The Complexity of Mining Maximal Frequent Subgraphs

Benny Kimelfeld
IBM Research – Almaden
kimelfeld@us.ibm.com

Phokion G. Kolaitis
UC Santa Cruz &
IBM Research – Almaden
kolaitis@cs.ucsc.edu

ABSTRACT

A *frequent subgraph* of a given collection of graphs is a graph that is isomorphic to a subgraph of at least as many graphs in the collection as a given threshold. Frequent subgraphs generalize frequent itemsets and arise in various contexts, from bioinformatics to the Web. Since the space of frequent subgraphs is typically extremely large, research in graph mining has focused on special types of frequent subgraphs that can be orders of magnitude smaller in number, yet encapsulate the space of all frequent subgraphs. *Maximal* frequent subgraphs (i.e., the ones not properly contained in any frequent subgraph) constitute the most useful such type.

In this paper, we embark on a comprehensive investigation of the computational complexity of mining maximal frequent subgraphs. Our study is carried out by considering the effect of three different parameters: possible restrictions on the class of graphs; a fixed bound on the threshold; and a fixed bound on the number of desired answers. We focus on specific classes of connected graphs: general graphs, planar graphs, graphs of bounded degree, and graphs of bounded tree-width (trees being a special case). Moreover, each class has two variants: the one in which the nodes are unlabeled, and the one in which they are uniquely labeled. We delineate the complexity of the enumeration problem for each of these variants by determining when it is solvable in (total or incremental) polynomial time and when it is NP-hard. Specifically, for the labeled classes, we show that bounding the threshold yields tractability but, in most cases, bounding the number of answers does not, unless P=NP; an exception is the case of labeled trees, where bounding either of these two parameters yields tractability. The state of affairs turns out to be quite different for the unlabeled classes. The main (and most challenging to prove) result concerns unlabeled trees: we show NP-hardness, even if the input consists of two trees, and both the threshold and the number of desired answers are equal to just two. In other words, we establish that the following problem is NP-complete: given two unlabeled trees, do they have more than one maximal subtree in common?

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data mining*; G.2.2 [**Discrete Mathematics**]: Graph Theory—*Graph algorithms*

General Terms

Algorithms, Theory

Keywords

Graph mining, maximal frequent subgraphs, enumeration complexity

1. INTRODUCTION

The discovery and generation of frequent patterns has occupied a central place in the area of data mining. Much of the earlier work on this topic focused on frequent itemsets and frequent sequences. In the past decade, however, the research has expanded to include the study of more complex frequent patterns, such as trees and graphs. The need to consider more complex patterns arises in applications that span a wide spectrum; examples include mining molecular data and classification of chemical compounds in bioinformatics [4, 25], behavior and link analysis in social networks [24, 28], and workflow analysis [8, 9]. In particular, graphs form a rich data structure that can capture complicated relationships between data encountered in such applications.

Given a finite sequence $\mathbf{g} = \langle g_1, \dots, g_n \rangle$ of graphs and a positive integer τ as threshold, a *frequent subgraph* of $\mathbf{g}$ is a graph that is isomorphic to a subgraph of at least τ graphs in $\mathbf{g}$. Ideally, one would like to be able to generate all frequent subgraphs; several different algorithms for this task have been proposed, including those reported in [10, 14, 15, 20, 21, 34]. However, since the space of frequent subgraphs is typically extremely large, research in graph mining has considered special types of frequent subgraphs that provide a more compact representation of this space. Maximal frequent subgraphs are arguably the most useful such type, where a *maximal frequent subgraph* is a frequent subgraph that is not properly contained in any frequent subgraph. Maximal frequent subgraphs can be orders of magnitude smaller in number than frequent subgraphs, yet they encapsulate the entire space of frequent subgraphs, as the frequent subgraphs are precisely the subgraphs of the maximal ones. By now, several different algorithms for generating all maximal frequent subgraphs have been proposed and evaluated experimentally, including SPIN [13] and MARGIN [29]. However, with the exception of some earlier work on maximal frequent itemsets [3, 11] and on the counting complexity of maximal frequent subgraphs [36], not much is known about the computational complexity of the fundamental enumeration and decision problems concerning maximal frequent subgraphs.

In this paper, we use the lens of computational complexity to systematically investigate the problem of mining maximal frequent subgraphs. More formally, we investigate the *enumeration problem* for maximal frequent subgraphs: given a sequence $\mathbf{g} = \langle g_1, \dots, g_n \rangle$ of graphs and a threshold τ, produce all maximal frequent subgraphs of $\mathbf{g}$, up to isomorphism and without repetitions. Our study is carried out by considering the effect of three different parameters that we now discuss. First and foremost, we focus on specific classes of connected graphs of algorithmic and combinatorial significance, namely, arbitrary connected graphs, trees, planar graphs, graphs of bounded degree, and graphs of bounded treewidth. These classes are listed in Table 1. Each such a class $\mathcal{P}$ (e.g., the class $\mathbf{T}$ of all trees) consists of *unlabeled* graphs, but also has a *labeled* variant, denoted by $\mathcal{P}_{\mathsf{L}}$ (e.g., the class $\mathbf{T}_{\mathsf{L}}$ of all labeled trees), where the nodes of each graph are uniquely labeled. Note that, for each of these classes, the input to the aforementioned enumeration problem consists of a sequence $\mathbf{g}$ of graphs and a positive integer τ as threshold. We will also consider the restriction of this problem obtained by imposing a fixed bound on the threshold τ; this is the second parameter. The third parameter is a fixed bound k on the number of desired maximal frequent subgraphs; in other words, instead of enumerating all maximal frequent subgraphs, the goal is relaxed to producing just k maximal frequent subgraphs.

Even though the number of maximal frequent subgraphs (up to isomorphism) is often much smaller than the number of frequent subgraphs, it can still be exponential in the size of the given sequence of graphs. Johnson at al. [16] introduced three different yardsticks of goodness for algorithms that solve enumeration problems where the number of answers is, in the worst case, exponential in the size of the input. The weakest one is *polynomial total time*: the running time of the algorithm is polynomial in the combined size of the input and the output. The strongest and most desirable one is *polynomial delay*: the time between every two consecutive answers is polynomial in the size of the input. In between lies the notion of *incremental polynomial time*: the time between every two consecutive answers is polynomial in the combined size of the input and output up to that point. We will use these yardsticks to gauge the complexity of mining maximal frequent subgraphs.

What tools do we have to establish lower bounds for the complexity of enumeration problems? Concretely, how can one show that, under standard complexity assumptions, a particular enumeration problem cannot be solved in polynomial total time? A simple technique is to show that the underlying *non-emptiness* decision problem is intractable. For example, assuming P $\neq$ NP, no algorithm can enumerate all satisfying assignments of a Boolean formula in polynomial total time, since, otherwise, such an algorithm could be used to decide whether a satisfying assignment exists. However, this technique is limited, as there are intractable enumeration problems whose non-emptiness problem (and even the problem of producing a single answer) is solvable in polynomial time. For this reason, a more powerful technique has been used, namely, lower bounds on the complexity of enumeration problems have been established by showing that the associated *extendibility* problem is intractable [3, 17, 19]. The extendibility problem is the decision problem in which the input consists of an instance of the enumeration problem at hand together with a set of answers, and the goal is to decide whether there is an answer that is not among the given ones. It is easy to see that if an enumeration problem is solvable in polynomial total time, then its associated extendibility problem can be solved in polynomial time.

In this paper, we will study the extendibility problem as a vehicle for establishing lower bounds on the complexity of mining maximal frequent subgraphs. Actually, in Section 2, we will introduce the property of *enumeration self-reducibility* and show that, in the presence of enumeration self-reducibility, enumeration in total polynomial time is equivalent to the polynomial-time solvability of the extendibility problem. Thus, in the presence of enumeration self-reducibility, the analysis of the complexity of the enumeration problem at hand amounts to the analysis of a standard decision problem. This will be of great interest to us, because, in Section 3, we will show that all labeled variants $\mathcal{P}_{\mathsf{L}}$ of the classes $\mathcal{P}$ in Table 1, as well as the class $\mathbf{T}$ of all (unlabeled) trees and the class $\mathbf{BDG}^2$ of all (unlabeled) graphs of degree at most 2, are enumeration self-reducible.

We can now summarize our results about mining maximal frequent subgraphs for the classes in Table 1. We first consider the labeled classes. Boros et al. [3] showed that the extendibility problem for maximal frequent itemsets is NP-complete. This implies that the extendibility problem is NP-hard for all labeled classes, except for the class $\mathbf{BDG}^2_{\mathsf{L}}$ of labeled graphs of degree at most 2; for this class, the enumeration problem can be easily solved in polynomial time. In view of this, we consider the restrictions of the enumeration problem for the labeled classes, where either the threshold τ is bounded or the number k of desired answers is bounded. We show that bounding τ yields tractability. In contrast, if we bound k, then NP-hardness persists in all cases, except for the case of labeled trees for which we give a polynomial-time algorithm for extendibility. As a special case, the latter algorithm can be used for generating, in polynomial time, any bounded number k of maximal frequent itemsets. Note that this task would be intractable had we desired to enumerate *by decreasing size*, since determining whether there is a frequent set of cardinality at least as big as a given number is NP-complete [11].

The state of affairs turns out to be quite different for the unlabeled classes. To begin with, the subgraph isomorphism problem is NP-hard for all classes of Table 1, except for the class $\mathbf{T}$ of unlabeled trees and the class $\mathbf{BDG}^2$ of unlabeled graphs of degree at most 2. The intractability of subgraph isomorphism easily implies the intractability of the enumeration problem for maximal frequent subgraphs, even if τ and k are fixed small integers. The class $\mathbf{BDG}^2$ can be dealt with in a straightforward manner; in fact, the extendibility problem is in polynomial time, even when τ and k are unbounded. The main and technically most challenging result concerns unlabeled trees. We show that the extendibility problem is NP-complete, even if the input consists of only two trees and both τ and k are equal to just 2. In other words, we establish that the following problem is NP-complete: given two unlabeled trees, do they have more than one maximal subtree in common?

