Efficient Mining of Subsample-Stable Graph Patterns

Aleksey Buzmakov, Sergei O. Kuznetsov, Amedeo Napoli

Efficient Mining of Subsample-Stable Graph Patterns

Aleksey Buzmakov
National Research University
Higher School of Economics
Gagarina bulvar 37a,
Perm, Russia
avbuzmakov@hse.ru

Sergei O. Kuznetsov
National Research University
Higher School of Economics
3 Kochnovsky Proezd,
Moscow, Russia
skuznetsov@hse.ru

Amedeo Napoli
LORIA
(CNRS – Inria – U. of Lorraine),
BP 239,
Vandœuvre-lès-Nancy, France
amedeo.napoli@loria.fr

***Abstract*—A scalable method for mining graph patterns stable under subsampling is proposed. The existing subsample stability and robustness measures are not antimonotonic according to definitions known so far. We study a broader notion of anti-monotonicity for graph patterns, so that measures of subsample stability become antimonotonic. Then we propose gSOFIA for mining the most subsample-stable graph patterns. The experiments on numerous graph datasets show that gSOFIA is very efficient for discovering subsample-stable graph patterns.**

I. Introduction

The idea of mining interesting patterns is at the heart of knowledge discovery. Graph mining faces more serious resource challenges than itemset mining, so interestingness measures allowing for efficient computation of most interesting graph patterns are of big value. In this paper we concentrate on mining interesting closed graph patterns.

As in frequent itemset mining, a number of special pattern types are useful for graph mining [1]. Provably efficient (polynomial-delay) mining of most interesting patterns was shown to be possible for antimonotonic measures, such as support and predicate "to be a minimal generator" [2], as well as for local antimonotonic measures, where for any pattern p at least one immediate subpattern should have higher measure than p, as in accessible systems [3].

However, many important interestingness measures are not antimonotonic in any of these senses, among them stability [4] and robustness [5], which express invariance of a pattern w.r.t. subsampling. As we show below, these measures are closely related to resampling strategies such as bootstrapping. Moreover, these measures are important for justifying mining heuristics based on sampling. In the case of non-(anti)monotonic measures interesting patterns can be selected by postpruning [5], [6], [7]. More elaborated approaches allow constructing a measure from (anti-)monotonic primitives [8] by finding a good anti-monotonic relaxation of the measure for the dataset in hand. Non-(anti)monotonic constraints can also be efficiently processed if one finds a good closure operator on the set of patterns [9], however this is usually a nontrivial task.

A broader class of antimonotonic constraints can be based on projection-antimonotonic measures [10]. This type of antimonotonicity is defined in terms of projections in pattern structures [11], which give a convenient formalism for treating closed descriptions of various types [12]. In this paper we propose algorithm gSOFIA inspired by $\Sigma o \varphi \iota \alpha$ [10]. gSOFIA is able to efficiently mine subsample-stable graph patterns based on projection-antimonotonicity. This approach rely on existing state-of-the-art graph mining canonical orders. Our algorithm determines which graphs should be expanded in order to produce new graph patterns with the subsample stability higher than a given threshold. Moreover, gSOFIA can be even faster than producing just frequent graphs w.r.t. the canonical order.

II. Preliminaries

In graph mining a dataset is considered to be a set of graphs $\mathbb{G}$ [1]. A unique name, also called *transaction identifier* (tid), can be attached to any graph form $\mathbb{G}$. We denote the set of tids by $\mathcal{T}$ and the set of all possible graphs with labeled vertices and edges by $\mathcal{G}$, i.e. $\mathbb{G} \subset \mathcal{G}$. $\mathcal{G}$ is ordered w.r.t. subgraph isomorphism $\leq$. Let $\delta : \mathcal{T} \to \mathcal{G}$ map graph names to graphs. A *graph dataset* can be denoted by a pair $(\mathcal{T}, \delta)$ with $\mathbb{G} = \delta(\mathcal{T})$. A *tidset* of g: $\mathbb{t}(g) = \{t \in \mathcal{T} \mid g \leq \delta(t)\}$ is the set of all ids of graphs that are supergraphs of g. The cardinality of $\mathbb{t}(g)$ is called *support* of g. If the support of a graph g is strictly larger than support of any supergraph of g, then g is called a *closed graph* [13]. Given a tidset $t \subseteq \mathcal{T}$ a maximal common subgraph of $\delta(t)$ is not always unique.

However, it is always possible to define the set of maximal common subgraphs for any $t \subseteq \mathcal{T}$. By *pattern space* $\mathcal{D}$ we denote the set of all possible sets of graphs from $\mathcal{G}$ incomparable w.r.t. $\leq$, elements of $\mathcal{D}$ are called *graph patterns*. Given two graph patterns $D_1, D_2 \in \mathcal{D}$ one has $D_1 \sqsubseteq D_2$ (D_2 *subsumes* D_1) if for every graph $g_1 \in D_1$ there is a graph $g_2 \in D_2$ such that $g_1 \leq g_2$. Subsumption $\sqsubseteq$ defines a partial order on $\mathcal{D}$ and the ordered set $(D, \sqsubseteq)$ is a semilattice [14], [15], i.e., any two graph patterns $D_1, D_2 \in \mathcal{D}$ have an infimum in $\mathcal{D}$ w.r.t. $\sqsubseteq$. For singletons $\{g_1\}$ and $\{g_2\}$, $g_1, g_2 \in \mathcal{G}$, their infimum, denoted by $\{g_1\} \sqcap \{g_2\}$, is the graph pattern consisting of all maximal common subgraphs of g_1 and g_2. In this case $\delta(t)$ maps a graph name to the corresponding singleton.

