ is split into a line graph consisting of n vertices v_i, where each v_i has an additional non-line-graph-edge in G' iff vertices v and i share an edge in G.

Formally, for each vertex $v \in V$, we insert vertices $v_1, \ldots, v_n$ into V' with edges (v_i, v_{i+1}) for $i = 1, \ldots, n-1$. Each vertex v_i is labeled by $(\mathit{label}(v), i)$. For each edge $(u, v) \in E$, we insert an edge $(u_{\mathit{label}(v)}, v_{\mathit{label}(u)})$ into G'.

Observe that the resulting graph $G' = f(G)$ indeed has bounded degree 3: Consider any vertex $v_i \in V'$. The vertex has at most 2 neighbors from the line graph $(v_1, \ldots, v_n)$. The only additional edge it could have is to vertex $i_{label(v)}$.

Maximality-preserving. Let $p \in \textsc{Max}(D_{\textsc{MaxFS}(\mathbf{G})}, \tau)$. We need to show that $f(p) \in \textsc{Max}(D_{\textsc{MaxFS}(\mathbf{BDG}^3)}, \tau)$. By construction of f, we have that $\mathrm{supp}(f(p), D_{\textsc{MaxFS}(\mathbf{BDG}^3)}) = \mathrm{supp}(p, D_{\textsc{MaxFS}(\mathbf{G})})$; hence, $f(p)$ is frequent in $D_{\textsc{MaxFS}(\mathbf{BDG}^3)}$. We need to show that $f(p)$ is also maximal. For the sake of contradiction, suppose there exists a maximal frequent pattern q with $f(p) \sqsubset q$ in $D_{\textsc{MaxFS}(\mathbf{BDG}^3)}$. Then q must contain an edge (u_i, v_j) with $i = \mathit{label}(v)$, $j = \mathit{label}(u)$, which is not contained in $f(p)$.

Case 1: $u_i \in f(p)$ and $v_j \in f(p)$. Consider the graph $q' = f(p) \cup (u_i, v_j)$. Then $f^{-1}(q')$ exists and must be frequent in $D_{\textsc{MaxFS}(\mathbf{G})}$ by Property 2. This contradicts the maximality of p.

Case 2: W.l.o.g. assume that $u_i \in f(p)$ and $v_j \notin f(p)$. Then since q is maximal and by construction of f and $D_{\textsc{MaxFS}(\mathbf{BDG}^3)}$, q must contain the line graph L with vertices $v_1, \ldots, v_n$. Consider the graph $q' = f(p) \cup (u_i, v_j) \cup L$. Again by construction of f and $D_{\textsc{MaxFS}(\mathbf{BDG}^3)}$, q' has a preimage $p' = f^{-1}(q')$ which is frequent and satisfies $p \sqsubset p'$. This is a contradiction to the maximality of p.

Case 3: $u_i \notin f(p)$ and $v_j \notin f(p)$. Since q is connected and $f(p) \sqsubset q$, we only need to consider the first two cases.

Observe that the second part of Property 4 is implied by the previous three case distinctions. Proving that $f(p) \in \textsc{Max}(D_{\textsc{MaxFS}(\mathbf{BDG}^3)}, \tau)$ implies $p \in \textsc{Max}(D_{\textsc{MaxFS}(\mathbf{G})}, \tau)$ can be done similarly to above. □

5 Constraining the Set of Patterns

In this section, we generalize frequency-based problems by allowing to constrain the set of patterns using a feasibility function. We introduce maximality-preserving reductions for this class of problems and prove that all problems discussed in this paper exhibit exactly the same hardness after introducing the feasibility function.

5.1 Feasible Frequency-Based Problems

A *feasible frequency-based problem* (FFBP) $\mathcal{P}$ is a frequency-based problem with an additional polynomial-time computable operation $\phi\colon \mathit{patterns}(\mathcal{P}) \to \{0,1\}$ which can be described using constant space. Note that the operation ϕ is part of the input for the problem; this is the reason for restricting the description length of the function to constant size (otherwise, the description length of the function might be larger than the database for the problem). We call ϕ the *feasibility function.*

Given a feasible frequency-based problem $\mathcal{P}$, a pattern $p \in \mathit{patterns}(\mathcal{P})$ is a *feasible frequent pattern* (FFP) if p is frequent and $\phi(p) = 1$. The goal is to find all maximal FFP s; we denote the set of all FFP s by $\textsc{Max}(D_{\mathcal{P}}, \tau, \phi_{\mathcal{P}})$. We define MaxFFIS, MaxFSQS, and MaxFFS($\mathcal{G}$) for a graph class $\mathcal{G}$ as before for maximal frequency-based problems.

Note that FFBP s are generalizations of frequency-based problems since setting $\phi_{\mathcal{P}}$ to the function which is identical to 1, we obtain the underlying frequency-based problem.

The main result of this section is given in the following theorem.