Finally, we consider the associated counting problem: given a sequence $\mathbf{g}$ of graphs and a threshold τ, compute the number of maximal frequent subgraphs. Yang's work [36] establishes #P-completeness for almost all cases. We focus on the effect of bounding τ, and show that the hardness of the counting complexity is aligned with the hardness of the extendibility problem. Our main contribution in this part concerns unlabeled trees and fixed τ: we prove that our reduction for extendibility can be made parsimonious, hence counting maximal frequent subtrees is #P-complete.

2. PRELIMINARIES

2.1 Enumeration Problems

An *enumeration relation* is a (possibly infinite) set $\mathcal{R}$ of pairs (x, y) of strings x and y, such that, for all strings x, the set $\mathcal{R}(x) = \{y \mid (x, y) \in \mathcal{R}\}$ is finite. A string $y \in \mathcal{R}(x)$ is called a *witness for x*. An enumeration relation $\mathcal{R}$ is said to be an NP*-relation* if the following hold:

1. There is a polynomial p such that $|y| \leq p(|x|)$, for all pairs $(x, y) \in \mathcal{R}$;
2. There is a polynomial-time algorithm for deciding membership of a given pair (x, y) in $\mathcal{R}$.

With every enumeration relation $\mathcal{R}$, we associate three algorithmic problems.

- $\mathcal{R}$-$\mathsf{enumerate}$ is the following function problem: given a string x as input, enumerate the set $\mathcal{R}(x)$; that is, produce all the witnesses for x without repetition.
- $\mathcal{R}$-extend is the following function problem: given a string x and a set $Y \subseteq \mathcal{R}(x)$ as input, generate a new witness $y \in \mathcal{R}(x) \setminus Y$ or declare that none exist.
- $\mathcal{R}$-$\mathsf{extendible}$ is the following decision problem: given a string x and a set $Y \subseteq \mathcal{R}(x)$ as input, decide whether $\mathcal{R}(x) \setminus Y \neq \emptyset$.

We also consider variants of these problems that are parameterized by a fixed bound k on the number of witnesses. Formally, let $\mathcal{R}$ be an enumeration relation and let k be a natural number.

- The problem $\mathcal{R}$-$\mathsf{enumerate}\langle k\rangle$ is similar to the problem $\mathcal{R}$-$\mathsf{enumerate}$, except that the goal is to generate a subset (any subset) of $\mathcal{R}(x)$ of size $\min(k, |\mathcal{R}(x)|)$.
- The problems $\mathcal{R}$-$\mathsf{extend}\langle k\rangle$ and $\mathcal{R}$-$\mathsf{extendible}\langle k\rangle$ are similar to the problems $\mathcal{R}$-extend and $\mathcal{R}$-$\mathsf{extendible}$, respectively, except that the condition $|Y| < k$ is imposed on the input.

As a special case, the problem $\mathcal{R}$-$\mathsf{extendible}\langle 1\rangle$ is the *non-emptiness* problem for $\mathcal{R}$: given a string x, decide whether $\mathcal{R}(x) \neq \emptyset$.

2.1.1 Enumeration Complexity

Let $\mathcal{R}$ be an NP-relation. An algorithm that solves the problem $\mathcal{R}$-$\mathsf{enumerate}$ is called an *enumeration algorithm*. The number of witnesses an enumeration algorithm is required to produce (i.e., $|\mathcal{R}(x)|$) can be exponential in the size of the input. In particular, *polynomial running time* in the size of the input x may be a wrong yardstick of efficiency when analyzing the execution cost of such an algorithm, because just writing the output may require exponential time. For this reason, Johnson et al. [16] introduced several different notions of efficiency for enumeration algorithms. The one most commonly used is *polynomial total time*, which means that the running time is polynomial in the combined size of the input and the output (in other words, the running time is polynomial in $|x|+|\mathcal{R}(x)|$). Two stricter notions measure the time it takes to output a new answer, after a subset Y of $\mathcal{R}(x)$ has already been produced: under *polynomial delay*, this time is polynomial in the size $|x|$ of the input; under *incremental polynomial time*, this time is polynomial in $|x| + |Y|$. The following fact is immediate from the definitions; we record it here for later reference.

FACT 2.1. *Let $\mathcal{R}$ be an* NP*-relation.*

1. *$\mathcal{R}$-$\mathsf{enumerate}$ is in incremental polynomial time if and only if $\mathcal{R}$-extend is in polynomial time.*
2. *If k is a natural number, then $\mathcal{R}$-$\mathsf{enumerate}\langle k\rangle$ is in polynomial time if and only if $\mathcal{R}$-$\mathsf{extend}\langle k\rangle$ is in polynomial time.*

It is well known (and easy to show) that a polynomial-total-time algorithm for $\mathcal{R}$-$\mathsf{enumerate}$ can be used to solve the non-emptiness problem for $\mathcal{R}$. Hence, a simple technique to prove that *no* polynomial-total-time algorithm exists (under standard complexity assumptions) is to prove hardness (e.g., NP-hardness) of the non-emptiness problem (e.g., [22, 37]). A more powerful technique is to show hardness of $\mathcal{R}$-extend. This technique has been used to prove that no polynomial total time algorithm exists in cases in which the non-emptiness problem is tractable or even trivial [17, 19].

Figure 1 depicts the implications among tractability yardsticks for the problems we consider. The strongest yardstick, polynomial delay, implies incremental polynomial time, and so on, until the weakest—polynomial-time non-emptiness.

2.1.2 Self Reducibility

Let $\mathcal{R}$ be an enumeration relation. Clearly (and as depicted in Figure 1), if the problem $\mathcal{R}$-extend is in polynomial time, then so is the problem $\mathcal{R}$-$\mathsf{extendible}$. We say that $\mathcal{R}$ is *enumeration self-reducible* if the other direction also holds, or, more precisely, if there is a polynomial-time algorithm for $\mathcal{R}$-extend using a solver for $\mathcal{R}$-$\mathsf{extendible}$ as an oracle. Similarly, if k is a natural number, then $\mathcal{R}$ is k-*enumeration self-reducible* if there is a polynomial-time algorithm for $\mathcal{R}$-$\mathsf{extend}\langle k\rangle$ that uses a solver for $\mathcal{R}$-$\mathsf{extendible}\langle k\rangle$ as an oracle. Using Fact 2.1, it is easy to prove the following proposition.

PROPOSITION 2.2. *Let $\mathcal{R}$ be an* NP*-relation that is enumeration self-reducible. The following statements are equivalent.*

- *$\mathcal{R}$-$\mathsf{extendible}$ is in polynomial time.*
- *$\mathcal{R}$-$\mathsf{enumerate}$ is in incremental polynomial time.*
- *$\mathcal{R}$-$\mathsf{enumerate}$ is in polynomial total time.*

Furthermore, for every $k \in \mathbb{N}$, if $\mathcal{R}$ is k-enumeration self-reducible, then the preceding equivalence holds for the versions of the problems parameterized by k.

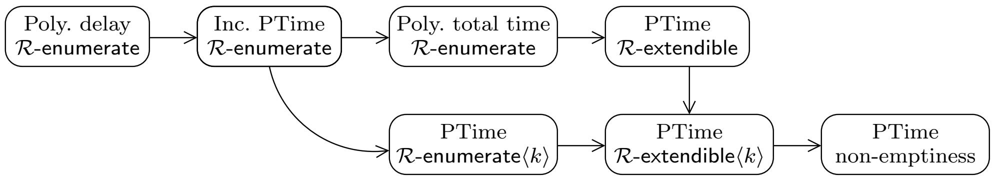
Poly. delay $\mathcal{R}$-enumerate
Inc. PTime $\mathcal{R}$-enumerate
Poly. total time $\mathcal{R}$-enumerate
PTime $\mathcal{R}$-extendible
PTime $\mathcal{R}$-enumerate$\langle k\rangle$
PTime $\mathcal{R}$-extendible$\langle k\rangle$
PTime non-emptiness

Figure 1: Levels of tractability for enumeration problems for an NP-relation $\mathcal{R}$

2.1.3 Counting

For an enumeration relation $\mathcal{R}$, the function $\#\mathcal{R}$ returns the number of witnesses $|\mathcal{R}(x)|$ for a given string x. We consider the following relevant concepts from computational complexity.

- FP is the class of functions that are computable in polynomial time.
- #P [31] is the class of functions $\#\mathcal{R}$, where $\mathcal{R}$ is an NP-relation. A function F is #P-*hard* if there is Turing reduction from every function in #P to F.

Recall that, with access to an oracle for a #P-hard function, one can efficiently solve every problem in the polynomial hierarchy [30]. Also, observe that if $\mathcal{R}$ is an NP-relation, then there is a straightforward polynomial-time reduction from $\mathcal{R}$-extendible to $\#\mathcal{R}$. Hence, from Proposition 2.2, we conclude that in the presence of self-reducibility, the ability to count witnesses implies the ability to enumerate them.

PROPOSITION 2.3. *Let $\mathcal{R}$ be an* NP*-relation that is enumeration self reducible. If $\#\mathcal{R} \in$* FP*, then the problem* $\mathcal{R}$-enumerate *is in incremental polynomial time.*

Note that, under the assumption that FP $\neq$ #P, the other direction of Proposition 2.3 is generally false. For example, the maximal independent sets[1] of a graph can be enumerated with polynomial delay [16] (in particular, we have enumeration self reducibility), yet counting the number of such sets is a #P-hard problem [26].

2.2 Graphs and Canonized Classes

In this paper, we focus on undirected graphs to which, from now on, we refer to as, simply, graphs. If g is a graph, then the set of its nodes is denoted by $\mathsf{nodes}(g)$, while the set of its edges is denoted by $\mathsf{edges}(g)$. We assume that, for every graph g, the set $\mathsf{nodes}(g)$ of its nodes is a subset of some fixed countably infinite set, say, the set $\mathbb{N}$ of all natural numbers. We distinguish between two types of graphs: *unlabeled* and *labeled*. An unlabeled graph is just a graph. For the labeled graphs, we assume a countably infinite alphabet of *labels*. In a *labeled* graph g, every node v of g is associated with a *unique* label (i.e., no two nodes have the same label in g), which we denote by $\lambda_g(v)$. When the graph g is clear from the context, we may omit it from $\lambda_g(v)$ and, instead, write just $\lambda(v)$.