For a graph pattern the tidset is defined similarly to the tidset of a graph: $\mathbb{t}(d \in \mathcal{D}) = \{t \in \mathcal{T} \mid d \sqsubseteq \delta(t)\}$. Moreover,

	i_1	i_2	i_3	i_4	i_5	i_6
t_1	x		x			
t_2		x	x			
t_3			x	x		
t_4			x		x	
t_5						x

Fig. 1. A toy binary dataset.

for tidset $T \subseteq \mathcal{T}$ there is always a unique graph pattern corresponding to it:

$$\mathbb{d}(T) := \max_{p \in \mathcal{D}, (\forall x \in t) p \sqsubseteq \delta(x)} p.$$

The infimum $D_1 \sqcap D_2$ of two graph patterns is defined as

$$D_1 \sqcap D_2 := \max_{\sqsubseteq} \{\{g_1\} \sqcap \{g_2\} \mid g_1 \in D_1, g_2 \in D_2\}.$$

Now a *graph pattern structure* is a triple $(\mathcal{T}, (\mathcal{D}, \sqcap), \delta)$. A *closed graph pattern* (or *graph intent* [14]) is the largest (w.r.t. $\sqsubseteq$) graph pattern among the graph patterns with the same tidset. As shown in [15] a closed graph pattern (graph intent) consists of closed graphs.

III. Subsample-Stable Graph Patterns

It is desirable to exclude spurious patterns. Stability [4] and robustness [5] measure the degree of spuriousness as stability of a pattern closedness w.r.t. dataset subsampling.

Given $\mathbb{D} = (\mathcal{T}, (\mathcal{D}, \sqcap), \delta)$, a triple $(S, (\mathcal{D}, \sqcap), \delta)$ where $S \subseteq \mathcal{T}$ is called a *subdataset* of $\mathbb{D}$. Giving a weight to every subdataset of $\mathbb{D}$, we find the sum of weights of all subdatasets of $\mathbb{D}$ where a pattern $p \in \mathcal{D}$ is closed. Depending on weights, this sum gives us stability or robustness of a closed pattern p. In the case of *stability* the weights w of all subdatasets $\mathbb{D}_s$ of $\mathbb{D}$ are equal, i.e.,

$$Stab(d) = \frac{|\{S \subseteq \mathbb{t}(d) \mid \mathbb{d}(S) = d\}|}{2^{|\mathbb{t}(d)|}}.$$

Example 1. *Let us consider the dataset in Figure 1 with 5 transactions and 6 items. In this case the dataset is* $\mathbb{D} = (\mathcal{T}, (2^I, \cap), \delta)$*, where* I *is the set of items and* δ *takes every transaction to its itemset. The patterns in this case are itemsets ordered by set inclusion. Consider itemset* $\{i_3\}$*, which is closed. There are* $2^{|\mathcal{T}|} = 32$ *subdatasets. Itemset* $\{i_3\}$ *is closed in 22 subdatasets, thus,* $\texttt{Stab}(\{i_3\}) = 22 \cdot \frac{1}{32} = 0.69$.

For *robustness* the weights w depend on a parameter $0 \leq \alpha \leq 1$. Then, the weight of a subdataset $\mathbb{D}_s = (S, \mathcal{D}, \delta)$ is the probability of obtaining $\mathbb{D}_s$ by removing transactions with probability $1 - \alpha$: $w(\mathbb{D}_s) = \alpha^{|S|} \cdot (1 - \alpha)^{|\mathcal{T}| - |S|}$.

A similar way of defining closed pattern independence of a particular dataset is to measure *closedness by bootstrapping*. Bootstrapping [16] refers to computing the average value of a function on random samples of a given dataset with replacement (so a sample is likely to contain duplicates of transactions). If we remove all duplicate transactions from a dataset a closed pattern remains closed. Accordingly, in any bootstrap sample we can remove all duplicate transactions and obtain a subdataset. Thus, we should assign a weight to this dataset equal to the probability of obtaining subsamples that are mapped to this subdataset. Computing any of these three measures is #P-complete [4], so to avoid intractability we rely on tractable bounds for all measures.

A. Tractable Bounds for Subsample Stability

Let us consider closed patterns $p, q \in \mathcal{D}$ such that $p \sqsubset q$. Let $\Delta(p, q) = |\mathbb{t}(p) \setminus \mathbb{t}(q)|$. This set is not empty since $p \neq q$ and p and q are closed.

Example 2. *Let* $p = \{i_3\}$ *and* $q = \{i_1, i_3\}$*, then* $\Delta(p, q) = |\mathbb{t}(p) \setminus \mathbb{t}(q)| = |\{t_1, t_2, t_3, t_4\} \setminus \{t_1\}| = 3$.