Theorem 9. *The* FFBP*-version of all problems discussed in this paper exhibit exactly the same hardness w.r.t. the notions of Sections 2.2 and 2.3. More concretely, let $\mathcal{P}$ be any* FFBP*-problem discussed in this paper. Then the following statements are true:*

- $\mathcal{P}$-ENUMERATE *is* #P*-hard.*
- $\mathcal{P}$-EXTENDIBLE *is* NP*-hard.*
- *For $k > 2$, the problem $\mathcal{P}$-EXTENDIBLE$\langle k \rangle$ is* NP*-hard.*
- *For fixed τ, the problem $\mathcal{P}^{\tau}$-ENUMERATE is solvable in polynomial time.*

Theorem 9 shows that the hierarchy given in Figure 1 for frequency-based problems completely collapses when a feasibility function is introduced to the problem. Note that many practical algorithms (like the Apriori algorithm) for finding maximal frequent patterns allow to add such a feasibility function. Hence, our reductions give a theoretical justification why many of these algorithms can be extended to a broader range of problems.

The proof of the theorem follows from the reductions presented later in this section and the theorems from Section 3.

5.2 Maximality-Preserving Reductions for FFPPs

We start by defining maximality-preserving reductions between two FFBP s $\mathcal{P}$ and $\mathcal{Q}$.

Definition 2. Let $\mathcal{P}$ and $\mathcal{Q}$ be two FFBP s. Let $D_{\mathcal{P}}$ be a database for $\mathcal{P}$, let $\phi_{\mathcal{P}}$ be the feasibility function for $\mathcal{P}$, and let τ be a support threshold.

A *maximality-preserving reduction* from $\mathcal{P}$ to $\mathcal{Q}$ defines an instance $(D_{\mathcal{Q}}, \tau, \phi_{\mathcal{Q}})$ using a polynomial-time computable injective function $f\colon \mathit{transactions}(\mathcal{P}) \to \mathit{transactions}(\mathcal{Q})$ with the following properties:

1. $f(\mathit{patterns}(\mathcal{P})) \subseteq \mathit{patterns}(\mathcal{Q})$.
2. For all $p, p' \in \mathit{transactions}(\mathcal{P})$, $p \sqsubseteq_{\mathcal{P}} p'$ if and only if $f(p) \sqsubseteq_{\mathcal{Q}} f(p')$.
3. The inverse $f^{-1}\colon \mathit{transactions}(\mathcal{Q}) \to \mathit{transactions}(\mathcal{P})$ of f can be computed in polynomial time.
4. $p \in \mathrm{MAX}(D_{\mathcal{P}}, \tau, \phi_{\mathcal{P}})$ if and only if $f(p) \in \mathrm{MAX}(D_{\mathcal{Q}}, \tau, \phi_{\mathcal{Q}})$, where $D_{\mathcal{Q}} = f(D_{\mathcal{P}}) = \{f(t) : t \in D_{\mathcal{P}}\}$. Additionally, for all $q \in \mathrm{MAX}(D_{\mathcal{Q}}, \tau, \phi_{\mathcal{Q}})$ the preimage $f^{-1}(q)$ exists.

Note that compared to Definition 1, we only had to change Property 4 to assert that the maximal patterns are feasible. Further observe that in general the function $\phi_{\mathcal{Q}} = \phi_{\mathcal{Q}}(\phi_{\mathcal{P}}, f, f^{-1})$ constructed in the reduction will depend on $\phi_{\mathcal{P}}$, f and f^{-1}.

Properties. The rest of this subsection is devoted to proving properties of maximality-preserving reductions for FFBP s. First, we show that maximality-preserving reductions are transitive, which is the crucial property to argue that one can use multiple reductions in a row. Second, we show that maximality-preserving reductions for frequency-based problems imply maximality-preserving reductions for FFBP s.

The following lemma shows that maximality-preserving reductions for FFBP s are transitive. The main challenge will be the construction of the feasibility function.

Lemma 10. *Let $\mathcal{P}, \mathcal{Q}, \mathcal{R}$ be FFBP s. Assume there exist maximality-preserving reductions from $\mathcal{P}$ to $\mathcal{Q}$ via a function g and $\phi_{\mathcal{Q}}$, and from $\mathcal{Q}$ to $\mathcal{R}$ via a function h and $\phi_{\mathcal{R}}$. Then there exists a maximality-preserving reduction from $\mathcal{P}$ to $\mathcal{R}$.*

Proof. Let $D_{\mathcal{P}}$ and $\phi_{\mathcal{P}}$ be an instance for $\mathcal{P}$. We construct an instance (D^*, ϕ_*) for $\mathcal{R}$: We set $f\colon \mathit{transactions}(\mathcal{P}) \to \mathit{transactions}(\mathcal{R})$ to $f(p) = h(g(p))$ for $p \in \mathit{transactions}(\mathcal{P})$. For a pattern $r \in \mathit{patterns}(\mathcal{R})$, we set $\phi_*(r) = 1$ if and only if the following four conditions are satisfied: (1) $h^{-1}(r)$ and $f^{-1}(r)$ exist; (2) $\phi_{\mathcal{R}}(r) = 1$; (3) $\phi_{\mathcal{Q}}(h^{-1}(r)) = 1$; and (4) $\phi_{\mathcal{P}}(f^{-1}(r)) = 1$.

