The Complexity of Mining Maximal Frequent Subgraphs

BENNY KIMELFELD, LogicBlox, Inc.
PHOKION G. KOLAITIS, University of California, Santa Cruz and IBM Research – Almaden

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 article, 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 treewidth (trees being a special case). Moreover, each class has two variants: that in which the nodes are unlabeled, and that 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

Additional Key Words and Phrases: Graph mining, maximal frequent subgraphs, enumeration complexity

ACM Reference Format:
Benny Kimelfeld and Phokion G. Kolaitis. 2014. The complexity of mining maximal frequent subgraphs. ACM Trans. Datab. Syst. 39, 4, Article 32 (December 2014), 33 pages.
DOI: http://dx.doi.org/10.1145/2629550

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

The research of B. Kimelfeld was done while he was at IBM Research, Almaden. The research of P. Kolaitis was partially supported by NSF grant IIS-1217869.
Authors' addresses: B. Kimelfeld (corresponding author), LogicBlox, Inc., 1349 West Peachtree Street NW, Atlanta, GA 30309; email: bennyk@gmail.com; P. G. Kolaitis, University of California at Santa Cruz, 1156 High Street, Santa Cruz, CA 95064 and IBM Research Almaden, 650 Harry Road, San Jose, CA 95120-6099.

DOI: http://dx.doi.org/10.1145/2629550

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 [Nijssen and Kok 2004; Deshpande et al. 2005], behavior and link analysis in social networks [Mooney et al. 2004; Stoica and Prieur 2009], and workflow analysis [Greco et al. 2005, 2007]. In particular, graphs form a rich data structure that can capture complicated relationships between data encountered in such applications.

Given a finite sequence $\mathbf{g} = \langle g_1, \ldots, g_n \rangle$ 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 Gudes et al. [2006], Inokuchi et al. [2000, 2003], Kuramochi and Karypis [2001, 2004], and Yan and Han [2002]. 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 [Huan et al. 2004] and MARGIN [Thomas et al. 2010]. However, with the exception of some earlier work on maximal frequent itemsets [Boros et al. 2003; Gunopulos et al. 2003] and on the counting complexity of maximal frequent subgraphs [Yang 2004], not much is known about the computational complexity of the fundamental enumeration and decision problems concerning maximal frequent subgraphs.

In this article, we use the lens of computational complexity to systematically investigate the problem of mining maximal frequent subgraphs. 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 II. Each such a class $\mathcal{C}$ (e.g., the class $\mathbf{T}$ of all trees) consists of *unlabeled* graphs, but also has a *labeled* variant, denoted by $\mathcal{C}_\mathsf{L}$ (e.g., the class $\mathbf{T}_\mathsf{L}$ of all labeled trees), where the nodes of each graph are uniquely labeled. A *graph mining setting* is a pair $(\mathcal{P}, \mathcal{D})$ of such classes, where $\mathcal{P}$ (the *pattern class*) and $\mathcal{D}$ (*the data class*) are both labeled or both unlabeled; in the former case, we say that $(\mathcal{P}, \mathcal{D})$ is a *labeled* graph mining setting, while in the latter we say it is an *unlabeled* one. Each graph mining setting $(\mathcal{P}, \mathcal{D})$ gives rise to the following *enumeration problem*: given a sequence $\mathbf{g} = \langle g_1, \ldots, g_n \rangle$ of graphs from the data class $\mathcal{D}$ and a threshold τ, produce all $\mathcal{P}$-maximal frequent subgraphs of $\mathbf{g}$, up to isomorphism and without repetitions, that is, produce (up to isomorphism and without repetitions) all frequent subgraphs h of $\mathbf{g}$ such that h is in the pattern class $\mathcal{P}$ and there is no frequent subgraph h' in $\mathcal{P}$ such that h is (isomorphic to) a strict subgraph of h'. We denote this problem by MaxFS$[\mathcal{P}, \mathcal{D}]$-enumerate. An important and natural special case is that in which the pattern class $\mathcal{P}$ is the same as the data class $\mathcal{D}$.

Note that, for each graph mining setting $(\mathcal{P}, \mathcal{D})$, the input to the associated enumeration problem MaxFS$[\mathcal{P}, \mathcal{D}]$-enumerate consists of a sequence $\mathbf{g}$ of graphs from $\mathcal{D}$ 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. A different restriction of this problem is obtained by imposing a fixed bound k on the number of desired $\mathcal{P}$-maximal frequent subgraphs; in other words, instead of enumerating all $\mathcal{P}$-maximal frequent subgraphs, the goal is relaxed to producing just k $\mathcal{P}$-maximal

frequent subgraphs. Thus, our investigation of the problem of mining maximal frequent subgraphs will be carried out by considering the effect of the aforementioned three parameters: the graph mining setting $(\mathcal{P}, \mathcal{D})$, the threshold τ, and the bound k.

Even though the number of maximal frequent subgraphs (up to isomorphism) is often much smaller than that of frequent subgraphs, it can still be exponential in the size of the given sequence of graphs. Johnson et al. [1988] 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 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 *nonemptiness* 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 nonemptiness 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 [Kimelfeld and Sagiv 2007; Khachiyan et al. 2008; Boros et al. 2003]. 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 article, 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 that of a standard decision problem. This will be of great interest to us because in Section 3 we will show that, for data mining settings $(\mathcal{P}, \mathcal{D})$ satisfying certain reasonable hypotheses, the problem MaxFS$[\mathcal{P}, \mathcal{D}]$-enumerate is enumeration self-reducible. In particular, enumeration self-reducibility holds for all data mining settings $(\mathcal{P}, \mathcal{D})$ such that both $\mathcal{P}$ and $\mathcal{D}$ are labeled variants $\mathcal{C}_{\mathsf{L}}$ of the classes $\mathcal{C}$ in Table II, as well when $\mathcal{D}$ is the class **T** of all (unlabeled) trees or the class $\mathbf{BDG}^2$ of all (unlabeled) graphs of degree at most 2.

The following is a short summary of the complexity results for the enumeration problem that we establish in this article.

—In the labeled case, every considered graph mining setting is intractable, except for the case where the pattern class (or the data class) is the class of graphs with degree bounded by 2 (i.e., the class of simple paths and simple cycles). Bounding the number

Table I. Complexity Results for Graph Mining Settings $(\mathcal{P}, \mathcal{D})$ in the Case where $\mathcal{P} = \mathcal{D}$

$\mathcal{P}$	Unrestr.	$k = 2$	Fix $k > 2$	Fix $\tau \geq 2$	$k = 2$, Fix $\tau \geq 2$
$\mathbf{T}$	NPc (#Pc)	NPc	NPc	NPc (#Pc)	NPc
$\mathbf{T}_{\mathsf{L}}$	NPc (#Pc)	FP	FP	FP (FP)	FP
$\mathbf{BDG}^2$, $\mathbf{BDG}^2_{\mathsf{L}}$	FP (FP)	FP	FP	FP (FP)	FP
$\mathbf{G}_{\mathsf{L}}$, $\mathbf{PLN}_{\mathsf{L}}$, $\mathbf{BDG}^{d>2}_{\mathsf{L}}$, $\mathbf{BTW}^{w>1}_{\mathsf{L}}$	NPc (#Pc)	FP	NPc	FP (FP)	FP

"FP" and "PD" refer to the complexity of enumeration (where "FP" is the class of functions computable in polynomial time, and "PD" is the class of enumeration problems solvable with polynomial delay) and "NPc" means that the extendibility problem is NP-complete. Complexity classes in parentheses are those of the corresponding counting problem (where "#Pc" stands for "#P-complete").

Table II. Unlabeled Canonized Classes of Graphs

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

Each class $\mathcal{C}$ has a corresponding labeled class, which is denoted by $\mathcal{C}_{\mathsf{L}}$; except for $\mathcal{C} = \mathbf{G}$, all canonizations are assumed computable in polynomial time.

of answers gives a tractability result for the important case of trees, but does not help the other cases. However, quite a few cases become tractable when bounding the threshold. The results concerning the complexity of the labeled case are depicted in Table III.

—In the unlabeled case, we again have tractability in the scenario where the pattern class (or the data class) is the class of graphs with degree bounded by 2. The remaining cases are intractable and, unlike the labeled case, intractability remains even when bounding the number of answers, the threshold, or both. The most striking and technically most challenging result concerns the graph mining setting $(\mathbf{T}, \mathbf{T})$ in which both the pattern class and the data class is the class of all 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? The results concerning the complexity of the unlabeled case are depicted in Table IV.

We also investigate the complexity of the counting problem associated with a graph mining setting $(\mathcal{P}, \mathcal{D})$: given a sequence $\mathbf{g}$ of graphs from $\mathcal{D}$ and a threshold τ, compute the number of $\mathcal{P}$-maximal frequent subgraphs. For this problem, the obvious tractable cases are those where the enumeration problem is in polynomial time (and not just polynomial *total* time). For those cases where the enumeration problem is intractable, we show how our proofs of hardness for the enumeration problem can be used to prove #P-completeness of counting (by making the reductions parsimonious). We also establish cases where the enumeration problem is solvable with polynomial delay, but still counting is #P-complete.

As a sample of the complexity results established in this article, we present here Table I, which covers the important case in which the pattern class $\mathcal{P}$ is the same as the data class $\mathcal{D}$. The meaning of the symbols for the classes $\mathcal{P}$ (e.g., $\mathbf{T}$, $\mathbf{T}_{\mathsf{L}}$, and so on) is given in Table II.

This article extends a conference publication of the authors [Kimelfeld and Kolaitis 2013]. The main extension is the differentiation between the pattern class of graphs and the data class of graphs—the conference publication is restricted to the case where those classes are the same. In particular, that publication has established the complexity results of Table I. Due to this differentiation, the present article contains a number of new results. For example, the polynomial-delay results (Theorems 4.4, 4.13, and 4.17), as well as the generic algorithm underneath (Section 3.3), are presented here for the first time. As another example, this article establishes new results (Theorems 6.2 and 6.3) on the counting complexity of the problem we study. In Section 3, we restrict the discussion to graph classes that are *closed under connected subgraphs*, instead of using the more general "incremental progression" property we used in the conference paper. We do so to simplify the presentation (and indeed, we believe that significant simplification has been achieved), as all the concrete graph classes we consider here are closed under connected subgraphs.

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.

—$\mathcal{R}$-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 $y \in \mathcal{R}(x) \backslash Y$ or declare that none exists.
—$\mathcal{R}$-extendible is the following decision problem: given a string x and a set $Y \subseteq \mathcal{R}(x)$ as input, decide whether $\mathcal{R}(x) \backslash Y \neq \emptyset$.

In the three algorithmic problems, the string x represents an instance of a problem, and the witnesses are the results that we wish to produce for the input x. In the next sections, x will be the specification of a graph mining task (that is, a collection of graphs and a threshold), and the witnesses are the maximal frequent subgraphs (see Section 3). Our ultimate goal here will be to analyze the complexity of $\mathcal{R}$-enumerate. The problem $\mathcal{R}$-extend captures a natural incremental strategy of solving $\mathcal{R}$-enumerate: starting with the empty set of witnesses, repeatedly create new witnesses until none remains. As we explain in Section 2.1.1, complexity analysis for $\mathcal{R}$-enumerate may seem inherently different from the analysis of decision algorithms (through decision algorithms, reductions, etc.) since the number of witnesses that a solver is required to print may be exponential in the size of the input. The problem $\mathcal{R}$-extendible, on the other hand, is just a standard decision problem. Nevertheless, we will draw a connection between the problems $\mathcal{R}$-enumerate and $\mathcal{R}$-extendible (*self-reducibility*), and we will often use this connection to translate complexity analysis for the former into complexity analysis for the latter.

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.

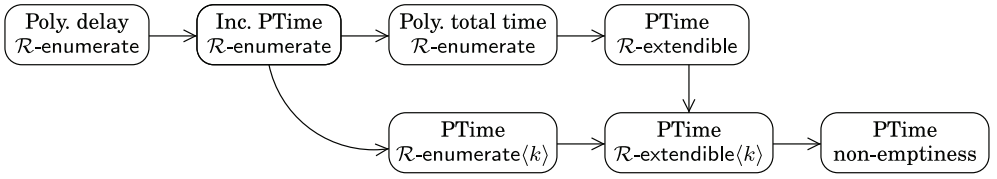

Fig. 1. Levels of tractability for enumeration problems for an NP-relation $\mathcal{R}$.

—The problem $\mathcal{R}$-enumerate$\langle k\rangle$ is similar to the problem $\mathcal{R}$-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 $\mathcal{R}$-extend$\langle k\rangle$ and $\mathcal{R}$-extendible$\langle k\rangle$ are similar to the problems $\mathcal{R}$-extend and $\mathcal{R}$-extendible, respectively, except that the condition $|Y| < k$ is imposed on the input.

As a special case, the problem $\mathcal{R}$-extendible$\langle 1\rangle$ is the *nonemptiness* 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 $\mathcal{R}$-enumerate is called an *enumeration algorithm*. As said earlier, 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. [1988] 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*) $\mathcal{R}$-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* $\mathcal{R}$-enumerate$\langle k\rangle$ *is in polynomial time if and only if* $\mathcal{R}$-extend$\langle k\rangle$ *is in polynomial time.*

It is well known (and easy to show) that a polynomial-total-time algorithm for $\mathcal{R}$-enumerate can be used to solve the nonemptiness 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 nonemptiness problem (e.g., Makino and Ibaraki [1996]). 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 nonemptiness problem is tractable or even trivial (e.g., Khachiyan et al. [2008] and Kimelfeld and Sagiv [2007]).

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 nonemptiness.

