MaNIACS: Approximate Mining of Frequent Subgraph Patterns through Sampling

Giulia Preti
ISI Foundation
Turin, Italy
giulia.preti@isi.it

Gianmarco De Francisci Morales
ISI Foundation
Turin, Italy
gdfm@acm.org

Matteo Riondato
Amherst College
Amherst, MA, USA
mriondato@amherst.edu

"And we're zany to the max!" — *Animaniacs* theme song

ABSTRACT

We present MaNIACS, a sampling-based randomized algorithm for computing high-quality approximations of the collection of the subgraph patterns that are frequent in a single, large, vertex-labeled graph, according to the Minimum Node Image-based (MNI) frequency measure. The output of MaNIACS comes with strong probabilistic guarantees, obtained by using the empirical Vapnik-Chervonenkis (VC) dimension, a key concept from statistical learning theory, together with strong probabilistic tail bounds on the difference between the frequency of a pattern in the sample and its exact frequency. MaNIACS leverages properties of the MNI-frequency to aggressively prune the pattern search space, and thus to reduce the time spent in exploring subspaces containing no frequent patterns. In turn, this pruning leads to better bounds to the maximum frequency estimation error, which leads to increased pruning, resulting in a beneficial feedback effect. The results of our experimental evaluation of MaNIACS on real graphs show that it returns high-quality collections of frequent patterns in large graphs up to two orders of magnitude faster than the exact algorithm.

CCS CONCEPTS

• **Mathematics of computing** → **Graph enumeration**; **Approximation algorithms**; • **Information systems** → **Data mining**; • **Theory of computation** → **Sketching and sampling**.

KEYWORDS

Minimum Node Image, Pattern mining, VC-dimension

ACM Reference Format:
Giulia Preti, Gianmarco De Francisci Morales, and Matteo Riondato. 2021. MaNIACS: Approximate Mining of Frequent Subgraph Patterns through Sampling. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3447548.3467344

KDD '21, August 14–18, 2021, Virtual Event, Singapore

ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467344

1 INTRODUCTION

A subgraph pattern (sometimes called "graphlet") is a small graph, possibly with labeled vertices. Frequent Subgraph Pattern Mining (FSPM), i.e., finding the patterns that appear frequently in a single graph, has many applications, from the discovery of protein functionality in computational biology [36, 51], to the development of recommender systems for video games [2], to social media marketing [19], to software engineering [24]. It is also a primitive for graph mining tasks such as classification [17] and clustering [22].

The FSPM task is computationally challenging, for two main reasons: *(i)* the number of possible patterns experiences a combinatorial explosion with the maximum number of vertices in a pattern and with the number of possible vertex labels; and *(ii)* the subgraph isomorphism operation needed to find a pattern in the graph is in general NP-complete. Ingenious exact algorithms exist, but they tend to scale poorly with the size of the graph and with the maximum size of a pattern.

The use of *random sampling* is a common solution to speed up time-consuming data analytics tasks, from approximate database query processing [16], to itemset mining [47], to other tasks on graphs [48]. It however comes at the price of obtaining an *approximate solution* to the task at hand. Such solutions are acceptable when they come with *stringent theoretical guarantees on their quality*. A typical approach for sampling algorithms relies on evaluating the function of interest only on a randomly chosen subset of the input domain. In FSPM, one can, e.g., create a small random sample of vertices, and evaluate the presence of subgraph isomorphisms between patterns and only those subgraphs of the graph that include at least one of the sampled vertices.

Random sampling and approximate solutions are *necessary* when access to the graph is restricted, as in online networks, where one cannot inspect the whole graph, but only query a vertex and its neighborhood through an API. By using vertex sampling schemes [14, 15], an approximation algorithm enables FSPM in this scenario.

The key challenge in using random sampling is understanding the trade-off between the sample size, the time needed to analyze the sample (which depends on the sample size, but also on the analytics task at hand), and the quality that can be obtained from a sample of the specific size. Large deviation bounds can be applied when there is only one function to be estimated, but in FSPM, like in most data analytics tasks, we need to accurately estimate the frequencies of many patterns from the same sample. Classic simultaneous deviation bounds tools such as the union bound, if applied naïvely, are inherently loose, so more sophisticated techniques must

be employed. For FSPM, having tight bounds to the maximum estimation error is particularly important, as they are used not only to decide what patterns to include in the approximate solution, but also to *prune the search space* via an apriori-like argument, thus avoiding the expensive step of evaluating the frequency of patterns that are not sufficiently frequent to be included in the output.

Contributions. We present MaNIACS (for "MNI Approximate Computation through Sampling"), an algorithm to compute high-quality approximations of the collection of frequent subgraph patterns from a single, large, vertex-labeled graph, according to the MNI-frequency measure [10] (see (3)).

- MaNIACS relies on *uniform random sampling* of vertices and on computing the patterns to which these vertices belong. MaNIACS is scalable w.r.t. the size of the graph and is the first FSPM algorithm on graphs with restricted access. Sampling allows MaNIACS to be easily parallelized on, e.g., Arabesque [55].
- MaNIACS is the first sampling-based algorithm for the task of FSPM that comes with *strong probabilistic guarantees* on the quality of its output. These guarantees are obtained by leveraging *sample-dependent quantities*: MaNIACS extracts information from the sample to determine the quality of the approximation in terms of the maximum frequency estimation error, which is used to avoid false negatives. The estimated quality is output with the approximate collection of frequent patterns. To upper bound the maximum estimation error, MaNIACS relies on the *empirical Vapnik-Chervonenkis (eVC) dimension* [58], a fundamental concept from statistical learning theory [53]. The eVC-dimension leads to much better quality guarantees than could be obtained by using classic approaches such as the union bound. We show that the eVC-dimension of the task at hand is *independent from the number of vertex labels*, and we show how to efficiently compute a tight upper bound to this quantity.
- MaNIACS aggressively leverages the *anti-monotonicity* of the MNI-frequency measure (Facts 2 and 3), to prune parts of the search space that provably do not contain any frequent pattern, and to focus the exploration only on the "promising" subspaces, therefore avoiding expensive-but-useless computations. Pruning also leads to better bounds to the maximum frequency estimation error, which enables additional pruning, thus creating a virtuous cycle that improves both the computational and statistical properties of MaNIACS.
- The results of our experimental evaluation of MaNIACS on real datasets show that it returns a high-quality output very fast, with even better error than guaranteed by the theory.

2 RELATED WORK

There is a vast body of work on subgraph extraction and counting. Due to space constraints, we focus on the single, static graph setting, and we omit others (e.g., transactional, dynamic, or stream). For a discussion of these many others areas, we refer the reader to the tutorial by Seshadhri and Tirthapura [52].

The patterns we consider are *connected, unweighted, undirected, vertex-labeled graphs with up to k vertices*. The assignment of the labels to the vertices of the pattern is important, and different assignments (up to automorphisms of the patterns) generate different patterns (see formal definitions in Sect. 3.1). The collection of patterns is therefore different from the collections of *colored graphlets* [45] and *heterogeneous graphlets* [50], which respectively only consider the set or the multiset of vertex labels. Graphlets [42] are a special case of patterns with a single label.

FSPM requires finding patterns with a *global* frequency, for instance as quantified by the popular *Minimum Node Image (MNI)* frequency measure [10] (see (3)), at least as large as a user-specified minimum threshold (see (4)). MaNIACS can be adapted to many other frequency measures [20, 29, 34, 35, 57], but due to space limitations, we postpone this discussion to an extended version.

The presence of a minimum frequency threshold and the use of the MNI measure distinguish this task from the well-studied task of *counting graphlets or motifs*, which require to compute the global number of vertex- or edge-induced instances of a pattern [4, 8, 9, 23, 25, 38, 41, 44, 59–61]. FSPM is also different from computing the *local* counts i.e., the number of instances of each pattern in which an edge/vertex participates [39, 50]. The techniques used in these tasks cannot be easily adapted to FSPM.

Algorithms. Elseidy et al. [18] present GraMi, an exact algorithm for FSPM. GraMi transforms the subgraph isomorphism problem into a constraint-satisfaction problem, and uses ingenious computation organization to speed up finding the edge-induced frequent patterns, although not their frequencies. Frequencies are important in pattern mining: since the minimum threshold is often set somewhat arbitrarily, it is important to be able to distinguish between patterns with frequency much greater than the threshold and those that are "barely" frequent. We define the FSPM task to include their exact frequencies (see (4)), which is inherently more difficult. GraMi requires complete access to the whole graph. This assumption is often unrealistic when dealing with online social networks, in addition to being extremely time consuming. In this setting, approximations of the collection of frequent patterns are necessary, and sufficient when they come with stringent quality guarantees, such as the ones provided by MaNIACS (see Thm. 4.5).

Parallel and distributed systems for FSPM try to address the scalability issue of mining frequent patterns from very large graphs or when the pattern search space is huge [1, 12, 26, 54, 55, 62]. MaNIACS can be used as a primitive inside these systems, similarly to how sampling-based approximation algorithms for frequent itemset mining [47] have been integrated in MapReduce [46].

Early works in approximate FSPM include the use of graph summaries [21] or heuristics for space pruning [28], but they offer no guarantees. Other works tackled the problem via graph sampling [1, 5, 43], but they also come with no quality guarantees.

Our algorithm samples a set of vertices, but it does *not* use them to build a graph from the sample. Neither does it subgraphs, which is the approach taken by other works on subgraph counting [3, 6–9, 37] or focusing on *output sampling* [11]. To the best of our knowledge, our work is the first to use concepts from statistical learning theory [58] for FPSM. Other works used VC-dimension or other concepts from statistical learning theory for centrality computations [48], for subgraph counting [37], or for itemsets mining [49], but these approaches cannot be easily adapted to FSPM, because this problem is clearly very different from centrality computation, and because the itemsets space is less complex and much easier to

"navigate" than the subgraph space that we consider. In particular, the evaluation of the frequency of an itemset is straightforward and much cheaper than computing the frequency of a subgraph pattern (see Sect. 4). Thus, approaches relying on Rademacher averages for generic pattern families [40] do not perform well for FPSM.

3 PRELIMINARIES

Let us now formally define the important concepts used throughout this work, and the task we are interested in.