Let g_1 and g_2 be graphs, where we assume that either both are unlabeled or both are labeled. We say that g_1 and g_2 are *isomorphic*, denoted $g_1 \equiv g_2$, if there is a bijection $\mu : \mathsf{nodes}(g_1) \to \mathsf{nodes}(g_2)$ that preserves edges, non-edges, and labels (when g_1 and g_2 are labeled). We write $g_1 \sqsubseteq g_2$ to denote that g_1 is isomorphic to a subgraph of g_2. We also write $g_1 \sqsubsetneq g_2$ to denote that $g_1 \sqsubseteq g_2$ and $g_1 \not\equiv g_2$.

A *class* $\mathcal{G}$ of labeled (or unlabeled) graphs is a collection of labeled (or unlabeled) graphs that is *closed under isomorphisms*; that is, if g_1 is in $\mathcal{G}$ and $g_1 \equiv g_2$, then g_2 is also in $\mathcal{G}$. We say that $\mathcal{G}$ is a *labeled* class if it consists of labeled graphs; similarly, we say that $\mathcal{G}$ is an *unlabeled* class if it consists of unlabeled graphs. Let $\mathcal{G}$ be a labeled or an unlabeled class. A *canonization for* $\mathcal{G}$ is a function $\mathcal{C} : \mathcal{G} \to \mathcal{G}$ possessing the following properties:

1. $\mathcal{C}(g) \equiv g$, for every g in $\mathcal{G}$;
2. $\mathcal{C}(g_1) = \mathcal{C}(g_2)$ for every $g_1, g_2 \in \mathcal{G}$ with $g_1 \equiv g_2$.

A *canonized class of graphs* is a pair $(\mathcal{G}, \mathcal{C})$ where $\mathcal{G}$ is a labeled class or an unlabeled class, and $\mathcal{C}$ is a canonization for $\mathcal{G}$. For a canonized class $\mathcal{P} = (\mathcal{G}, \mathcal{C})$ of graphs, we write $dom(\mathcal{P})$ to denote $\mathcal{G}$, and we write $\mathcal{C}_{\mathcal{P}}$ to denote $\mathcal{C}$. We say that $\mathcal{P}$ is *labeled* if $dom(\mathcal{P})$ is labeled, and *unlabeled* if $dom(\mathcal{P})$ is unlabeled. To simplify the presentation, we often abuse the notation and identify $dom(\mathcal{P})$ with $\mathcal{P}$; thus, we may write $g \in \mathcal{P}$, instead of $g \in dom(\mathcal{P})$.

In this paper, we study specific canonized classes of graphs. Table 1 lists five such unlabeled classes $\mathcal{P}$ by specifying $dom(\mathcal{P})$ in each case. Our work here is restricted to *connected* graphs; for simplicity of presentation, connectedness is implicit in the notation of the classes of Table 1. Although Table 1 does not mention the canonization of each class $\mathcal{P}$, we do assume that each class $\mathcal{P}$ comes with a fixed canonization $\mathcal{C}_{\mathcal{P}}$. Moreover, building on known results [2, 12, 32], we assume that if $\mathcal{P}$ is one of $\mathbf{T}$, $\mathbf{PLN}$, $\mathbf{BDG}^b$ and $\mathbf{BTW}^b$, then $\mathcal{C}_{\mathcal{P}}$ is computable in polynomial time. It should be pointed out that it is not known whether a polynomial-time computable canonization exists for the class $\mathbf{G}$ of all unlabeled connected graphs. Observe also that the existence of a polynomial-time computable canonization $\mathcal{C}_{\mathcal{P}}$ implies that $\mathcal{P}$ has polynomial-time graph isomorphism testing; thus, for a class of graphs, canonization is not easier than graph isomorphism. Note that some containment relationships hold between the domains of the classes of Table 1; for example, $\mathbf{T} \subseteq \mathbf{PLN}$ and $\mathbf{T} = \mathbf{BTW}^1$. Note also that $\mathbf{BTW}^b \subsetneq \mathbf{BTW}^{b+1}$.

For each unlabeled canonized class $\mathcal{P}$ of graphs in Table 1, we write $\mathcal{P}_{\mathsf{L}}$ to denote the corresponding labeled canonized class. As an example, $\mathbf{BTW}^3_{\mathsf{L}}$ consists of the labeled graphs of treewidth 3. For each $\mathcal{P}$ in the table, we assume that the canonization $C_{\mathcal{P}_{\mathsf{L}}}$ is computable in polynomial time. We can make this assumption because of the straightforward observation that there is a polynomial-time computable canonization for the class of all labeled graphs.

We now define some properties of canonized classes $\mathcal{P}$ of graphs. We say that $\mathcal{P}$ is *monotone* if $dom(P)$ is closed un-

[1]A *maximal independent set* is an independent set that is not strictly contained in any other independent set.

Table 1: Unlabeled canonized classes of graphs; each class $\mathcal{P}$ has a corresponding labeled class, which is denoted by $\mathcal{P}_{\mathsf{L}}$; except for $\mathcal{P} = \mathbf{G}$, all canonizations are assumed to be computable in polynomial time.

$\mathcal{P}$	$dom(\mathcal{P})$
$\mathbf{G}$	All connected graphs
$\mathbf{T}$	All trees (acyclic connected graphs)
$\mathbf{PLN}$	All connected planar graphs
$\mathbf{BDG}^b$	All connected graphs with degree $\leq b$
$\mathbf{BTW}^b$	All connected graphs with treewidth $\leq b$

der taking subgraphs.[2] We say that $\mathcal{P}$ is *connected monotone* if $dom(P)$ is closed under taking connected subgraphs. As an example, each of the classes of Table 1 is connected monotone.

Next, we consider properties that capture different aspects of tractable behavior.

We say that a canonized class $\mathcal{P}$ of graphs has a *tractable realization* if the following two tasks can be executed in polynomial time:

- Given a graph g, determine whether $g \in \mathcal{P}$.
- Given $g \in \mathcal{P}$, compute $\mathcal{C}_{\mathcal{P}}(g)$.

We say that a canonized class $\mathcal{P}$ of graphs has *tractable subgraph isomorphism* if the subgraph isomorphism problem for $\mathcal{P}$ (i.e., given $g_1, g_2 \in \mathcal{P}$, decide whether $g_1 \sqsubseteq g_2$) can be tested in polynomial time. As an example, if $\mathcal{P}$ is labeled, then $\mathcal{P}$ has tractable subgraph isomorphism. Moreover, it is well known that $\mathbf{T}$ has tractable subgraph isomorphism, and so does $\mathbf{BDG}^2$. But, unless P = NP, none of $\mathbf{G}$, $\mathbf{PLN}$, $\mathbf{BDG}^b$ with $b > 2$ and $\mathbf{BTW}^b$ with $b > 1$ has tractable subgraph isomorphism [7,23].

The next property of a canonized class $\mathcal{P}$ of graphs has to do with the ability to efficiently traverse the graphs of $\mathcal{P}$; we formalize it as follows. Let $g \in \mathcal{P}$ be a graph. A graph $h \in \mathcal{P}$ is an *immediate $\mathcal{P}$-extension* of g if $g \sqsubsetneq h$ and no graph $h' \in \mathcal{P}$ satisfies $g \sqsubsetneq h' \sqsubsetneq h$. Similarly, $h \in \mathcal{P}$ is an *immediate $\mathcal{P}$-reduction* of g if $h \sqsubsetneq g$ and no graph $h' \in \mathcal{P}$ satisfies $h \sqsubsetneq h' \sqsubsetneq g$. We say that $\mathcal{P}$ has *tractable progression* if there is a polynomial-time algorithm that, given a graph $g \in \mathcal{P}$, returns a set of graphs that contains all the immediate $\mathcal{P}$-extensions and immediate $\mathcal{P}$-reductions of g up to isomorphism; if $\mathcal{P}$ is labeled, then the algorithm is also given a set Σ of labels, and we require the labels of all returned graphs to have labels only from Σ. Note that, besides covering all the immediate $\mathcal{P}$-extensions and immediate $\mathcal{P}$-reductions of g up to isomorphism, the algorithm for realizing tractable progression is allowed to produce any other additional graph.

The following proposition is fairly straightforward.

Proposition 2.4. *Let $\mathcal{P}$ be a canonized class of graphs. If $\mathcal{P}$ is monotone or connected monotone, then $\mathcal{P}$ has tractable progression. Hence, all classes in Table 1 have tractable progression.*

[2]There are multiple, different definitions of monotone classes of graphs in the literature; here we follow the definition of Alon and Shapira [1] (which is different from, e.g., that of Friedgut and Kalai [6]).

3. MAXIMAL FREQUENT SUBGRAPHS

In this section, we formalize the enumeration problem of finding the maximal frequent subgraphs. We also provide some initial insights on the complexity of this problem and its variants.

Definition 3.1. Let $\mathbf{g} = \langle g_1, \dots, g_n \rangle$ be a sequence of graphs and let h be a graph.

- The *support of h in* $\mathbf{g}$, denoted $supp(h|\mathbf{g})$, is the cardinality $|\{i \mid h \sqsubseteq g_i\}|$.
- Let τ be a natural number, which is viewed as a threshold. A *τ-frequent subgraph of* $\mathbf{g}$ is a graph h such that $supp(h|\mathbf{g}) \geq \tau$.

When $\mathbf{g}$ and τ are clear from the context, we will often use the term "frequent subgraph", instead of "τ-frequent subgraph of $\mathbf{g}$". $\square$

Definition 3.2. Let $\mathcal{P}$ be a canonized class of graphs.

- Let τ be a natural number and let $h, g_1, \dots, g_n$ be (not necessarily distinct) graphs in $\mathcal{P}$. We say that h is a *$\mathcal{P}$-maximal* τ-frequent subgraph of $\mathbf{g} = \langle g_1, \dots, g_n \rangle$ if h is a frequent subgraph, and there is no frequent subgraph $h' \in \mathcal{P}$ such that $h \sqsubsetneq h'$. If $\mathbf{g}$, τ and $\mathcal{P}$ are all clear from the context, then we will often say that h is a *maximal frequent subgraph.*
- The enumeration relation $\mathsf{MaxFS}[\mathcal{P}]$ consists of all pairs (x, y) such that
  - $x = \langle \mathbf{g}, \tau \rangle$, for some sequence $\mathbf{g}$ of graphs in $\mathcal{P}$ and some threshold τ;
  - $y = \mathcal{C}_{\mathcal{P}}(h)$, for some $\mathcal{P}$-maximal τ-frequent subgraph h of $\mathbf{g}$.