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

$\mathcal{R}$-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 $\mathcal{R}$-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 $\mathcal{R}$-extend$\langle k\rangle$ that uses a solver for $\mathcal{R}$-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.*

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

Throughout this article, we will implicitly use the concept of enumeration self-reducibility and also Proposition 2.2 by interchangeably analyzing $\mathcal{R}$-extendible and $\mathcal{R}$-enumerate. For example, we will prove that $\mathcal{R}$-extendible is NP-complete in order to establish a lower bound for $\mathcal{R}$-enumerate that, under standard complexity-theoretic assumptions, rules out the existence of polynomial-total-time solutions for $\mathcal{R}$-enumerate.

2.1.3. Counting. In Section 6, we will study the complexity of counting all maximal frequent subgraphs. Here, we provide some basic terminology and make a connection to the enumeration problem and to enumeration self-reducibility.

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 [Valiant 1979a] 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 [Toda and Ogiwara 1992]. 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 [Johnson et al. 1988] (in particular, we have enumeration self-reducibility), yet counting the number of such sets is a #P-hard problem [Okamoto et al. 2008].

2.2. Graphs and Canonized Classes

In this article, we focus on undirected graphs which from now on we refer to as, simply, graphs. If g is a graph, then the set of its nodes is denoted by nodes(g), while the set of its edges is denoted by edges(g). We assume that, for every graph g, the set

[1]A *maximal independent set* is an independent set that is not strictly contained in any other independent 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, nonedges, 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. We say that $\mathcal{G}$ is *connected* if every graph in $\mathcal{G}$ is connected. We say that $\mathcal{G}$ is *closed under connected subgraphs* if, for all g in $\mathcal{G}$, every connected subgraph of g is also in $\mathcal{G}$.

Let $\mathcal{G}$ be a graph class. A *canonization for* $\mathcal{G}$ is a function $\kappa : \mathcal{G} \to \mathcal{G}$ possessing the following properties:

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

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

The focus of this article is on graph classes that are connected and closed under connected subgraphs. We will establish complexity results for specific such classes. For this we will use properties that capture different aspects of tractable behavior. Let $\mathcal{C}$ be a canonized graph class, and let $\mathcal{D}$ be a graph class.

—We say that $\mathcal{C}$ has a *tractable membership testing* if there is a polynomial-time algorithm that tests whether a given graph g is in $\mathcal{C}$.
—We say $\mathcal{C}$ has *tractable canonization* if there is a polynomial-time algorithm that, given $g \in \mathcal{C}$, computes $\kappa_{\mathcal{C}}(g)$.
—The *subgraph isomorphism of* $\mathcal{C}$ *in* $\mathcal{D}$ is the problem of testing, given $h \in \mathcal{C}$ and $g \in \mathcal{D}$, whether $h \sqsubseteq g$. If this problem can be solved in polynomial time, then we say that $\mathcal{C}$ has *tractable subgraph isomorphism* in $\mathcal{D}$.

Table II lists five specific unlabeled classes $\mathcal{C}$ by specifying $dom(\mathcal{C})$ in each case. For simplicity of presentation, connectedness is implicit in the notation of the classes of Table II. Although Table II does not mention the canonization of each class $\mathcal{C}$, we assume that each class $\mathcal{C}$ comes with a fixed canonization $\kappa_{\mathcal{C}}$. Note that some containment relationships hold between the domains of the classes of Table II; 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 $\mathcal{C}$ of graphs in Table II, we write $\mathcal{C}_{\mathsf{L}}$ to denote the corresponding labeled canonized class. As an example, $\mathbf{BTW}^3_{\mathsf{L}}$ consists of the labeled graphs of treewidth 3.

Note that all the classes in Table II, as well as their labeled variants, are connected, closed under connected subgraphs, and have tractable membership testing. We will assume that if, $\mathcal{C}$ is one of $\mathbf{T}$, $\mathbf{PLN}$, $\mathbf{BDG}^d$, and $\mathbf{BTW}^w$, then its associated canonization is computable in polynomial time (hence, $\mathcal{C}$ has tractable canonization); we can do so due to known results [Hopcroft and Tarjan 1972; Babai and Luks 1983; Wagner 2011]. 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 if $\mathcal{C}$ has tractable canonization, then $\mathcal{C}$ has polynomial-time graph isomorphism testing, thus for a class of graphs, canonization is not easier than graph isomorphism. We also assume that each labeled class $\mathcal{C}$ in the table has tractable canonization; we can make such an assumption because of the straightforward observation that there is a polynomial-time computable canonization for the class of all labeled graphs. Finally, note that if $\mathcal{C}$ and $\mathcal{D}$ are labeled, then $\mathcal{C}$ has tractable subgraph isomorphism in $\mathcal{D}$. In the case of unlabeled classes, it is known that $\mathcal{C}$ has tractable subgraph isomorphism in $\mathcal{D}$ if $\mathcal{D}$ is $\mathbf{T}$ or $\mathbf{BDG}^2$, as well as when both $\mathcal{C} = \mathbf{BDG}^d$ and $\mathcal{D} = \mathbf{BTW}^w$ for some d and w [Matousek and Thomas 1992]. But, unless P = NP, none of other combinations has tractable subgraph isomorphism [Matousek and Thomas 1992; Garey et al. 1976].

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 $\mathbf{g} = \langle g_1, \ldots, g_n \rangle$ 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 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 $\mathbf{g}$".

Definition 3.2. A *graph mining setting* is a pair $(\mathcal{P}, \mathcal{D})$ where $\mathcal{P}$, the *pattern class*, is a canonized graph class, and $\mathcal{D}$, the *data class*, is a graph class. We require $\mathcal{P}$ and $\mathcal{D}$ to be both labeled or both unlabeled; in the former case we say that the setting $(\mathcal{P}, \mathcal{D})$ is *labeled*, and in the latter it is *unlabeled*. Let $(\mathcal{P}, \mathcal{D})$ be a graph mining setting.

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

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

In Definition 3.2, the sequence $\mathbf{g} = \langle g_1, \ldots, g_n \rangle$ 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 4.6), 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, $\mathsf{MaxFS}^{\tau}[\mathcal{P}, \mathcal{D}]$ is defined similarly to $\mathsf{MaxFS}[\mathcal{P}, \mathcal{D}]$, 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 article, we will explore the effect of these parameters on the complexity of the enumeration problems.

3.1. Tractable Verification

Our ultimate goal is to study the complexity of the problems $\mathsf{MaxFS}[\mathcal{P}, \mathcal{D}]$-enumerate, as well as their parameterized versions, for different (combinations of) canonized classes $\mathcal{P}$ and $\mathcal{D}$ of graphs (and with main focus on the classes of Table II). To avoid immediate intractability hurdles, we must make some assumptions about the complexity entailed in the classes $\mathcal{P}$ and $\mathcal{D}$. To begin with, we need to be able to test the validity of involved graphs and to produce canonized witnesses; therefore, our first requirement is that both $\mathcal{P}$ and $\mathcal{D}$ should have tractable membership testing and that $\mathcal{P}$ should have tractable canonization. Unfortunately, this does not suffice for the tractability of solving $\mathsf{MaxFS}[\mathcal{P}, \mathcal{D}]$-enumerate. The following straightforward proposition implies that, in the cases we focus on, subgraph isomorphism of $\mathcal{P}$ in $\mathcal{D}$ is necessary—it reduces, via a trivial inspection of the input and output, to $\mathsf{MaxFS}[\mathcal{P}, \mathcal{D}]$-enumerate, even if both τ and k are fixed small integers.

PROPOSITION 3.3. *Let $(\mathcal{P}, \mathcal{D})$ be a graph mining setting such that $\mathcal{P}$ is connected and that $\mathcal{D}$ is closed under connected subgraphs. Let h and g be two graphs in $\mathcal{P}$ and $\mathcal{D}$, respectively, such that $|\mathsf{nodes}(h)| \leq |\mathsf{nodes}(g)|$ and $|\mathsf{edges}(h)| \leq |\mathsf{edges}(g)|$. Then $h \sqsubseteq g$ if and only if both of the following are true.*

—*$h \in \mathcal{D}$.*
—*$\mathsf{MaxFS}^2[\mathcal{P}, \mathcal{D}](h, g)$ is a singleton $\{h'\}$, such that h' has the same number of nodes and edges as h.*

Proposition 3.3 justifies our additional requirement that $\mathcal{P}$ should have tractable subgraph isomorphism in $\mathcal{D}$. 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 $(\mathcal{P}, \mathcal{D})$ be a graph mining setting such that $\mathcal{P}$ is connected and $\mathcal{D}$ is closed under connected subgraphs. Suppose $\mathcal{P}$ and $\mathcal{D}$ have tractable membership testing and that $\mathcal{P}$ has tractable canonization. Suppose also that subgraph isomorphism of $\mathcal{P}$ in $\mathcal{D}$ is* NP*-hard. The following problem is both* NP*-hard and* coNP*-hard: given h and $\mathbf{g}$, decide whether $(\mathbf{g}, h) \in \mathsf{MaxFS}^2[\mathcal{P}, \mathcal{D}]$. In particular, neither $\mathsf{MaxFS}^2[\mathcal{P}, \mathcal{D}]$ nor $\mathsf{MaxFS}[\mathcal{P}, \mathcal{D}]$ is an* NP*-relation unless* P = NP.

PROOF. Let $h \in \mathcal{P}$ and $g \in \mathcal{D}$ be two graphs that are input to the subgraph isomorphism problem, that is, the goal is to determine whether $h \sqsubseteq g$. If $h \notin \mathcal{G}$, then $h \not\sqsubseteq g$, because $\mathcal{G}$ is closed under connected subgraphs (and hence contains every connected subgraph of g). Since $\mathcal{G}$ has tractable membership testing, we can assume that $h \in \mathcal{G}$. Similarly, we can assume $|\mathsf{nodes}(h)| \leq |\mathsf{nodes}(g)|$ and $|\mathsf{edges}(h)| \leq |\mathsf{edges}(g)|$ (otherwise $h \not\sqsubseteq g$). We may also assume that h and g are not isomorphic (otherwise, $h \sqsubseteq g$). Note that h and g are isomorphic if and only if $g \in \mathcal{P}$ and $\kappa_{\mathcal{P}}(h) = \kappa_{\mathcal{P}}(g)$, hence we can test in polynomial time whether h and g are not isomorphic.

With the prior assumptions, we can test whether $h \sqsubseteq g$ using the following reductions.

—Let $\mathbf{g} = \langle h, g \rangle$. For $h' = \kappa_{\mathcal{P}}(h)$, we have that $h' \in \mathsf{MaxFS}^2[\mathcal{P}, \mathcal{D}](\mathbf{g})$ if and only if $h \sqsubseteq g$.
—Let $\mathbf{g} = \langle h, h, g, g \rangle$. For $h' = \kappa_{\mathcal{P}}(h)$, we have that $h' \in \mathsf{MaxFS}^2[\mathcal{P}, \mathcal{D}](\mathbf{g})$ if and only if $h \not\sqsubseteq g$. Indeed, if $h \sqsubseteq g$ then h' is isomorphic to a strict subgraph of g (since we assume h and g are not isomorphic), and hence h' is not maximal; on the other hand, if $h \not\sqsubseteq g$, then h' is frequent and no strict supergraph thereof is also frequent. □

The property of *tractable verification* captures the tractability properties we listed thus far.

Definition 3.5. A graph mining setting $(\mathcal{P}, \mathcal{D})$ has *tractable verification* if $\mathcal{P}$ and $\mathcal{D}$ have tractable membership testing, $\mathcal{P}$ has tractable canonization, and $\mathcal{P}$ has tractable subgraph isomorphism in $\mathcal{D}$.

The following proposition lists the combinations of classes from Table II that yield graph mining settings with tractable verification. Note that these are exactly the pairs in the table that feature 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. *$(\mathcal{P}, \mathcal{D})$ has tractable verification in the following cases.*

—*Both $\mathcal{P}$ and $\mathcal{D}$ are labeled variants of the classes in Table II.*
—*$\mathcal{D}$ is* $\mathbf{T}$.
—*$\mathcal{D}$ is* $\mathbf{BDG}^2$.
—*$\mathcal{P} = \mathbf{BDG}^d$ and $\mathcal{D} = \mathbf{BTW}^w$ for some natural numbers d and w.*

Proposition 3.6 is true because of the aforementioned known results on canonization and subgraph isomorphism.

The following result states that, in the case in which $\mathcal{P}$ is connected and closed under connected subgraphs, if $(\mathcal{P}, \mathcal{D})$ has tractable verification then $\mathsf{MaxFS}[\mathcal{P}, \mathcal{D}]$ and $\mathsf{MaxFS}^\tau[\mathcal{P}, \mathcal{D}]$ are NP-relations.

PROPOSITION 3.7. *Let $(\mathcal{P}, \mathcal{D})$ be a graph mining setting such that $\mathcal{P}$ is connected and closed under connected subgraphs, and let τ be a natural number. If $(\mathcal{P}, \mathcal{D})$ has tractable verification, then both* $\mathsf{MaxFS}[\mathcal{P}, \mathcal{D}]$ *and* $\mathsf{MaxFS}^\tau[\mathcal{P}, \mathcal{D}]$ *are* NP*-relations.*

PROOF. Consider the input (x, y) for $\mathsf{MaxFS}[\mathcal{P}, \mathcal{D}]$. We first need to verify that $x = \langle \mathbf{g}, \tau \rangle$, for some sequence $\mathbf{g}$ of graphs in $\mathcal{D}$, and for this we use the assumption that $\mathcal{D}$ has tractable membership testing. Next, we need to verify that $y = \kappa_{\mathcal{D}}(h)$ for some graph h in $\mathcal{P}$, and for this we use the assumption that $\mathcal{P}$ has tractable membership testing and tractable canonization. Next, we need to verify that h is a τ-frequent subgraph $\mathbf{g}$, and for this we use the assumption that $\mathcal{P}$ has tractable subgraph isomorphism in $\mathcal{D}$. Finally, we need to verify that h is maximal; for this, we use the assumption that $\mathcal{P}$ is connected and closed under connected subgraphs—we verify that no addition of any edge for h results in a frequent subgraph. □

3.2. Self-Reducibility

We complete this section with the following theorem stating that, in the case where both the pattern and data classes are connected and closed under connected subgraphs, tractable verification implies enumeration self-reducibility.