3.1 Graph theory concepts

Any graph G we consider is simple (no self loops, no multi-edges), unweighted, undirected, and vertex-labeled, i.e., $G = (V, E, \mathsf{L})$ where L is a function that assigns labels from a fixed set $L = \{\lambda_1, \ldots, \lambda_m\}$ to vertices (unlabeled graphs can be seen as labeled graphs with a single label). For brevity, we usually drop L from the notation, and do not repeat "labeled", but all the graphs we consider are labeled, unless otherwise specified. A graph G is connected iff, for each pair of vertices $v \neq u \in V$, there exists a sequence of vertices $u, w_1, \ldots, w_n, v \in V$ and a sequence of edges $(u, w_1), \ldots, (w_i, w_{i+1}), (w_n, v) \in E$ for $1 \leq i \leq n-1$.

For a fixed $k \in \mathbb{N}$, let $\mathcal{P}$ be the set of all possible *connected* graphs with up to k vertices and whose vertices have labels in L. We call *patterns* the elements of $\mathcal{P}$. Let $S \subseteq V$ be a subset of vertices of a graph $G = (V, E)$, and let $E(S) \doteq \{(u, v) \in E : u, v \in S\}$. We say that $G_S \doteq (S, E(S))$ is the subgraph of G *induced by* S.[1] For $k > 0$, we define $\mathcal{C}$ to be the set of all *connected induced subgraphs with up to k vertices* in G.[2] All subgraphs we consider are connected induced subgraphs, unless stated otherwise.

Two graphs $G' = (V', E', \mathsf{L}')$ and $G'' = (V'', E'', \mathsf{L}'')$ are *isomorphic* if there exists a bijection $\mu : V' \to V''$ such that $(u, v) \in E'$ iff $(\mu(u), \mu(v)) \in E''$ and the mapping μ *preserves the vertex labels*, i.e., $\mathsf{L}'(u) = \mathsf{L}''(\mu(u))$, for all $u \in V'$. Isomorphisms from a graph G' to itself are called *automorphisms* and their set is denoted as $\mathsf{Aut}(G')$.

Given a pattern $P = (V_P, E_P)$ in $\mathcal{P}$ and a vertex $v \in V_P$, the *orbit* $B_P(v)$ *of* v *in* P is the subset of V_P that is mapped to v by any automorphism of P, i.e.,

$$B_P(v) \doteq \{u \in V_P : \exists \mu \in \mathsf{Aut}(P) \text{ s.t. } \mu(u) = v\} .$$

The orbits of P form a partitioning of V_P, for each $u \in B_P(v)$, it holds $B_P(u) = B_P(v)$, and all vertices in $B_P(v)$ have the same label.

3.2 Frequent patterns

Among the many measures of frequency for subgraphs [20, 34, 35], we adopt the *minimum node image-based* (MNI) support [10] metric to count the occurrences of the patterns. MNI is *anti-monotonic*: any pattern (e.g., a triangle) has MNI support no larger than any of its subgraphs (e.g., an edge) (see Sect. 4.1), which avoids counter-intuitive results. Computationally, anti-monotonicity enables apriori-like algorithms [56] to prune the pattern space.

Let $G = (V, E)$ be a graph, and let $S \subseteq V$ be a subset of vertices. For any orbit A of any pattern $P \in \mathcal{P}$, let the *image set* $\mathsf{Z}_S(A)$ *of* A *on* S be the subset of S containing all and only the vertices $v \in S$ for which there exists an isomorphism μ from an induced subgraph $G' = (V', E') \in \mathcal{C}$ with $v \in V'$ to P such that $\mu(v) \in A$. Formally,

$$\mathsf{Z}_S(A) \doteq \{v \in S : \exists \text{ isomorphism } \mu : (V', E') \to P \text{ s.t. } (V', E') \in \mathcal{C} \wedge v \in V' \wedge \mu(v) \in A\} . \quad (1)$$

The *orbit frequency* $\mathsf{c}_S(A)$ *of* A *on* S is the ratio between the size of its image set $\mathsf{Z}_S(A)$ and the size of S, i.e.,

$$\mathsf{c}_S(A) \doteq \frac{|\mathsf{Z}_S(A)|}{|S|} . \quad (2)$$

The *(relative) MNI-frequency* $\mathsf{f}_S(P)$ *of* $P \in \mathcal{P}$ *on* S is the minimum orbit frequency on S for any orbit of P, i.e.,

$$\mathsf{f}_S(P) \doteq \min\{\mathsf{c}_S(A) : A \text{ is an orbit of } P\} . \quad (3)$$

When dealing with approximations, it is more straightforward to reason about this quantity than about the (absolute) MNI-support (i.e., the minimum size of the image set of any orbit of P).[3] Given a (large) graph $G = (V, E)$, and a *minimum frequency threshold* $\tau \in (0, 1)$, for any $S \subseteq V$, the set $\mathsf{FP}_S(\tau)$ *of* τ*-frequent patterns on* S contains all and only the patterns with frequency on S greater than or equal to τ, *together with their frequencies*, i.e.,

$$\mathsf{FP}_S(\tau) \doteq \{(P, \mathsf{f}_S(P)) : P \in \mathcal{P} \wedge \mathsf{f}_S(P) \geq \tau\} . \quad (4)$$

The task we are interested in requires finding $\mathsf{FP}_V(\tau)$. Due to the exponential number of candidate patterns, and to the hardness of evaluating the subgraph isomorphisms, finding this collection is challenging. An *approximate solution* $\mathcal{Q}$ is sufficient, in many cases, provided it comes with stringent quality guarantees, such as *(i)* the lack of false negatives, i.e., every pattern in $\mathsf{FP}_V(\tau)$ also appears in $\mathcal{Q}$, and *(ii)* guarantees on the frequency estimation error. MaNIACS, outputs a set $\mathcal{Q}$ with such guarantees (see Thm. 4.5), by sampling a subset of vertices from V, which are then used to approximate the frequency of patterns of increasing size, while exploiting the anti-monotonicity of the frequency measure to prune the search space. To understand the trade-off between the sample size and the accuracy ε of the approximation, we use concepts and results from statistical learning theory [58], described next.

3.3 Empirical VC-dimension and η-samples

We give here the main definitions and results about empirical VC-dimension, tailored to our setting. For a general discussion, see the textbook by Shalev-Shwartz and Ben-David [53, Ch. 6].

A *range space* is a pair $(\mathcal{D}, \mathcal{R})$ where $\mathcal{D}$ is a finite ground set of elements called *points* and $\mathcal{R}$ is a family of subsets of $\mathcal{D}$ called *ranges*. For any $A \subseteq \mathcal{D}$, let the *projection* $\mathsf{P}_{\mathcal{R}}(A)$ *of* $\mathcal{R}$ *on* A be the set $\mathsf{P}_{\mathcal{R}}(A) \doteq \{r \cap A : r \in \mathcal{R}\} \subseteq 2^A$. When $\mathsf{P}_{\mathcal{R}}(A) = 2^A$, i.e., when the projection contains all the proper and improper subsets of A, then we say that A is *shattered* by $\mathcal{R}$. Given a subset $Y \subseteq \mathcal{D}$, the *empirical Vapnik-Chervonenkis (eVC) dimension* $\mathsf{E}_Y(\mathcal{R})$ *of* $\mathcal{R}$ *on* Y is the size of the *largest shattered subset of* Y [58]. The *VC-dimension* of $\mathcal{R}$ is the empirical VC-dimension of $\mathcal{R}$ on $\mathcal{D}$.

The concept of η*-sample for* $(\mathcal{D}, \mathcal{R})$ is crucial for our work. For $0 < \eta < 1$, a subset $A \subseteq \mathcal{D}$ is an η*-sample for* $(\mathcal{D}, \mathcal{R})$ if it holds

$$\left| \frac{|R|}{|\mathcal{D}|} - \frac{|A \cap R|}{|A|} \right| \leq \eta, \text{ for every } R \in \mathcal{R} . \quad (5)$$

[1] Our algorithm can also handle *edge-induced* subgraphs, with minor modifications, but we do not discuss them here due to space limitations.

[2] $\mathcal{C}$ depends on k and G but we do not use them in the notation to keep it light.

[3] Henceforth, we use "frequency" to refer to the MNI-frequency.

Given an η-sample A, we can estimate the relative sizes of any range $R \in \mathcal{R}$ w.r.t. the domain (i.e., the first term on the l.h.s.) with its relative size w.r.t. A (the second term on the l.h.s.), and the estimate is guaranteed to be no more than η-far from its exact value.

Given a sample size s, let $\mathcal{T}$ be a collection of s points sampled from $\mathcal{D}$ independently and uniformly at random (with or without replacement). Knowing an upper bound d to the empirical VC-dimension of $(\mathcal{D}, \mathcal{R})$ on $\mathcal{T}$ allows the computation of an η such that, probabilistically, $\mathcal{T}$ is an η-sample for $(\mathcal{D}, \mathcal{R})$.

Theorem 3.1 (31). *Let $\phi \in (0,1)$ be an acceptable failure probability. For $(\mathcal{D}, \mathcal{R})$, s, $\mathcal{T}$, and d as above, it holds that, with probability at least $1-\phi$ (over the choice of $\mathcal{T}$), $\mathcal{T}$ is an η-sample for $(\mathcal{D}, \mathcal{R})$ for*

$$\eta = \sqrt{\frac{c\left(d + \ln \frac{1}{\phi}\right)}{s}}, \tag{6}$$

where c is a universal constant.[4]

When the upper bound d to $\mathsf{E}_{\mathcal{T}}(\mathcal{R})$ is computed from $\mathcal{T}$, the value η from (6) depends only on $\mathcal{T}$ and on ϕ, i.e., it is a *sample-dependent* upper bound to the maximum difference, over all ranges, between the relative sizes of the ranges w.r.t. the sample and the relative sizes w.r.t. the domain, i.e., to the l.h.s. of (5).

4 APPROXIMATE FSPM

We now present MaNIACS, our algorithm for mining high-quality approximations to $\mathsf{FP}_V(\tau)$ through sampling.[5] At a very high level, MaNIACS draws a sample S from V and uses the orbit frequencies, the frequency of the patterns on S, and the eVC-dimension of appropriately-designed range spaces, to derive the output quality guarantees. MaNIACS does *not* consider the subgraph of G induced by S. Rather, it always considers the whole graph G when checking the existence of isomorphisms from patterns to induced subgraphs of G. The sample is instead used to compute $\mathsf{f}_S(P)$ as an estimation of $\mathsf{f}_V(P)$, as obtaining the former is faster given that $|S| \ll |V|$.

The following fact is at the basis of MaNIACS, and it is immediate from the definition of MNI-frequency (see (3)).