Proposition 1. *Stability, robustness, and bootstrap closedness are bounded as follows, :*

$$1 - \sum_{q \succ p} 2^{-\Delta(p,q)} \leq \texttt{Stab}(p) \leq 1 - 2^{-\Delta(p,q)} \quad (1)$$

$$1 - \sum_{q \succ p} (1 - \alpha)^{\Delta(p,q)} \leq \texttt{Rbst}(q) \leq 1 - (1 - \alpha)^{\Delta(p,q)} \quad (2)$$

$$1 - \sum_{q \succ p} \left(1 - \frac{\Delta(p,q)}{|\mathcal{T}|}\right)^{|\mathcal{T}|} \leq \texttt{bootstrapping}(p) \leq$$

$$\leq 1 - \left(1 - \frac{\Delta(p,q)}{|\mathcal{T}|}\right)^{|\mathcal{T}|} \quad (3)$$

The proof is similar to the one for stability bounds [17]. These bounds can be computed in polynomial time if the immediate superpatterns are known, so we can use the bounds as proxies to the original measures. The upper bound of stability, robustness, and bootstrapping rank the patterns in the same way as $\Delta(p) = \min_{q \succ p} \Delta(p, q)$, called Δ-measure.

Example 3. *Let us compute* $\Delta(\{i_3\})$*. The nearest superitemsets are* $\{i_1, i_3\}$*,* $\{i_2, i_3\}$*,* $\{i_3, i_4\}$*, and* $\{i_3, i_5\}$*, all having support 1. Thus,* $\Delta(\{i_3\}) = 4 - 1 = 3$.

IV. Description of Algorithm gSOFIA

A. The main idea of algorithm Σοφια

We propose an efficient way for computing Δ-stable graph patterns w.r.t. a threshold θ. It is inspired by abstract algorithm Σοφια [10]. The idea of Σοφια is presented here. Later we discuss its instantiation for graphs.

Most of the pattern mining approaches start from small patterns, e.g., graphs with one vertex, etc. Then they iteratively expand smaller patterns to larger ones trying to avoid duplicate generation by using a canonical order.

Let $\mathcal{D}$ be a pattern space. At one iteration a pattern $d_0 \in \mathcal{D}$ is expanded to patterns $d_1, ..., d_k \in \mathcal{D}$. *Ancestor function* $\psi : \mathcal{D} \to \mathcal{D}$ returns, for a given pattern d, the pattern it was generated from, e.g., $\psi(d_i) = d_0$ for all $1 \leq i \leq k$. One function ψ can encode a large class of expansions, e.g. expansion from graphs of size $\leq k$ to graphs of size $\leq k + 1$.

Since pattern mining is an iterative procedure, we have a chain of ancestor functions $\psi_1, \psi_2, ...$ Given a pattern d and function ψ, the result of expansion of d is given by the set of preimages of d for ψ, i.e., patterns $\tilde{d}$ such that $\psi(\tilde{d}) = d$. The set of preimages of all patterns for the first ancestor function is the image for the second ancestor function and so on. For simplicity we agree that every ancestor function is idempotent,

i.e., the set of preimages of patterns from the domain is a subset of the domain. Thus, functions ψ_i are idempotent and contractive ($\psi_i(x) \sqsubseteq x$).

Example 4. *In itemset data, an ancestor function ψ can correspond to the removal of an item a from an itemset. However, during the pattern expansion, ψ "adds" the attribute a to patterns (the preimages are found). More precisely, if ψ corresponds to the removal of items Y, the preimages of an itemset X (we assume $X \cap Y = \emptyset$) for ψ form the set of itemsets $\{Z \mid \psi(Z) = X\}$. It can be seen that the set of preimages is given by* $\texttt{Preimages}_\psi(X) = \{Z \subseteq \mathcal{I} \mid X \subseteq Z \subseteq X \cup Y\}$.

If $(\forall d \in D)\Delta(d) \leq \Delta(\psi(d))$, then any non Δ-stable pattern d can be ignored, since no Δ-stable patterns can be generated from d by ψ. This property is called antimonotonicity w.r.t. the ancestor function ψ. So, the idea of $\Sigma o\varphi\iota\alpha$ is to start from the smallest pattern, e.g., zero-sized graph. This pattern is considered to be Δ-stable. Then, it passes through the chain of ancestor functions. Given a set of found Δ-stable patterns and an ancestor function ψ, it computes the set of preimages for every Δ-stable pattern from the set w.r.t. ψ. Then, it computes Δ-measure for all preimages and if Δ-measure is less than the threshold θ, $\Sigma o\varphi\iota\alpha$ removes this pattern. Otherwise, the preimage is included in the current set of Δ-stable patterns. The formal description of $\Sigma o\varphi\iota\alpha$ can be found in [10]. It was also shown there that Δ-measure is antimonotonic w.r.t. any projection, a special case of ancestor function that is monotone, i.e., $(\forall x, y \in D)x \sqsubseteq y \Rightarrow \psi(x) \sqsubseteq \psi(y)$.

For dealing with closed graph patterns, any pattern is substituted by its tidset. The tidset is used instead of the pattern in $\Sigma o\varphi\iota\alpha$. Details can be found in [10].