THEOREM 3.8. *Let $(\mathcal{P}, \mathcal{D})$ be a graph mining setting. Suppose $\mathcal{P}$ and $\mathcal{D}$ are connected and closed under connected subgraphs. If $(\mathcal{P}, \mathcal{D})$ has tractable verification, then*

MaxFS$[\mathcal{P}, \mathcal{D}]$ *and* MaxFS$^{\tau}[\mathcal{P}, \mathcal{D}]$ *are both enumeration self-reducible and k-enumeration self-reducible, for all* $\tau, k \in \mathbb{N}$.

PROOF. We will prove that the problem MaxFS$[\mathcal{P}, \mathcal{D}]$ is enumeration self-reducible by designing a polynomial-time algorithm for the problem MaxFS$[\mathcal{P}, \mathcal{D}]$-extend that uses as an oracle solver for MaxFS$[\mathcal{P}, \mathcal{D}]$-extendible. The proofs for the combinations of fixed τ and k are similar and hence omitted. So, for the remainder of the proof, we fix an input $\langle \mathbf{g}, \tau \rangle$ and $h_1, \ldots, h_k$ for MaxFS$[\mathcal{P}, \mathcal{D}]$-extend, where $h_1, \ldots, h_k$ are maximal frequent subgraphs. We need to either produce a maximal frequent subgraph h such that $h \not\sqsubseteq h_i$ for all $i = 1, \ldots, k$, and then return $\kappa_{\mathcal{P}}(h)$, or determine that no such h exists. For brevity, if such h exists, then we will say that $(\mathbf{g}, \tau, h_1, \ldots, h_k)$ is *extendible*. Of course, using our oracle for a solver of MaxFS$[\mathcal{P}, \mathcal{D}]$-extendible, we can efficiently test whether $(\mathbf{g}, \tau, h_1, \ldots, h_k)$ is extendible, and so we will assume that this is indeed the case.

Our algorithm has two main steps. In the first step, we construct a frequent subgraph $h' \in \mathcal{P}$ that is not necessarily maximal, but is not isomorphic to any subgraph of any h_i, $1 \leq i \leq k$. We call h' an *intermediate solution*. In the second step, we extend h' to a maximal frequent subgraph h and return $\kappa_{\mathcal{P}}(h)$. Recall that $\mathcal{P}$ has tractable canonization, hence $\kappa_{\mathcal{P}}(h)$ can be created in polynomial time once h has been constructed in polynomial time. So, it remains to show how to construct h', and then how to construct h.

Construction of h' Assume $\mathbf{g}$ is the sequence $\langle g_1, \ldots, g_n \rangle$. We say that a graph h' is a *trivial solution* if there are τ distinct indices $i_1, \ldots, i_\tau$, such that h' is isomorphic to each of the g_{i_j} and, moreover, h' is not isomorphic to any subgraph of any one of the h_is. Note that if h' is a trivial solution, then h' is also an intermediate solution. Note also that a trivial solution exists precisely when there is some $m \leq n$ such that g_m is a frequent subgraph and, for every $i \leq k$, we have that g_m is not isomorphic to any subgraph of h_i. Furthermore, if a trivial solution exists, then one can be found in polynomial time (because $\mathcal{P}$ has tractable subgraph isomorphism in $\mathcal{D}$). We say that $(\mathbf{g}, \tau, h_1, \ldots, h_k)$ is *immediately reducible* if, for some $j \leq n$, the graph g_j has a subgraph g'_j with the property that g'_j is obtained from g_j by removing an edge (and possibly a resulting isolated node), the sequence $(\mathbf{g}', \tau, h_1, \ldots, h_k)$ is extendible, where $\mathbf{g}'$ is the sequence of graphs obtained from $\mathbf{g}$ by replacing g_j with g'_j; in this case, we say that $\mathbf{g}'$ is an *immediate reduction of* $\mathbf{g}$.

An easy observation is that at least one of the following statements is necessarily true (given our assumption that $(\mathbf{g}, \tau, h_1, \ldots, h_k)$ is extendible).

—$(\mathbf{g}, \tau, h_1, \ldots, h_k)$ has a trivial solution.
—$(\mathbf{g}, \tau, h_1, \ldots, h_k)$ is immediately reducible.

This observation gives rise to the following algorithm for finding the desired g. If $(\mathbf{g}, \tau, h_1, \ldots, h_k)$ has a trivial solution, find such a trivial solution g as described before. Otherwise, the observation implies that $(\mathbf{g}, \tau, h_1, \ldots, h_k)$ is immediately reducible. So, we apply an immediate reduction of $\mathbf{g}$ and repeat the process (in a loop). This process must terminate after a number of iterations that is linear in the size of $\mathbf{g}$, and produces an intermediate solution h'. Consequently, we obtain the sought h' in polynomial time.

Construction of h It remains to show how h' is extended to a maximal frequent subgraph h. We simply extend h' incrementally, where in each step we extend h' with a single new edge and test whether each such extension h'' is a frequent subgraph. If so, we take h'' as our new h'. Once h' cannot be further extended, we take it as the sought h. □

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

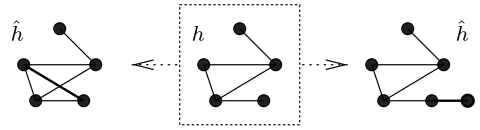

Fig. 2. Immediate extensions $\hat{h}$ of a graph h.

COROLLARY 3.9. *Let $(\mathcal{P}, \mathcal{D})$ be one of the graph mining settings listed in Proposition 3.6. For every τ and for every k, both* MaxFS$[\mathcal{P}, \mathcal{D}]$ *and* MaxFS$^{\tau}[\mathcal{P}, \mathcal{D}]$ *are* NP*-relations, enumeration self-reducible, and k-enumeration self-reducible.*

3.3. Generic Algorithm

In this section we describe a generic algorithm, GENERICMAXFS($\mathbf{g}, \tau$), for solving MaxFS$[\mathcal{P}, \mathcal{D}]$-enumerate in the case where $\mathcal{P}$ is connected and closed under connected subgraphs. This algorithm reduces the problem to a restricted variant that we describe later in this section. In Section 4, we will use this algorithm to establish polynomial-delay upper bounds in some specific cases by showing an upper bound (polynomial time) on the complexity of the restricted variant. Beyond its role as a proof technique, we believe this algorithm can be used as the basis of practical solutions for MaxFS$[\mathcal{P}, \mathcal{D}]$-enumerate. The algorithm is an adaptation of that of Cohen et al. [2006] for computing the full disjunction of database relations.

Figure 3 depicts the algorithm. Before we describe the algorithm itself, we explain the notation and subroutines in use. We denote by $FS(\mathbf{g}, \tau)$ the set of all the τ-frequent subgraphs $f \in \mathcal{P}$. By subgraphs(f) we denote the set of all the subgraphs of f. The subroutine MAXIMIZEFS$(f, \mathbf{g}, \tau)$ takes as input a graph $f \in FS(\mathbf{g}, \tau)$, the sequence $\mathbf{g}$, and the threshold τ, and returns as output an arbitrary canonized $\mathcal{P}$-maximal τ-frequent subgraph h of $\mathbf{g}$ such that $f \sqsubseteq h$. If $h \in \mathcal{P}$ is a graph, then an *immediate extension* of h is a graph $\hat{h}$ that is obtained from h by either adding a new node, adding a new node with an edge to/from an existing node, or adding an edge between two existing nodes. See Figure 2 for an illustration. When adding new nodes, we always use labels from $\mathbf{g}$.

The algorithm first solves the trivial case where $\mathbf{g}$ has fewer than τ graphs; in this case, there are no frequent subgraphs and therefore the algorithm terminates. So, assume this is not the case.

The algorithm maintains two sets, Q and R, that are initially empty. When generated, each maximal frequent subgraph is inserted into Q. During the execution of the algorithm, graphs are removed from Q, then printed, and then added into R. We begin by adding into Q an arbitrary maximal frequent subgraph (line 4) by calling MAXIMIZEFS$(f, \mathbf{g}, \tau)$ with f being the empty graph (denoted as $\emptyset$). In the main loop (lines 5–13), we remove from Q a graph h and print it. Then, we consider every immediate extension h^+ of h, and enumerate all the maximal graphs $\hat{h}$ among the frequent subgraphs that are also subgraphs of h^+. Note that, in the figure, the notation max(G) denotes the maximal graphs, under subgraph isomorphism, of the set G of graphs. We extend each such $\hat{h}$ into a maximal frequent subgraph h', and insert h' into Q if we have not generated h' already (i.e., h' is in neither Q nor R). The correctness of the algorithm is stated in the following lemma.

LEMMA 3.10. *Let $(\mathcal{P}, \mathcal{D})$ be a graph mining setting, and suppose that $\mathcal{P}$ is connected and closed under connected subgraphs. The following hold in the execution of* GENERICMAXFS($\mathbf{g}, \tau$).

(*1*) *Every printed graph is a canonized maximal frequent subgraph.*
(*2*) *Every canonized maximal frequent subgraph is printed.*
(*3*) *No graph is printed more than once.*

Algorithm GENERICMAXFS(g, τ)

```
1: if g contains fewer than τ graphs then
2:    terminate
3: initialize empty sets Q and R
4: Q.add(MAXIMIZEFS(∅, g, τ))
5: while Q ≠ ∅ do
6:    h ← Q.pop()
7:    print h
8:    R.add(h)
9:    for all immediate extensions h⁺ of h do
10:      for all ĥ ∈ max(FS(g, τ) ∩ subgraphs(h⁺)) do
11:         h′ ← MAXIMIZEFS(ĥ, g, τ)
12:         if h′ ∉ Q ∪ R then
13:            Q.add(h′)
```

Fig. 3. A generic algorithm for MaxFS$[\mathcal{P}, \mathcal{D}]$-enumerate where $\mathcal{P}$ and $\mathcal{D}$ are connected and closed under connected subgraphs.

PROOF. We assume the condition tested in line 1 is false (i.e., $\mathbf{g}$ has at least τ graphs). We first prove Part 1. We need to show that every graph inserted into Q is a maximal frequent subgraph. From the definition of the subroutine MAXIMIZEFS$(f, \mathbf{g}, \tau)$ we have that the return value of this subroutine is a canonized maximal frequent subgraph whenever f is a frequent subgraph. So, we need to prove that the argument f in lines 4 and 11 is a frequent subgraph. Since $\mathcal{P}$ is closed under connected subgraphs, the empty graph is in $\mathcal{P}$ and, of course, frequent. As for line 11, the fact that $\hat{h}$ is a frequent subgraph is due to the definition of $\hat{h}$ in line 10 (i.e., $f \in \mathit{FS}(\mathbf{g}, \tau)$).

We now prove Part 2. Suppose by way of contradiction that there is a canonized maximal frequent subgraph not printed by the algorithm. Let h_0 be such a graph. Let h_0' be a maximal connected subgraph of h_0, such that $h_0' \sqsubseteq h$ for some graph h that the algorithm prints. Note that $h_0' \subsetneq h_0$ (since h_0' is not isomorphic to h_0) and $h_0' \subsetneq h$ (since h_0' is not maximal). Since h_0 is connected (as $\mathcal{P}$ is connected), we conclude there is an immediate extension h^+ of h with a connected subgraph $\hat{h}_0$, such that $h_0' \subsetneq \hat{h}_0 \sqsubseteq h_0$. From the fact that $\hat{h}_0 \sqsubseteq h_0$ and that $\mathcal{P}$ is closed under connected subgraphs, we conclude $\hat{h}_0$ is a frequent subgraph. Consequently, in the execution of line 10 when h and h^+ are selected, $\hat{h}_0$ is in $\mathit{FS}(\mathbf{g}, \tau) \cap \mathrm{subgraphs}(h^+)$. Hence, some graph $\hat{h}$ generated in line 10 is such that $\hat{h}_0 \sqsubseteq \hat{h}$, and then some maximal frequent subgraph h' that is generated in line 11 is such that $\hat{h}_0 \sqsubseteq h'$. Note that the algorithm does print h'. But then we get a contradiction to our assumption that h_0' is a maximal connected subgraph of h_0 that is printed (up to isomorphism) as part of another graph.

Finally, Part 3 is true due to the test of line 12, namely it guarantees that every graph inserted into Q has never been generated previously during the execution. □

3.3.1. Complexity. Let $(\mathcal{P}, \mathcal{D})$ be a graph mining setting. The *restricted variant* of MaxFS$[\mathcal{P}, \mathcal{D}]$-enumerate is the following problem. Given $\mathbf{g}$, τ and an immediate extension $\hat{h}$ of some maximal frequent subgraph h, enumerate $\max(\mathit{FS}(\mathbf{g}, \tau) \cap \mathrm{subgraphs}(\hat{h}))$.