Fact 1. *Given $P \in \mathcal{P}$ and $S \subseteq V$, let ε be such that it holds*

$$|\mathsf{c}_S(A) - \mathsf{c}_V(A)| \leq \varepsilon, \text{ for every orbit } A \text{ of } P. \tag{7}$$

Then it must be $|\mathsf{f}_S(P) - \mathsf{f}_V(P)| \leq \varepsilon$. .

This corollary suggests how to identify patterns that cannot be frequent, and that can therefore be pruned.

Corollary 4.1. *Let P, S, and ε as in Fact 1. If it holds $\mathsf{f}_S(P) < \tau - \varepsilon$, then it must be $\mathsf{f}_V(P) < \tau$, i.e., $P \notin \mathsf{FP}_V(\tau)$.*

Statistical learning theory gives us the tools to compute values ε which satisfy the condition from (7). Given the exponential number of patterns, it would be unfeasible to compute $\mathsf{f}_S(P)$ for every $P \in \mathcal{P}$. Thus, we rely on properties of the orbit frequency and of the MNI-frequency functions (see Sect. 4.1) to *prune the space of patterns*, in an *apriori-like way*, and therefore to avoid computing the frequencies of orbits whose pattern is not in $\mathsf{FP}_V(\tau)$.

[4] In our experiments, we follow Löffler and Phillips [32] and use $c = 0.5$.
[5] Due to space constraints, all the proofs are in Appendix A.2.

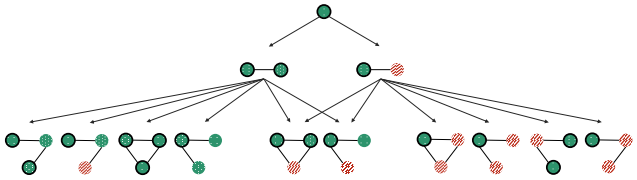

Figure 1: Examples of parent-child relations for orbits (labels represented as colors). We represent each orbit using its pattern with the vertices of the orbit in a thicker border.

4.1 Search space and frequency properties

We now define a partial order between patterns in $\mathcal{P}$: we say that P'' is a child of P' if *(i)* P'' has exactly one more vertex than P'; and *(ii)* there exists an isomorphism between P' and some induced subgraph of P''. When P'' is a child of P' we say that P' is a *parent* of P''. A pattern may have multiple parents, while patterns with a single vertex have no parent. The anti-monotone property of the MNI-frequency gives the following fact:

Fact 2 ([10]). *For any pattern $P \in \mathcal{P}$, any pattern $Q \in \mathcal{P}$ that is a child of P, and any $S \subseteq V$, it holds that $\mathsf{f}_S(Q) \leq \mathsf{f}_S(P)$.*

We define a similar parent-child relation between pairs of orbits. Given two distinct patterns $P, Q \in \mathcal{P}$ and two orbits B_P and B_Q of each respectively, we say that B_Q is the *child* of B_P iff Q is a child of P and there is a subgraph isomorphism from P to Q that maps at least one vertex of B_P to a vertex of B_Q. When B_Q is the child of B_P, we say that B_P is the *parent* of B_Q, and denote all the children of B_P as $\mathsf{C}(B_P)$. Figure 1 shows some examples of the parent-child relationships. An orbit can have multiple parents, and the orbits of patterns containing a single vertex have no parent. Our algorithm leverages the following important property of this relationship, which is immediate from the definition, to quickly prune the search space of patterns.

Fact 3. *Let A and D be two orbits such that D is a child of A. Then, for any $S \subseteq V$, it holds $\mathsf{Z}_S(D) \subseteq \mathsf{Z}_S(A)$.*

4.2 The frequent patterns range spaces

We now define an appropriate set of range spaces and show how to compute bounds to their eVC-dimensions. Given $\tau \in (0,1]$, let $\mathcal{F}_i$ for $i = 1, \ldots, k$ be the set of patterns with i vertices that belong to $\mathsf{FP}_V(\tau)$.[6] Let $\mathcal{R}_i$ be the set whose elements are the image sets on V of all the orbits of all the patterns in $\mathcal{F}_i$, $1 \leq i \leq k$, i.e.,

$$\mathcal{R}_i \doteq \{\mathsf{Z}_V(A) \ : \ A \text{ is an orbit of } P \in \mathcal{F}_i\} \ .$$

Henceforth, we use the range spaces $(V, \mathcal{R}_i)$, $1 \leq i \leq k$. The relevance of these range spaces is clear when looking at Equation (2). We now show novel results to upper bound the eVC-dimension of $(V, \mathcal{R}_i)$ on any $S \subseteq V$. MaNIACS computes such bounds to derive the approximation guarantees and to prune the search space.

The following two results are presented in the most general form because they hold for *any* range space. We later tailor them for our case, and discuss how to compute the presented bounds efficiently.

[6] The sets $\mathcal{F}_i$, $1 \leq i \leq k$, depend on G and on τ, but the notation does not reflect these dependencies to keep it light.

Algorithm 1: MaNIACS

Input: Graph $G = (V, E)$, maximum pattern size k, frequency threshold τ, sample size s, failure probability δ
Output: A set $\mathcal{Q}$ with the properties from Thm. 4.5

1 $S \leftarrow$ `drawSample`(V, s)
2 $\mathcal{Q} \leftarrow \emptyset$; $i \leftarrow 1$
3 $\mathcal{H}_1 \leftarrow \{P \in \mathcal{P} : P \text{ has a single vertex}\}$
4 **while** $i \leq k$ **and** $\mathcal{H}_i \neq \emptyset$ **do**
5 $\quad \mathcal{Z}_i \leftarrow$ `getImageSets`$(\mathcal{H}_i, S, \tau)$
6 $\quad$ **do**
7 $\quad\quad b_i^* \leftarrow$ `getEVCBound`$(\mathcal{Z}_i)$
8 $\quad\quad \varepsilon_i \leftarrow$ `getEpsilon`$(b_i^*, \delta/k)$
9 $\quad\quad \mathcal{H}'_i \leftarrow \mathcal{H}_i$
10 $\quad\quad \mathcal{H}_i \leftarrow \{P \in \mathcal{H}_i : \mathsf{f}_S(P) \geq \tau - \varepsilon_i\}$
11 $\quad$ **while** $\mathcal{H}'_i \neq \mathcal{H}_i$ **and** $\mathcal{H}_i \neq \emptyset$
12 $\quad \mathcal{Q} \leftarrow \mathcal{Q} \cup \{(P, \mathsf{f}_S(P), \varepsilon_i) : P \in \mathcal{H}_i\}$
13 $\quad$ **if** $i < k$ **then** $\mathcal{H}_{i+1} \leftarrow$ `createChildren`$(\mathcal{H}_i, \mathcal{Z}_i)$
14 $\quad i \leftarrow i + 1$
15 **return** $\mathcal{Q}$

Lemma 4.2. *Let $(\mathcal{D}, \mathcal{R})$ be a range space, and let $\mathcal{T} \subseteq \mathcal{D}$. Consider the set $\mathcal{R}_\mathcal{T} \doteq \{\mathcal{T} \cap R : R \in \mathcal{R}\}$. Let g^* be the maximum g such that $\mathcal{T}$ contains at least g points each appearing in at least 2^{g-1} sets from $\mathcal{R}_\mathcal{T}$. If, for at least one set B of such g^* points, there exist a set $Z_B \in \mathcal{R}_\mathcal{T}$ such that $B \subseteq Z_B$, then $\mathsf{E}_\mathcal{T}(\mathcal{R})$ is at most g^*, otherwise it is at most $g^* - 1$.*

Lemma 4.3. *Let $(\mathcal{D}, \mathcal{R})$ and $\mathcal{T}$ as in Lemma 4.2. Let $R_1, \ldots, R_{|\mathcal{R}|}$ be a labeling of the ranges in $\mathcal{R}$ such that $|R_w \cap \mathcal{T}| \geq |R_u \cap \mathcal{T}|$ for $1 \leq w < u \leq |\mathcal{R}|$. Let $(a_j)_{j=1}^{\ell}$ be a non-increasing sequence of $\ell \geq |\mathcal{R}|$ naturals such that $a_j \geq \left|R_j \cap \mathcal{T}\right|$ for $1 \leq j < |\mathcal{R}|$ and $a_j \geq \left|R_{|\mathcal{R}|} \cap \mathcal{T}\right|$ for $|\mathcal{R}| \leq j \leq \ell$.*

Let h^* be the maximum natural h such that, for every $0 \leq j < h$, if we let $c_j = \sum_{z=0}^{j} \binom{h}{z}$,[7] it holds $a_{c_j} \geq h - j$. Then, $\mathsf{E}_\mathcal{T}(\mathcal{R}) \leq h^*$.

While the lemma above may seem complex at first, its proof is essentially an application of the pigeonhole principle, and the procedure to compute the bound h^* from the sequence $(a_i)_{i=1}^{\ell}$ is straightforward, as we discuss in Sect. 4.3.

For $\lambda \in L$, let $\mathcal{R}_{i,\lambda}$ be the subset of $\mathcal{R}_i$ containing all and only the image sets of the orbits whose vertices have all label λ. Clearly each $(V, \mathcal{R}_{i,\lambda})$ is a range space. The following result ties the empirical VC-dimension of these range spaces to that of $(V, \mathcal{R}_i)$.

Lemma 4.4. *For any $S \subseteq V$, it holds $\mathsf{E}_S(\mathcal{R}_i) = \max_{\lambda \in L} \mathsf{E}_S(\mathcal{R}_{i,\lambda})$.*

Lemma 4.4 says that $\mathsf{E}_S(\mathcal{R}_i)$ is, in some sense, *independent from the number $|L|$ of labels*, which is surprising, from a theoretical point of view. MaNIACS computes upper bounds to $\mathsf{E}_S(\mathcal{R}_{i,\lambda})$, $\lambda \in L$, using Lemmas 4.2 and 4.3, and then leverages Lemma 4.4 to derive an upper bound to $\mathsf{E}_S(\mathcal{R}_i)$.