We check the properties from Definition 2. Property 1 and Property 2 are satisfied since f is the composition g and h. Property 3 holds since $f^{-1} = g^{-1} \circ h^{-1}$ and both g^{-1} and h^{-1} can be computed in polynomial time.

The rest of the proof is devoted to proving Property 4.

Let $p \in \textsc{Max}(D_{\mathcal{P}}, \tau, \phi_{\mathcal{P}})$. Then p is feasible w.r.t. $\phi_{\mathcal{P}}$. By the reduction from $\mathcal{P}$ to $\mathcal{Q}$, $g(p) \in \textsc{Max}(D_{\mathcal{Q}}, \tau, \phi_{\mathcal{Q}})$, where $D_{\mathcal{Q}} = g(D_{\mathcal{P}})$. Note that $g(p)$ is feasible w.r.t. $\phi_{\mathcal{Q}}$. Using the reduction from $\mathcal{Q}$ to $\mathcal{R}$, we obtain $r := h(g(p)) \in \textsc{Max}(D_{\mathcal{R}}, \tau, \phi_{\mathcal{R}})$, where $D_{\mathcal{R}} = h(D_{\mathcal{Q}})$; additionally, r is feasible w.r.t. $\phi_{\mathcal{R}}$. Now observe that $r = f(p)$ and that r is feasible w.r.t. the operation ϕ_* defined above. Note that r is frequent in D^* since for each transaction $t \in D_{\mathcal{P}}$ with $p \sqsubseteq_{\mathcal{P}} t$, $r = f(p) \sqsubseteq_{\mathcal{R}} f(t)$ by Property 2 of f. To prove that $r \in \textsc{Max}(D^*, \tau, \phi_*)$, it remains to show that r is maximal. Suppose not. Then there exists a pattern $r' \in \textsc{Max}(D^*, \tau, \phi_*)$ such that $r \sqsubset_{\mathcal{R}} r'$. Since r' is feasible, let $p' = f^{-1}(r')$. By Property 2 of f, we have that $p \sqsubset_{\mathcal{P}} p'$ and that p' is frequent since $p' \sqsubset_{\mathcal{P}} t$ for $t \in D_{\mathcal{P}}$ if and only if $f(p') = r' \sqsubset_{\mathcal{R}} f(t)$. This contradicts the maximality of p. Hence, we proved that $r \in \textsc{Max}(D^*, \tau, \phi_*)$.

Let $r \in \textsc{Max}(D^*, \tau, \phi_*)$. Since r is feasible w.r.t. ϕ_*, there exists $p = f^{-1}(r) \in \mathit{patterns}(\mathcal{P})$ that is feasible w.r.t. $\phi_{\mathcal{P}}$. By Property 2, p is frequent in $D_{\mathcal{P}}$. It remains to show that p is maximal. We argue by contradiction. Suppose there exists a frequent pattern p' with $p \sqsubset p'$. Then $f(p') \in \textsc{Max}(D_*, \tau, \phi_*)$ by the previous paragraph, and $r \sqsubset f(p')$ by Property 2 of f. This contradicts the maximality of r. Hence, $p \in \textsc{Max}(D_{\mathcal{P}}, \tau, \phi_{\mathcal{P}})$. ☐

The next lemma shows that if for two frequency-based problems $\mathcal{P}$ and $\mathcal{Q}$

there exists a maximality-preserving reduction from $\mathcal{P}$ to $\mathcal{Q}$, then there also exists a reduction between the FFBP-version of these problems.

Lemma 11. *Let $\mathcal{P}$ and $\mathcal{Q}$ be two frequency-based problems, and let $\mathcal{P}'$ and $\mathcal{Q}'$ be the* FFBP*-versions of those problems. Suppose there exists a maximality-preserving reduction from $\mathcal{P}$ to $\mathcal{Q}$ via a mapping g.*

Then there exists a maximality-preserving reduction from $\mathcal{P}'$ to $\mathcal{Q}'$.

Proof. *Construction of f.* We set $f \equiv g$. Given a pattern $q \in \mathit{patterns}(\mathcal{Q})$, we set $\phi_{\mathcal{Q}'}(q) = 1$ iff $f^{-1}(q)$ exists and $\phi_{\mathcal{P}'}(f^{-1}(q)) = 1$.

Maximality-preserving. Note that Properties 1–3 of maximality-preserving reductions for f are satisfied since they are satisfied for g. We prove Property 4 of f.