THEOREM 3.11. *Let $(\mathcal{P}, \mathcal{D})$ be a graph mining setting. Suppose $\mathcal{P}$ is connected and closed under connected subgraphs and that $(\mathcal{P}, \mathcal{D})$ has tractable verification. The problem* MaxFS$[\mathcal{P}, \mathcal{D}]$-enumerate *is solvable with polynomial delay whenever its restricted variant is solvable in polynomial time.*

Table III. Complexity Results for Labeled Graph Classes

$\mathcal{P}$ \ $\mathcal{D}$	$\mathbf{BDG}_{\mathsf{L}}^{2}$	$\mathbf{BDG}_{\mathsf{L}}^{d>2}$	$\mathbf{T}_{\mathsf{L}}$	$\mathbf{BTW}_{\mathsf{L}}^{w>1}$	$\mathbf{PLN}_{\mathsf{L}}$	$\mathbf{G}_{\mathsf{L}}$
$\mathbf{BDG}_{\mathsf{L}}^{2}$	FP / FP / FP	PD / FP / PD	FP / FP / FP	PD / FP / PD	PD / FP / PD	PD / FP / PD
$\mathbf{BDG}_{\mathsf{L}}^{d'>2}$	FP / FP / FP	NPc / NPc / PD	NPc / FP / PD	NPc / NPc / PD	NPc / NPc / PD	NPc / NPc / PD
$\mathbf{T}_{\mathsf{L}}$	FP / FP / FP	NPc / FP / PD	NPc / FP / FP	NPc / FP / PD	NPc / FP / PD	NPc / FP / PD
$\mathbf{BTW}_{\mathsf{L}}^{w'>1}$	FP / FP / FP	NPc / NPc	NPc / FP / FP	NPc / NPc	NPc / NPc	NPc / NPc
$\mathbf{PLN}_{\mathsf{L}}$	FP / FP / FP	NPc / NPc	NPc / FP / FP	NPc / NPc	NPc / NPc / FP	NPc / NPc
$\mathbf{G}_{\mathsf{L}}$	FP / FP / FP	NPc / NPc / FP	NPc / FP / FP	NPc / NPc / FP	NPc / NPc / FP	NPc / NPc / FP

See Table I for the meaning of the complexity classes. In a triple "c_1 / c_2 / c_3," the symbol c_1 refers to the unbounded variant, c_2 assumes that the number of answers is bounded, and c_3 assumes a bounded threshold. In some cases c_3 is unknown, and then only "c_1 / c_2" is specified.

PROOF. The proof is a straightforward combination of the following observations. First, each of Q and R can store at most exponentially many items (since there are at most exponentially many subgraphs of graphs in $\mathbf{g}$). Using standard data structures, the operations on Q and R (pop in line 6, insertions in lines 8 and 13, and membership testing in line 12) can be done in logarithmic time in the number of elements, hence polynomial time in the input size. Second, as mentioned in the proof of Theorem 3.8, the assumptions on $(\mathcal{P}, \mathcal{D})$ imply MAXIMIZEFS$(f, \mathbf{g}, \tau)$ can be implemented so that it takes polynomial time. Finally, there is at most a polynomial number of immediate extensions of a maximal frequent subgraph h. □

4. LABELED GRAPH CLASSES

Recall that our goal is to study the complexity of enumerating the maximal frequent subgraphs for graph mining settings $(\mathcal{P}, \mathcal{D})$. Furthermore, we would like to investigate the effect on complexity of various parameters:

—the class $\mathcal{P}$;
—the class $\mathcal{D}$;
—whether or not a bounded number of answers suffices;
—whether or not the threshold is bounded.

In this section, we focus on the case where $\mathcal{P}$ and $\mathcal{D}$ are among the labeled classes of Table II (recall from Proposition 3.6 that every graph mining setting $(\mathcal{P}, \mathcal{D})$ considered here has tractable verification). We do not consider here every possible combination of parameters, but rather study a few that we view as important and/or exposing fundamental insights on the effect of the parameters. In our study, we covered almost all of these combinations; the results are shown in Table III (see the caption for an explanation of the notation). Some of the cases not covered in this section are simple consequences of the results of this section. Other results are proved in Section 4.4, while the remaining few cases are left as open problems.

4.1. The Unbounded Case

In this section we give complexity results that make no assumptions on the threshold τ or the number k of results. We begin with a general hardness result (Theorem 4.2) and then show a special case where a polynomial-delay algorithm exists (Theorem 4.5).

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 not properly contained in any other frequent subset.

The following result by Boros et al. [2003] will be of interest to us.

THEOREM 4.1 [BOROS ET AL. 2003]. MaxFIS-extendible *is* NP*-complete.*

One of the consequences of this result is that most of the graph mining settings obtained from the labeled classes of Table II have intractable enumeration.

THEOREM 4.2. *Let $\mathcal{P}$ and $\mathcal{D}$ be any two classes among $\mathbf{G}_{\mathsf{L}}$, $\mathbf{T}_{\mathsf{L}}$, $\mathbf{PLN}_{\mathsf{L}}$, $\mathbf{BDG}^d_{\mathsf{L}}$ and $d \geq 3$, or $\mathbf{BTW}^w_{\mathsf{L}}$ with $w \geq 1$. Then the problem* MaxFS$[\mathcal{P}, \mathcal{D}]$-extendible *is* NP*-complete.*

PROOF. Membership in NP is an immediate consequence of the fact that all of the graph mining settings $(\mathcal{P}, \mathcal{D})$ referred to in the theorem have tractable verification, as stated in Proposition 3.6. To prove hardness, we give a reduction from MaxFIS-extendible. Consider the input $\langle \mathbf{s}, \tau', Y' \rangle$ for MaxFIS-extendible, where Y' is a set of maximal frequent subsets. Suppose that $\mathbf{s} = \langle s_1, \ldots, s_n \rangle$ and that the union of the sets in $\mathbf{s}$ consists of the elements $a_1, \ldots, a_m$. We construct in polynomial time a labeled binary tree g that has n leaves labeled $a_1, \ldots, a_m$. Observe that g is in each of the classes mentioned in the corollary. Now, for every set $s \subseteq \{a_1, \ldots, a_m\}$, we consider the tree g_s obtained from g by removing all leaves except for those labeled by some element a_i in s. Our input for MaxFS$[\mathcal{P}, \mathcal{D}]$-extendible will then be $(\mathbf{g}, \tau, Y)$, where $\mathbf{g} = \langle g_{s_1}, \ldots, g_{s_n} \rangle$, $\tau = \tau'$, and $Y = \{g_s \mid s \in Y'\}$. Observe that every maximal frequent subgraph contains the graph g' obtained from g by removing all the leaves. Consequently, the mapping $s \mapsto g_s$ is a bijective mapping from the maximal frequent itemsets of $\mathbf{s}$ to the maximal frequent graphs of $\mathbf{g}$. □

After Theorem 4.2, the only remaining cases (for unbounded enumeration with labeled graphs) are those where either $\mathcal{P}$ or $\mathcal{D}$ is $\mathbf{BDG}^d_{\mathsf{L}}$ for $d \leq 2$. Note that $\mathbf{BDG}^2_{\mathsf{L}}$ is the class of all simple labeled paths and cycles. We cover these cases by showing how the generic algorithm of Section 3.3 can be used to obtain a tractability result in the case where $\mathcal{P} = \mathbf{BDG}^2_{\mathsf{L}}$ and $\mathcal{G} = \mathbf{G}_{\mathsf{L}}$. Specifically, we will apply Theorem 3.11 by proving the following lemma.

LEMMA 4.3. *The restricted variant of* MaxFS$[\mathbf{BDG}^2_{\mathsf{L}}, \mathbf{G}_{\mathsf{L}}]$-enumerate *is solvable in polynomial time.*

PROOF. The lemma is straightforward from the fact that a graph h in $\mathbf{BDG}^2_{\mathsf{L}}$ with n nodes has $O(n^2)$ connected subgraphs, and so does every immediate extension $\hat{h}$ of h. □

By combining Theorem 3.11 with Lemma 4.3, we get the following theorem.

THEOREM 4.4. MaxFS$[\mathbf{BDG}^2_{\mathsf{L}}, \mathbf{G}_{\mathsf{L}}]$-enumerate *is solvable with polynomial delay.*

From Theorem 4.4 we get immediately the following theorem.

THEOREM 4.5. *If $\mathcal{P}$ and $\mathcal{D}$ are labeled classes in Table II and either $\mathcal{P}$ or $\mathcal{D}$ is $\mathbf{BDG}^2_{\mathsf{L}}$, then* MaxFS$[\mathcal{P}, \mathcal{D}]$-enumerate *is solvable with polynomial delay.*

4.2. Bounded Number of Witnesses

In this section, we consider the effect of bounding the number k of witnesses of interest. We will assume $k > 2$ (in Section 4.4 we show a general tractability result, applying to all the combinations of labeled $\mathcal{P}$ and $\mathcal{D}$ in Table II, for $k \leq 2$). We show two main results. The first is a hardness result that applies to most of the settings $(\mathcal{P}, \mathcal{D})$ we consider here (Theorem 4.6). The second result is a polynomial-time upper bound for the case where either $\mathcal{P}$ or $\mathcal{D}$ is the class $\mathbf{T}_{\mathsf{L}}$ of all labeled trees (Corollary 4.9) or a subclass of $\mathbf{T}_{\mathsf{L}}$ (Theorem 4.8). In conjunction with Theorem 4.2, we conclude that bounding the number of witnesses does not make the problem tractable in most of the cases of consideration, but does in the case where the patterns are trees.

We begin with the hardness result. In what follows we assert that the problem $\mathsf{MaxFS}[\mathcal{P}, \mathcal{D}]\text{-}\mathsf{extendible}\langle k\rangle$ is NP-hard when $\mathcal{P}$ and $\mathcal{D}$ are among the labeled variants of the classes in Table II, except for the classes $\mathbf{BDG}^d_{\mathsf{L}}$ with $d \leq 2$ (that was covered in Theorem 4.4) and $\mathbf{T}_{\mathsf{L}}$ (equivalently, $\mathbf{BTW}^1_{\mathsf{L}}$), that we discuss later.

THEOREM 4.6. *Let $\mathcal{P}$ and $\mathcal{D}$ be among $\mathbf{G}_{\mathsf{L}}$, $\mathbf{PLN}_{\mathsf{L}}$, $\mathbf{BDG}^d_{\mathsf{L}}$ with $d > 2$, or $\mathbf{BTW}^w_{\mathsf{L}}$ with $w > 1$. Then, for every $k > 2$, the problem $\mathsf{MaxFS}[\mathcal{P}, \mathcal{D}]\text{-}\mathsf{extendible}\langle k\rangle$ 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 $(\mathcal{P}, \mathcal{D})$ has tractable verification (Proposition 3.6). 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 f has no isolated nodes. We construct an input $\mathbf{g}$, τ, and Y to $\mathsf{MaxFS}[\mathcal{P}, \mathcal{D}]\text{-}\mathsf{extendible}\langle 3\rangle$, as follows.

Suppose $\{1, \ldots, N\}$ is the node set of f. For each $i = 1, \ldots, N$, 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 $i = 1, \ldots, N$, and $\{b_i, a_{i+1}\}$, for all $i = 1, \ldots, N-1$.

Our construction is illustrated with a concrete example in Figure 4. 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 4 show h_1 and h_2, respectively.

Next, for $i = 1, \ldots, N$, 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 4 depict the graphs g^1 and g^4 in our example.

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

—$\mathbf{g} = g'_1, \ldots, g'_m, g''_1, \ldots, g''_m, g^1, \ldots, g^N$, 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 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.

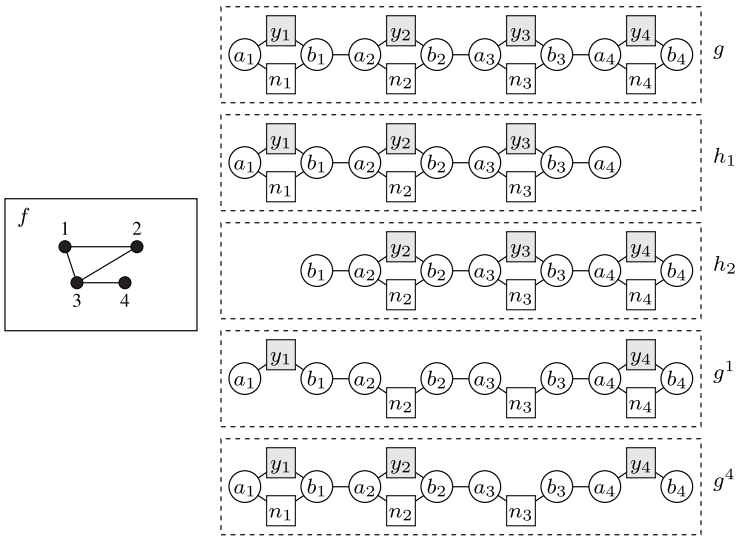

Fig. 4. Reducing maximum independent set to MaxFS$[\mathcal{P}, \mathcal{D}]$-extendible$\langle 2\rangle$.

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 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 $\mathbf{G}_{\mathsf{L}}$, $\mathbf{PLN}_{\mathsf{L}}$, $\mathbf{BDG}^d_{\mathsf{L}}$ with $d > 2$, 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 [Downey and Fellows 1999]. □

We are left with the case where either $\mathcal{P}$ or $\mathcal{D}$ is the class $\mathbf{T}_{\mathsf{L}}$ of all trees. We will show that, unlike the previous studied case, bounding the number k of witnesses yields tractability of MaxFS$[\mathcal{P}, \mathcal{D}]$-extendible$\langle k\rangle$. We will use the following lemma.

LEMMA 4.7. *Let $(\mathcal{P}, \mathcal{D})$ be a labeled graph mining setting. Suppose that $\mathcal{P}$ is connected and closed under connected subgraphs and that either $\mathcal{P}$ or $\mathcal{D}$ is a subclass of*

$\mathbf{T}_{\mathsf{L}}$. *Let* $(\mathbf{g}, \tau, Y)$ *be an input instance for the problem* MaxFS$[\mathcal{P}, \mathcal{D}]$-extendible$\langle k \rangle$. *Let* Z *be the set of all the canonized frequent subgraphs that are subgraph isomorphic to none of the graphs in* Y. *The following are equivalent.*

(*1*) $Z \neq \emptyset$ *(hence* $(\mathbf{g}, \tau, Y)$ *is a "yes" instance).*
(*2*) *There are graphs* $h \in Y$ *and* $h' \subseteq h$, *such that* h' *is a tree with at most* $k+1$ *leaves, and some immediate extension of* h' *is in* Z.