4.3 MaNIACS, the algorithm

The intuition behind MaNIACS is the following. It creates a sample S by drawing vertices independently and uniformly at random without replacement from V. Then it computes from S a value ε_i such that S is an ε_i-sample for the range space $(V, \mathcal{R}_i)$, for $1 \leq i \leq k$. For such an ε_i, thanks to (6) and (2), it holds, for any $P \in \mathcal{F}_i$, that

$$\mathsf{c}_S(A) \geq \mathsf{c}_V(A) - \varepsilon_i \geq \tau - \varepsilon_i \text{ for any orbit } A \text{ of } P,$$

which implies that $\mathsf{f}_S(P) \geq \tau - \varepsilon_i$. This lower bound to the possible frequency of $P \in \mathcal{F}_i \subseteq \mathsf{FP}_V(\tau)$ on S allows us to determine which patterns may actually belong to $\mathsf{FP}_V(\tau)$ and which ones cannot.

Unfortunately, the sets $\mathcal{F}_i$, $1 \leq i \leq k$, are *not known a priori*, as if they were, we could use them to exactly obtain $\mathsf{FP}_V(\tau)$. MaNIACS therefore computes a *superset* $\mathcal{H}_i$ of each set. It uses the sizes of the image sets on S of the orbits of the patterns in $\mathcal{H}_i$ to compute an upper bound to the eVC-dimension of $(V, \mathcal{R}_i)$, thanks to Lemmas 4.2 to 4.4. By plugging this upper bound in (6), it gets a value ε_i such that S is (probabilistically) an ε_i-sample for $\mathcal{F}_i$.

We first present a simplified version of MaNIACS (pseudocode in Alg. 1), and discuss more details in Sect. 4.3.2. MaNIACS takes as input a graph $G = (V, E)$, a maximum pattern size k, a minimum frequency threshold τ, a sample size s, and an acceptable failure probability δ. It outputs a set $\mathcal{Q}$ with the following properties (proof in App. A.2).

Theorem 4.5. *With probability at least $1 - \delta$ over the choice of S, the output $\mathcal{Q}$ of MaNIACS contains a triplet $(P, \mathsf{f}_S(P), \varepsilon_P)$ for every $P \in \mathsf{FP}_V(\tau)$ such that $|\mathsf{f}_S(P) - \mathsf{f}_V(P)| \leq \varepsilon_P$.*

Algorithm 2: `getEVCBound`

Input: Bag $\mathcal{Z}_i$ of image sets $\mathsf{Z}_A(S)$, $\forall$ orbit A of each pattern in $\mathcal{H}_i$
Output: A value $b_i^* \geq \mathsf{E}_S(\mathcal{R}_i)$

1 **foreach** $\lambda \in L$ **do**
2 $\quad D_\lambda \leftarrow$ set of image sets in $\mathcal{Z}_i$ of orbits of vertices with label λ
3 $\quad M \leftarrow |S|$-vector with element $(v, |\{Z \in D_\lambda : v \in Z\}|), \forall v \in S$
4 $\quad$ sort M in decreasing order of the 2^{nd} component
$\quad$ `// Denote with` (v_i, q_i) `the` i`-th element of` M
5 $\quad g_\lambda^* \leftarrow \max\{g : v_g \geq 2^{g-1}\}$
6 $\quad \gamma \leftarrow \max\{i : v_i > 2^{g_\lambda^* - 1}\}$
7 $\quad$ **if** $\nexists Q \subseteq \{v_1, \ldots, v_\gamma\}$, $|Q| = g_\lambda^*$, *s.t.* $\exists Z \in D_\lambda$ *s.t.* $Q \subseteq Z$ **then** $g_\lambda^* \leftarrow g_\lambda^* - 1$
8 $\quad N \leftarrow |D_\lambda|$-vector with element $|Z|, \forall Z \in D_\lambda$
9 $\quad$ sort N in decreasing order
$\quad$ `// Denote with` a_i `the` i`-th element of` N
10 $\quad h_\lambda^* \leftarrow \min\{a_1, \lfloor \log_2(|D_\lambda| + 1) \rfloor\}$
11 $\quad$ **while** $h_\lambda^* > 1$ **do**
12 $\quad\quad$ **foreach** $j \in \{0, \ldots, h_\lambda^* - 1\}$ **do** $c_j \leftarrow \sum_{z=0}^{j} \binom{h_\lambda^*}{z}$
13 $\quad\quad$ **if** $\nexists j \in \{0, \ldots, h_\lambda^* - 1\}$ *s.t.* $a_{c_j} < h_\lambda^* - j$ **then break**
14 $\quad\quad$ **else** $h_\lambda^* \leftarrow h_\lambda^* - 1$
15 **return** $\max_{\lambda \in L} \min\{g_\lambda^*, h_\lambda^*\}$

The algorithm starts by initializing the empty set $\mathcal{Q}$, which will contain the output (line 2) and by creating the sample $S = \{v_1, \ldots, v_s\}$ of s vertices by drawing them independently and uniformly at random from V (line 1).

MaNIACS keeps, for every $1 \leq i \leq k$, a superset $\mathcal{H}_i$ of the set $\mathcal{F}_i$. The first such superset $\mathcal{H}_1$ is initialized to contain every pattern of a single vertex (line 3). The algorithm then enters a loop (lines 4–14) which is repeated until i is greater than k or until $\mathcal{H}_i$ is empty. At every iteration, MaNIACS first calls `getImageSets` to obtain the collection $\mathcal{Z}_i$ of the image sets $\mathsf{Z}_S(A)$ on S of every orbit A of every

[7] We define $\binom{q}{0} = 1$ for any q.

pattern $P \in \mathcal{H}_i$ (pseudocode in Alg. 3 in Appendix). Each set $\mathsf{Z}_S(A)$ is obtained by running an existence query (pseudocode in Alg. 4 in Appendix) for each vertex v in the sub-sample S_A, to determine whether v belongs to at least one subgraph isomorphic to P. The existence query is a recursive function that incrementally builds a dictionary M, by inserting, at each iteration, a new candidate match from a pattern vertex to a graph vertex. If a match can be found for each vertex of the pattern, the query returns true, and v is inserted into $\mathsf{Z}_S(A)$. A match z for a pattern vertex u is added to M only if it is consistent with the matches already in M, i.e., if the pattern vertices already matched and u are connected in the same way as the graph vertices mapped to them (see line 7 in Alg. 4 in Appendix). If we wish to find the edge-induced subgraph isomorphic to P, we just need to modify this consistency check.

MaNIACS then enters a do-while loop (lines 6–11), whose dual purpose is to compute an ε_i such that S is a ε_i-sample for $(V, \mathcal{R}_i)$, and to iteratively refine $\mathcal{H}_i$ as a superset of $\mathcal{F}_i$. To compute ε_i, we first need an upper bound to the eVC-dimension of $(V, \mathcal{R}_i)$ on S (line 7). This bound is computed by the `getEVCBound` function (pseudocode in Alg. 2). For each label $\lambda \in L$, let D_λ be the subset of $\mathcal{Z}_i$ containing the distinct sets associated to orbits of vertices with label λ (line 2 of Alg. 2). First, g^*_λ from Lemma 4.2 is computed (lines 3–7), by taking into account the number of image sets in D_λ in which each vertex $v \in S$ appears. Then, the value h^*_λ from Lemma 4.3 is computed (lines 8–14). The value b^*_i returned by `getEVCBound` is the *maximum*, over $\lambda \in L$, of $\min\{h^*_\lambda, g^*_\lambda\}$.

MaNIACS uses b^*_i in (6) together with $\eta = \delta/k$ to obtain ε_i (line 8 of Alg. 1). The value ε_i is used to refine $\mathcal{H}_i$ by removing from it any pattern whose frequency in S is lower than $\tau - \varepsilon_i$ (line 10), as they cannot belong to $\mathsf{FP}_V(\tau)$ (see proof of Thm. 4.5 in App. A.2). The frequencies can be obtained from $\mathcal{Z}_i$. This refinement process is iterated until no more patterns can be pruned, i.e., $\mathcal{H}'_i = \mathcal{H}_i$, or $\mathcal{H}_i$ becomes empty (line 11). At this point, the patterns still in $\mathcal{H}_i$ are added to the output set $\mathcal{Q}$, together with their frequencies on S and ε_i (line 12). If $i < k$, MaNIACS creates the set $\mathcal{H}_{i+1}$ to contain the patterns on $i+1$ vertices whose parents are *all* in $\mathcal{H}_i$, by calling the function `createChildren` (line 13). Thanks to Fact 2, this requirement ensures that $\mathcal{H}_i$ is the smallest superset of $\mathcal{F}_i$ that can be obtained on the basis of the currently available information. At this point, the current iteration of the while loop is completed. When the loop condition (line 4) is no longer satisfied, the algorithm returns the set $\mathcal{Q}$ (line 15).

4.3.1 Generating the next set of patterns. MaNIACS takes an apriori-like, level-wise approach that explores a subset of the "level" i of the pattern search space containing the patterns on i vertices, after having explored and pruned the level $i-1$. This subset is generated by the `createChildren` function on the basis of the non-pruned patterns at level $i-1$. In particular, this function extends each non-pruned pattern in the level $i-1$, by adding an edge to every possible position. As this procedure may generate the same pattern multiple times (a pattern can have multiple parents), we identify the canonical form of each pattern generated [27] and prune duplicate patterns. For each distinct extension, we need to compute its orbits, in order to compute their image sets (`getImageSets` function from Alg. 1). The generation of the orbits and patterns in MaNIACS follows steps similar to the procedure by Melckenbeeck et al. [33], adapted to take into consideration the fact that we are working with labeled graphs. We defer the details to the extended version of this work due to space limitations.

4.3.2 Additional Pruning. An efficient pattern mining algorithm must take any chance for pruning the search space. This requirement is particularly important when dealing with subgraphs, because computing the collection $\mathcal{Z}_i$ (line 5) of image sets of the orbits of a pattern $P \in \mathcal{H}_i$ is particularly expensive. We now describe how MaNIACS can prune as much as possible, as early as possible, without any effect on its quality guarantees.