B. Naïve Approach for Processing Graphs

A naïve approach is to construct a (huge) binary dataset where every item corresponds to a frequent graph. The tidset of this item is the tidset of the graph. This allows for applying algorithm $\Sigma o\varphi\iota\alpha$ [10] in the following way.

The frequent graphs are canonically ordered and, thus, we name them as $p_1, p_2, ..., p_k$. These graphs form a set P. In the corresponding binary dataset $\mathbb{D}(\mathcal{T}, 2^P, \delta)$ we consider the chain of projections $\psi_0 < \psi_1 < \psi_2 < ... < \psi_k = \mathbb{1}$, where $\psi_i(X \in 2^P) = X \backslash \{p_{i+1}, ..., p_k\}$. For this chain of projections we run $\Sigma o\varphi\iota\alpha$. In the pattern space $\mathcal{D}$, the projection is given by $\tilde{\psi}_i(p) = \{q \in P \mid q \sqsubseteq p\}$.

Proposition 2 ([15]). *Given a graph dataset $(\mathcal{T}, \delta)$ and frequent graphs $P = \{p_1, \ldots, p_k\}$, there is a bijection between*

- *Δ-stable closed graph patterns for $\tilde{\psi}_i((\mathcal{T}, \mathcal{D}, \delta))$;*
- *Δ-stable closed itemsets for $\psi_i((\mathcal{T}, 2^P, \delta))$.*

C. Algorithm gSOFIA

Let us now present gSOFIA (Algorithm 2).

1) Branch cutting: It is unfeasible for real datasets to generate the set P of all frequent graphs. Moreover, the frequency threshold is not known. So, we compute the set P_i of the first i mined graphs corresponding to ψ_i. If we found

```
1: FUNCTION Algorithm_gSOFIA((T, δ), θ)
2: IN: A graph dataset (T, δ), threshold θ for Δ-measure.
3: OUT: The set P of all graph patterns d such that Δ(d) ≤ θ.
4: P ⟵ {∅}
5: SetCurrGraphStable(true)
6: while HasNextSubgraph() do
7:    g ⟵ GetNextSubgraph(IsCurrGraphStable())
8:    t ⟵ 𝕥(g)
9:    P ⟵ ExtendProjection(P, θ, t)
10: return P
11: FUNCTION ExtendProjection(P_prev,θ,t)
12: IN: The set P_prev of current Δ-stable patterns, a threshold θ, and the
    tidset t for the next subgraph.
13: OUT: The set P of Δ-stable patterns d such that Δ(d) ≤ θ.
14: SetCurrGraphStable(false)
15: P ⟵ ∅
16: for all p ∈ P_prev do
17:    p_new ⟵ p ∩ t
18:    if p_new = p then
19:       SetCurrGraphStable(true)
20:       return {p}
21:    InitNewPattern(()t,p,p_new)
22:    if p_new.Δ ≥ θ then
23:       SetCurrGraphStable(true)
24:       P ⟵ P ∪ {p_new}
25:    p.Δ = min(p.Δ, |p| − |p_new|)
26:    if p.Δ ≥ θ then
27:       p.Children ⟵ p.Children ∪ {p_new}
28:       P ⟵ P ∪ {p}
29: return P
30: FUNCTION InitNewPattern(t, p, p_new)
31: IN: The tidset t, old p and new p_new patterns.
32: OUT: Children and Δ-measure are computed for p_new.
33: p_new.Δ ⟵ p.Δ
34: for all child ∈ p.Children do
35:    child_new ⟵ child ∩ t
36:    p_new.Δ ⟵ min(p_new.Δ, |p_new| − |child_new|)
37:    if p_new.Δ < θ then
38:       return
39:    p_new.Children ⟵ p_new.Children ∪ {child_new}
40: FilterChidren(p_new.Children)
```

Fig. 2. Algorithm gSOFIA for finding Δ-stable graph patterns.

the Δ-stable graph patterns for ψ_i, *how do we pass from them to Δ-stable patterns of $\psi_{i+1}(\mathbb{D})$?* First, since we deal with closed graph patterns, we take the tidset of the next graph $\mathbb{t}(p_{i+1})$ and intersect it with the tidsets corresponding to Δ-stable graph patterns from $\psi_i(\mathbb{D})$.

Thus, in lines 6-9 of Algorithm 2 we switch to the next projections while there are frequent subgraphs (line 6). The next graph is requested by function `GetNextGraph()` with a logical parameter discussed later (line 7). Finally it is translated to its tidset (line 8) and every Δ-stable graph pattern is extended to the new projection.

Not all graphs can be found in Δ-stable graph patterns. In particular, if a graph is not found in any Δ-stable graph pattern, then any supergraph cannot be found either. Since most of graph mining canonical orders are instances of either breadth-first or depth-first order, it is easy to stop expansion of a graph. Thus to go from P_i to P_{i+1}, we search for the next graph in the canonical order (line 7). However, if its supergraphs cannot be included into a Δ-stable pattern we backtrack to another frequent graph in the canonical order.