Proof. The direction $2 \Rightarrow 1$ is immediate, so we will prove the direction $1 \Rightarrow 2$. Let g be a graph in Z. The assumptions imply the g, as well as every frequent subgraph, is a tree. Since $(\mathcal{P}, \mathcal{D})$ is labeled, for every tree $f \in Y$ the tree g has to contain at least one item that is not in f, where by *item* we mean either a label or a pair of labels connected by an edge (otherwise, $g \sqsubseteq f$ for some $f \in Y$). Pick such an item a_f for each $f \in Y$, and let g_m be the smallest subtree of g that contains all the a_f. An easy case analysis can show that, since there are at most $|Y|$ such a_f and g_m is smallest, then g' has at most $k+1$ leaves. Now, let g' be the largest subtree of g_m such that $g' \sqsubseteq h$ for some $h \in Y$, and let $h \in Y$ be such that $g' \sqsubseteq h$. We pick as h' the subtree of h that is isomorphic to g'. Observe that g' has at most $k+1$ leaves (since it is a subtree of a tree with at most $k+1$ leaves), and so does h'. Also observe that $h' \subsetneq g$, and so there is an immediate extension $\hat{h}$ of h' that satisfies $\hat{h} \sqsubseteq g$. The facts that g is a frequent subgraph and $\mathcal{P}$ is closed under connected subgraphs imply that $\hat{h}$ is a frequent graph. And the fact that g' is largest implies that $\hat{h}$ is in Z. □

Using Lemma 4.7, we get the following theorem.

Theorem 4.8. *Let* $(\mathcal{P}, \mathcal{D})$ *be a labeled graph mining setting with tractable verification, and let* k *be a natural number. Suppose that* $\mathcal{P}$ *is connected and closed under connected subgraphs and that either* $\mathcal{P}$ *or* $\mathcal{D}$ *is a subclass of* $\mathbf{T}_{\mathsf{L}}$. *Then* MaxFS$[\mathcal{P}, \mathcal{D}]$-extendible$\langle k \rangle$ *is solvable in polynomial time.*

Proof. By Lemma 4.7, it suffices to look at every subgraph h' of a graph h in Y, such that h' has at most $k+1$ leaves, and then consider as candidate every immediate extension of h'. To complete the proof, we need to show that we can enumerate all those h' in polynomial time (and in particular that there are at most a polynomial number of such h'). To do so, observe that every $h \in Y$ is a tree. Therefore, there is a one-to-one correspondence between the subtrees h' of h and the sets of leaves of h'. In particular, the number of such h' is bounded by the number of subsets of size $k+1$ of the nodes of h, which is polynomial if k is bounded. □

Corollary 4.9. *Let* k *be a natural number. If* $\mathcal{P}$ *and* $\mathcal{D}$ *are among the labeled classes of Table II and either* $\mathcal{P}$ *or* $\mathcal{D}$ *is* $\mathbf{T}_{\mathsf{L}}$, *then* MaxFS$[\mathcal{P}, \mathcal{D}]$-extendible$\langle k \rangle$ *is solvable in polynomial time.*

4.3. Bounded Threshold

In this section, we consider the complexity of MaxFS$^{\tau}[\mathcal{P}, \mathcal{D}]$-enumerate where τ is fixed. We will show the assumption of a fixed τ makes the problem tractable in quite a few cases where, in the absence of the assumption, the problem is hard (as stated in Theorem 4.2). We will give two tractability results. The first is a general result for the case where $\mathcal{P}$ is a subclass of $\mathcal{D}$ (Theorem 4.10). The second result applies to the case where $\mathcal{P}$ is the class $\mathbf{T}_{\mathsf{L}}$ of all labeled trees (Theorem 4.13). In Section 4.4 we also consider the case where $\mathcal{P}$ is the class $\mathbf{BDG}_{\mathsf{L}}^{d}$ (for some d). We begin with the first result.

Theorem 4.10. *Let* $(\mathcal{P}, \mathcal{D})$ *be a labeled graph mining setting with tractable verification. Suppose* $\mathcal{P}$ *is connected and closed under connected subgraphs and that* $\mathcal{D} \subseteq \mathcal{P}$. *For*

every natural number τ, the problem MaxFS$^\tau[\mathcal{P}, \mathcal{D}]$-enumerate *is solvable in polynomial time.*

PROOF. In this proof, we identify the nodes of the graphs in $\mathbf{g}$ with their labels. Let $\mathbf{f}$ be a subsequence of $\mathbf{g}$ of length τ. We write $\cap\mathbf{f}$ to denote the graph that consists of the nodes (labels) and edges that are common to all the graphs in $\mathbf{f}$. Although every graph in $\mathbf{g}$ is connected, the graphs in $\cap\mathbf{f}$ may be disconnected. Let $CC(\cap\mathbf{f})$ be the set of graphs induced by the connected components of $\cap\mathbf{f}$. Clearly, every graph in $\cap\mathbf{f}$ is in $\mathcal{P}$ (since every graph in $\mathbf{g}$ is in $\mathcal{P}$, and $\mathcal{P}$ is closed under connected subgraphs) and is a frequent subgraph of $\mathbf{g}$. Also, every maximal frequent subgraph of $\mathbf{g}$ is contained in a graph of $\cap\mathbf{f}$ for some subsequence $\mathbf{f}$ of $\mathbf{g}$, since every graph in $\mathcal{P}$ is connected. It follows that the set of maximal frequent graphs is precisely the set of maximal subgraphs of the set M, where M is the union of $CC(\cap\mathbf{f})$ over all the subsequences $\mathbf{f}$ of $\mathbf{g}$ of length τ. Since τ is fixed, we can iterate in polynomial time through all the subsequences $\mathbf{f}$ of $\mathbf{g}$ of length τ. For each such $\mathbf{f}$, we can produce $CC(\cap\mathbf{f})$ in polynomial time. Hence, M can be computed in polynomial time, and consequently so can the set MaxFS$^\tau[\mathcal{P}, \mathcal{D}](\mathbf{g}, \tau)$. □

Next, we consider additional cases. Let $\mathcal{C}$ be a graph class. The *maximal $\mathcal{C}$-subgraphs* problem is the following: Given a graph g (where g is not necessarily in $\mathcal{C}$), enumerate all the maximal subgraphs g' of g, such that $g' \in \mathcal{C}$. In the *restricted* maximal $\mathcal{C}$-subgraphs problem, we assume g is an immediate extension of a graph $g' \in \mathcal{C}$. We will use the following lemma.

LEMMA 4.11. *Let $(\mathcal{P}, \mathcal{D})$ be a labeled graph mining setting with tractable verification, and let τ be a natural number. Suppose $\mathcal{P}$ is connected and closed under connected subgraphs. If the restricted maximal $\mathcal{P}$-subgraphs problem is solvable in polynomial time, then* MaxFS$^\tau[\mathcal{P}, \mathcal{D}]$-enumerate *is solvable with polynomial delay.*

PROOF. Like in the previous proof, we identify the node of a labeled graph with its label. We will apply Theorem 3.11. For this, we need to show that the restricted variant of MaxFS$^\tau[\mathcal{P}, \mathcal{D}]$-enumerate is solvable in polynomial time. So, let $\hat{h}$ be an immediate extension of a maximal frequent subgraph. We need to present a polynomial-time algorithm for $\max(FS(\mathbf{g}, \tau) \cap \text{subgraphs}(\hat{h}))$. The algorithm is based on the following observation. A graph h is in $\max(FS(\mathbf{g}, \tau) \cap \text{subgraphs}(\hat{h}))$ if and only if h is (isomorphic to) a connected subgraph of an intersection of $\tau + 1$ graphs $g'_1, \ldots, g'_\tau, h'$, where the g'_i are graphs in $\mathbf{g}$ and h' is a maximal $\mathcal{P}$-subgraph of $\hat{h}$. Since τ is fixed and since the maximal $\mathcal{P}$-subgraphs problem is solvable in polynomial time, we can actually enumerate all possible such intersections in polynomial time; to produce $\max(FS(\mathbf{g}, \tau) \cap \text{subgraphs}(\hat{h}))$, we take the maximal connected components. □

To use Lemma 4.11, we will prove the following lemma.

LEMMA 4.12. *The restricted maximal $\mathbf{T}_\mathsf{L}$-subgraphs problem is solvable in polynomial time.*

PROOF. Let h be a labeled tree and let $\hat{h}$ be an immediate extension of h. If $\hat{h}$ is acyclic, then the maximal $\mathcal{P}$-subgraphs of h are precisely the connected components of $\hat{h}$. So suppose that $\hat{h}$ is cyclic. Then it must be the case that $\hat{h}$ is obtained from h by connecting two nodes of h by a new edge. Let $e = \{v_1, v_2\}$ be this edge, and let c be a simple cycle in $\hat{h}$. Then c must include e (since h is acyclic). Moreover, besides e, the nodes v_1 and v_2 are connected through the unique simple path between them in h. It thus follows that the removal of every edge in e results in a tree. And since every subtree of $\hat{h}$ must exclude at least one edge of c, the maximal $\mathcal{P}$-subgraphs of $\hat{h}$ are precisely those trees that are obtained by removing from $\hat{h}$ any edge of c. □

Combining Theorem 3.11 and Lemmas 4.11 and 4.12, we conclude the following.

THEOREM 4.13. *If $\mathcal{D}$ is a labeled class in Table II and τ is a fixed natural number, then* MaxFS$^{\tau}$[$\mathbf{T}_{\mathsf{L}}, \mathcal{D}$]-enumerate *is solvable with polynomial delay.*

In Section 4.4 we also prove a lemma corresponding to Lemma 4.12 for the class $\mathbf{BDG}^{d}_{\mathsf{L}}$, thereby obtaining an additional tractability result.

4.4. Additional Complexity Results

In this section we give additional complexity results for labeled graphs.

LEMMA 4.14. *Let $(\mathcal{P}, \mathcal{D})$ be a labeled graph mining setting. Suppose $\mathcal{P}$ is closed under connected subgraphs and that $(\mathcal{P}, \mathcal{D})$ has tractable verification. For $k \leq 2$, the problem* MaxFS[$\mathcal{P}, \mathcal{D}$]-extendible$\langle k \rangle$ *is solvable in polynomial time.*

PROOF. Recall that in MaxFS[$\mathcal{P}, \mathcal{D}$]-extendible$\langle k \rangle$ the input consists of a sequence $\mathbf{g}$ of graphs in $\mathcal{D}$, a threshold τ, and a set Y of at most $k-1$ maximal frequent subgraphs (in $\mathcal{P}$). 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 that is a frequent subgraph because $\mathcal{P}$ is closed under connected subgraphs), 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 such 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 closed under connected subgraphs, implying that g' is also in $\mathcal{P}$. □

Lemma 4.14 and Proposition 3.6 yield the following result.

THEOREM 4.15. *Let $\mathcal{P}$ and $\mathcal{D}$ be any labeled variants of classes in Table II. For $k \leq 2$,* MaxFS[$\mathcal{P}, \mathcal{D}$]-extendible$\langle k \rangle$ *is solvable in polynomial time.*

LEMMA 4.16. *For all natural numbers d, the restricted maximal $\mathbf{BDG}^{d}_{\mathsf{L}}$-subgraphs problem is solvable in polynomial time.*

PROOF. Let h be a graph in $\mathcal{P}$, and let $\hat{h}$ be an immediate extension of h. If $\hat{h}$ has bounded degree d, then the maximal $\mathcal{P}$-subgraphs of $\hat{h}$ are precisely the connected components of $\hat{h}$. So suppose that $\hat{h}$ has at least one node of degree higher than d. Since $\hat{h}$ is an immediate extension of h, all the nodes of $\hat{h}$ have degree bounded by d, except for, possibly, one or two nodes with degree $d+1$. We will consider the case where two nodes v_1 and v_2 have degree $d+1$; the case of a single such node is handled by a similar argument. Every $\mathcal{P}$-subgraph of $\hat{h}$ excludes at least one edge that is incident to v_1 and at least one edge incident to v_2. Furthermore, every such exclusion of edges results in a subgraph of bounded degree d. Consequently, to find the maximal $\mathcal{P}$-subgraphs of $\hat{h}$, it suffices to take maximal among the connected components of all the graphs obtained by removing some edges e_1 and e_2 that are incident to v_1 and v_2, respectively. □

Combining Theorem 3.11 and Lemmas 4.11 and 4.16, we conclude the following.