Let $p \in \textsc{Max}(D_{\mathcal{P}}, \tau, \phi_{\mathcal{P}})$. We show that $f(p) \in \textsc{Max}(D_{\mathcal{Q}}, \tau, \phi_{\mathcal{Q}})$. Observe that $f(p)$ is feasible w.r.t. $\phi_{\mathcal{Q}}$ since $f^{-1}(f(p)) = p$ is feasible w.r.t. $\phi_{\mathcal{P}}$. Note that $f(p)$ is frequent in $D_{\mathcal{Q}}$ by Property 2 of f. We need to argue that $f(p)$ is also maximal. Suppose this is not the case. Then there exists a pattern $q \in \textsc{Max}(D_{\mathcal{Q}}, \tau, \phi_{\mathcal{Q}})$ such that $f(p) \sqsubset q$. Since q is feasible, there exists a feasible pattern $p' = f^{-1}(q) \in \mathit{patterns}(\mathcal{P})$. By Property 2, we have $p \sqsubset p'$. Additionally, the pattern p' is frequent in $D_{\mathcal{P}}$: for each transaction $t \in D_{\mathcal{Q}}$ with $q \sqsubset_{\mathcal{Q}} t$, $p' \sqsubset_{\mathcal{P}} f^{-1}(t)$ (by Property 2 of f and definition of $D_{\mathcal{Q}}$). This contradicts the maximality of p.

Let $q \in \textsc{Max}(D_{\mathcal{Q}}, \tau, \phi_{\mathcal{Q}})$. Since q is feasible, $p = f^{-1}(q)$ exists and is feasible w.r.t. $\phi_{\mathcal{P}}$. We show that $p \in \textsc{Max}(D_{\mathcal{P}}, \tau, \phi_{\mathcal{P}})$. Note that p is frequent in $D_{\mathcal{P}}$ by Property 2 of f. We prove the maximality of p by contradiction. Suppose there exists a pattern $p' \in \textsc{Max}(D_{\mathcal{P}}, \tau, \phi_{\mathcal{P}})$ with $p \sqsubset p'$. Then by the previous paragraph the pattern $f(p')$ is a feasible frequent pattern in $D_{\mathcal{Q}}$ with $q = f(p) \sqsubset f(p')$. This contradicts the maximality of q. □

5.3 Reductions

From graphs to feasible frequent itemsets. We show that any algorithm solving the MaxFFIS-problem can be used to mine maximal frequent subgraphs in general graphs.

Lemma 12. *There exists a maximality-preserving reduction from* MaxFFS(**G**) *to* MaxFFIS.

Proof. Let $D_{\textsc{MaxFFS}(\mathbf{G})}$ be a database consisting of labelled graphs from **G** with labels from $\{1, \ldots, n\}$, let τ be a support threshold, let $\phi_{\textsc{MaxFFS}(\mathbf{G})}$ be a feasibility function.

Construction of f. For MaxFFIS we use the labels $\mathcal{L} = \{1, \ldots, n\}^2$. Let $G = (V, E)$ be a graph from $D_{\textsc{MaxFFS}(\mathbf{G})}$. We construct an itemset $I(G) := f(G)$ by mapping the graph onto the labels of its edges, i.e., we construct an itemset $I(G) = \{(\mathit{label}(u), \mathit{label}(v)) : (u, v) \in E\}$.

Given an itemset $I \in \mathit{patterns}(\textsc{MaxFFIS})$, we set $\phi_{\textsc{MaxFFIS}}(I) = 1$ iff (1) $f^{-1}(I)$ exists and $\phi_{\textsc{MaxFFS}(\mathbf{G})}(f^{-1}(I)) = 1$, and (2) for each pair of tuples

$(a, b), (c, d) \in I$ there exists a sequence $(a, b) = (e_1, e'_1), \ldots, (e_k, e'_k) = (c, d)$ of tuples $(e_i, e'_i) \in I$ with the following property: For each pair of consecutive tuples (e_i, e'_i) and (e_{i+1}, e'_{i+1}), there exists some $\ell \in \{1, \ldots, n\}$ with $\ell \in \{e_i, e'_i\}$ and $\ell \in \{e_{i+1}, e'_{i+1}\}$. Intuitively, condition (2) of ϕ_{MaxFFIS} asserts that the graphs corresponding to the itemset I must be connected.

Maximality-preserving. Note that any feasible frequent itemset in D_{MaxFFIS} corresponds to a frequent *connected* graph in $D_{\text{MaxFFS}(\mathbf{G})}$ due to the choice of ϕ_{MaxFFIS}. Observe that there exists a bijection between connected subgraphs G and feasible itemsets $I(G) \subseteq \mathcal{L}'$. Further observe that for two frequent subgraphs G and H, $G \subseteq H$ if and only if $f(G) \subseteq f(H)$. It follows that a graph G and an itemset I must have the same supports in $D_{\text{MaxFFS}(\mathbf{G})}$ and D_{MaxFIS}, respectively. The maximality then follows from the subset-property we observed. □

Note that the reduction simplifies when $\phi_{\text{MaxFFS}(\mathbf{G})} \equiv 1$, i.e., when we consider the reduction from frequency-based problem MaxFS(**G**) to the FFBP MaxFFIS. Then the mapping f stays the same and ϕ_{MaxFFIS} only needs to check condition (2). We believe that many algorithms for mining itemsets can be augmented with this choice of ϕ_{MaxFFIS} function to mine graph patterns as we will discuss further in Section 6.