Before delving into pruning, we comment on the computation of the set $\mathsf{Z}_S(A)$ for an orbit A of a pattern $P \in \mathcal{H}_i$. Computing $\mathsf{Z}_S(A)$ does not require to explicitly verify whether $v \in \mathsf{Z}_S(A)$ for every $v \in S$. Rather, the algorithm can create, when initializing $\mathcal{H}_i$, a subset $S_A \subseteq S$ for every orbit as above such that it holds

$$\mathsf{Z}_S(A) \subseteq S_A \ . \tag{8}$$

For $i = 1$, this set contains all and only the vertices in S whose label is the same as the label of the single vertex of the patterns. For $1 < i \leq k$, we can use Fact 3: when creating $\mathcal{H}_i$ on line 13, the algorithm can associate to each orbit A of a pattern in the set $\mathcal{H}_i$ returned by `createChildren`, a set S_A obtained by taking the intersections of the image sets $\mathsf{Z}_S(B)$ on S of every parent B of the orbit A, which are available from $\mathcal{Z}_i$, i.e.,

$$S_A \doteq \bigcap_{B \text{ parent of } A} \mathsf{Z}_S(B) \ .$$

The computation of these sets can be done in the call to the `createChildren` function on line 13 of Alg. 1, for $1 < i \leq k$, and just before the starting of the loop on line 4 for $i = 1$. The properties of the orbit child-parent relation (Fact 3) therefore enable a faster computation of the collection $\mathcal{Z}_i$ because $\mathsf{Z}_S(A) = \mathsf{Z}_{S_A}(A)$, and we only need to check for subgraph isomorphisms involving S_A, which may be much smaller than S. We remark that, thanks to Equation (1), we need to find only one subgraph isomorphism for each vertex in S_A, rather than enumerating all of them.

Maintaining the sets S_A for every orbit A of a pattern $P \in \mathcal{H}_i$ allows for pruning $\mathcal{H}_i$ *before* even computing the collection $\mathcal{Z}_i$ of the image sets. The idea is that the sets S_A can be used in place of the exact image set $\mathsf{Z}_S(A)$ to compute an upper bound to the eVC-dimension of $(V, \mathcal{R}_i)$ on S. It holds by definition that $S_A \supseteq \mathsf{Z}_S(A)$ for every orbit A, so a call to `getEVCBound` (with the minor tweak of not getting rid of duplicated sets on line 2 of Alg. 2) using the collection of these supersets would return a valid upper bound $\hat{b}^*_i$ to the eVC-dimension of $(V, \mathcal{R}_i)$ on S. Thus, a call to `getEpsilon` with parameters $\hat{b}^*_i$ and δ/k, would return a value $\hat{\varepsilon}_i$ that is not smaller than the value ε_i that would be returned if we used b^*_i. We can then further improve MaNIACS by adding a do-while loop as the first step of every iteration of the loop on lines 4–14. This inner loop is exactly the same as the do-while loop on lines 6–11, but with $\hat{b}^*_i$ being used in place of b^*_i. At each iteration of this loop, some orbits and therefore some patterns may be pruned from $\mathcal{H}_i$ because not frequent enough, resulting potentially in a lower bound to the eVC-dimension, thus in a lower $\hat{\varepsilon}_i$, creating a positive feedback loop. The improved algorithm has exactly the same properties as the vanilla MaNIACS, i.e., Thm. 4.5 holds.

The pruning strategies above can be incorporated in the call to the `createChildren` function. The call to `getImageSets` on line 5 also offers opportunities for pruning. MaNIACS computes one image set $\mathsf{Z}_S(A) = \mathsf{Z}_{S_A}(A)$ at a time, and it evaluates whether $v \in S_A$ belongs to the image set, one vertex v at a time. With the goal of maximizing pruning, we can first sort the orbits of a pattern by increasing size of their sets, and then compute the image sets of the orbits according to the obtained order. We can stop *early* the identification of the image set of an orbit A, if the sum between the number of vertices in S_A that are left to be examined, and the number of vertices in S_A that we found to belong to $\mathsf{Z}_{S_A}(A)$, divided by the size of S, is less than $\tau - \varepsilon_i$. Thus, we can also skip computing the image sets of the remaining orbits of the same pattern, and we can remove the pattern from $\mathcal{H}_i$.

Pruning is extremely important for MaNIACS, not only for *computational* efficiency reasons, but also for *statistical* efficiency reasons, as aggressive pruning leads to better bounds to the eVC-dimension, and therefore to a smaller bound to the maximum estimation error, i.e., to a better approximation quality guarantee.

5 EXPERIMENTAL EVALUATION

In this section, we aim to (1) show in which cases a sampling-based approximate algorithm can be preferred to an exact algorithm, (2) discuss how much information on the graph is required to obtain accurate results, (3) assess how strict our theoretical upper bounds are with respect to the actual estimation errors, and (4) evaluate the scalability of MaNIACS. Additional results are reported in App. A.3.

Datasets. We consider 5 real world networks, whose characteristics are summarized in Table 2 in App. A.1. All the datasets are publicly available. Citeseer [18] is a citation network where nodes are publications and edges are citations. Node labels denote Computer Science areas. Phy-Cit [30] is a citation network covering e-print arXiv HEP-PH papers from 1993 to 2003. Node labels corresponds to year of publication of the paper. MiCo [18] is a co-authorship network: nodes represent authors and edges collaborations between them. Node labels indicate each author's research field. Patents[30] is a network of all the citations in utility patents granted between 1975 and 1999. Each node label indicates the year the patent was granted. YouTube [13] is a network with nodes representing videos. Two videos are connected if they are related. The vertex label is a combination of a video's rating and length.

Experimental Environment. We run our experiments on a 32-Core (2.54 GHz) AMD EPYC 7571 Amazon AWS instance, with 250GB of RAM, and running Amazon Linux 2. MaNIACS and the exact algorithm are implemented in Java 1.8, and we made the code publicly available (see App. A.1).

Parameter Configuration. We test several sample sizes and frequency thresholds, while the parameters k, c,and δ were always set to 5, 0.5, and 0.1, respectively. The last two have minimal impact on the performance (see (6)). Given that the sample extracted from the graph highly affects the quality of the results, we perform each test 5 times and report the averages. The exact algorithm searches for the frequent patterns in the whole graph, without the need to compute any ε_i. We do not compare with other exact Java implementations such as GraMi [18] because they search for edge-induced patterns, and do not compute the pattern exact frequencies.

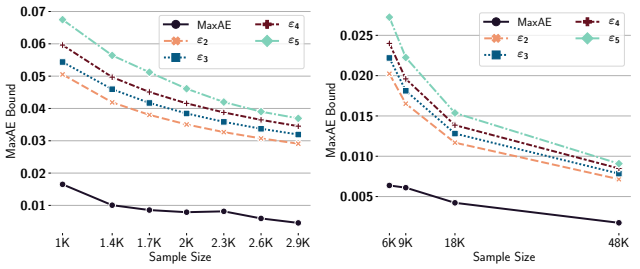

Figure 2: MaxAE, ε_2, ε_3, ε_4, ε_5, for various sample sizes, min. freq. threshold $\tau = 0.16$, in Citeseer (left) and MiCo (right).

Evaluation Metrics. We evaluate the output quality in terms of:

- the *Maximum Absolute Error (MaxAE)* in the frequency estimations of the patterns (i.e., what MaNIACS guarantees (Thm. 4.5));
- the *Mean Absolute Error (MAE)* in the frequency estimations;
- *Precision*, i.e., the fraction of returned patterns that are actually frequent; and
- *Kendall's rank correlation coefficient*, i.e., the correlation between the ordered vectors of frequency estimates and actual frequencies. Values close to 1 indicate strong agreement between the two rankings, while values close to -1 indicate disagreement. When $\mathsf{FP}_V(\tau) = \emptyset$, we set its value to 0.

We do not report the *Recall*, i.e., the fraction of frequent patterns returned by MaNIACS, because Thm. 4.5 guarantees a perfect recall.

MaxAE vs epsilon values. Figure 2 displays the MaxAE of MaNIACS in Citeseer (left) and MiCo (right), together with its theoretical upper bounds ε_i computed by MaNIACS, at varying sample size, and with $\tau = 0.16$. At this frequency threshold, the upper bounds are quite large, and in the worst case, they lead to values of $\tau - \varepsilon_i$ close to 0.07 in Citeseer, and 0.03 in MiCo. As a consequence, MaNIACS explores a large number of unnecessary patterns. Nonetheless, the actual MaxAE achieved by MaNIACS is at least 2.5 times lower than the upper bound, which implies that MaNIACS works even better in practice than what is guaranteed by the analysis. This fact is not surprising as some of the bounds we use are pretty loose, and improving them is an important direction for future work. We observe similar results for the other datasets.

Accuracy. According to (6), the upper bounds ε_i decrease as the sample size grows: with larger samples, MaNIACS obtains lower upper bounds and can perform a better pruning of superfluous patterns. As expected, for small graphs, and especially when the patterns have small frequencies, the sample size required to achieve good frequency estimations can be close to the graph size. In Citeseer for example, where the patterns have frequencies below 0.173, for $\tau = 0.13$, a sample size equal to 1.4k (roughly 42% of the graph size) leads to a precision of 0.72, but a sample size close to 80% of the graph size is required to discover that no pattern of size greater than 2 is actually frequent (see Table 1, line 4). In this case, MaNIACS achieves a perfect precision. Small graphs are not really the target for a sampling algorithm anyway. In contrast, on Patents, where the patterns have frequencies below 0.367, MaNIACS achieves a precision of 0.71 with a sample size of 9k, which is roughly 0.3% of the graph size, i.e., a very small sample.

A different situation can be observed in Phy-Cit (Table 1, lines

Table 1: MAE, MaxAE, Precision, and Kendall's correlation, for all datasets at fixed freq. thres. τ and varying sample size.

Dataset	τ	s	MAE	MaxAE	Precision	Kendall
Citeseer	0.13	1K	0.003	0.015	0.538	0.760
		1.4k	0.004	0.010	0.720	0.760
		2k	0.004	0.008	0.843	0.800
		2.6k	0.003	0.006	1.000	0.800
Phy-Cit	0.18	9K	0.001	0.006	0.278	0.893
		15k	0.001	0.004	0.390	0.918
		18k	0.001	0.003	0.437	0.929
		21k	0.001	0.002	0.436	0.978
Patents	0.25	6K	0.003	0.006	0.667	0.333
		9k	0.003	0.005	0.717	0.349
		12k	0.002	0.004	0.708	0.374
		18k	0.001	0.003	0.750	0.349
MiCo	0.09	6K	0.002	0.006	0.488	0.952
		9k	0.002	0.006	0.518	0.938
		18k	0.002	0.004	0.612	0.973
		48k	0.001	0.002	0.714	1.000
YouTube	0.09	60K	0.001	0.002	0.632	0.976
		120k	0.000	0.002	0.716	0.988
		240k	0.000	0.001	0.760	0.982
		600k	0.000	0.001	0.823	1.000

5–8). Here MaNIACS returns up to 4 times more patterns than the exact algorithm, and hence the precision is low. However, the frequency estimations are very close to the exact frequencies (max error is 0.001), and therefore the Kendall's correlation is high, with values greater than 0.89 even for a sample size of 9k. MaNIACS does not and cannot offer guarantees on the precision, because the precision depends on the distribution of the exact pattern frequencies around the threshold τ, which is unknown to the algorithm.