THEOREM 4.17. *If $\mathcal{D}$ is a labeled class in Table II and τ and d are fixed natural numbers, then* MaxFS$^{\tau}$[$\mathbf{BDG}^{d}_{\mathsf{L}}, \mathcal{D}$]-enumerate *is solvable with polynomial delay.*

5. UNLABELED GRAPH CLASSES

In this section, we study the complexity of enumerating the maximal frequent subgraphs for the unlabeled classes in Table II. We refer the reader to Proposition 3.6, where we list the graph mining settings $(\mathcal{P}, \mathcal{D})$ that have tractable verification. The

Table IV. Complexity Results for Unlabeled Graph Classes

$\mathcal{P}$ \ $\mathcal{D}$	$\mathbf{BDG}^2$	$\mathbf{T}$	$\mathbf{BTW}^{w>1}$
$\mathbf{BDG}^2$	FP	FP	FP
$\mathbf{BDG}^{d>2}$	FP	NPc	NPc
$\mathbf{BTW}^{w>0}$, $\mathbf{PLN}$, $\mathbf{G}$	FP	NPc	

See Table I for the meaning of the complexity classes. The upper bound FP applies to MaxFS$[\mathcal{P}, \mathcal{D}]$-enumerate and makes no assumption on τ and k, whereas the lower bound NPc (NP-completeness) applies to MaxFS$[\mathcal{P}, \mathcal{D}]$-extendible and is true for all $\tau > 1$ and $k > 1$. The gray cell is for cases where $(\mathcal{P}, \mathcal{D})$ does not have tractable verification (unless $\mathcal{P} = \text{NP}$).

complexity results we establish in this section are listed in Table IV. These results are obtained by two claims that we prove in this section. The first is a polynomial-time upper bound in a special case where the pattern class $\mathcal{P}$ is $\mathbf{BDG}^2$ (Proposition 5.1). The second claim covers the intractability results by proving the NP-hardness of a basic graph problem (Theorem 5.2).

We begin with a case with simple analysis, namely where $\mathcal{P}$ is $\mathbf{BDG}^2$. Among the unlabeled classes $\mathcal{D}$ in Table II and assuming NP $\neq$ P, the only classes such that $\mathcal{P}$ has tractable subgraph isomorphism in $\mathcal{D}$ are $\mathbf{BDG}^2$ and $\mathbf{BTW}^k$, with $\mathbf{T}$ as a special case. Observe that we can generate, in polynomial time, the set of all possible graphs in $\mathbf{BDG}^2$ (up to isomorphism) with a number of nodes bounded by the maximal number of nodes in the data graphs. Consequently, we get the following proposition.

PROPOSITION 5.1. *If $\mathcal{P}$ is $\mathbf{BDG}^2$ and $\mathcal{D}$ is one of $\mathbf{BDG}^2$ and $\mathbf{BTW}^k$, then* MaxFS$[\mathcal{P}, \mathcal{D}]$-enumerate *is solvable in polynomial time.*

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

THEOREM 5.2. *The following is an* NP*-complete problem: Given two unlabeled trees of t_1 and t_2, decide whether they have more than one maximal subtree in common (i.e., whether the set* MaxFS$^2[\mathbf{T}, \mathbf{T}](t_1, t_2)$ *has at least two members). The problem remains* NP*-hard even if t_1 and t_2 are assumed to be of degree bounded by* 3.

Observe that obtaining one maximal frequent subgraph is in polynomial time for the graph mining settings mentioned in Proposition 3.6. Consequently, the decision problem of Theorem 5.2 reduces in polynomial time to the extendibility problem with $k = 2$. As a result, we obtain the following corollary of Theorem 5.2.

COROLLARY 5.3. *If $\mathcal{P}$ and $\mathcal{D}$ are in $\{\mathbf{T}, \mathbf{BDG}^3\}$ and $\tau \geq 2$ is a fixed threshold, then* MaxFS$^\tau[\mathcal{P}, \mathcal{D}]$-extendible$\langle 2 \rangle$ *is* NP*-complete.*

PROOF. Membership in NP is due to the fact that $(\mathcal{P}, \mathcal{D})$ has tractable verification, and each of $\mathcal{P}$ and $\mathcal{D}$ is closed under connected subgraphs. Theorem 5.2 establishes NP-hardness in the case of $\tau = 2$. 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. □

5.1. Proof of Theorem 5.2

In this section we give the proof of Theorem 5.2. The proof is by a reduction from CNF-satisfiability; in what follows, we will show how to construct, given a CNF-formula, the trees t_1 and t_2 of Theorem 5.2; we will also prove the correctness of the reduction. The construction is illustrated in Figure 5.

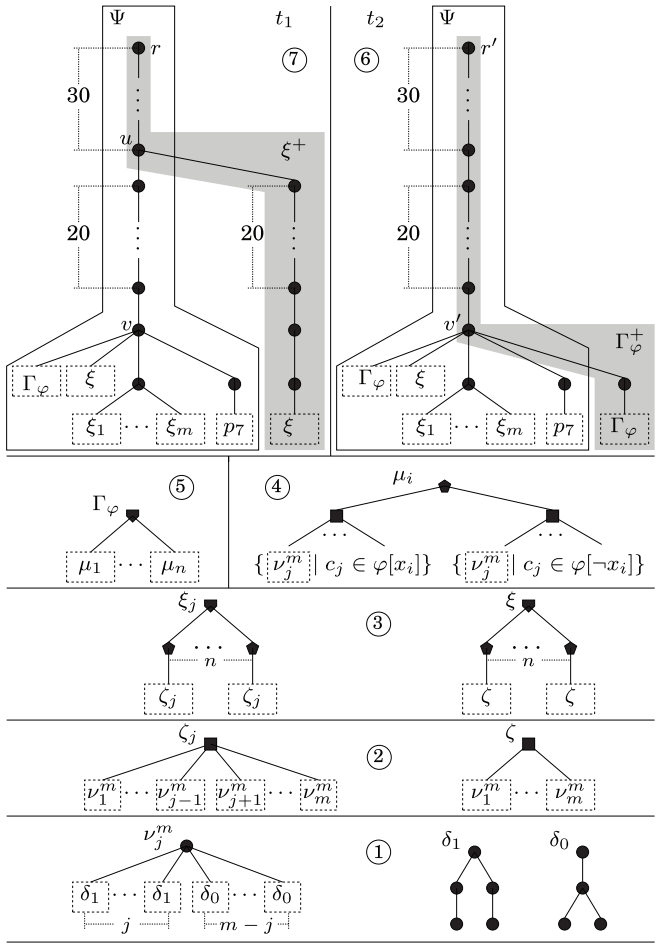

Fig. 5. Constructions in the reduction.

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 before 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 $\{c_1, \ldots, c_m\}$ of its clauses. If a is an atomic formula over $x_1, \ldots, x_n$ (i.e., $a = x_i$ or $a = \neg x_i$ for some $i \in \{1, \ldots, n\}$), then $\varphi[a]$ denotes the set of all those clauses c_j that have a as a disjunct. A *truth assignment* (*for* φ) is a mapping $\alpha : \{x_1, \ldots, x_n\} \to \{\textbf{true}, \textbf{false}\}$. 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 5 shows the construction of two trees, t_1 and t_2. The construction is bottom up—each tree is built from subtrees defined in previous phases. We view each tree in Figure 5 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 ζ has every tree ν_j^m, where $j = 1, \ldots, m$, placed under the root.
—The tree ζ_j is obtained from ζ by removing the tree ν_j^m.
—For $i = 1, \ldots, n$, 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 seven 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 5, 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. *The tree Ψ is a maximal common subtree of t_1 and t_2, that is, $\Psi \in \mathsf{MaxFS}^2[\mathbf{T}, \mathbf{T}](t_1, t_2)$.*

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. □

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

LEMMA 5.5. *Let j and k be two numbers in $\{1, \ldots, m\}$. 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 5). Note that in Γ_φ, the children of the root are the trees μ_i, for $i = 1, \ldots, n$. 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 $i = 1, \ldots, n$, 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 5. 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.*

PROOF. We begin with the "if" direction. Suppose α is not a satisfying assignment, then $\varphi[\alpha]$ excludes at least one clause, say c_k. Consequently, in the construction of Γ_α we removed every occurrence of ν_k^m. As a result, by following the construction in Figure 5 one can easily verify that $\Gamma_\alpha \sqsubseteq \xi_k$. Hence, $\Gamma_\alpha^+ \sqsubseteq \Psi$ can be witnessed by the mapping that maps the root of Γ_α to that of ξ_k.

We now show the "only if" direction. We assume that $\Gamma_\alpha^+ \sqsubseteq \Psi$. Let μ be a subgraph isomorphism from Γ_α^+ to Ψ. Due to the height of Γ_α^+ and Ψ (and to an easy node-degree argument), it must be the case that μ maps the root of Γ_α^+ to the root of Ψ. Consequently, via a similar argument, we can conclude the root of Γ_α is mapped to the root of one of

the ξ_j, say ξ_k. We conclude $\Gamma_\alpha \subseteq \xi_k$. We complete the proof by showing that c_k is not satisfied by α. By way of contradiction, suppose otherwise. So, c_k is in $\varphi[a]$ for some atom a that is satisfied by α. Suppose x_i is the variable of a. Then, in the construction of Γ_α, the subtree of μ_i obtained after pruning contains v_k^m. It thus follows that $v_k^m \sqsubseteq t$ for some tree t under the root of κ_k, but then we contradict Lemma 5.5, since κ_k does not have v_k^m under the root. □

We can now establish the following lemma.

LEMMA 5.7. *If φ is satisfiable, then* $\mathsf{MaxFS}^2[\mathbf{T}, \mathbf{T}](t_1, t_2)$ *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 5), for all truth assignments α, due to the construction of ξ.

It remains to prove the other direction of Lemma 5.7, namely $|\mathsf{MaxFS}^2[\mathbf{T}, \mathbf{T}](t_1, t_2)| > 1$ 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 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. $h \sqsubseteq_\mathsf{r} \xi^+$.

PROOF. We first prove that $h \sqsubseteq \xi^+$. We do so by showing that h_1 is a subtree of ξ^+. Consider the node u of t_1, as depicted in Figure 5. If h_1 does not contain u, then h_1 is either a subtree of Ψ, contradicting our assumption that $h \not\sqsubseteq \Psi$, or h_1 is a subtree of ξ^+, which proves our claim. Hence we assume h_1 contains u. We further assume that the degree of u in h_1 is at least 2 or, otherwise, we again get that h_1 is a subtree of either Ψ or ξ^+. Let $d_{h_1}(u)$ denote the degree of u in h_1, so $d_{h_1}(u)$ is either 2 or 3. We first show that $d_{h_1}(u)$ is necessarily 2.

Assume by way of contradiction that $d_{h_1}(u) = 3$. Let u_h be the node of h such that $\mu_1(u_h) = u$. Since the degree of u_h in h is 3, we get that the node $\mu_2(u_h)$ of t_2 must be in the subtree rooted by v' (since every other node of t_2 has a degree smaller than 3). Let h_1^1, h_1^2, and h_1^3 be the three subtrees of h_1 that are obtained from h_1 by removing u. Due to the lengths of the paths emanating from u in t_1 and the fact that the downward paths under v' are length at most 9, we conclude that each of the three h_1^i is a simple path. Moreover, at least two of the three paths are of length at most 9 since at least two are downward paths. Moreover, the third path is of length at most 30, since they emanate from u in the tree t_1. We conclude that h can be embedded in the subtree of h_1 that is obtained by combining: *(1)* the path from r to v; *(2)* the downward path from v that goes through p_7; *and (3)* another downward path from v (of length 9) that goes through any of the ξ_i. Consequently $h \sqsubseteq \Psi$, contradicting our assumption that $h \not\sqsubseteq \Psi$.

So, we conclude that $d_{h_1}(u) = 2$. Let h_1^1 and h_1^2 be the two subtrees of h_1 obtained from h_1 by removing u. Since $h_1 \sqsubseteq t_2$, we have that at least one of h_1^1 and h_1^2 is a simple path. This is true since, otherwise, we will have that h_1 contains two nodes w_1 and w_2 such that the degree of both w_i is larger than 2 and the distance (i.e., the length of the path)

between w_1 and w_2 is at least 40; and it is easy to observe that t_2 does not contain such two nodes. So, suppose that h_1^1 is a simple path. Then in h_1, we can replace h_1 with a path from u to some node on the way to the root r. Consequently, we get a tree h_1' that is isomorphic to h_1 and is a subtree of either Ψ, which is an impossible scenario, or of ξ^+, which proves our claim that $h \sqsubseteq \xi^+$.

Finally, to see that $h \sqsubseteq_{\mathsf{r}} \xi^+$, it suffices to show that h_1 necessarily contains the root r of t_1. By way of contradiction, suppose that h_1 does not contain r. Since h_1 is a subtree of ξ^+, it is also a subtree of the subtree $\hat{t}_1$ of t_1, where $\hat{t}_1$ is the tree obtained by connecting the path from r to v with the subtree ξ (see step 7 in Figure 5). However, we get that h_1 is a subtree of Ψ, which contradicts our assumption that $h \not\sqsubseteq \Psi$. □

LEMMA 5.9. $h \sqsubseteq_{\mathsf{r}} \Gamma_\varphi^+$.

PROOF. From Lemma 5.8 we get that h_1 is a root subtree of ξ^+. And since $h_1 \not\sqsubseteq \Psi$, the tree h_1 must contain at least one node of the ξ part of ξ^+. Let h_1^ξ be subtree of h_1 that comes from ξ, and let p_1 be the remaining path. Hence, h_1 is obtained by connecting an endpoint of p_1 to h_1^ξ. Let p_2 and h_2^ξ be the subtrees of h_2 that correspond to p_1 and h_1^ξ, respectively, according to the isomorphism $\mu_2 \circ \mu_1^{-1}$. Note that p_2 has 52 nodes. Since we can connect p_2 and h_2^ξ into a subtree of t_2, it must be the case that p_2 has one endpoint strictly above v', and the other endpoint strictly below v'. Consequently, h_2^ξ is a subtree of one of the subtrees strictly below v'. Therefore, h_2^ξ must be a subtree of Γ_φ, or otherwise h_2 is a subtree of Ψ, contradicting our assumption that $h \not\sqsubseteq \Psi$. Hence we get that $h_1 \sqsubseteq \Gamma_\varphi^+$.

From $h_1 \sqsubseteq \Gamma_\varphi^+$, we can infer that $h_1 \sqsubseteq_{\mathsf{r}} \Gamma_\varphi^+$ using an argument similar to the one in the last paragraph in the proof of Lemma 5.8, except that this time we use the subtree Γ_φ (instead of the subtree ξ) under the node v of Ψ. □

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

LEMMA 5.10. *There is a satisfying assignment α such that h is isomorphic to Γ_α^+.*

PROOF. We will prove 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 $h \not\sqsubseteq \Psi$.