Observe that while condition (2) looks rather technical, it can be easily implemented using a graph traversal. Additionally, when computing the union of two feasible patterns, an algorithm only needs to check if both patterns share any label.

Note also that the reduction above works as well for directed graphs (we just need to distinguish between edge labels $(\mathit{label}(u), \mathit{label}(v))$ and $(\mathit{label}(v), \mathit{label}(u))$). This immediately gives us the following lemma.

Lemma 13. *There exists a maximality-preserving reduction from* MaxFFS(**DirG**) *to* MaxFFIS.

From sequences to feasible DAGs. To finish the hierarchy of Figure 1, we need one more reduction, from MaxFSQS to MaxFFS(**DAG**).

Lemma 14. *There exists a maximality-preserving reduction from* MaxFSQS *to* MaxFFS(**DAG**).

Proof. Let D_{MaxFSQS} be a database of sequences over labels from $\mathcal{L}$, let τ be a support threshold, and let ϕ_{MaxFSQS} be a feasibility function. Recall that a sequence contains each label at most once.

Construction of f. For MaxFFS(**DAG**) we use the same labels $\mathcal{L}$. Consider a sequence $S \in \mathcal{L}^r$ of length r such that $S_i \neq S_j$ for all $i \neq j$. This sequence is mapped to the graph $G(S)$ with vertices $V(S) = \{S_1, \ldots, S_r\}$, where each vertex S_i is labelled by $\mathit{label}(S_i)$. The graph contains the edges

$$E(S) = \{(S_i, S_j) : i \in \{1, \ldots, k-1\}, j > i\}.$$

Given a DAG $p \in \mathit{patterns}(\text{MaxFFS}(\mathbf{DAG}))$, we set $\phi_{\text{MaxFFS}(\mathbf{DAG})}(p) = 1$ iff $f^{-1}(p)$ exists and $\phi_{\text{MaxFSQS}}(f^{-1}(p)) = 1$.

Maximality-preserving. Note that Properties 1–3 of maximality-preserving reductions for f are trivially satisfied. We prove Property 4.

Let S be sequence from $\textsc{Max}(D_{\textsc{MaxFSQS}}, \tau, \phi_{\textsc{MaxFSQS}})$ of length r. We show that $G := f(S) \in \textsc{Max}(D_{\textsc{MaxFFS}(\mathbf{DAG})}, \tau, \phi_{\textsc{MaxFFS}(\mathbf{DAG})})$. By construction of $D_{\textsc{MaxFFS}(\mathbf{DAG})}$ and due to Property 2, G is frequent in $D_{\textsc{MaxFFS}(\mathbf{DAG})}$. We need to argue that G is also maximal; we do this by contradiction. Suppose there exists a feasible graph H such that $G \subset H$. Observe that adding any edge to G would introduce a cycle. Hence, H must contain more vertices than G. Since H is also feasible, it corresponds to a sequence $S' = f^{-1}(H)$ of length at least $r+1$. By Property 2, S' is frequent and $S \sqsubset S'$. This contradicts the maximality of S.

Consider any maximal feasible frequent DAG $G = (V, E)$ in $D_{\textsc{MaxFFS}(\mathbf{DAG})}$. Since G is feasible, let $S = f^{-1}(G)$. Then the sequence $S = \langle v_1, \ldots, v_r \rangle$ must be frequent in $D_{\textsc{MaxFSQS}}$ by the choice of f and the construction of $D_{\textsc{MaxFFS}(\mathbf{DAG})}$. Additionally, S must be maximal. Assume it is not. Then there exists a maximal sequence T with $S \sqsubset T$. By the argument of the previous paragraph, the graph $H = f(T)$ is maximal and frequent. But then we also have $G = f(S) \sqsubset f(T) = H$, which contradicts the maximality of G. □

6 Algorithms and Experiments

In this section, we discuss the practical consequences of our reductions and show that the reductions can be used to develop efficient real-world algorithms.

6.1 Reductions as Algorithms

In addition to providing us the theoretical understanding of the relationships between the problems, the reductions also provide us a direct way to solve a maximal frequent pattern mining problem in one domain by using a solver from the other domain. As an example, consider the reduction from the *frequency-based* problem $\textsc{MaxFS}(\mathbf{G})$ to the FFBP $\textsc{MaxFFIS}$ (Lemma 12) and let D_G be the graph database for an instance of $\textsc{MaxFS}(\mathbf{G})$ and D_T be the transaction database built by the reduction.