MaNIACS performs well at small sample sizes also on MiCo and YouTube. Table 1, lines 9–16, reports the results achieved using a minimum frequency threshold equal to 0.09, for which there are roughly a dozen frequent patterns in both datasets. On MiCo, with a sample size as little as 18k (1.8% of the graph size), the precision is 0.612 and Kendall's correlation is 0.973, in $^2/_7$ of the time of the exact algorithm. Similarly, in YouTube, by sampling only 1.3% of the graph vertices, the precision is 0.632, the Kendall's correlation is 0.976, and the running time is $^1/_{17}$ of the time required by the exact algorithm. These results prove that, even when the number of labels is large, MaNIACS can find good approximations of the frequent patterns, while saving a very significant amount of time. As the pattern size i increases, the number of candidate patterns to explore grows significantly, so the empirical VC-dimension and the corresponding upper bound ε_i are likely to increase as well.

Sampling-based vs exact algorithm. Figure 3 shows the running time of MaNIACS when using different sample sizes, varying τ, compared with the exact algorithm, on Patents (left), and Phy-Cit (right). The advantages of MaNIACS are evident when processing larger graphs such as Patents. Figure 3 (left) shows that MaNIACS is up to 2 orders of magnitude faster than the exact algorithm. The algorithm can achieve this performance because it has to examine fewer than 0.6% of the graph vertices. For small graphs such as Phy-Cit, especially for low frequency thresholds, there is no gain from MaNIACS compared to the exact algorithm. At lower values of τ, the frequency thresholds $\tau - \varepsilon_i$ used to prune the pattern space are close to 0, hence almost all the patterns are deemed frequent.

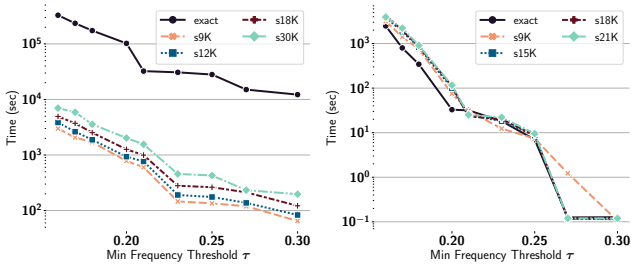

Figure 3: Running times of MaNIACS and exact algorithm on Patents (left) and Phy-Cit (right).

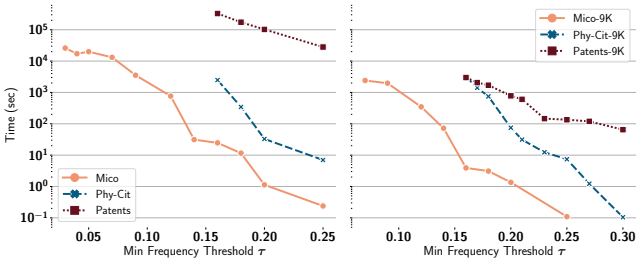

Figure 4: Scalability of the exact algorithm (left), and of MaNIACS with sample size 9000 (right).

The time saved by MaNIACS by examining a smaller number of vertices is counterbalanced by the time required to examine a larger number of candidate patterns, which, in this case, is up to 3 times larger than that of the exact algorithm. Conversely, for larger values of τ, the pruning capacity of MaNIACS is similar to that of the exact algorithm, and therefore the running times are comparable. Once again, these results are expected: sampling algorithms are meant for large datasets, i.e., graphs, not small ones.

Scalability. Figure 4 shows the running time of the exact algorithm (left), and of MaNIACS at $s = 9k$ (right) (YouTube and Citeseer are not reported because respectively too large or too small to have meaningful results at this sample size). The running time of the exact algorithm increases with lower frequency thresholds τ, and is inversely proportional to the number of labels. E.g., even though MiCo is far larger than Phy-Cit, the exact algorithm takes less time at the same frequency threshold. This difference is due to the fact that MiCo has 29 distinct labels, and thus, the patterns have lower frequencies than those in Phy-Cit (6 labels). MaNIACS allows us to control the complexity of the mining process while achieving good results. Fig. 4 (right) shows that with a sample size to 9k, the running time of MaNIACS on, e.g., Patents is two orders of magnitude faster, with a maximum error in the range $[0.004, 0.007]$.

6 CONCLUSIONS

We presented MaNIACS, a sampling-based algorithm that outputs high-quality approximations of the collection of frequent subgraph patterns in a large graph according to the MNI frequency. To compute the quality of the approximation, MaNIACS relies on the empirical VC-dimension, a concept from statistical learning theory that ties the maximum frequency estimation error to *sample-dependent* properties. We showed how to compute an upper bound on the

eVC-dimension and how to use the resulting bound on the estimation error to prune the pattern search space to avoid expensive-but-worthless computations. The results of our experimental evaluation showed that MaNIACS achieves high-precision results, up to two orders of magnitude faster than an exact algorithm.

ACKNOWLEDGMENTS

We thank Cigdem Aslay and Muhammad Anis Uddin Nasir for their help in the preliminary phase of this work. Part of this work is supported by the National Science Foundation grant 2006765. The authors acknowledge the support from Intesa Sanpaolo Innovation Center. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