In particular, a subgraph p can be expanded only if p is

contained in a Δ-stable graph pattern of $\psi_i(\mathbb{D})$. We process closed graph patterns by their tidsets. When we compute intersection of the graph tidset $\mathbb{t}(p_i)$ and the tidset t_j (line 17) of a Δ-stable graph pattern d there are three options. First, if $t_j \cap \mathbb{t}(p_i) = t_j$ (lines 18–20), i.e., the graph is in the closure of d, p_i can be further expanded. Second, if a new Δ-stable graph pattern is generated, p_i can also be further expanded (lines 22–24). Thus, if $t_j \cap \mathbb{t}(p_i)$ generates only non-stable graph patterns for all current Δ-stable graph patterns, then p_i and all its supergraphs are rejected. In our algorithm we use a special flag (`SetCurrGraphStable` and `IsCurrGraphStable` functions) that is raised (lines 19 and 23) if at least one Δ-stable graph pattern includes the current frequent subgraph.

Proposition 3. *gSOFIA is sound and complete.*

Proof. Indeed, $\Sigma o \varphi \iota \alpha$ is sound and complete [10]. By means of Proposition 2 we can reduce the proof of the proposotion to a binary dataset. This binary dataset contains all subgraphs that are included into at least one Δ-stable graph pattern. Since Δ-measure is projection-antimonotonic, then addition of new items (subgraphs) to this binary dataset cannot increase the Δ-measure of Δ-stable graph patterns. Later (Proposition 6) we show that such new subgraphs do not change Δ-measure of Δ-stable graph patterns. □

2) Efficient Computation of Δ-measure: Δ-measure is the minimal difference in the size of the tidset of a graph pattern and its immediate closed superpatterns. Thus, if we are able to find the set of all children for a graph pattern d, we can compute Δ-measure for d. So, we maintain the set of all children for Δ-stable graph patterns for any projection.

We can reduce the graph dataset to a binary dataset (Proposition 2). After each iteration there are two types of graph patterns: "old graph patterns" (found in previous iterations) and "new graph patterns" (found in the current iteration). A new graph pattern d is computed as intersection of the tidset of an old graph pattern with the tidset of the current frequent graph. This old graph pattern is called a *graph pattern generator* for d. There can be several graph pattern generators for d and there is only one maximal graph pattern.

Proposition 4. *For any new Δ-stable graph pattern there is exactly one maximal old Δ-stable graph pattern generator.*

Proof. Δ-stable graph patterns can be found in a binary dataset (Proposition 2). For mining closed itemsets there is only one maximal closed itemset generator for any new closed itemset [18]. Let p be the graph added in projection ψ_{i+1} w.r.t. ψ_i. If $p_{max.gen.}$ is not Δ-stable, then $\mathbb{d}(\mathbb{t}(p_{max.gen.}) \cap \mathbb{t}(p)) \equiv p_{new}$ is not Δ-stable (projection-antimonotonicity). But p_{new} is Δ-stable. □

A new Δ-stable graph pattern is a child for the old one. Thus, the set of immediate children of the old one should be updated with the generated new graph pattern (line 27). Now we can update Δ-measure for the old Δ-stable graph pattern (line 25). Let us show how to compute the set of immediate descendant graph patterns for a new Δ-stable graph pattern. The proof is based on [18] and Proposition 2.

Proposition 5. *Tidsets of all immediate closed superpatterns of a new Δ-stable graph pattern p_{new} can be found by intersecting tidsets of the superpatterns of the maximal graph pattern generator of p_{new} with the tidset of the graph g added in the current iteration.*

Thus, for any Δ-stable graph pattern we maintain the set of immediate closed superpatterns. If a new graph pattern is generated, we intersect the tidsets of all immediate closed children of the maximal graph pattern generator with the tidset of the newly added frequent graph (lines 34–39). The resulting set of tidsets is a superset (filtered in line 40) of immediate closed children of the new graph pattern. Now it is easy to find Δ-measure for the new graph pattern.

Example 5. *Consider the example in Figure 1 and the projection chain $\psi_2 \leq \psi_3 \leq \psi_6 = \mathbb{1}$ (recall that $\psi_k(X) = X \setminus \{i_{k+1}, \cdots, i_6\}$). Let $\theta = 2$. For ψ_2 there is only one Δ-stable description: $\emptyset$. It has two immediate closed superitemsets: $\{i_1\}$ and $\{i_2\}$ that are not stable.*

Passing from ψ_2 to ψ_3 we intersect the tidsets of Δ-stable itemsets with the tidset of $\{i_3\}$. Thus, we get the new candidate Δ-stable itemset $\{i_3\}$. Now we check the Δ-measure. For that we compute the immediate closed superitemsets of $\{i_3\}$ by intersecting the immediate closed superitemsets of $\emptyset$, the itemset generator of $\{i_3\}$, with $\mathbb{t}(i_3)$. This produces $\{i_1, i_3\}$ and $\{i_2, i_3\}$. Thus, Δ-measure for $\{i_3\}$, it is equal to 3.