The mapping of patterns f is straight forward, as we only need to generate a transaction for each graph, and an item for each unique edge label. The crux of the reduction lies in the feasibility function ϕ: it has to ensure that the returned frequent itemsets correspond to *connected* frequent subgraphs in the original problem. As the feasible frequent itemsets are a strict subset of all of the frequent subsets,[2] we could simply prune out the results at the very end. A naïve algoritm for solving $\textsc{MaxFS}(\mathbf{G})$ could then work as follows: (1) build D_T following Lemma 12; (2) compute all frequent itemsets from D_T; (3) prune out the non-feasible frequent itemsets; (4) prune out the non-maximal feasible frequent itemsets.

[2]Note, however, that the feasible *maximal* itemsets are not necessarily a subset of all maximal itemsets.

More efficient implementations are possible, however. In particular, we can add the feasibility constraint in the mining process, thus reducing the number of candidates to consider in each iteration. The connectedness constraint is not monotone, though: it is possible that two itemsets A and B do not correspond to connected subgraphs, while their union does (e.g., $A = \{(a, b), (c, d)\}$ and $B = \{(b, c), (d, e)\}$). On the other hand, if C is a feasible (connected) frequent itemset in D_T, then it can be split into subsets of any size that are frequent and feasible. This means that we can prune all infeasible itemsets at the same time when we prune away all infrequent itemsets. In other words, we can in fact work with *less* candidates (or at least with no more) than if we would be doing standard frequent itemset mining.

The final question in our example is how to implement the feasibility check efficiently. Let us abuse the notation slightly and denote by $\mathit{label}(A)$ the set of unique (vertex) labels in an itemset A, that is $\mathit{label}(A) = \{l :$ edge $(l, \cdot)$ or $(\cdot, l)$ is an item in $A\}$. Then $A \cup B$ is a connected (i.e., feasible) itemset if and only if $\mathit{label}(A) \cap \mathit{label}(B) \neq \emptyset$ and both A and B are connected (i.e., feasible). Hence, if we store the sets $\mathit{label}(A)$ together with the candidate itemsets, we only need to test the disjointness of these two sets to test the feasibility of $A \cup B$.

The above example should make clear that the reductions we present in this paper can yield practical algorithms, and it is not too hard to see that similarly efficient algorithm can be designed following the reduction of Lemma 14. However, note that in this reduction it would not be a good idea to add single edges during the candidate generation; an efficient implementation would ensure that whole nodes with edges to all over vertices are added. This ensures that the preimage of the reduction exists at all times and that fewer infeasible candidates are generated.

To further validate our approach, we present some experimental evaluation of the above algorithm in the next subsection. Before that, let us however discuss a bit on the general approaches for using the maximality-preserving reductions.

The first observation is that the type of the feasibility constraint obviously has a big impact on the efficiency of the final algorithm. The study of constrained frequent pattern mining is well established (see, e.g., [15] or Section 3), and that research gives characterizations of constraints that can be implemented efficiently in standard algorithms. Similarly, the constraint-programming algorithms for data analysis can often be easily adapted for the feasibility constraints used in frequency-based reductions.

The second observation concerns the number of (non-maximal) frequent itemsets. Our reductions are only guaranteed to preserve the maximality, and can, in principle, yield an exponentially larger number of non-maximal frequent (and feasible) itemsets. This would, naturally, make it practically infeasible to use the reductions together with standard frequent pattern mining algorithms. There are a few possible solutions to this. First, many reductions do not grow the number of feasible frequent patterns. This is, for example, the case with the reductions in Lemmas 6, 7, and 12. Second, a clever implementation of a reduction would only generate candidates which may be generated by the mapping from the reduction. This can dramatically decrease the number of possible candidates. In fact, if the implementation manages not to generate any candidates which have no preimage under the mapping from the reduction, then the number of possible candidates will not increase at all. We believe that this is possible for all reductions we present in this paper. Third, the maximal frequent patterns can also be found by first finding all the maximal frequent and minimal infrequent patterns [13]. Unfortunately for this approach, we do not yet know the behaviour of minimal infrequent patterns under our reductions. We leave further studies in this for future work.

6.2 Experimental Evaluation

For the experimental evaluation, we implemented the reduction from MaxFS($\mathbf{G}$) to MaxFIS (Lemma 12) in a custom version of the Apriori algorithm [2]. The constraint on the feasible patterns was straight forward to implement, as discussed above.[3]

We tested our approach on a discussion forum data from the StackExchange forums.[4] The data contains 161 different question-answering forums (we excluded the meta-forums). We concentrated on the most recent year's activity, and constructed one graph for each forum where the users are the vertices and there is an edge between two users if one has answered or commented to the other's question or answer. The vertices are labelled uniquely using the global user-id. The data has 1 627 946 different users, and in total 8 264 675 uniquely-labelled edges. Hence, the dataset does not pose a significant problem for frequent itemset mining algorithms.

We wanted to study the effects the constraint has for the number of candidates. Recall that the constraint is used to enforce that we find only connected subgraphs. In Figure 2, we show the number of frequent itemsets and feasible frequent itemsets of different sizes with minimum frequency 3.