REFERENCES

[1] E. Abdelhamid, I. Abdelaziz, P. Kalnis, Z. Khayyat, and F. Jamour. 2016. Scalemine: Scalable Parallel Frequent Subgraph Mining in a Single Large Graph. In *SC*.
[2] I. Alobaidi, J. Leopold, and A. Allami. 2019. The Use of Frequent Subgraph Mining to Develop a Recommender System for Playing Real-Time Strategy Games. In *ICDM*. 146–160.
[3] Çigdem Aslay, Muhammad Anis Uddin Nasir, Gianmarco De Francisci Morales, and Aristides Gionis. 2018. Mining Frequent Patterns in Evolving Graphs. In *CIKM*. 923–932.
[4] S.K. Bera and C. Seshadhri. 2020. How to Count Triangles, without Seeing the Whole Graph. In *KDD*. 306–316.
[5] V. Bhatia and R. Rani. 2018. Ap-FSM: A parallel algorithm for approximate frequent subgraph mining using Pregel. *Exp. Sys. Appl.* 106 (2018), 217–232.
[6] M.A. Bhuiyan, M. Rahman, and M. Al Hasan. 2012. Guise: Uniform sampling of graphlets for large graph analysis. In *ICDM*. 91–100.
[7] M. Bressan, F. Chierichetti, R. Kumar, S. Leucci, and A. Panconesi. 2017. Counting Graphlets: Space vs Time. In *WSDM*. 557–566.
[8] M. Bressan, F. Chierichetti, Ravi Kumar, Stefano Leucci, and Alessandro Panconesi. 2018. Motif Counting Beyond Five Nodes. *TKDD* 12, 4 (2018).
[9] M. Bressan, S. Leucci, and A. Panconesi. 2019. Motivo: Fast Motif Counting via Succinct Color Coding and Adaptive Sampling. *PVLDB* 12, 11 (2019), 1651–1663.
[10] B. Bringmann and S. Nijssen. 2008. What is frequent in a single graph?. In *PAKDD*. 858–863.
[11] M.H. Chehreghani, T. Abdessalem, A. Bifet, and M. Bouzbila. 2020. Sampling informative patterns from large single networks. *FGCS* 106 (2020), 653–658.
[12] X. Chen, J .and Qian. 2020. DwarvesGraph: A High-Performance Graph Mining System with Pattern Decomposition. arXiv:2008.09682 [cs.DC]
[13] X. Cheng, C. Dale, and J. Liu. 2008. Statistics and social network of YouTube videos. In *IWQoS*. 229–238.
[14] F. Chierichetti, A. Dasgupta, R. Kumar, S. Lattanzi, and T. Sarlós. 2016. On sampling nodes in a network. In *WWW*. 471–481.
[15] F. Chierichetti and S. Haddadan. 2018. On the Complexity of Sampling Vertices Uniformly from a Graph. In *ICALP*.
[16] G. Das. 2009. Sampling Methods in Approximate Query Answering Systems. In *Encyclopedia of Data Warehousing and Mining*. 1702–1707.
[17] M. Deshpande, M. Kuramochi, N. Wale, and G. Karypis. 2005. Frequent substructure-based approaches for classifying chemical compounds. *TKDE* 17, 8 (2005), 1036–1050.
[18] M. Elseidy, E. Abdelhamid, S. Skiadopoulos, and P. Kalnis. 2014. Grami: Frequent subgraph and pattern mining in a single large graph. *PVLDB* 7, 7 (2014), 517–528.
[19] W. Fan, X. Wang, Y. Wu, and J. Xu. 2015. Association Rules with Graph Patterns. *PVLDB* 8, 12 (2015), 1502–1513.
[20] M. Fiedler and C. Borgelt. 2007. Subgraph support in a single large graph. In *ICDMW*. 399–404.
[21] S. Ghazizadeh and S.S. Chawathe. 2002. SEuS: Structure extraction using summaries. In *DS*. 71–85.
[22] V. Guralnik and G. Karypis. 2001. A scalable algorithm for clustering sequential data. In *ICDM*. 179–186.
[23] G. Han and H. Sethu. 2016. Waddling random walk: Fast and accurate sampling of motif statistics in large graphs. In *ICDM*. 181–190.
[24] T.A.D. Henderson. 2017. *Frequent Subgraph Analysis and its Software Engineering Applications.* Ph.D. Dissertation. Case Western Reserve University.
[25] A.P. Iyer, Z. Liu, X. Jin, S. Venkataraman, V. Braverman, and I. Stoica. 2018. ASAP: Fast, Approximate Graph Pattern Mining at Scale. In *OSDI*. 745–761.
[26] K. Jamshidi, R. Mahadasa, and K. Vora. 2020. Peregrine: A Pattern-Aware Graph Mining System. In *EuroSys*.
[27] T. Junttila and P. Kaski. 2007. Engineering an efficient canonical labeling tool for large and sparse graphs. In *ALENEX*. 135–149.
[28] M. Kuramochi and G. Karypis. 2004. Grew-a scalable frequent subgraph discovery algorithm. In *ICDM*.
[29] M. Kuramochi and G. Karypis. 2005. Finding frequent patterns in a large sparse graph. *DMKD* 11, 3 (2005), 243–271.
[30] J. Leskovec, J. Kleinberg, and C. Faloutsos. 2005. Graphs over time: densification laws, shrinking diameters and possible explanations. In *KDD*. 177–187.
[31] Y. Li, P.M. Long, and A. Srinivasan. 2001. Improved Bounds on the Sample Complexity of Learning. *J. Comput. System Sci.* 62, 3 (2001), 516–527.
[32] M. Löffler and J.M. Phillips. 2009. Shape Fitting on Point Sets with Probability Distributions. In *ESA*. 313–324.
[33] I. Melckenbeeck, P. Audenaert, T. Van Parys, Y. Van De Peer, D. Colle, and M. Pickavet. 2019. Optimising orbit counting of arbitrary order by equation selection. *BMC bioinformatics* 20, 1 (2019), 1–13.
[34] J. Meng, N. Pitaksirianan, and Y. Tu. 2019. Generalizing Design of Support Measures for Counting Frequent Patterns in Graphs. In *BigData*. 533–542.
[35] J. Meng, N. Pitaksirianan, and Y.-C. Tu. 2020. Counting frequent patterns in large labeled graphs: a hypergraph-based approach. *DMKD* (2020), 1–42.
[36] A. Mrzic, P. Meysman, W. Bittremieux, P. Moris, B. Cule, B. Goethals, and K. Laukens. 2018. Grasping frequent subgraph mining for bioinformatics applications. *BioData Mining* 11, 20 (2018).
[37] M.A.U. Nasir, Ç. Aslay, G. De Francisci Morales, and M. Riondato. 2021. TipTap: Approximate Mining of Frequent k-Subgraph Patterns in Evolving Graphs. *TKDD* (2021).
[38] K. Paramonov, D. Shemetov, and J. Sharpnack. 2019. Estimating Graphlet Statistics via Lifting. In *KDD*. 587–595.
[39] N. Pashanasangi and C. Seshadhri. 2020. Efficiently Counting Vertex Orbits of All 5-Vertex Subgraphs, by EVOKE. In *WSDM*. 447–455.
[40] L. Pellegrina, C. Cousins, F. Vandin, and M. Riondato. 2020. MCRapper: Monte-Carlo Rademacher Averages for Poset Families and Approximate Pattern Mining. In *KDD*. 2165–2174.
[41] A. Pinar, C. Seshadhri, and V. Vishal. 2017. ESCAPE: Efficiently Counting All 5-Vertex Subgraphs. In *WWW*. 1431–1440.
[42] N. Pržulj, D.G. Corneil, and I. Jurisica. 2004. Modeling interactome: scale-free or geometric? *Bioinformatics* 20, 18 (2004), 3508–3515.
[43] S. Purohit, S. Choudhury, and L. B. Holder. 2017. Application-specific graph sampling for frequent subgraph mining and community detection. In *Big Data*.
[44] P. Ribeiro, P. Paredes, M.E.P. Silva, D. Aparicio, and F. Silva. 2019. A Survey on Subgraph Counting: Concepts, Algorithms and Applications to Network Motifs and Graphlets. arXiv:1910.13011 [cs.DS]
[45] P. Ribeiro and F. Silva. 2014. Discovering colored network motifs. In *Complex Networks V*. Springer, 107–118.
[46] M. Riondato, J.A. DeBrabant, R. Fonseca, and E. Upfal. 2012. PARMA: A Parallel Randomized Algorithm for Association Rules Mining in MapReduce. In *CIKM*.
[47] M. Riondato and E. Upfal. 2014. Efficient Discovery of Association Rules and Frequent Itemsets through Sampling with Tight Performance Guarantees. *TKDD* 8, 4 (2014), 20.
[48] M. Riondato and E. Upfal. 2018. ABRA: Approximating Betweenness Centrality in Static and Dynamic Graphs with Rademacher Averages. *TKDD* 12, 5 (2018).
[49] M. Riondato and F. Vandin. 2014. Finding the True Frequent Itemsets. In *SDM*.
[50] R. A. Rossi, N. K. Ahmed, A. Carranza, D. Arbour, A. Rao, S. Kim, and E. Koh. 2020. Heterogeneous Graphlets. *TKDD* 15, 9 (2020).
[51] T. K. Saha, A. Katebi, W. Dhifli, and M. Al Hasan. 2019. Discovery of Functional Motifs from the Interface Region of Oligomeric Proteins Using Frequent Subgraph Mining. *TCBB* 16, 5 (2019), 1537–1549.
[52] C. Seshadhri and S. Tirthapura. 2019. Scalable Subgraph Counting: The Methods Behind The Madness. In *WWW*.
[53] S. Shalev-Shwartz and S. Ben-David. 2014. *Understanding Machine Learning: From Theory to Algorithms.* Cambridge University Press.
[54] N. Talukder and M.J. Zaki. 2016. A distributed approach for graph mining in massive networks. *DMKD* 30, 5 (2016), 1024–1052.
[55] C.H.C. Teixeira, A.J. Fonseca, M. Serafini, G. Siganos, M.J. Zaki, and A. Aboulnaga. 2015. Arabesque: A System for Distributed Graph Mining. In *SOSP*. 425–440.
[56] N. Vanetik, E. Gudes, and S. E. Shimony. 2002. Computing frequent graph patterns from semistructured data. In *ICDM*. 458–465.
[57] N. Vanetik, S.E. Shimony, and E. Gudes. 2006. Support measures for graph data. *DMKD* 13, 2 (2006), 243–260.
[58] Vladimir N. Vapnik. 1998. *Statistical learning theory.* Wiley.
[59] J. Wang, Y. Wang, W. Jiang, Y. Li, and K. Tan. 2020. Efficient Sampling Algorithms for Approximate Temporal Motif Counting. In *CIKM*. 1505–1514.
[60] P. Wang, J. Lui, B. Ribeiro, D. Towsley, J. Zhao, and X. Guan. 2014. Efficiently estimating motif statistics of large networks. *TKDD* 9, 2 (2014), 8.
[61] P. Wang, J. Lui, D. Towsley, and J. Zhao. 2016. Minfer: A method of inferring motif statistics from sampled edges. In *ICDE*. 1050–1061.
[62] X. Zhao, Y. Chen, C. Xiao, Y. Ishikawa, and J. Tang. 2016. Frequent subgraph mining based on Pregel. *Comput. J.* 59, 8 (2016), 1113–1128.

A SUPPLEMENTARY MATERIAL

A.1 Reproducibility

The implementation of our algorithm is publicly available at https://github.com/lady-bluecopper/MaNIACS, complete with instructions on how to run it, and a Jupyter Notebook *results.ipynb* with the full results of our experimental evaluation. This notebook contains all the plots omitted here due to space limitations.

Datasets description.

Table 2: Characteristics of the datasets.

	Citeseer	Phy-Cit	MiCo	Patents	YouTube
Vertices	3.3K	30K	100K	2.7M	4.5M
Edges	4.5K	347K	1M	13M	43M
Labels	6	6	29	4	12
Density	$8.37\ 10^{-4}$	$7.46\ 10^{-4}$	$2.16\ 10^{-4}$	$3.67\ 10^{-6}$	$4.17\ 10^{-6}$
Avg Label Freq	552	5K	3.4K	689K	382K
Med Label Freq	593	5.9K	2.1K	672K	472K
Avg Edge Freq	218.6	16K	2.4K	1.5M	563K
Med Edge Freq	86	12K	1.0K	1.2M	535K

Pseudocode for the computation of the image sets.

Algorithm 3: GETIMAGESETS

Input: Set of patterns $\mathcal{H}_i$, sample S, frequency threshold τ
Output: The image sets $\mathcal{Z}_i$ of the patterns in $\mathcal{H}_i$

```
1  Z_i ← ∅
2  foreach P ∈ H_i do
3      foreach orbit A of P do
4          n ← a vertex of P in A
5          S_A ← vertices in S with the label of A
6          Z_S(A) ← ∅, remain ← |S_A|
7          foreach v ∈ S_A do
8              M ← ∅; M[n] ← v
9              if existsIsomorphism(P, M) then
10                 Z_S(A) ← Z_S(A) ∪ {v}
11             remain ← remain − 1
12             if (remain + |Z_S(A)|)/|S| < τ − ε_i then
13                 prune P and go to next pattern
14         Z_i ← Z_i ∪ {Z_S(A)}
15 return Z_i
```

Algorithm 4: EXISTSISOMORPHISM

Input: Pattern P, Partial match M
Output: **true** iff M contains a match for each vertex of P

```
1  if |M| = |V_P| then return true
2  u ← vertex of P not already in M
3  cands ← ⋂_{w∈u.Γ∩M} M[w].Γ
4  foreach z ∈ cands with the same label as u do
5      isMatch ← true
6      foreach w ∈ M do
7          if w ∉ u.Γ and M[w] ∈ z.Γ then
8              isMatch ← false; break
9      if isMatch then
10         M[u] ← z
11         if existsIsomorphism(P, M) then return true
12 return false
```

A.2 Missing proofs

We now give the proofs for all our theoretical results, restated here for convenience.

LEMMA 4.2. *Let $(\mathcal{D}, \mathcal{R})$ be a range space, and let $\mathcal{T} \subseteq \mathcal{D}$. Consider the set $\mathcal{R}_\mathcal{T} \doteq \{\mathcal{T} \cap R : R \in \mathcal{R}\}$. Let g^* be the maximum g such that $\mathcal{T}$ contains at least g points each appearing in at least 2^{g-1} sets from $\mathcal{R}_\mathcal{T}$. If, for at least one set B of such g^* points, there exist a set $Z_B \in \mathcal{R}_\mathcal{T}$ such that $B \subseteq Z_B$, then $\mathsf{E}_\mathcal{T}(\mathcal{R})$ is at most g^*, otherwise it is at most $g^* - 1$.*

PROOF OF LEMMA 4.2. Each point a in a shattered set A of size $|A| = z$ must belong to at least 2^{z-1} distinct sets in $\mathcal{R}_\mathcal{T}$, as it belongs to this number of non-empty subsets of A. Additionally, there must be a set in $\mathcal{R}_\mathcal{T}$ that contains the whole A. □

LEMMA 4.3. *Let $(\mathcal{D}, \mathcal{R})$ and $\mathcal{T}$ as in Lemma 4.2. Let $R_1, \ldots, R_{|\mathcal{R}|}$ be a labeling of the ranges in $\mathcal{R}$ such that $|R_w \cap \mathcal{T}| \geq |R_u \cap \mathcal{T}|$ for $1 \leq w < u \leq |\mathcal{R}|$. Let $(a_j)_{j=1}^{\ell}$ be a non-increasing sequence of $\ell \geq |\mathcal{R}|$ naturals such that $a_j \geq \left|R_j \cap \mathcal{T}\right|$ for $1 \leq j < |\mathcal{R}|$ and $a_j \geq \left|R_{|\mathcal{R}|} \cap \mathcal{T}\right|$ for $|\mathcal{R}| \leq j \leq \ell$.*

Let h^* be the maximum natural h such that, for every $0 \leq j < h$, if we let $c_j = \sum_{z=0}^{j} \binom{h}{z}$,[8] it holds $a_{c_j} \geq h - j$. Then, $\mathsf{E}_\mathcal{T}(\mathcal{R}) \leq h^*$.