Note that $\mathsf{MaxFS}[\mathcal{P}]$ is indeed an enumeration relation, since, given $\mathbf{g}$ and τ, there are only finitely many (maximal) frequent subgraphs up to isomorphism. $\square$

In Definition 3.2, the sequence $\mathbf{g} = \langle g_1, \dots, g_n \rangle$ is allowed to include repetitions (and isomorphic copies), since this is what may happen in data mining applications. We note that all complexity results of Sections 4–6 hold even if one requires that $g_i \not\equiv g_j$, whenever $i \neq j$. In particular, while we will use repetitions in one of the NP-hardness proofs (namely, the one of Theorem 4.11), that proof can be easily adjusted to avoid repetitions.

We will also consider the variant of this enumeration relation in which τ is a fixed parameter; specifically, $\mathsf{MaxFS}^{\tau}[\mathcal{P}]$ is defined similarly to $\mathsf{MaxFS}[\mathcal{P}]$, except that x is now just $\mathbf{g}$ (while τ is fixed). For the enumeration problems introduced in the previous section, there are actually two parameters: k (the restriction on the number of witnesses) and τ (the frequency threshold). Later in this paper, we will explore the effect of these parameters on the complexity of the enumeration problems.

3.1 Strong Tractability

Our ultimate goal is to study the complexity of the problems $\mathsf{MaxFS}[\mathcal{P}]$-$\mathsf{enumerate}$, as well as their parameterized versions, for different canonized classes $\mathcal{P}$ of graphs (and with main focus on the classes of Table 1). To avoid immediate intractability hurdles, we must make some assumptions about the complexity entailed in the canonized class

$\mathcal{P}$. To begin with, we need to be able to test the validity of our input, and to produce canonical witnesses; therefore, our first requirement is that $\mathcal{P}$ should have tractable realization. However, this is not enough. Indeed, the following straightforward proposition implies that subgraph isomorphism reduces, via a trivial inspection of the output, to $\mathsf{MaxFS}[\mathcal{P}]$-$\mathsf{enumerate}$, even if both τ and k are fixed small integers.

Proposition 3.3. *Let $\mathcal{P}$ be a canonized class of graphs, and let g_1 and g_2 be two graphs in $\mathcal{P}$ with $|\mathsf{nodes}(g_1)| \leq |\mathsf{nodes}(g_2)|$ and $|\mathsf{edges}(g_1)| \leq |\mathsf{edges}(g_2)|$.*

- *If $\tau = 1$, then $g_1 \sqsubseteq g_2$ if and only if $\mathsf{MaxFS}^{\tau}[\mathcal{P}](g_1, g_2)$ is a singleton.*
- *If $\tau = 2$, then $g_1 \sqsubseteq g_2$ if and only if $\mathsf{MaxFS}^{\tau}[\mathcal{P}](g_1, g_2)$ is a singleton $\{h\}$, such that h has the same number of nodes and edges as g_1.*

Therefore, our second requirement is that $\mathcal{P}$ should have tractable subgraph isomorphism. The next proposition shows that tractable subgraph isomorphism is also a necessary condition, if one desires the underlying enumeration relation to be an NP-relation.

Proposition 3.4. *Let $\mathcal{P}$ be a canonized class of graphs with tractable realization, but where subgraph isomorphism is* NP*-hard. The following problem is both* NP*-hard and* coNP*-hard: given h and $\mathbf{g}$, decide whether $(\mathbf{g}, h) \in \mathsf{MaxFS}^2[\mathcal{P}]$. In particular, neither $\mathsf{MaxFS}^2[\mathcal{P}]$, nor $\mathsf{MaxFS}[\mathcal{P}]$ is an* NP*-relation, unless* P $=$ NP.

Finally, as our goal is to have algorithms for producing maximal frequent subgraphs, we need the property of tractable progression in order to create new graphs from existing ones. The property of *strong tractability* is the conjunction of the three tractability properties we introduced thus far.

Definition 3.5. A canonized class $\mathcal{P}$ of graphs is *strongly tractable* if $\mathcal{P}$ has tractable realization, tractable subgraph isomorphism, and tractable progression. □

The following proposition lists the classes of Table 1 that are strongly tractable. Note that these are exactly the classes in the table that have tractable subgraph isomorphism (assuming P $\neq$ NP). Therefore, in view of Proposition 3.3, these are the only classes in the table for which we can hope to obtain tractability results.

Proposition 3.6. *If $\mathcal{P}$ is one of the labeled variants of the classes in Table 1, or one of $\mathbf{T}$ and $\mathbf{BDG}^2$, then $\mathcal{P}$ is strongly tractable.*

The proof of the preceding Proposition 3.6 is by a straightforward combination of Proposition 2.4 and the aforementioned known results on canonization and subgraph isomorphism.

The following result states that, in the case in which $\mathcal{P}$ is strongly tractable, $\mathsf{MaxFS}[\mathcal{P}]$ and $\mathsf{MaxFS}^{\tau}[\mathcal{P}]$ are NP-relations.

Proposition 3.7. *If $\mathcal{P}$ is a strongly tractable canonized class of graphs and τ is a natural number, then both $\mathsf{MaxFS}[\mathcal{P}]$ and $\mathsf{MaxFS}^{\tau}[\mathcal{P}]$ are* NP*-relations.*

3.2 Self Reducibility

We complete this section with the following theorem, stating that strong tractability of $\mathcal{P}$ also implies enumeration self-reducibility. The proof describes a polynomial-time algorithm for the problem $\mathsf{MaxFS}[\mathcal{P}]$-$\mathsf{extend}$, using an oracle for $\mathcal{R}$-$\mathsf{extendible}$. The algorithm consists of two main steps. In the first step, we construct a frequent subgraph $g \in \mathcal{P}$ that is not necessarily maximal, but is not isomorphic to any subgraph of any of the provided maximal subgraphs. In the second step, we extend g to a maximal frequent subgraph h and return $\mathcal{C}_{\mathcal{P}}(h)$. Both steps apply incremental and greedy constructions, invoking the algorithms for progression and testing subgraph isomorphism; the first step also invokes the oracle for $\mathcal{R}$-$\mathsf{extendible}$. The complete proof will be given in the full version of the paper.

Theorem 3.8. *If $\mathcal{P}$ is a strongly tractable canonized class of graphs, then $\mathsf{MaxFS}[\mathcal{P}]$ and $\mathsf{MaxFS}^{\tau}[\mathcal{P}]$ are both enumeration self-reducible and k-enumeration self-reducible, for all $\tau, k \in \mathbb{N}$.*

Combining Theorem 3.8 with Propositions 3.6 and 3.7, we obtain the following corollary.

Corollary 3.9. *Let $\mathcal{P}$ be one of the labeled variants of the classes in Table 1, or one of the classes $\mathbf{T}$ and $\mathbf{BDG}^2$. For every τ and for every k, both $\mathsf{MaxFS}[\mathcal{P}]$ and $\mathsf{MaxFS}^{\tau}[\mathcal{P}]$ are* NP*-relations, enumeration self-reducible and k-enumeration self-reducible.*

4. LABELED GRAPH CLASSES

In this section, we study the complexity of enumerating the maximal subgraphs for the labeled classes of Table 1. Recall from Proposition 3.6 that each of these classes is strongly tractable. We begin with the case in which both τ and k are unbounded.

4.1 The Unbounded Case

The *frequent-itemset* problem is one of the most extensively studied data mining problems. In the context of the framework described in Section 2.1, the frequent-itemset problem amounts to the enumeration problem associated with the enumeration relation MaxFIS of all pairs (x, y) such that

- $x = \langle \mathbf{s}, \tau \rangle$, for some finite sequence $\mathbf{s}$ of finite sets and some threshold τ.
- $y = s$, for some *maximal frequent subset* s.

Here, a set s is a *frequent subset* of $x = \langle \mathbf{s}, \tau \rangle$ if s is a subset of at least τ elements of $\mathbf{s}$; furthermore, s is a *maximal frequent subset* if it is a frequent subset that is not properly contained in any other frequent subset.

The following result by Boros et al. [3] will be of interest to us.

Theorem 4.1. [3] MaxFIS-$\mathsf{extendible}$ *is* NP*-complete.*

One of the consequences of this result is that essentially all labeled classes of Table 1 have intractable enumeration.

Corollary 4.2. *Let $\mathcal{P}$ be one of the classes $\mathbf{G}_{\mathsf{L}}$, $\mathbf{T}_{\mathsf{L}}$, $\mathbf{PLN}_{\mathsf{L}}$, $\mathbf{BDG}^b_{\mathsf{L}}$ with $b \geq 3$, or $\mathbf{BTW}^w_{\mathsf{L}}$ with $w \geq 1$. Then the problem $\mathsf{MaxFS}[\mathcal{P}]$-$\mathsf{extendible}$ is* NP*-complete.*

The proof of Corollary 4.2 reduces MaxFIS-extendible to MaxFS[$\mathcal{P}$]-extendible, so that each of the associated graphs in the construction belongs to every class among those mentioned in the corollary. The complete proof will be given in the full version of the paper.

The only remaining case is MaxFS[$\mathbf{BDG}^b_{\mathsf{L}}$] with $b \leq 2$. The case of $b = 1$ is trivial, so we consider the case of $b = 2$. Let $\mathbf{g}$ and τ be an input to MaxFS[$\mathbf{BDG}^2_{\mathsf{L}}$]-enumerate. Observe that a graph of degree bounded by 2 is either a path or a simple cycle. Therefore, each of the graphs of $\mathbf{g}$, as well as any (maximal) frequent subgraph, is either a path or simple cycle. Moreover, every graph in $\mathbf{BDG}^2_{\mathsf{L}}$ has only a polynomial number of connected subgraphs. It follows that we can solve MaxFS[$\mathbf{BDG}^2_{\mathsf{L}}$]-enumerate in polynomial time via the following brute force algorithm: enumerate all subgraphs of the graphs in $\mathbf{g}$, filter out those that are not frequent, and then select the maximal among the remaining ones. Consequently, we obtain the following proposition.