As can be seen from Figure 2, the total number of frequent itemsets is approximately ten times the number of feasible candidates, indicating that the feasibility constraint allows us to prune significant amounts of candidates (there are no feasible candidates of size 17 or 18). In total, the data has 265 111 frequent itemsets, of which 29 752 were feasible and 549 were maximal feasible itemsets.

The number of maximal frequent itemsets and maximal feasible frequent itemsets with respect to different minimum thresholds is presented in Figure 3. We can see that their numbers are mostly aligned, with the number of maximal itemsets dropping almost exponentially as the minimum threshold increases. No pattern has support higher than 9.

[3] The code and sample data are available from `https://people.mpi-inf.mpg.de/~pmiettin/frequency-based-reductions/`.

[4] `https://archive.org/details/stackexchange`

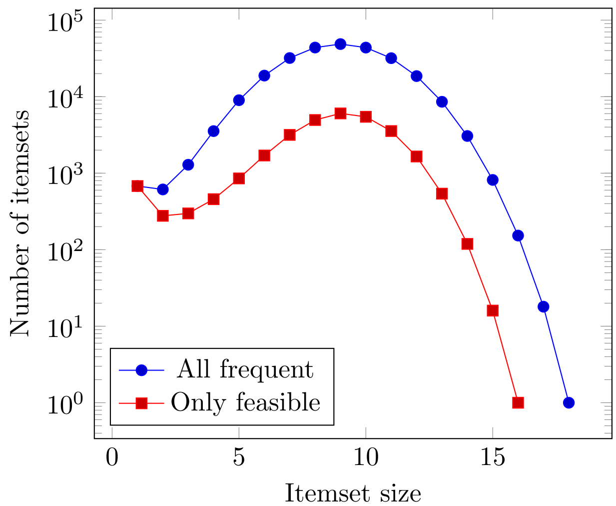

Figure 2: The number of frequent itemsets and feasible frequent itemsets when solving the MaxFS($\mathbf{G}$) problem using MaxFIS algorithms. The y-axis is in logarithmic scale.

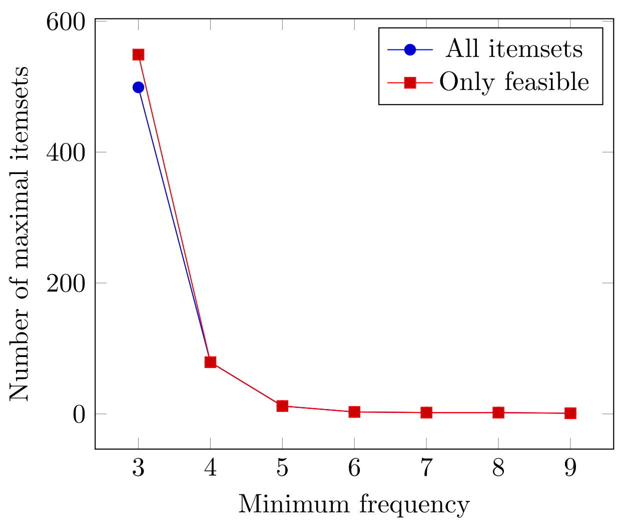

Figure 3: The number of maximal feasible frequent itemsets with different minimum support thresholds.

7 Conclusion

We showed that when considering a generalized version of frequency-based problems, FFBP, the computational hardness of many frequency-based problems collapses. Hence, our reductions provide a unifying framework for the existing computational hardness results of fundamental data mining problems. Additionally, our reductions give a formal explanation why algorithms similar to the Apriori algorithm can be used for such a wide range of problems by only slightly adjusting the candidate generation.

In the future it will be interesting to study the computational complexity of frequency-based problems in which labels can appear multiple times. A daunting question is whether the following two problems exhibit the same hardness: Mining subsequences without the restriction that each label appears only once, and mining graphs with possibly multiple vertices of the same label.

The reductions we provide hint that many practical algorithms for frequency-based problems can be augmented to solve more complicated problems. We provided such an example in Section 6. It will be interesting to see if our insights can lead to more efficient algorithms for the problems we considered or to algorithms which can solve a wider range of problems.

References

[1] Charu C. Aggarwal. *Data Mining - The Textbook*. Springer, 2015.

[2] Rakesh Agrawal, Heikki Mannila, Srikant Ramakrishnan, Hannu Toivonen, and A Inkeri Verkamo. Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, pages 307–328. MIT Press, 1996.

[3] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In *ICDE*, pages 3–14, 1995.

[4] Francesco Bonchi, Fosca Giannotti, Claudio Lucchese, Salvatore Orlando, Raffaele Perego, and Roberto Trasarti. A constraint-based querying system for exploratory pattern discovery. *Inf. Syst.*, 34(1):3–27, 2009.

[5] Francesco Bonchi, Fosca Giannotti, Alessio Mazzanti, and Dino Pedreschi. Exante: Anticipated data reduction in constrained pattern mining. In *PKDD*, pages 59–70, 2003.

[6] Francesco Bonchi and Claudio Lucchese. On closed constrained frequent pattern mining. In *ICDM*, pages 35–42, 2004.