Proposition 6. *Any child of a Δ-stable graph pattern contains only frequent subgraphs from Δ-stable graph patterns or immediate supergraphs of them.*

Proof. Let us assume that there is a Δ-stable graph pattern d, a graph g that is neither contained into any Δ-stable graph pattern nor an immediate supergraph of it, and g generates a new child of d. Since in the canonical order we go from small graphs to large graphs, then before finding a graph g we should find a graph g_Δ that is included into a Δ-stable graph pattern and an immediate supergraph of g_Δ, denoted by $g_{n\Delta}$, which is not contained in a Δ-stable graph patteam. Then, the pattern with $\mathbb{t}(d) \cap \mathbb{t}(g_{n\Delta})$ has already been generated. It corresponds to a more specific graph pattern than d, and thus is already a child of d, i.e., g cannot generate a new child of d. □

3) Finding K-top patterns: Till now we show how to find all Δ-stable patterns w.r.t. a threshold θ. Let us called it θ-gSOFIA. Its complexity is given by $|\mathcal{G}_d| \cdot \max |\mathcal{P}_i| \cdot |\mathcal{T}|^2$, where $\mathcal{G}_d$ is the set of discovered graphs. Now we can limit the number of stored Δ-stable graph patterns in each iteration, i.e., $|\mathcal{P}_i| <= K$, by automatically adjusting θ. It gives us gSOFIA with the complexity $O(|\mathcal{G}_d| \cdot K \cdot |\mathcal{T}|^2)$.

V. Experiments

We used *Intel(R) Pentium(R) CPU G2120 @ 3.10GHz* with 4GB of RAM under XUbuntu 14.04 for evaluation. The

TABLE I
THE SUMMARY INFORMATION ABOUT THE USED DATASETS

Dataset	Size	Avg. Vertices	Avg. Edges
PTC[5]	417	25.6	26.0
C340[6]	340	27.0	27.4
C422[6]	422	39.6	42.3
AIDS[7]	42687	45.7	47.7
NCI330[8]	23050	24.8	26.5
NCI81[8]	24061	29.2	31.6
DD[8]	1178	284.3	715.6
NCI-Ch[9]	280816	22.5	23.9
NCI-all[10]	160366	29.0	31.4
PDB[11]	189771	12.3	12.3

canonical orders of `Gaston` [19][1] and `gSpan` [20][2] are used. gSOFIA is implemented in the framework of FCAPS[3].

Here we focus on computational efficiency of gSOFIA, since the importance of Δ-stable patterns is already highlighted [6], [21]. We see three options to evaluate the performance of gSOFIA. First, we can compare the runtime of gSOFIA with the runtime of frequent graph mining in the corresponding canonical order. This allows us to study the efficiency of branch-cutting in gSOFIA. Second we can compare runtime of gSOFIA and $\Sigma o \varphi \iota \alpha$ for binary attributes, thus we encode a dataset as a huge binary dataset and run $\Sigma o \varphi \iota \alpha$. And third, we can process the equivalent binary dataset with an algorithm constructing the whole set of frequent closed itemsets. Since we need to know all children for any graph pattern to compute Δ-measure, we rely on very efficient algorithm `Charm-L` [22][4].

We performed experiments on datasets of various sizes. The summary of the datasets is given in Table I.For example, the first row of the table describes a dataset from PTC challenge. It contains 417 molecular graphs with 25.6 vertices and 26.0 edges on average.

Results and Discussion: In the first series of experiments we run gSOFIA and its competitors on all datasets. The support threshold is set to 7% since it allows one to run `gSpan` and `Gaston` on all datasets. The computation results are given in Table II. For example, in `PTC` dataset there are only 2850 graph patterns with frequency larger than 7%. The computation of gSOFIA takes from 0.1 to 4.2 seconds depending on the canonical order and the number (K) of stored Δ-stable patterns. The computation of frequent graphs by `gSpan` and `Gaston` takes 1.7 seconds and 0.2 seconds, respectively. The computation of $\Sigma o \varphi \iota \alpha$ takes from 0.1 to 0.7 for $K \in \{10, 1000\}$. Pure `Charm-L` takes 1.6 seconds.

First, in mining Δ-stable patterns `Charm-L` is an outsider. Moreover, it was not able to finish computation within 10 hours in 6 experiments. It was outperformed by both gSOFIA and $\Sigma o \varphi \iota \alpha$. It is not surprising, since `Charm-L` uses no optimization regarding the search of Δ-stable patterns. For the next experiment we do not report on `Charm-L`. Among the implementations of frequent graph mining algorithms `Gaston` is faster than `gSpan`. So, we compare the runtime of gSOFIA based on `Gaston` with that of `Gaston` alone and not with that of `gSpan`. Note that we should add the time of `Gaston` or `gSpan` to the time of $\Sigma o \varphi \iota \alpha$ and `Charm-L` since the latter rely on the output of the former.

On `PDB` dataset $\Sigma o \varphi \iota \alpha$ is not able to terminate, since it does not fit into the memory. gSOFIA does not require to fit any context into the memory, so it can process this dataset. By contrast, $\Sigma o \varphi \iota \alpha$ is more efficient for high frequency thresholds. We vary the frequency threshold in the next experiment.