PROPOSITION 4.3. MaxFS[$\mathbf{BDG}^b_{\mathsf{L}}$]-enumerate *is solvable in polynomial time if* $b \leq 2$.

4.2 Bounded Threshold

In this section, we show that, once τ is fixed, all labeled variants in Table 1 have tractable enumeration.

LEMMA 4.4. *Let $\mathcal{P}$ be a canonized class of labeled connected graphs that is connected monotonic and has tractable realization. For every integer τ,* MaxFS$^\tau$[$\mathcal{P}$]-enumerate *is solvable in polynomial time.*

The proof of Lemma 4.4 shows how to obtain in polynomial time the set of all maximal frequent graphs, by applying a form of *intersection* over each subsequence of length τ from the input sequence of graphs. The complete proof will be given in the full version of the paper. Combining the lemma with Proposition 3.6, we get following result.

THEOREM 4.5. *Let $\mathcal{P}$ be the labeled variant of one of the classes in Table 1. For every integer τ,* MaxFS$^\tau$[$\mathcal{P}$]-enumerate *is solvable in polynomial time.*

4.3 Bounded Number of Witnesses

In this section, we consider the effect of bounding the number of witnesses of interest. As we shall see, the state of affairs turns out to be more involved than the case of bounding the threshold.

4.3.1 Labeled Trees

We begin by showing that, in the case of trees, bounding the number of witnesses yields tractability.

THEOREM 4.6. MaxFS[$\mathbf{T}_{\mathsf{L}}$]-extendible$\langle k\rangle$ *is solvable in polynomial time for every fixed k.*

In what follows, we prove Theorem 4.6 by describing an algorithm for solving MaxFS[$\mathbf{T}_{\mathsf{L}}$]-extendible$\langle k\rangle$.

Let t be a labeled tree. Recall that $\lambda(v)$ denotes the label of a node v of t. For an edge $e = \{v_1, v_2\}$ of t, we write $\lambda(e)$ to denote the set $\{\lambda(v_1), \lambda(v_2)\}$. We also write $\lambda\mathsf{edges}(t)$ to denote the set $\{\lambda(e) \mid e \in \mathsf{edges}(t)\}$.

To describe our algorithm, we need the following simple observation. If t and t' are labeled trees and t has two or more nodes, then $t \sqsubseteq t'$ is equivalent to $\lambda\mathsf{edges}(t) \subseteq \lambda\mathsf{edges}(t')$. We then obtain the following lemma.

Algorithm LTEXT$\langle k\rangle(\mathbf{g}, \tau, Y)$

```
1: if there is a label that occurs at least $\tau$ times in $\mathbf{g}$ and
   does not occur in $Y$ then
2:   return true
3: let $\mathbf{g} = \langle t_1, \dots, t_n\rangle$
4: for all subsets $F \subseteq \cup_{i=1}^{n} \lambda\mathsf{edges}(t_i)$ with $|F| \leq k$ do
5:   if $F \not\subseteq \lambda\mathsf{edges}(p)$ for all $p \in Y$ then
6:     for all $p' \in \mathbf{g}[F]$ do
7:       if $supp(p'|\mathbf{g}) \geq \tau$ then
8:         return true
9: return false
```

Figure 2: MaxFS[$\mathbf{T}_{\mathsf{L}}$]-extendible$\langle k\rangle$ **solver** LTEXT$\langle k\rangle$

LEMMA 4.7. *If $p', p_1, \dots, p_k$ are labeled trees and p' has two or more nodes, then the following statements are equivalent.*

1. *$p' \not\sqsubseteq p_i$, for every $i \in \{1, \dots, k\}$.*
2. *There is a set $F \subseteq \lambda\mathsf{edges}(p')$ with $|F| \leq k$, such that $F \not\subseteq \lambda\mathsf{edges}(p_i)$, for every $i \in \{1, \dots, k\}$.*

Let t be a labeled tree and F a subset of $\lambda\mathsf{edges}(t)$. Another straightforward observation is that, among the subtrees t' of t, there is exactly one tree f that is minimal among all those t' that contain F:

$$\forall t' \left[F \subseteq \lambda\mathsf{edges}(t') \Rightarrow f \sqsubseteq t'\right]$$

We denote that subtree f by $t[F]$. The notion of $t[F]$ is similar to that of a *reduced subtree* [18]. If $\mathbf{t} = t_1, \dots, t_n$ is a sequence of labeled trees, then $\mathbf{t}[F]$ denotes the sequence that is obtained from $\mathbf{t}$ in two steps:

1. Remove every tree t_i such that $F \not\subseteq \lambda\mathsf{edges}(t_i)$.
2. Replace every remaining tree t_i with $t_i[F]$.

The algorithm LTEXT$\langle k\rangle$, depicted in Figure 2, solves the problem MaxFS[$\mathbf{T}_{\mathsf{L}}$]-extendible$\langle k\rangle$. As expected, the algorithm takes as input a sequence $\mathbf{g}$ of labeled trees, a threshold τ, and a set Y of maximal frequent subtrees. The idea behind this algorithm is as follows. The case where Y can be extended by a single-node frequent subtree is detected by a straightforward traversal over the labels of $\mathbf{g}$. So, suppose that p' is a frequent tree with at least two nodes, and that p' is contained in none of the trees in Y. From Lemma 4.7, we conclude that there is a set $F \subseteq \lambda\mathsf{edges}(p')$, such that $|F| \leq k$ and, for every $p \in Y$, F is not a subset $\lambda\mathsf{edges}(p)$. Then $p'[F]$ is isomorphic to $t[F]$ for at least τ trees t in $\mathbf{g}$. On the other hand, if some set F of label pairs is such that τ trees t in $\mathbf{g}$ share an isomorphic copy of $t[F]$, then $\mathbf{g}$ is indeed extendible. So, LTEXT$\langle k\rangle$ simply considers every possible F (or more precisely, every F that consists of k or fewer edges from $\mathbf{g}$), and every possible $t[F]$ (where t is in $\mathbf{g}$) and tests whether F is a subset of none of the $\lambda\mathsf{edges}(p)$, for $p \in Y$, and that $t[F]$ is frequent. The correctness of the algorithm LTEXT$\langle k\rangle$ is obvious from the above description. Note that, since k is fixed, the number of possible candidates F is polynomial in $\mathbf{g}$.

Lemma 4.8. $\mathrm{LTExt}\langle k\rangle(\mathbf{g},\tau,Y)$ *terminates in polynomial time, and returns* **true** *if and only if there is a frequent subtree that is contained in none of the trees in* Y.

Observe that a frequent subtree that is contained in none of the trees in Y can be extended (in polynomial time) to a maximal frequent subtree that is not isomorphic to any tree in Y. Therefore, Lemma 4.8 implies that $\mathrm{LTExt}\langle k\rangle(\mathbf{g},\tau,Y)$ if and only if $\mathbf{g}$, τ and Y form a positive instance of the problem $\mathsf{MaxFS}[\mathbf{T}_{\mathsf{L}}]\text{-}\mathsf{extendible}\langle k\rangle$. This completes the proof of Theorem 4.6.

4.3.2 Beyond Labeled Trees

We continue the exploration of the effect of bounding the number k of witnesses. The first result is the special case in which $k \leq 2$.

Lemma 4.9. *Let $\mathcal{P}$ be a canonized class of labeled connected graphs. Assume that $\mathcal{P}$ is connected monotonic, and that $\mathcal{P}$ has tractable realization. For $k \leq 2$, the problem $\mathsf{MaxFS}[\mathcal{P}]\text{-}\mathsf{extendible}\langle k\rangle$ is solvable in polynomial time.*

Proof. Recall that in $\mathsf{MaxFS}[\mathcal{P}]\text{-}\mathsf{extendible}\langle k\rangle$, the input consists of a sequence $\mathbf{g}$ of graphs in $\mathcal{P}$, a threshold τ, and a set Y of at most $k-1$ maximal frequent subgraphs. The goal is to determine whether there is a (maximal) frequent subgraph h that is contained in none of the graphs in Y. For $k = 1$ the answer is always "yes" (e.g., take h to be the empty graph), unless $|\mathbf{g}| < \tau$ (in which case the answer is always "no"). So, we consider the case in which $k = 2$ and Y consists of exactly one graph h_Y. In that case, the answer is "yes" if and only if there is a frequent graph g' that consists of either a single node or a single edge, among the nodes/edges of the graphs in $\mathbf{g}$, such that $g' \not\sqsubseteq h_Y$. Here we are using the fact that $\mathcal{P}$ is connected monotonic, implying that g' is also in $\mathcal{P}$. $\Box$

Lemma 4.9 and Proposition 3.6 yield the following result.

Theorem 4.10. *Let $\mathcal{P}$ be the labeled variant of a class in Table 1. For $k \leq 2$, $\mathsf{MaxFS}[\mathcal{P}]\text{-}\mathsf{extendible}\langle k\rangle$ is solvable in polynomial time.*

Finally, we show that the preceding results cover all cases where we obtain tractability by fixing k. More precisely, in what follows we assert that $\mathsf{MaxFS}[\mathcal{P}]\text{-}\mathsf{extendible}\langle k\rangle$ is NP-hard, where $\mathcal{P}$ is any one of the labeled variants of the classes in Table 1, except for the classes $\mathbf{T}_{\mathsf{L}}$ (which is also $\mathbf{BTW}^1_{\mathsf{L}}$) and $\mathbf{BDG}^b_{\mathsf{L}}$ with $b \leq 2$ (which was covered in Proposition 4.3).