Combined, Lemmas 5.8 and 5.9 state 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 (that 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. These 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 5). 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 $h \equiv h_2$. □

It remains to show how the proof is modified to restrict the trees into ones of degree 3. This is done in a standard fashion of encoding trees as binary ones. In particular, we will assume that n and m are powers of 2. Each of the trees ζ, ζ_j and the subtrees under μ_i are now obtained by taking the complete binary tree with m leaves, ordering the leaves from left to right, and attaching each v_j^m to the jth leaf. We similarly reconstruct v_j^m, ξ, ξ_i, Γ_φ and the subtrees underlying v and v' (Figure 5).

Table V. Complexity Results for Counting the Maximal Frequent Subgraphs over Labeled Graph Classes

$\mathcal{P}$ \ $\mathcal{D}$	$\mathbf{BDG}_{\mathsf{L}}^{2}$	$\mathbf{BDG}_{\mathsf{L}}^{d>2}$	$\mathbf{T}_{\mathsf{L}}$	$\mathbf{BTW}_{\mathsf{L}}^{w>1}$	$\mathbf{PLN}_{\mathsf{L}}$	$\mathbf{G}_{\mathsf{L}}$
$\mathbf{BDG}_{\mathsf{L}}^{2}$	FP / FP	#Pc / #Pc	FP / FP	#Pc / #Pc	#Pc / #Pc	#Pc / #Pc
$\mathbf{BDG}_{\mathsf{L}}^{d'>2}$	FP / FP	#Pc	#Pc	#Pc	#Pc	#Pc
$\mathbf{T}_{\mathsf{L}}$	FP / FP	#Pc / #Pc	#Pc / FP	#Pc / #Pc	#Pc / #Pc	#Pc / #Pc
$\mathbf{BTW}_{\mathsf{L}}^{w'>1}$	FP / FP	#Pc	#Pc / FP	#Pc	#Pc	#Pc
$\mathbf{PLN}_{\mathsf{L}}$	FP / FP	#Pc	#Pc / FP	#Pc	#Pc	#Pc / FP
$\mathbf{G}_{\mathsf{L}}$	FP / FP	#Pc / FP	#Pc / FP	#Pc / FP	#Pc / FP	#Pc / FP

See Table I for the meaning of the complexity classes. In a pair "c_1 / c_2," the symbol c_1 refers to the unbounded variant, while c_2 assumes a bounded threshold. In some cases, c_2 is unknown and then only c_1 is specified.

6. COUNTING COMPLEXITY

In this section we study the complexity of counting the number of maximal frequent subgraphs; more precisely, we study the complexity of the problem #$\mathcal{R}$ for the enumeration relations $\mathcal{R}$ considered in this article. Our results for the unlabeled graph classes are given in Theorem 6.2, and those for the labeled graph classes are listed in Table V. Note that Table V leaves a few cases as open questions.

For the graph mining settings $(\mathcal{P}, \mathcal{D})$ of interest, our proofs of complexity results are of three kinds.

—MaxFS$[\mathcal{P}, \mathcal{D}]$-enumerate is solvable in polynomial time. Then, counting is obviously in FP.
—MaxFS$[\mathcal{P}, \mathcal{D}]$-extendible is NP-complete, and then we show that our previous reductions are either parsimonious or can be made so.
—MaxFS$[\mathcal{P}, \mathcal{D}]$-enumerate is solvable with polynomial delay (but not polynomial time since the number of answers can be exponential). Then we construct reductions that show #P-hardness.

The problem of counting maximal frequent subgraphs has already been studied by Yang [2004], who established #P-completeness for both labeled and unlabeled trees.

THEOREM 6.1 [YANG 2004]. *If $\mathcal{R}$ is* MaxFS$[\mathbf{T}, \mathbf{T}]$ *or* MaxFS$[\mathbf{T}_{\mathsf{L}}, \mathbf{T}_{\mathsf{L}}]$*, then* #$\mathcal{R}$ *is* #P*-complete. Moreover,* #P*-completeness holds even if the inputs are binary trees (equivalently, trees of degree at most* 3*).*

Yang's results do not cover all the cases we consider here. Most 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.

6.1. Unlabeled Graph Classes

We start with the unlabeled case. Observe that in all the cases where MaxFS$[\mathcal{P}, \mathcal{D}]$-enumerate is solvable in polynomial time, so is the associated counting problem.

It turns out that in our study of specific graph classes, those graph mining settings $(\mathcal{P}, \mathcal{D})$ where MaxFS$[\mathcal{P}, \mathcal{D}]$-enumerate is solvable in polynomial time are exactly the ones with tractable verification, such that the number of maximal frequent subgraphs is bounded by a polynomial. For the remaining cases, we prove #P-hardness by making the reduction of Section 5.1 parsimonious.

THEOREM 6.2. *For the (unlabeled) graph mining settings* $(\mathcal{P}, \mathcal{D})$ *from Table IV:*

(*1*) *if* MaxFS$[\mathcal{P}, \mathcal{D}]$-enumerate *is in* FP*, then* #MaxFS$[\mathcal{P}, \mathcal{D}]$ *is solvable in polynomial time;*
(*2*) *otherwise,* #MaxFS$^{\tau}[\mathcal{P}, \mathcal{D}]$ *is* #P*-complete for every fixed* $\tau \geq 2$.

PROOF. The first part is straightforward, so we prove only the second. Membership in #P is immediate from the fact that MaxFS$^{\tau}[\mathcal{P}, \mathcal{D}]$ is an NP-relation for every τ (Corollary 3.9). We need to prove #P-hardness.

We first show #P-hardness for $\tau = 2$, building on the proof of Theorem 5.2 (Section 5.1). In that proof, we gave a reduction from CNF satisfiability. Specifically, given a CNF formula φ, we showed there are exactly two kinds of frequent subgraphs (Lemmas 5.4, 5.6, and 5.10):

—the tree Ψ;
—the tree Γ_{α}^{+}, where α is a satisfying truth assignment for φ.

Moreover, Lemma 5.4 states that Ψ is a maximal frequent subgraph. Let α be a satisfying truth assignment. Lemma 5.6 states that $\Gamma_{\alpha}^{+} \not\sqsubseteq \Psi$. By a careful inspection of the construction in Figure 5, one can observe that Γ_{α}^{+} is a maximal frequent subgraph as well. We would like our reduction to be *parsimonious*; in our case, this means that, for all satisfying assignments α and β, if $\alpha \neq \beta$, then $\Gamma_{\alpha}^{+} \not\equiv \Gamma_{\beta}^{+}$ (or, equivalently, $\Gamma_{\alpha}^{+} \not\sqsubseteq \Gamma_{\beta}^{+}$ whenever $\alpha \neq \beta$). Indeed, had our reduction been parsimonious, we would have MaxFS$^{2}[\mathcal{P}, \mathcal{D}](t_1, t_2) - 1 = \#\varphi$, where $\#\varphi$ is the number of satisfying assignments for φ. As computing $\#\varphi$ is #P-complete, we would obtain #P-hardness of #MaxFS$^{2}[\mathcal{P}, \mathcal{D}]$. However, our reduction is not necessarily parsimonious.

Recall that the variables of φ are $x_1, \ldots, x_n$, and that the clauses of φ are $c_1, \ldots, c_m$. If α is a truth assignment for φ, then we write $\mathcal{S}(\alpha)$ to denote the set $\{S_1, \ldots, S_n\}$, where each S_i is the set of clauses that are satisfied due to the assignment of α to x_i. Thus

$$S_i = \begin{cases} \varphi[x_i] & \text{if } \alpha(x_i) = \textbf{true}; \\ \varphi[\neg x_i] & \text{if } \alpha(x_i) = \textbf{false}. \end{cases}$$

We say that an assignment α is φ-*subsumed by* an assignment β if there is a bijection $\pi : \mathcal{S}(\alpha) \to \mathcal{S}(\beta)$ such that $S_i \subseteq \pi(S_i)$ for all $S_i \in \mathcal{S}(\alpha)$. The construction of Figure 5 implies that, for two satisfying assignments α and β, we have $\Gamma_{\alpha}^{+} \sqsubseteq \Gamma_{\beta}^{+}$ if and only if α is φ-subsumed by β. We complete this part of the proof by showing how to (efficiently) construct from φ a CNF formula φ' with the following properties.

(1) φ and φ' have the same number of satisfying assignments.
(2) For all assignments α and β for φ', if α is φ'-subsumed by β, then $\alpha = \beta$.

We obtain φ' from φ by adding $3n$ clauses, consisting of the following three clauses for every variable x_i.

$$c_i^0 \stackrel{\text{def}}{=} x_i \vee \neg x_i \quad c_i^1 \stackrel{\text{def}}{=} x_i \vee x_i' \quad c_i^2 \stackrel{\text{def}}{=} \neg x_i \vee \neg x_i'$$

Here, x_i' is a new, fresh variable that is unique for each i. It is easy to verify that Property 1 holds and in particular that, in every satisfying assignment α, the value $\alpha(x_i')$ is determined by (in particular, is the negation of) $\alpha(x_i)$. To see that Property 2 holds as well, suppose α is φ'-subsumed by β. We will show that $\alpha = \beta$, that is, $\alpha(x_i) = \beta(x_i)$, for each variable x_i. So, consider a variable x_i. If $\alpha(x_i) = \textbf{true}$, then $\mathcal{S}(\alpha)$ has a set S_i that contains both c_i^0 and c_i^1. Consequently, $\mathcal{S}(\beta)$ has also a set that contains both c_i^0 and c_i^1, which implies that $\beta(x_i) = \textbf{true}$. Hence $\alpha(x_i) = \textbf{true}$ implies that $\beta(x_i) = \textbf{true}$. Similarly, $\alpha(x_i) = \textbf{false}$ implies that $\beta(x_i) = \textbf{false}$, thus $\alpha = \beta$, as claimed.

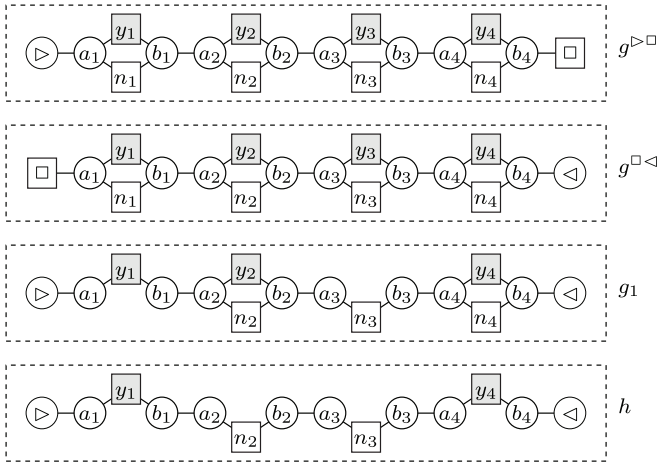

Fig. 6. Reducing #DNF to #MaxFS$^\tau$[$\mathbf{BDG}^2_{\mathsf{L}}$, $\mathcal{D}$].

We extend the result from $\tau = 2$ to any $\tau \geq 2$ exactly as done in the proof of Corollary 5.3. □

6.2. Labeled Graph Classes

We now proceed to the labeled case. As said previously, in those cases where MaxFS[$\mathcal{P}$, $\mathcal{D}$]-enumerate can be solved in polynomial time, so can #MaxFS[$\mathbf{T}$, $\mathbf{T}$]. These cases are mentioned in Table V and we do not further consider them.

The next two theorems show cases where the problem #MaxFS[$\mathbf{T}$, $\mathbf{T}$] can be solved with polynomial delay, but #MaxFS$^\tau$[$\mathcal{P}$, $\mathcal{D}$] is #P-complete for all $\tau > 0$ (including $\tau = 1$). In the first theorem, the pattern class $\mathcal{P}$ is the class $\mathbf{BDG}^2_{\mathsf{L}}$ of all labeled paths and simple cycles; in the second theorem $\mathcal{P}$ is the class $\mathbf{T}_{\mathsf{L}}$ of all labeled trees.

THEOREM 6.3. *Suppose that $\mathcal{D}$ is one of* $\mathbf{BDG}^d$, $\mathbf{BTW}^w$, **PLN**, *and* **PLN** *for $d > 2$ and $w > 1$. Then* #MaxFS$^\tau$[$\mathbf{BDG}^2_{\mathsf{L}}$, $\mathcal{D}$] *is* #P*-complete for all thresholds $\tau > 0$.*

PROOF. The proof follows the general idea of that of Theorem 4.6, but uses a (Turing) reduction from a different problem—counting the number of satisfying assignments of a DNF formula. This counting problem is denoted by #DNF and is known to be #P-complete [Valiant 1979b]. In both proofs, the reduction gets as input a DNF formula $\varphi = c_1 \vee \cdots \vee c_m$ over the variables $x_1, \ldots, x_n$, where each c_m is the conjunction of literals (where a literal has the form x_i or $\neg x_i$). We will produce inputs $(\mathbf{g}, \tau)$ for #MaxFS$^\tau$[$\mathbf{BDG}^2_{\mathsf{L}}$, $\mathcal{D}$] with $\tau = 1$. For generalizing to arbitrary $\tau \geq 1$, we simply extend $\mathbf{g}$ by taking τ copies thereof.

We define the graphs $g^{\triangleright\square}$ and $g^{\square\triangleleft}$ as depicted in Figure 6. This graph has the labels (identified as nodes here) a_i, b_i, y_i, and n_i for each variable x_i. In addition, $g^{\triangleright\square}$ has the labels $\triangleright$ (leftmost) and $\square$ (rightmost), and $g^{\square\triangleleft}$ has the labels $\square$ (leftmost) and $\triangleleft$ (rightmost). For each clause c_j, the graph g_j is similar to $g^{\triangleright\square}$, except for the following differences.