[7] Endre Boros, Vladimir Gurvich, Leonid Khachiyan, and Kazuhisa Makino. On maximal frequent and minimal infrequent sets in binary matrices. *Ann. Math. Artif. Intell.*, 39(3):211–221, 2003.

[8] Douglas Burdick, Manuel Calimlim, Jason Flannick, Johannes Gehrke, and Tomi Yiu. MAFIA: A maximal frequent itemset algorithm. *IEEE Trans. Knowl. Data Eng.*, 17(11):1490–1504, 2005.

[9] Minos N. Garofalakis, Rajeev Rastogi, and Kyuseok Shim. SPIRIT: sequential pattern mining with regular expression constraints. In *VLDB*, pages 223–234, 1999.

[10] Gösta Grahne, Laks V. S. Lakshmanan, and Xiaohong Wang. Efficient mining of constrained correlated sets. In *ICDE*, pages 512–521, 2000.

[11] Gianluigi Greco, Antonella Guzzo, and Luigi Pontieri. Mining taxonomies of process models. *Data Knowl. Eng.*, 67(1):74–102, 2008.

[12] Dimitrios Gunopulos, Roni Khardon, Heikki Mannila, Sanjeev Saluja, Hannu Toivonen, and Ram Sewak Sharm. Discovering all most specific sentences. *ACM Trans. Database Syst.*, 28(2):140–174, 2003.

[13] Dimitrios Gunopulos, Heikki Mannila, Roni Khardon, and Hannu Toivonen. Data mining, hypergraph transversals, and machine learning. In *PODS '97*, pages 209–216, 1997.

[14] Tias Guns, Siegfried Nijssen, and Luc De Raedt. k-pattern set mining under constraints. *IEEE Trans. Knowl. Data Eng.*, 25(2):402–418, 2013.

[15] Jiawei Han, Hong Cheng, Dong Xin, and Xifeng Yan. Frequent pattern mining: current status and future directions. *Data Min. Knowl. Discov.*, 15(1):55–86, 2007.

[16] Jiawei Han, Jian Pei, Yiwen Yin, and Runying Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Min. Knowl. Discov.*, 8(1):53–87, 2004.

[17] Harry B. Hunt, III, Madhav V. Marathe, Venkatesh Radhakrishnan, and Richard Edwin Stearns. The complexity of planar counting problems. *SIAM J. Comput.*, 27(4):1142–1167, 1998.

[18] David S Johnson, Mihalis Yannakakis, and Christos H Papadimitriou. On generating all maximal independent sets. *Inf. Proc. Lett.*, 27(3):119–123, 1988.

[19] Roberto J. Bayardo Jr. Efficiently mining long patterns from databases. In *SIGMOD*, pages 85–93, 1998.

[20] Benny Kimelfeld and Phokion G. Kolaitis. The complexity of mining maximal frequent subgraphs. In *PODS*, pages 13–24, 2013.

[21] Benny Kimelfeld and Phokion G. Kolaitis. The complexity of mining maximal frequent subgraphs. *ACM Trans. Database Syst.*, 39(4):32:1–32:33, 2014.

[22] Michihiro Kuramochi and George Karypis. Frequent subgraph discovery. In *ICDM*, pages 313–320, 2001.

[23] Stefan Neumann and Pauli Miettinen. Reductions for frequency-based data mining problems. In *ICDM '17*, 2017. To appear.

[24] Raymond T. Ng, Laks V. S. Lakshmanan, Jiawei Han, and Alex Pang. Exploratory mining and pruning optimizations of constrained association rules. In *SIGMOD*, pages 13–24, 1998.

[25] Jian Pei, Jiawei Han, and Wei Wang. Mining sequential patterns with constraints in large databases. In *CIKM*, pages 18–25, 2002.

[26] J. Scott Provan and Michael O. Ball. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM J. Comput.*, 12(4):777–788, 1983.

[27] Luc De Raedt, Tias Guns, and Siegfried Nijssen. Constraint programming for itemset mining. In *KDD*, pages 204–212, 2008.

[28] Seinosuke Toda and Mitsunori Ogiwara. Counting classes are at least as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 21(2):316–328, 1992.

[29] Salil P. Vadhan. The complexity of counting in sparse, regular, and planar graphs. *SIAM J. Comput.*, 31(2):398–427, 2001.

[30] Leslie G. Valiant. The complexity of computing the permanent. *Theor. Comput. Sci.*, 8:189–201, 1979.

[31] Yongqiao Xiao, Jenq-Foung Yao, Zhigang Li, and Margaret H. Dunham. Efficient data mining for maximal frequent subtrees. In *ICDM*, pages 379–386, 2003.

[32] Guizhen Yang. The complexity of mining maximal frequent itemsets and maximal frequent patterns. In *KDD*, pages 344–353, 2004.

[33] Mohammed Javeed Zaki. Efficiently mining frequent trees in a forest: Algorithms and applications. *IEEE Trans. Knowl. Data Eng.*, 17(8):1021–1035, 2005.