Theorem 4.11. *Let $\mathcal{P}$ be one of $\mathbf{G}_{\mathsf{L}}$, $\mathbf{PLN}_{\mathsf{L}}$, $\mathbf{BDG}^b_{\mathsf{L}}$ with $b > 2$, or $\mathbf{BTW}^w_{\mathsf{L}}$ with $w > 1$. Then, for every $k > 2$, the problem $\mathsf{MaxFS}[\mathcal{P}]\text{-}\mathsf{extendible}\langle k\rangle$ is NP-complete; moreover, this problem is $W[1]$-hard with respect to the parameter τ.*

Proof. We first prove NP-completeness. Membership in NP is immediate from the fact $\mathsf{MaxFS}[\mathcal{P}]$ is strongly tractable. So, we prove NP-hardness, and it is enough to do so only for $k = 3$. We will describe a polynomial-time reduction from *Maximum Independent Set*: given a graph f and a number m, determine whether f has an independent set (i.e., a set of nodes inducing an edge-free subgraph) of size m. So, let f and m be a given input to Maximum Independent Set. Without loss of generality, we may assume that f has

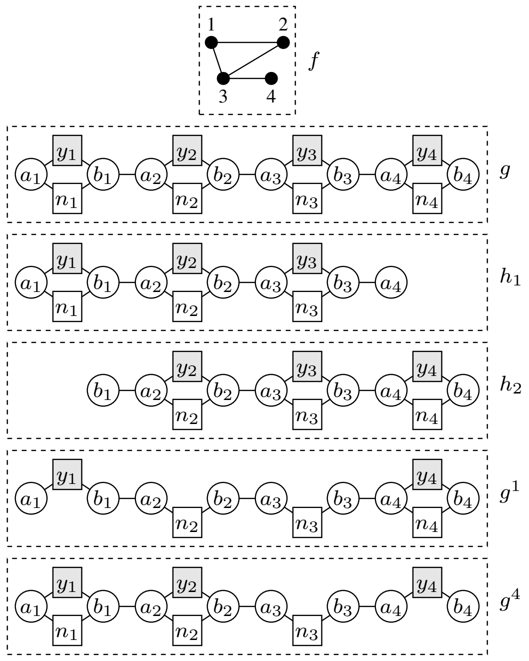

Figure 3: Reducing Maximum Independent Set to $\mathsf{MaxFS}[\mathbf{G}_{\mathsf{L}}]\text{-}\mathsf{extendible}\langle 2\rangle$

no isolated nodes. We construct an input $\mathbf{g}$, τ, and Y to $\mathsf{MaxFS}[\mathbf{G}_{\mathsf{L}}]\text{-}\mathsf{extendible}\langle 3\rangle$, as follows.

Suppose that $\{1,\dots,n\}$ is the node set of f. For each $i = 1,\dots,n$, we define four labels: a_i, b_i, y_i, and n_i (where y and n stand for *yes* and *no*, respectively). For simplicity, when we construct a graph over the labels we introduced, we do not distinguish between a node and its label (recall that here our graphs are uniquely labeled). We define the graph g to be the one over these $4n$ labels and with the edges $\{a_i, y_i\}$, $\{a_i, n_i\}$, $\{y_i, b_i\}$, $\{n_i, b_i\}$ for all $i = 1,\dots,n$, and $\{b_i, a_{i+1}\}$, for all $i = 1,\dots,n-1$.

Our construction is illustrated with a concrete example in Figure 3. The graphs f and g are on the top row in the figure. Note that $n = 4$ in this example.

We denote by h_1 the subgraph of g obtained by removing the nodes y_n, n_n and b_n. We also denote by h_2 the subgraph of g obtained by removing the nodes a_1, y_1 and n_1. In our example, the second top and third top rows in Figure 3 show h_1 and h_2, respectively.

Next, for $i = 1,\dots,n$, let g^i denote the subgraph of g obtained by deleting n_i and also deleting y_j, for every neighbor j of i in f. The bottom two rows of Figure 3 depict the graphs g^1 and g^4 in our example.

We are now ready to define the input $\mathbf{g}$, τ, and Y:

- $\mathbf{g} = g'_1,\dots,g'_m,g''_1,\dots,g''_m,g^1,\dots,g^n$, where each g'_i is h_1 and each g''_i is h_2.
- $\tau = m$ (i.e., the size in the given instance of Maximum Independent Set).
- $Y = \{h_1, h_2\}$.

Observe that the graphs h_1 and h_2 are indeed frequent; moreover, they are maximal since we assumed that neither 1 nor n is an isolated node. So, $(\mathbf{g},\tau,Y)$ is a legal instance indeed.

The reduction is now complete; it remains to establish correctness, that is, some frequent graph is contained in neither h_1 nor h_2 (i.e., $(\mathbf{g}, \tau, Y)$ is a "yes" instance) if and only if f has an m-node independent set.

We say that a path p in g is a $(1, n)$-*path* if it is of the form $v_1 - b_1 - a_2 - v_2 - b_2 \cdots - b_{n-1} - a_n - v_n$, where each v_i is one of $\{y_i, n_i\}$. Observe that no graph in Y contains a $(1, n)$-path; moreover, every frequent subgraph not contained in a graph of Y must include some $(1, n)$-path. Hence, $(\mathbf{g}, \tau, Y)$ is a "yes" instance if and only if there is a frequent $(1, n)$-path. So, we will prove that there is a frequent $(1, n)$-path p if and only if f has an independent set I of size m.

For the "if" direction, suppose that I is an independent set of size m. Let p_I be the $(1, n)$-path $v_1 - b_1 - a_2 - v_2 - b_2 \cdots - b_{n-1} - a_n - v_n$, where $v_i = y_i$ if $i \in I$, and $v_i = n_i$, otherwise. For $i \in I$, the graph g^i contains the path p_I because I is an independent set. Therefore, p_I is a subgraph of m members of $\mathbf{g}$, or, equivalently, p_I is a frequent subgraph.

For the "only if" direction, suppose that $p = v_1 - b_1 - a_2 - v_2 - b_2 \cdots - b_{n-1} - a_n - v_n$ is a frequent $(1, n)$-path. Since p is a subgraph of neither h_1 nor h_2, there is a set I of m indices among $1, \ldots, n$ such that p is a subgraph of g^i, for all $i \in I$. We claim that I is an independent set of f. Indeed, if i and j are such that p is a subgraph of both g^i and g^j, then v_i and v_j must both coincide with y_i, and consequently, v_j cannot be a neighbor of v_i, since, otherwise, g^j does not have y_i as a node.

This completes the proof of correctness for the reduction. Observe that g (and, hence, every graph in $\mathbf{g}$ and Y) is in each of the classes $\mathbf{G}_{\mathsf{L}}$, $\mathbf{PLN}_{\mathsf{L}}$, $\mathbf{BDG}^b_{\mathsf{L}}$ with $b > 2$, or $\mathbf{BTW}^w_{\mathsf{L}}$ with $w > 1$.

Finally, $W[1]$-hardness with respect to the parameter τ follows immediately from our reduction, since τ is equal to m, and it is known that Maximum Independent Set is $W[1]$-hard with respect to the size of the independent set [5]. $\square$

5. UNLABELED GRAPH CLASSES

In this section, we study the complexity of enumerating the maximal frequent subgraphs for the unlabeled classes in Table 1. Following Proposition 3.3 and Corollary 3.9, the classes that we can hope for tractable complexity results are $\mathbf{T}$ and $\mathbf{BDG}^2$.

We begin with the easy case, $\mathbf{BDG}^2$. By the same argument as that preceding Proposition 4.3, we obtain the following result.

PROPOSITION 5.1. $\mathsf{MaxFS}[\mathbf{BDG}^2]$-$\mathsf{enumerate}$ *is solvable in polynomial time.*

We now turn to the result with the most intricate proof in the paper. The proof itself is presented in the next section.

THEOREM 5.2. *The following is an* NP*-complete problem: Given two unlabeled trees t_1 and t_2, decide whether they have more than one maximal subtree in common (i.e., whether the set $\mathsf{MaxFS}^2[\mathbf{T}](t_1, t_2)$ has at least two members).*

Observe that obtaining one maximal frequent subgraph is in polynomial time for the class $\mathbf{T}$ (as it is for any strongly tractable canonized class of graphs). Thus, the decision problem of Theorem 5.2 reduces to $\mathsf{MaxFS}^2[\mathbf{T}]$-$\mathsf{extendible}\langle 2\rangle$. And, of course, the problem $\mathsf{MaxFS}[\mathbf{T}]$-$\mathsf{extendible}$ is in NP. Consequently, we obtain the following corollary of Theorem 5.2.

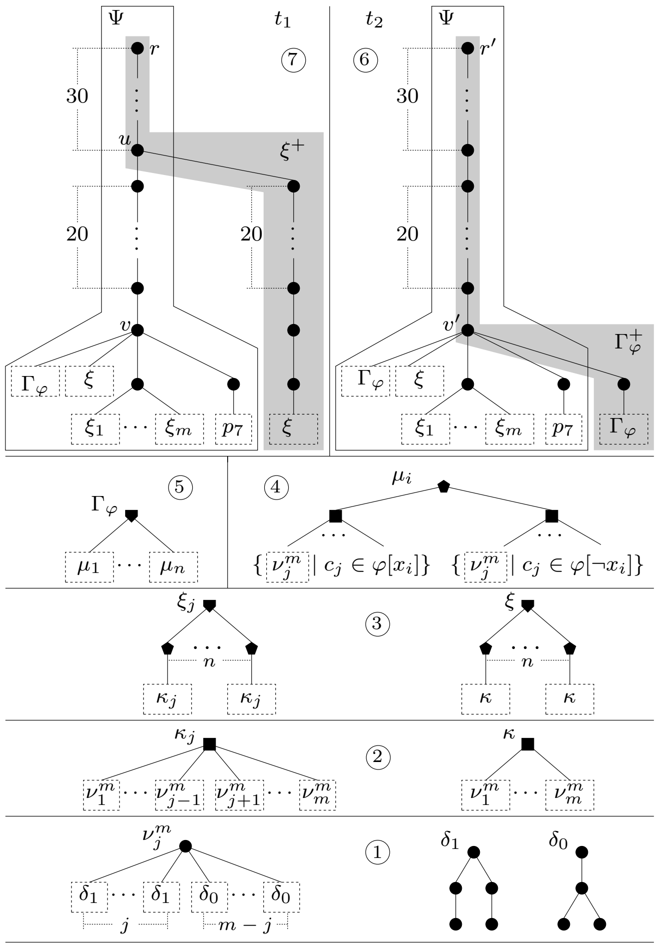

Figure 4: Constructions in the reduction

COROLLARY 5.3. *For all fixed thresholds $\tau \geq 2$, the problem $\mathsf{MaxFS}^\tau[\mathbf{T}]$-$\mathsf{extendible}\langle 2\rangle$ is* NP*-complete.*

PROOF. Theorem 5.2 gives the case $\tau = 2$. For $\tau > 2$, we still use Theorem 5.2, except that now our input $\mathbf{g}$ has $\tau - 1$ copies of t_1 (or of t_2) instead of just 1. $\square$

By combining Corollary 5.3 with Proposition 2.2 and Corollary 3.9, we conclude that in the case of unlabeled trees and unless P = NP, it is impossible to enumerate the maximal frequent subtrees in polynomial total time, or even obtain more than one in polynomial time, even if the frequency is fixed to 2 and the input consists of just two trees.