PROOF OF LEMMA 4.3. Let $z = \mathsf{E}_\mathcal{T}(\mathcal{R})$. Then there is a set $A \subseteq \mathcal{T}$ with $|A| = z$ that is shattered by $\mathcal{R}$. For a set A with $|A| = z$ to be shattered by $\mathcal{R}$, there must be, for every $0 \leq i < z$, $\binom{z}{i}$ distinct ranges $H_{i,1}, \ldots, H_{i,\binom{z}{i}} \in \mathcal{R}$ such that $\left|H_{i,j} \cap A\right| = z - i$, as A has $\binom{z}{i}$ subsets of size $z - i$. It must then also hold that $\left|H_{i,j} \cap \mathcal{T}\right| \geq z - i$.

If $\ell = |\mathcal{R}|$ and $a_j = \left|R_j \cap \mathcal{T}\right|$ for every $1 \leq j \leq |\mathcal{R}|$, it follows from the definition of h^* that it must be $z \leq h^*$. For a generic sequence $(a_j)_{j=1}^{\ell}$, the thesis follows from the fact that the value h^* computed on this generic sequence cannot be smaller than the value h^* computed on the specific sequence for which $\ell = |\mathcal{R}|$ and $a_j = \left|R_j \cap \mathcal{T}\right|$ for every $1 \leq j \leq |\mathcal{R}|$. □

LEMMA 4.4. *For any $S \subseteq V$, it holds $\mathsf{E}_S(\mathcal{R}_i) = \max_{\lambda \in L} \mathsf{E}_S(\mathcal{R}_{i,\lambda})$.*

Lemma 4.4 is an immediate corollary of the following result.

LEMMA A.1. *No $S \subseteq V$ containing vertices with different labels can be shattered by $\mathcal{R}_i$.*

PROOF. The statement is immediate from the definition of image set (see (1)). For any orbit A, its image set $\mathsf{Z}_V(A)$ on V only contains vertices with the same label, thus there would be no range in $\mathcal{R}_i$ that would contain, for example, the whole S, thus $S \notin \mathsf{P}_{\mathcal{R}_i}(S)$, which implies that S cannot be shattered by $\mathcal{R}_i$. □

THEOREM 4.5. *With probability at least $1 - \delta$ over the choice of S, the output $\mathcal{Q}$ of MANIACS contains a triplet $(P, \mathsf{f}_S(P), \varepsilon_P)$ for every $P \in \mathsf{FP}_V(\tau)$ such that $|\mathsf{f}_S(P) - \mathsf{f}_V(P)| \leq \varepsilon_P$.*

PROOF OF THM. 4.5. For $1 \leq i \leq k$, let η_i be the value η computed as in Thm. 3.1 for $\phi = \delta/k$, $(\mathcal{D}, \mathcal{R}) = (V, \mathcal{R}_i)$, $\mathcal{T}$ chosen as S on line 1, and d being the eVC-dimension of $(V, \mathcal{R}_i)$ on S. It follows from Thm. 3.1 and an application of the union bound over the k sets (hence the use of δ/k), that, with probability at least $1 - \delta$, it

[8] We define $\binom{q}{0} = 1$ for any q.

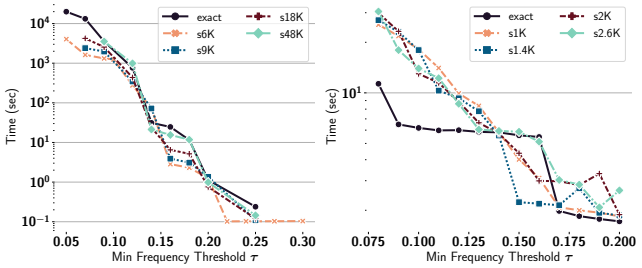

Figure 6: Running time of MaNIACS and the exact algorithm, varying sample size and min frequency threshold τ, on Mico (left) and Citeseer (right).

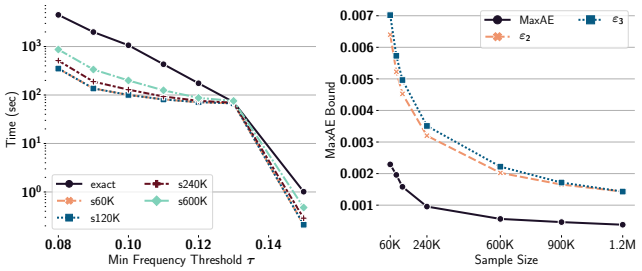

Figure 7: YouTube: running time of MaNIACS and the exact algorithm, varying sample size, and min frequency threshold τ (left); and Max Absolute Error (MaxAE), ε_2, ε_3, ε_4, and ε_5, for various sample size and min frequency threshold $\tau = 0.16$ (right).

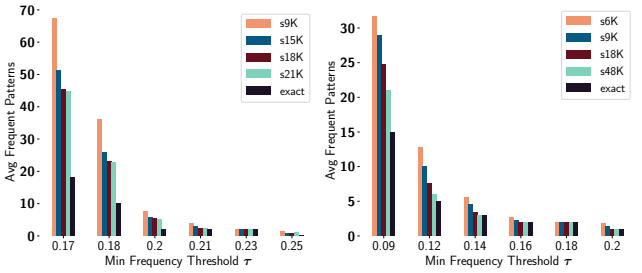

Figure 8: Average number of patterns found by MaNIACS, together with the exact number of frequent patterns, varying minimum frequency threshold τ, in Phy-Cit (left) and Mico (right).

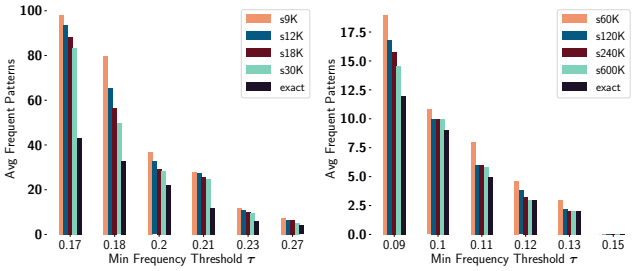

Figure 9: Average number of patterns found by MaNIACS, together with the exact number of frequent patterns, varying minimum frequency threshold τ, in Patents (left) and YouTube (right).

holds that S is, simultaneously, an η_i-sample for $(V, \mathcal{R}_i)$ for every $1 \le i \le k$. Assume for the rest of the proof that that is the case.

We show inductively that, at the end of every iteration of the "main" loop of MaNIACS (lines 4–14), it holds that

(1) Q contains a triplet $(P, \mathrm{f}_S(P), \varepsilon_i)$ for each $P \in \mathcal{F}_i$, and the triplet is such that

$$|\mathrm{f}_V(P) - \mathrm{f}_S(P)| \le \varepsilon_i \ ;$$

(2) $\mathcal{F}_i \subseteq \mathcal{H}_i$, for $i \le k$.

At the beginning of the first iteration, i.e., for $i = 1$, it obviously holds $\mathcal{F}_1 \subseteq \mathcal{H}_1$ from the definition of $\mathcal{H}_1$ (line 3). Thus, at the first iteration of the do-while loop on lines 6–11, the value ε_1 computed on line 8 using Thm. 3.1 is not smaller than η_1, because b_1^* is an upper bound to the eVC-dimension of $(V, \mathcal{R}_1)$ on S, thanks to Lemmas 4.2 to 4.4, and the value η on the l.h.s. of (6) is monotonically increasing with the value d used on the r.h.s. of the same equation. It then follows, from this fact and from Corol. 4.1, that no pattern $P \in \mathrm{FP}_\tau(V)$ may have $\mathrm{f}_S(P) < \tau - \varepsilon_1$, therefore the refinement of $\mathcal{H}_1$ on line 10 is such that it still holds $\mathcal{F}_1 \subseteq \mathcal{H}_1$ at the end of the first iteration of the do-while loop. Following the same reasoning one can show that this condition and the fact that $\varepsilon_1 \ge \eta_1$ throughout every iteration of the do-while loop.

The set Q, updated on line 12, therefore contains, among others, a triplet for every pattern $P \in \mathrm{FP}_\tau(V)$, and the properties from the thesis hold because of this fact and the fact that $\varepsilon_1 \ge \eta_1$, thus completing the base case for point (1) in the list above. Point (2), i.e., that $\mathcal{F}_2 \subseteq \mathcal{H}_2$, then follows from the anti-monotone property of the MNI-frequency (Fact 2).

Assume now that points (1) and (2) hold at every iteration of the while loop from $i = 1, \ldots, i^* < k$. The proof that they hold at the end of iteration $i^* + 1$ follows the same reasoning as above. □

A.3 Additional experiments

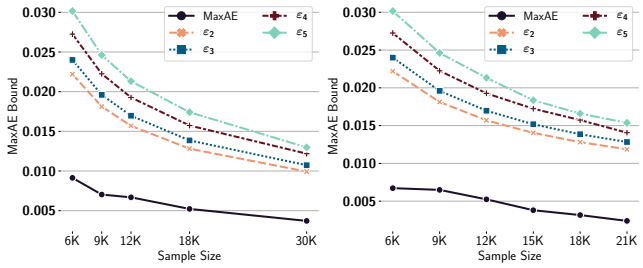

Figure 5: Max Absolute Error (MaxAE), ε_2, ε_3, ε_4, and ε_5, for various sample size, min frequency threshold $\tau = 0.16$, in Patents (left) and in Phy-Cit (right).