In Table III the runtime is shown w.r.t. the support threshold. We report `ERR` in the table, when the corresponding procedure cannot terminate. In the case of $\Sigma o \varphi \iota \alpha$ it cannot fit large contexts to the memory. In the case of gSOFIA and `Gaston` the latter tries to allocate too much memory. If the error appears we do not run the more complex experiments. If an error for $\Sigma o \varphi \iota \alpha$ appeared then we do not run `Gaston` either, since it cannot compute the graph patterns. Here we set $K = 10$. We give runtime for gSOFIA with the canonical order of `Gaston`, since it is faster. We also report time for extracting frequent graphs and processing the result with $\Sigma o \varphi \iota \alpha$. We can see that for most of the datasets the runtime for gSOFIA is quite stable and a small growth is explained by the fact that more and more irrelevant graphs are passed from the canonical order. And they are rejected immediately. In contrast to gSOFIA, `Gaston` is obliged to find all frequent graphs. Thus the time increases exponentially. The number of graphs is also increasing and $\Sigma o \varphi \iota \alpha$ slows down.

VI. CONCLUSIONS

In this paper we have proposed an efficient algorithm for finding subsample-stable graph patterns. For numerous benchmarks the algorithm was shown to be faster than the existing ones, as well as faster than frequent graph mining algorithms based on the same canonical orders.

VII. ACKNOWLEDGMENTS

Sections 1-3 were written by Sergei O. Kuznetsov supported by the Russian Science Foundation under grant 17-11-01294 and performed at National Research University Higher School of Economics, Russia.

[1] https://github.com/AlekseyBuzmakov/LibGastonForSofia
[2] https://github.com/AlekseyBuzmakov/LibgSpanForSofia
[3] https://github.com/AlekseyBuzmakov/FCAPS
[4] http://www.cs.rpi.edu/~zaki/www-new/pmwiki.php/Software/Software.
[5] http://oldwww.comlab.ox.ac.uk/oucl/groups/machlearn/PTE
[6] `gSpan` dataset https://www.cs.ucsb.edu/~xyan/software/gSpan.htm
[7] https://wiki.nci.nih.gov/display/NCIDTPdata/AIDS+Antiviral+Screen+Data
[8] http://www.dbs.ifi.lmu.de/cms/Publications/Discriminative_Frequent_Subgraph_Mining_with_Optimality_Guarantees
[9] https://wiki.nci.nih.gov/display/NCIDTPdata/Chemical+Data
[10] All NCI dataset into one from ([8])
[11] http://www.rcsb.org/pdb/download/download.do

REFERENCES

[1] H. Cheng, X. Yan, and J. Han, "Mining Graph Patterns," in *Freq. Pattern Min.* Cham: Springer, 2014, pp. 307–338.
[2] Z. Zeng, J. Wang, J. Zhang, and L. Zhou, "FOGGER : An Algorithm for Graph Generator Discovery," in *Proc. 12th Int. Conf. Extending Database Technol. Adv. Database Technol. - EDBT '09*. New York, New York, USA: ACM Press, mar 2009, pp. 517–528.

TABLE II
RUNTIME FOR 7% SUPPORT THRESHOLD

		gSOFIA						Σοφια		
		gSpan		Gaston						
dataset	#	K=10	K=1000	K=10	K=1000	gSpan	Gaston	K=10	K=1000	Charm-L
PTC	2850	2.3	4.2	0.1	3.9	1.7	0.2	0.1	11	1.6
C340	1845	1.2	2.1	< 0.1	1.3	0.9	< 0.1	< 0.1	5.9	0.4
C422	65252	23	31	0.1	18	16	1.4	6.0	1415	3.5
AIDS	10993	736	867	13	188	626	53	20	232	–
NCI330	3856	49	93	2.7	40	40	10	2.9	33	–
NCI81	6406	112	236	8.8	155	87	19	5.0	64	–
DD	301415	7380	32580	166	12300	255	66	44	244	3862
NCI-Ch	267	68	72	15	50	58	13	13	38	–
NCI-All	3411	360	421	27	39	299	73	19	226	–
PDB	21333	600	–	24	–	520	90	–	–	–

TABLE III
RUNTIME W.R.T. THE SUPPORT THRESHOLD.

Dataset	Algo	4.7%	3.5%	2.3%	1.7%	1.2%	0.1%
PTC	gΣ+G	0.12	0.12	0.13	0.13	0.14	ERR
	G	0.4	0.7	4.3	15	325	ERR
	Σ	1.0	2.5	15	60	217	ERR
C340	gΣ+G	0.1	0.1	0.1	0.1	0.1	ERR
	G	0.2	0.4	2.3	15	1475	ERR
	Σ	0.2	0.6	5.2	28	570	ERR
C420	gΣ+G	0.1	0.1	0.1	0.1	0.1	ERR
	G	83	1770				
	Σ	203	ERR				
AIDS	gΣ+G	14	14	14	15	15	21
	G	113					
	Σ	ERR					
NCI330	gΣ+G	2.8	3.1	3.2	3.2	3.4	3.6
	G	18.7	344				
	Σ	6.7	ERR				
NCI81	gΣ+G	8.8	9.6	10.8	12.7	12.9	25
	G	39	72	163			
	Σ	9.1	18	ERR			
DD	gΣ+G	235	306	410	480	554	–
	G	128					
	Σ	ERR					
NCI-Ch	gΣ+G	16	16	16	17	17	20
	G	16	18	22	25	30	248
	Σ	25	26	27	27	30	ERR
NCI-all	gΣ+G	32	32	34	36	48	
	G	135					
	Σ	ERR					
PDB	gΣ+G	24	24	25	26	26	28
	G	739					
	Σ	ERR					

[3] M. Boley, T. Horváth, A. Poigné, and S. Wrobel, "Listing closed sets of strongly accessible set systems with applications to data mining," *Theor. Comput. Sci.*, vol. 411, no. 3, pp. 691–700, jan 2010.