5.1 Proof of Theorem 5.2

In this section we give the proof of Theorem 5.2. Due to space limitations, the proofs of some of the lemmas we state are omitted. The complete details will be given in the full version of the paper.

The *CNF satisfiability* problem is as follows. The input consists of a formula $\varphi = c_1 \wedge \cdots \wedge c_m$ over the free variables $x_1, \ldots, x_n$, where each c_i is a disjunction of atomic formulas (where an atomic formula is a variable or a negated variable) that we call a *clause*. The goal is to determine whether φ is satisfiable. In the remainder of this section, we prove Theorem 5.2 by giving a polynomial-time reduction from

CNF satisfiability. Specifically, we fix a formula φ as above, and show how t_1 and t_2 are constructed from φ.

We use the following notation. When there is no risk of ambiguity, we may identify φ with the set $\{c_1, \dots, c_m\}$ of its clauses. If a is an atomic formula over $x_1, \dots, x_n$ (i.e., $a = x_i$ or $a = \neg x_i$ for some $i \in \{1, \dots, n\}$), then $\varphi[a]$ denotes the set of all the clauses c_j that have a as a disjunct. A *truth assignment* (*for* φ) is a mapping $\alpha : \{x_1, \dots, x_n\} \to \{\mathbf{true}, \mathbf{false}\}$. If α is a truth assignment, then $\varphi[\alpha]$ denotes the set of the clauses c_j that are satisfied by α. Hence, in our notation α is *satisfying* if $\varphi[\alpha] = \varphi$, and the goal is to determine whether such an α exists.

Figure 4 shows the construction of two trees, t_1 and t_2. The construction is bottom up—each tree is built from subtrees that are defined in previous phases. We view each tree in Figure 4 as *rooted*, which means that one node is distinguished as the *root*. In the figure, the root of a tree t is always the topmost (highest) node in the corresponding figure. The figure is self explanatory, except for a few points that we explain as we go along.

- The tree ν_j^m, where $0 < j \leq m$, represents the selection of the number j out of m; in our reduction, it represents the clause c_j.
- The tree κ has every tree ν_j^m, where $j = 1, \dots, m$, placed under the root.
- The tree κ_j is obtained from κ by removing the tree ν_j^m.
- For $i = 1, \dots, n$, the tree μ_i represents the clauses satisfied by x_i and its negation: the set $\varphi[x_i]$ is represented (through the corresponding ν_j^m) under the left child, and the set $\varphi[\neg x_i]$ is represented under the right child.
- The tree p_7 is simply a path consisting of 7 nodes; it is easy to verify that p_7 is of the same height as each ξ_i.

Consider the subtree Ψ of t_1. This subtree is also a subtree of t_2. In Figure 4, both occurrences of Ψ are surrounded by polygons (with a flashlight-like shape). The following lemma states that Ψ is a maximal common subtree of t_1 and t_2.

Lemma 5.4. *Ψ is a maximal common subtree of t_1 and t_2, that is, $\Psi \in \mathsf{MaxFS}^2[\mathbf{T}](t_1, t_2)$.*

Proof. The proof is straightforward in view of the following two observations.

1. t_1 has exactly one subtree that is isomorphic to Ψ.
2. If t is a subtree of t_2 that is isomorphic to Ψ, then t must contain the root of t_2.

Indeed, due to the first observation, if Ψ is not maximal then the node of distance 30 from the root must have degree 3, which cannot be true due to the second observation. $\square$

A key property in our reduction is the following simple lemma.

Lemma 5.5. *Let j and k be two numbers in $\{1, \dots, m\}$. Then $\nu_j^m \sqsubseteq \nu_k^m$ if and only if $j = k$.*

Consider the tree Γ_φ constructed in the fifth step of the reduction (numbered 5 in Figure 4). Note that in Γ_φ, the children of the root are the trees μ_i, for $i = 1, \dots, n$. Let α be a truth assignment for φ. We denote by Γ_α the subtree of Γ_φ obtained by pruning subtrees in the following way. For all $i = 1, \dots, n$, if $\alpha(x_i) = \mathbf{true}$, then we prune away the right subtree of μ_i; otherwise (i.e., if $\alpha(x_i) = \mathbf{false}$), then we prune away the left subtree of μ_i. Observe that in Γ_α, the root has n children, as in Γ_φ, but unlike Γ_φ, in Γ_α each of these n children has precisely one child. Consider the subtree Γ_φ^+ of t_2, covered by a grey shade in Figure 4. We write Γ_α^+ to denote the subtree of Γ_φ^+ obtained by replacing Γ_φ with Γ_α. The following lemma states a key property of Γ_α^+.

Lemma 5.6. *Let α be a truth assignment for φ. Then $\Gamma_\alpha^+ \sqsubseteq \Psi$ if and only if α is **not** a satisfying assignment.*

We can now establish the following lemma.

Lemma 5.7. *If φ is satisfiable, then $\mathsf{MaxFS}^2[\mathbf{T}](t_1, t_2)$ contains at least two maximal frequent subgraphs.*

The proof of Lemma 5.7 is straightforward: combine Lemmas 5.4 and 5.6, and the observation that Γ_α^+ is a subtree of ξ^+ (the subtree of t_1 shaded in Figure 4), for all truth assignments α, due to the construction of ξ.

It remains to prove the other direction of Lemma 5.7, namely, $|\mathsf{MaxFS}^2[\mathbf{T}](t_1, t_2)| > 1$ implies that φ is satisfiable. We embark on this task next.

Let h be a maximal 2-frequent subtree of t_1 and t_2, such that $h \not\sqsubseteq \Psi$. We fix h for the rest of this section. We will prove that there is a satisfying assignment α such that h is isomorphic to Γ_α^+.

Fix a subtree h_1 of t_1 that is isomorphic to h, and an isomorphism $\mu_1 : \mathsf{nodes}(h) \to \mathsf{nodes}(h_1)$. Similarly, fix a subtree h_2 of t_2 that is isomorphic to h, and an isomorphism $\mu_2 : \mathsf{nodes}(h) \to \mathsf{nodes}(h_2)$.

If t and t' are two rooted trees, we write $t \sqsubseteq_{\mathsf{r}} t'$ to denote that there is an isomorphism μ from t to a subgraph of t', such that μ maps the root of t to the root of t'. Clearly, $t \sqsubseteq_{\mathsf{r}} t'$ implies $t \sqsubseteq t'$. The following lemma states that h is isomorphic to a subtree of both ξ^+ and Γ_φ^+, via root-preserving isomorphisms.

Lemma 5.8. *$h \sqsubseteq_{\mathsf{r}} \xi^+$ and $h \sqsubseteq_{\mathsf{r}} \Gamma_\varphi^+$.*

Finally, the following lemma implies that φ is satisfiable, thereby completing the proof of Theorem 5.2.

Lemma 5.9. *There is a satisfying assignment α such that h is isomorphic to Γ_α^+.*

Proof. We will prove that h is isomorphic to Γ_α^+ for some truth assignment α. We will then derive that α is necessarily a satisfying assignment by using Lemma 5.6 and the assumption that $h \not\sqsubseteq \Psi$.

Lemma 5.8 states that h is isomorphic to a root subtree of both ξ^+ and Γ_α^+. Since h is maximal, h_1 consists of the path from the root all the way to the root of ξ (which corresponds to the root of Γ_φ in h_2), which we denote by r_ξ. Moreover, h being maximal implies that r_ξ has n children in h_1. Those n children correspond to the roots of the μ_i in h_2. Hence, since h is maximal, we have that each of the subtrees of μ_i in h_2 is *precisely* the one obtained by pruning away one of the two subtrees (see Step 4 in Figure 4). From the definition of Γ_α^+, it follows that h_2 is necessarily Γ_α^+ for some truth assignment α. This completes the proof because $h \equiv h_2$. $\square$

Table 2: The complexity of $\mathsf{MaxFS}[\mathcal{P}]$-extendible and the restricted variants $\mathsf{MaxFS}[\mathcal{P}]$-extendible$\langle k\rangle$, $\mathsf{MaxFS}^{\tau}[\mathcal{P}]$-extendible and $\mathsf{MaxFS}^{\tau}[\mathcal{P}]$-extendible$\langle k\rangle$ for the strongly tractable classes of Table 1; $\surd$ stands for "polynomial time" (or "FP" when in parentheses), $\times$ stands for "NP-complete" (or "#P-complete" when in parentheses)

$\mathcal{P}$	Unrestricted (#)	$k = 2$	Fix $k > 2$	Fix $\tau \geq 2$ (#)	$k = 2$, Fix $\tau \geq 2$
$\mathbf{T}$	$\times$ ($\times$)	$\times$	$\times$	$\times$ ($\times$)	$\times$
$\mathbf{T}_\mathsf{L}$	$\times$ ($\times$)	$\surd$	$\surd$	$\surd$ ($\surd$)	$\surd$
$\mathbf{BDG}^2$, $\mathbf{BDG}^2_\mathsf{L}$	$\surd$ ($\surd$)	$\surd$	$\surd$	$\surd$ ($\surd$)	$\surd$
$\mathbf{G}_\mathsf{L}$, $\mathbf{PLN}_\mathsf{L}$, $\mathbf{BDG}^{b>2}_\mathsf{L}$, $\mathbf{BTW}^{w>1}_\mathsf{L}$	$\times$ ($\times$)	$\surd$	$\times$	$\surd$ ($\surd$)	$\surd$

6. COUNTING COMPLEXITY

We now study the complexity of counting the number of maximal frequent subgraphs; more precisely, we study the problem $\#\mathcal{R}$ for the enumeration relations $\mathcal{R}$ considered in this paper. As a matter of fact, the problem of counting maximal frequent subgraphs has already been studied by Yang [36], who established #P-completeness for both labeled trees and unlabeled trees.