—The label $\square$ is replaced with $\triangleleft$.
—For each variable x_i in c_j, node n_i is removed if x_i occurs positively, and y_i is removed if x_i occurs negatively.

For illustration, Figure 6 shows the graph g_1 assuming that $c_1 = x_1 \vee \neg x_3$. Observe that each of $g^{\triangleright\square}$, $g^{\square\triangleleft}$, and g_j is in all of the classes $\mathbf{BDG}^d$, $\mathbf{BTW}^w$, **PLN**, and **PLN** for $d > 2$ and $w > 1$.

The produced input $\mathbf{g}$ is $\langle g^{\triangleright\square}, g^{\square\triangleleft}, g_1, \ldots, g_m\rangle$. We will use the following observation. If h is a maximal (frequent) subgraph, then exactly one of the following is true: *(1)* h is isomorphic to a subgraph of one of $g^{\triangleright\square}$ and $g^{\square\triangleleft}$; *(2)* h is a simple path from $\triangleright$ to $\triangleleft$. In the second case, we say that h is a $(\triangleright, \triangleleft)$-path. We can identify a $(\triangleright, \triangleleft)$-path with a truth assignment τ for $x_1, \ldots, x_n$, by defining $\tau(x_i) = \mathbf{true}$ if h contains y_i and $\tau(x_i) = \mathbf{false}$ if h contains n_i. Observe that the $(\triangleright, \triangleleft)$-paths in g_j correspond to precisely those assignments that satisfy c_j. We conclude that the frequent $(\triangleright, \triangleleft)$-paths correspond precisely to the satisfying assignments for φ, which are what we aim to count.

So, to compute the number of satisfying assignments for φ, we only need to compute the number of all maximal frequent subgraphs for $(\mathbf{g}, \tau)$ and subtract from this the number of all maximal frequent subgraphs for $(\mathbf{g}', \tau)$ where $\mathbf{g}' = \langle g^{\triangleright\square}, g^{\square\triangleleft}\rangle$. We can compute both of these numbers using an oracle to $\#\mathsf{MaxFS}^\tau[\mathbf{BDG}^2_{\mathsf{L}}, \mathcal{D}]$. □

THEOREM 6.4. *Suppose that $\mathcal{D}$ is one of* $\mathbf{BDG}^d$, $\mathbf{BTW}^w$, **PLN**, *and* **PLN** *for $d > 2$ and $w > 1$. Then* $\#\mathsf{MaxFS}^\tau[\mathbf{T}_{\mathsf{L}}, \mathcal{D}]$ *is* #P*-complete for all thresholds $\tau > 0$.*

PROOF. The proof is basically an adaptation of that of Theorem 6.3. We will assume the formula is a 2DNF one (i.e., every clause has precisely two literals). This counting problem is denoted by #DNF and is again known to be #P-complete [Valiant 1979b].

As in the proof of Theorem 6.3, the reduction gets as input a 2DNF formula $\varphi = c_1 \vee \cdots \vee c_m$ over the variables $x_1, \ldots, x_n$, where each c_m is the conjunction of two literals. We will produce the input $(\mathbf{g}, \tau)$ for $\#\mathsf{MaxFS}^\tau[\mathbf{T}_{\mathsf{L}}, \mathcal{D}]$ with $\tau = 1$. For generalizing to arbitrary $\tau \geq 1$, we simply extend $\mathbf{g}$ by taking τ copies thereof.

We define $\mathbf{g}$ as follows. For each clause c_j, we take the graph g_j from the previous part and add back the two missing nodes (e.g., n_1 and y_3 in g_1 of Figure 6). For each of these two nodes, we add only one of the edges to a_j and b_j. As there are four ways of doing so (two ways per node), we take all four possibilities, denoted g^1_j, g^2_j, g^3_j, and g^4_j, to be members of $\mathbf{g}$. Figure 7 shows the graphs g^1_1, g^2_1, g^3_1, and g^4_1 assuming $c_1 = x_1 \vee \neg x_3$. Our sequence $\mathbf{g}$ consists of the $4m$ graphs g^l_js.

A simple observation is that every maximal (frequent) subtree h is a spanning tree of some g^l_j that consists of a $(\triangleright, \triangleleft)$-path along with extra edges connecting the path to the remaining nodes. See an illustration of h in the bottom of Figure 7. Our construction is such that, given a $(\triangleright, \triangleleft)$-path in g^l_j, there are precisely 2^n ways (up to isomorphism) of extending this into a maximal (frequent) subtree of $\mathbf{g}$. Moreover, the extension sets of two different $(\triangleright, \triangleleft)$-paths are disjoint (i.e., they do not contain isomorphic graphs), since we can restore the path from the extension (being the unique simple path between $\triangleright$ and $\triangleleft$). Consequently, the number of maximal subtrees is 2^n times the number of $(\triangleright, \triangleleft)$-paths. Using the same reasoning as in the proof of Theorem 6.3, we establish that the number of satisfying assignments for φ is equal to the number of $(\triangleright, \triangleleft)$-paths. Therefore, we can compute the number of satisfying assignments by dividing the number of maximal frequent subtrees by 2^n. □

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 article, we used the lens of computational complexity to obtain a fairly comprehensive

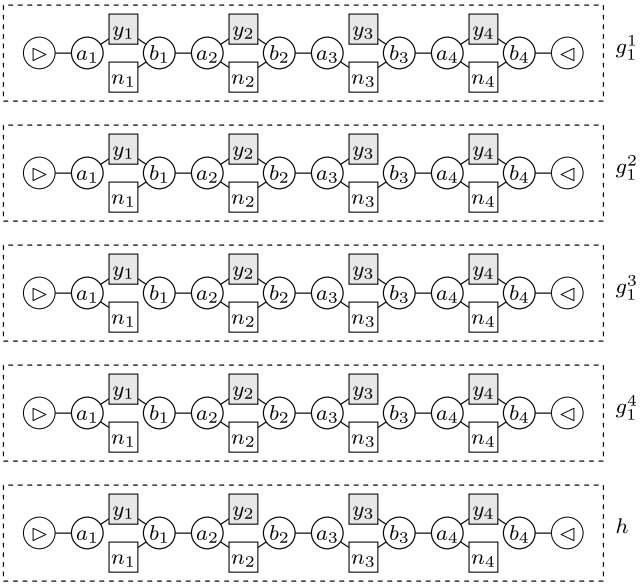

Fig. 7. Reducing #2DNF to #MaxFS$^{\tau}$[$\mathbf{T}$, $\mathcal{D}$].

picture, shown in Tables III, IV, and V, for the problem of mining maximal frequent subgraphs. Furthermore, we believe 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 where 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 [Pei et al. 2000; Wang et al. 2003; Yan and Han 2003; Zaki and Hsiao 2002]. To date, no systematic investigation of the complexity of mining closed frequent subgraphs has been carried out.

Finally, note that labeled 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.

REFERENCES

Laszlo Babai and Eugene M. Luks. 1983. Canonical labeling of graphs. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing (STOC'83)*. ACM Press, New York, 171–183.

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

Sara Cohen, Itzhak Fadida, Yaron Kanza, Benny Kimelfeld, and Yehoshua Sagiv. 2006. Full disjunctions: Polynomial-delay iterators in action. In *Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB'06)*. ACM Press, New York, 739–750.

Mukund Deshpande, Michihiro Kuramochi, Nikil Wale, and George Karypis. 2005. Frequent substructure-based approaches for classifying chemical compounds. *IEEE Trans. Knowl. Data Engin.* 17, 8, 1036–1050.

Rodney G. Downey and Michael R. Fellows. 1999. *Parameterized Complexity*. Springer.

Michael R. Garey, David S. Johnson, and Robert Endre Tarjan. 1976. The planar hamiltonian circuit problem is np-complete. *SIAM J. Comput.* 5, 4, 704–714.

Gianluigi Greco, Antonella Guzzo, Giuseppe Manco, and Domenico Sacca. 2005. Mining and reasoning on workflows. *IEEE Trans. Knowl. Data Engin.* 17, 4, 519–534.

Gianluigi Greco, Antonella Guzzo, Giuseppe Manco, and Domenico Sacca. 2007. Mining unconnected patterns in workflows. *Inf. Syst.* 32, 5, 685–712.

Ehud Gudes, Solomon Eyal Shimony, and Natalia Vanetik. 2006. Discovering frequent graph patterns using disjoint paths. *IEEE Trans. Knowl. Data Engin.* 18, 11, 1441–1456.

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

John E. Hopcroft and Robert Endre Tarjan. 1972. Isomorphism of planar graphs. In *Complexity of Computer Computations*. The IBM Research Symposia Series. Plenum Press, New York, 131–152.

Jun Huan, Wei Wang, Jan Prins, and Jiong Yang. 2004. SPIN: Mining maximal frequent subgraphs from graph databases. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04)*. ACM Press, New York, 581–586.

Akihiro Inokuchi, Takashi Washio, and Hiroshi Motoda. 2000. An apriori-based algorithm for mining frequent substructures from graph data. In *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD'00)*. Lecture Notes in Computer Science, vol. 1910, Springer, 13–23.

Akihiro Inokuchi, Takashi Washio, and Hiroshi Motoda. 2003. Complete mining of frequent patterns from graphs: Mining graph data. *Mach. Learn.* 50, 3, 321–354.

David S. Johnson, Mihalis Yannakakis, and Christos H. Papadimitriou. 1988. On generating all maximal independent sets. *Inf. Process. Lett.* 27, 119–123.

Leonid Khachiyan, Endre Boros, Konrad Borys, Khaled M. Elbassioni, and Vladimir Gurvich. 2008. Generating all vertices of a polyhedron is hard. *Discr. Comput. Geom.* 39, 1–3, 174–190.

Benny Kimelfeld and Phokion G. Kolaitis. 2013. The complexity of mining maximal frequent subgraphs. In *Proceedings of the 32nd Symposium on Principles of Database Systems (PODS'13)*. ACM Press, New York, 13–24.

Benny Kimelfeld and Yehoshua Sagiv. 2007. Maximally joining probabilistic data. In *Proceedings of the 26th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS'07)*. ACM Press, New York, 303–312.

Michihiro Kuramochi and George Karypis. 2001. Frequent subgraph discovery. In *Proceedings of the IEEE International Conference on Data Mining (ICDM'01)*. IEEE Computer Society, 313–320.

Michihiro Kuramochi and George Karypis. 2004. An efficient algorithm for discovering frequent subgraphs. *IEEE Trans. Knowl. Data Engin.* 16, 9, 1038–1051.

Kazuhisa Makino and Toshihide Ibaraki. 1996. Interior and exterior functions of boolean functions. *Discr. Appl. Math.* 69, 3, 209–231.

Jiri Matousek and Robin Thomas. 1992. On the complexity of finding iso- and other morphisms for partial k-trees. *Discr. Math.* 108, 1–3, 343–364.

Raymond J. Mooney, Prem Melville, Rupert Lapoon Tang, Jude Shavlik, Inłs Dutra, David Page, and Vitor Santos Costa. 2004. Relational data mining with inductive logic programming for link discovery. In *Data Mining: Next Generation Challenges and Future Directions,* AAAI Press, 239–254.

Siegfried Nijssen and Joost N. Kok. 2004. Frequent graph mining and its application to molecular databases. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC'04)*. Vol. 5, 4571–4577.

Yoshio Okamoto, Takeaki Uno, and Ryuhei Uehara. 2008. Counting the number of independent sets in chordal graphs. *J. Discr. Algor.* 6, 2, 229–242.

Jian Pei, Jiawei Han, and Runying Mao. 2000. CLOSET: An efficient algorithm for mining frequent closed itemsets. In *Proceedings of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery.* 21–30.

Alina Stoica and Christophe Prieur. 2009. Structure of neighborhoods in a large social network. In *Proceedings of the International Conference on Computational Science and Engineering (CSE'09)*. IEEE Computer Society, 26–33.

Lini T. Thomas, Satyanarayana R. Valluri, and Kamalakar Karlapalem. 2010. MARGIN: Maximal frequent subgraph mining. *ACM Trans. Knowl. Discov. Data* 4, 3.

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

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

Leslie G. Valiant. 1979b. The complexity of enumeration and reliability problems. *SIAM J. Comput.* 8, 3, 410–421.

Fabian Wagner. 2011. Graphs of bounded treewidth can be canonized in ac^1 computer science. In *Proceedings of the 6th International Conference on Computer Science: Theory and Applications (CSR'11)*. Lecture Notes in Computer Science, vol. 6651, Springer, 209–222.

Jianyong Wang, Jiawei Han, and Jian Pei. 2003. CLOSET+: Searching for the best strategies for mining frequent closed itemsets. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'03)*. ACM Press, New York, 236–245.

Xifeng Yan and Jiawei Han. 2002. gSpan: Graph-based substructure pattern mining. In *Proceedings of the IEEE International Conference on Data Mining (ICDM'02)*. IEEE Computer Society, 721–724.

Xifeng Yan and Jiawei Han. 2003. CloseGraph: Mining closed frequent graph patterns. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'03)*. ACM Press, New York, 286–295.

Guizhen Yang. 2004. The complexity of mining maximal frequent itemsets and maximal frequent patterns. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04)*. ACM Press, New York, 344–353.

Mohammed Javeed Zaki and Ching-Jiu Hsiao. 2002. CHARM: An efficient algorithm for closed itemset mining. In *Proceedings of the 2nd SIAM International Conference on Data Mining (SDM'02)*. 457–473.

Received October 2013; accepted February 2014