[4] S. O. Kuznetsov, "On stability of a formal concept," *Ann. Math. Artif. Intell.*, vol. 49, no. 1-4, pp. 101–115, 2007.

[5] N. Tatti, F. Moerchen, and T. Calders, "Finding Robust Itemsets under Subsampling," *ACM Trans. Database Syst.*, vol. 39, no. 3, pp. 1–27, 2014.

[6] C. Roth, S. A. Obiedkov, and D. G. Kourie, "On succinct representation of knowledge community taxonomies with formal concept analysis," *Int. J. Found. Comput. Sci.*, vol. 19, no. 02, pp. 383–404, apr 2008.

[7] F. Moerchen, M. Thies, and A. Ultsch, "Efficient mining of all margin-closed itemsets with applications in temporal knowledge discovery and classification by compression," *Knowl. Inf. Syst.*, vol. 29, no. 1, pp. 55–80, 2011.

[8] A. Soulet and B. Crémilleux, "Optimizing constraint-based mining by automatically relaxing constraints," in *Proc. 5th IEEE Inter- Natl. Conf. Data Min. (ICDM 2005)*. IEEE Computer Society, 2005, pp. 777–780.

[9] A. Soulet and B. Cremilleux, "Adequate condensed representations of patterns," *Data Min. Knowl. Discov.*, vol. 17, no. 1, pp. 94–110, 2008.

[10] A. Buzmakov, S. O. Kuznetsov, and A. Napoli, "Fast Generation of Best Interval Patterns for Nonmonotonic Constraints," in *Mach. Learn. Knowl. Discov. Databases*, ser. LNCS. Springer, 2015, vol. 9285, pp. 157–172.

[11] B. Ganter and S. O. Kuznetsov, "Pattern Structures and Their Projections," in *Concept. Struct. Broadening Base*, ser. LNCS. Springer, 2001, vol. 2120, pp. 129–142.

[12] M. Kaytoue, V. Codocedo, A. Buzmakov, J. Baixeries, S. O. Kuznetsov, and A. Napoli, "Pattern Structures and Concept Lattices for Data Mining and Knowledge Processing," in *Mach. Learn. Knowl. Discov. Databases*, ser. LNCS. Springer, 2015, vol. 9286, pp. 227–231.

[13] X. Yan and J. Han, "CloseGraph: mining closed frequent graph patterns," in *Proc. ninth ACM SIGKDD Int. Conf. Knowl. Discov. data Min.*, ser. KDD '03. USA: ACM, 2003, pp. 286–295.

[14] S. O. Kuznetsov, "Learning of Simple Conceptual Graphs from Positive and Negative Examples," in *Princ. Data Min. Knowl. Discov. SE - 47*, ser. LNCS. Springer, 1999, vol. 1704, pp. 384–391.

[15] S. O. Kuznetsov and M. V. Samokhin, "Learning Closed Sets of Labeled Graphs for Chemical Applications," in *Inductive Log. Program. SE - 12*, ser. LNCS. Springer, 2005, vol. 3625, pp. 190–208.

[16] B. Efron, "Bootstrap Methods: Another Look at the Jackknife," *Ann. Stat.*, vol. 7, no. 1, pp. 1–26, 1979.

[17] A. Buzmakov, S. O. Kuznetsov, and A. Napoli, "Scalable Estimates of Concept Stability," in *Form. Concept Anal.*, ser. LNCS. Springer, 2014, vol. 8478, pp. 161–176.

[18] D. G. Kourie, S. A. Obiedkov, B. W. Watson, and D. van der Merwe, "An incremental algorithm to construct a lattice of set intersections," *Sci. Comput. Program.*, vol. 74, no. 3, pp. 128–142, 2009.

[19] S. Nijssen and J. N. Kok, "The Gaston Tool for Frequent Subgraph Mining," *Electron. Notes Theor. Comput. Sci.*, vol. 127, no. 1, pp. 77–87, mar 2005.

[20] X. Yan and J. Han, "gSpan: Graph-Based Substructure Pattern Mining," in *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, 2002, pp. 721–724.

[21] J.-P. Métivier, A. Lepailleur, A. Buzmakov, G. Poezevara, B. Crémilleux, S. Kuznetsov, J. Le Goff, A. Napoli, R. Bureau, and B. Cuissart, "Discovering structural alerts for mutagenicity using stable emerging molecular patterns," *J. Chem. Inf. Model.*, vol. 55, no. 5, pp. 925–940, 2015.

[22] M. J. Zaki and C.-J. Hsiao, "Efficient algorithms for mining closed itemsets and their lattice structure," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 4, pp. 462–478, apr 2005.