THEOREM 6.1. [36] *If $\mathcal{R}$ is $\mathsf{MaxFS}[\mathbf{T}]$ or $\mathsf{MaxFS}[\mathbf{T}_\mathsf{L}]$, then $\#\mathcal{R}$ is #P-complete. Moreover, #P-completeness holds, even if the inputs are labeled or unlabeled binary trees (equivalently, trees of degree at most 3).*

Yang's results do not cover the case of graphs of degree bounded by 2; we will consider this case next. More importantly, the threshold is part of the input in Yang's results. Thus, in what follows, we will also examine the effect of bounding the threshold. Note that, as regards counting problems, it is not meaningful to bound the number of witnesses. We begin by discussing cases for which $\#\mathcal{R}$ is in polynomial time.

PROPOSITION 6.2. *The counting problem $\#\mathcal{R}$ is in FP if $\mathcal{R}$ is one of $\mathsf{MaxFS}[\mathbf{BDG}^2]$, $\mathsf{MaxFS}[\mathbf{BDG}^2_\mathsf{L}]$ and $\mathsf{MaxFS}^{\tau}[\mathcal{P}_\mathsf{L}]$, where τ is a fixed threshold and $\mathcal{P}_\mathsf{L}$ is the labeled variant of a class $\mathcal{P}$ in Table 1.*

The preceding proposition is immediate from the fact that, for the enumeration relations $\mathcal{R}$ considered, $\mathcal{R}$-enumerate is solvable in polynomial time (and not just, e.g., polynomial total time); hence, one can count the witnesses in polynomial time by actually producing them first.

It remains to examine the case of unlabeled trees and fixed threshold. In the final result of this paper, we establish #P-completeness even in the case of $\tau = 2$. The proof builds on the reduction we used for proving Theorem 5.2. Although that reduction is not parsimonious, we show how it can be made so by properly changing the CNF formula. The complete proof will be given in the full version of the paper.

THEOREM 6.3. *For every fixed $\tau \geq 2$, the counting problem $\#\mathsf{MaxFS}^{\tau}[\mathbf{T}]$ is #P-complete.*

7. CONCLUDING REMARKS

Our interest in mining maximal frequent subgraphs arose in a research project at the IBM Almaden Research Center, where we observed that a small threshold τ and a small number k of answers are often realistic assumptions in applications. In this paper, we used the lens of computational complexity to obtain a fairly comprehensive picture, shown in Table 2, for the problem of mining maximal frequent subgraphs. Furthermore, we believe that our main technical result about unlabeled trees is of independent interest as a new NP-complete problem: given two unlabeled trees, do they have more than one maximal common subtree in common?

The results presented here suggest several different directions for further research. We conclude by mentioning two such concrete directions.

In addition to maximal frequent subgraphs, a different representation of the space of all frequent subgraphs can be obtained by considering the collection of all *closed frequent subgraphs*, that is, those frequent subgraphs that no proper supergraph has the same set of containing graphs. Every maximal frequent subgraph is closed, while the converse need not be true. Thus, closed frequent subgraphs do not provide as compact a representation of the space of frequent subgraphs as the maximal ones do. On the other hand, closed frequent subgraphs can be used to compute the actual occurrences of all frequent subgraphs. Several different algorithms for enumerating all closed frequent subgraphs have been proposed [27, 33, 35, 38]. To date, no systematic investigation of the complexity of mining closed frequent subgraphs has been carried out.

Finally, note that labeled graphs and unlabeled graphs can be thought of as two extreme cases in the first of which each node has a distinct label, whereas in the second all nodes have the same label. This suggests considering an intermediate case in which every label has a bounded number of occurrences in the graph. Mining maximal frequent subgraphs in this case is a problem that remains to be explored.

Acknowledgment

The research of Phokion Kolaitis was partially supported by NSF Grant IIS-1217869.

8. REFERENCES

[1] N. Alon and A. Shapira. Every monotone graph property is testable. *SIAM J. Comput.*, 38(2):505–522, 2008.

[2] L. Babai and E. M. Luks. Canonical labeling of graphs. In *STOC*, pages 171–183. ACM, 1983.

[3] E. Boros, V. Gurvich, L. Khachiyan, and K. Makino. On maximal frequent and minimal infrequent sets in binary matrices. *Ann. Math. Artif. Intell.*, 39(3):211–221, 2003.

[4] M. Deshpande, M. Kuramochi, N. Wale, and G. Karypis. Frequent substructure-based approaches for classifying chemical compounds. *IEEE Trans. Knowl. Data Eng.*, 17(8):1036–1050, 2005.

[5] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, 1999.

[6] E. Friedgut and G. Kalai. Every monotone graph property has a sharp threshold. *Proc. Amer. Math. Soc.*, 124(10):2993–3002, 1996.

[7] M. R. Garey, D. S. Johnson, and R. E. Tarjan. The planar Hamiltonian circuit problem is NP-complete. *SIAM J. Comput.*, 5(4):704–714, 1976.

[8] G. Greco, A. Guzzo, G. Manco, and D. Saccà. Mining and reasoning on workflows. *IEEE Trans. Knowl. Data Eng.*, 17(4):519–534, 2005.

[9] G. Greco, A. Guzzo, G. Manco, and D. Saccà. Mining unconnected patterns in workflows. *Inf. Syst.*, 32(5):685–712, 2007.

[10] E. Gudes, S. E. Shimony, and N. Vanetik. Discovering frequent graph patterns using disjoint paths. *IEEE Trans. Knowl. Data Eng.*, 18(11):1441–1456, 2006.

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

[12] J. E. Hopcroft and R. E. Tarjan. Isomorphism of planar graphs. In *Complexity of Computer Computations*, The IBM Research Symposia Series, pages 131–152. Plenum Press, New York, 1972.

[13] J. Huan, W. Wang, J. Prins, and J. Yang. SPIN: mining maximal frequent subgraphs from graph databases. In *KDD*, pages 581–586, 2004.

[14] A. Inokuchi, T. Washio, and H. Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In *PKDD*, pages 13–23, 2000.

[15] A. Inokuchi, T. Washio, and H. Motoda. Complete mining of frequent patterns from graphs: Mining graph data. *Machine Learning*, 50(3):321–354, 2003.

[16] D. Johnson, M. Yannakakis, and C. Papadimitriou. On generating all maximal independent sets. *Information Processing Letters*, 27:119–123, 1988.

[17] L. Khachiyan, E. Boros, K. Borys, K. M. Elbassioni, and V. Gurvich. Generating all vertices of a polyhedron is hard. *Discrete & Computational Geometry*, 39(1-3):174–190, 2008.

[18] B. Kimelfeld and Y. Sagiv. Finding and approximating top-k answers in keyword proximity search. In *PODS*, pages 173–182. ACM, 2006.

[19] B. Kimelfeld and Y. Sagiv. Maximally joining probabilistic data. In *PODS*, pages 303–312. ACM, 2007.

[20] M. Kuramochi and G. Karypis. Frequent subgraph discovery. In *ICDM*, pages 313–320, 2001.

[21] M. Kuramochi and G. Karypis. An efficient algorithm for discovering frequent subgraphs. *IEEE Trans. Knowl. Data Eng.*, 16(9):1038–1051, 2004.

[22] K. Makino and T. Ibaraki. Interior and exterior functions of boolean functions. *Discrete Applied Mathematics*, 69(3):209–231, 1996.

[23] J. Matousek and R. Thomas. On the complexity of finding iso- and other morphisms for partial k-trees. *Discrete Mathematics*, 108(1-3):343–364, 1992.

[24] R. J. Mooney, P. Melville, L. R. Tang, J. Shavlik, I. Dutra, and D. Page. Relational data mining with inductive logic programming for link discovery. *Data Mining: Next Generation Challenges and Future Directions*, pages 239–254, 2004.

[25] S. Nijssen and J. N. Kok. Frequent graph mining and its application to molecular databases. In *SMC (5)*, pages 4571–4577. IEEE, 2004.

[26] Y. Okamoto, T. Uno, and R. Uehara. Counting the number of independent sets in chordal graphs. *J. Discrete Algorithms*, 6(2):229–242, 2008.

[27] J. Pei, J. Han, and R. Mao. CLOSET: An efficient algorithm for mining frequent closed itemsets. In *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 21–30, 2000.

[28] A. Stoica and C. Prieur. Structure of neighborhoods in a large social network. In *CSE (4)*, pages 26–33. IEEE Computer Society, 2009.

[29] L. T. Thomas, S. R. Valluri, and K. Karlapalem. Margin: Maximal frequent subgraph mining. *TKDD*, 4(3), 2010.

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

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

[32] F. Wagner. Graphs of bounded treewidth can be canonized in AC^1. In *CSR*, volume 6651 of *Lecture Notes in Computer Science*, pages 209–222. Springer, 2011.

[33] J. Wang, J. Han, and J. Pei. CLOSET+: searching for the best strategies for mining frequent closed itemsets. In *KDD*, pages 236–245, 2003.

[34] X. Yan and J. Han. gSpan: Graph-based substructure pattern mining. In *ICDM*, pages 721–724, 2002.

[35] X. Yan and J. Han. CloseGraph: mining closed frequent graph patterns. In *KDD*, pages 286–295, 2003.

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

[37] M. Yannakakis. Algorithms for acyclic database schemes. In *VLDB*, pages 82–94. IEEE Computer Society, 1981.

[38] M. J. Zaki and C.-J. Hsiao. CHARM: An efficient algorithm for closed itemset mining. In *SDM*, 2002.