# Mining and Ranking of Generalized Multi-Dimensional Frequent Subgraphs

***Abstract*—Frequent pattern mining is an important research field and can be applied to different labeled data structures ranging from itemsets to graphs. There are scenarios where a label can be assigned to a taxonomy and generalized patterns can be mined by replacing labels by their ancestors. In this work, we propose a novel approach to generalized frequent subgraph mining. In contrast to existing work, our approach considers new requirements from use cases beyond molecular databases. In particular, we support directed multigraphs as well as multiple taxonomies to deal with the different semantic meaning of vertices. Since results of generalized frequent subgraph mining can be very large, we use a fast analytical method of p-value estimation to rank results by significance. We propose two extensions of the popular gSpan algorithm thjat mine frequent subgraphs across all taxonomy levels. We compare both algorithms in an experimental evaluation based on a database of business process executions represented by graphs.**

## I. Introduction

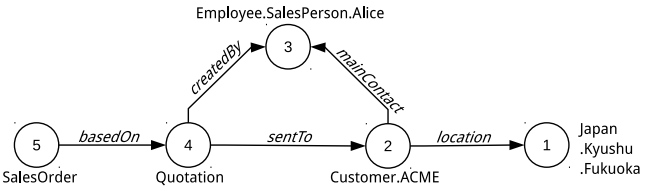
Frequent pattern mining is an important research problem and found much interest since the early nineties [1]. Generally speaking, a pattern is a collection of labels attached to a respective data structure such as itemsets, sequences, trees and graphs. In many applications labels can be assigned to taxonomies and mining patterns at different taxonomy levels [9] may reveal interesting patterns. For example, the pattern $\{bread, butter\}$ could be infrequent while the more general one $\{bakery\ product, milk\ product\}$ is frequent. In some cases, analysts also want to analyze patterns across levels [6]. For example, the pattern $\{wholegrain\ bread, butter\}$ can be more interesting than just $\{bread, butter\}$. Finally, users also want to analyze patterns in the context of multiple dimensions [28], for example, to find out that $\{bread, butter\}$ is mostly bough in the *morning* in *suburban stores*.

These simple examples show that an elaborate approach to frequent pattern mining must be capable to mine patterns across multiple levels of multiple dimensional taxonomies. However, an respective approach to generalized multi-dimensional pattern mining have only been studied for sequences [29]. With regard to graphs, generalization has already been investigated [13] but under the assumption that all vertices belong to the same semantic class (e.g., atoms). In this work, we propose the first approach to *generalized multi-dimensional frequent subgraph mining* (*GM-FSM*).

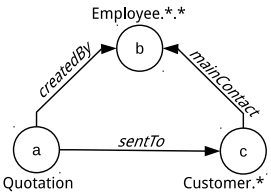
But let us first have a look on an example application of the resulting graph patterns: Data generated during business process executions can be represented by graphs [26]. Figure 1a shows an example subgraph of sales process data. Both vertices and edges show labels. Edges and most vertices show simple labels (e.g., *Quotation*, *createdBy*). Some vertices

Fig. 1: Example multi-dimensional subgraph and patterns.

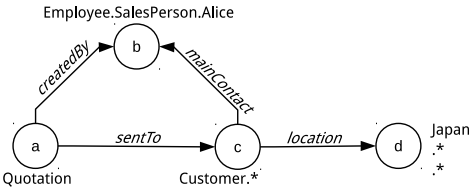
(a) Subgraph.




(b) Trivial pattern.



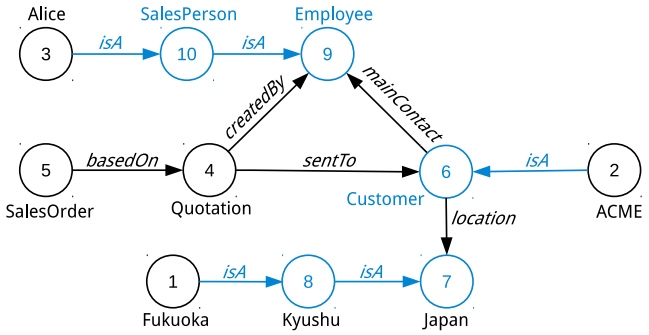

(c) Interesting pattern.




are labeled by *taxonomy paths* because they are attached to dimensional taxonomies. Taxonomy paths are represented by *[top-level].⋯.[bottom-level]*. For example, vertex 3 not only represents *Alice* but also more generally a *SalesPerson* and an *Employee*. This information is taken from a respective taxonomy to which *Alice* is assigned. Further on, vertex 4 represents *Fukuoka* a city on the island *Kyushu* and so on. Since the subgraph's vertices are associated to different taxonomies its contained patterns are considered to be *multi-dimensional*.

The problem of frequent subgraph mining is the extraction of graph patterns that occur in at least a minimum number of graphs in a collection (*minimum support*). In scenarios where bottom-level labels (e.g., *Alice*) may have low frequencies, mining frequent graph patterns on the bottom-level will barely lead to frequent results. On the other hand, considering only the top-levels (e.g., *Employee.*.*.*) leads only to very general patterns such as the one of Figure 1b. The pattern is expressing that a quotation was sent by an employee to a customer, where the latter has the sending employee as its main contact. However, this pattern might occur in all sales process executions

Fig. 2: Path-substitution method: Taxonomy paths are represented by dedicated vertices and edges (blue lines).




and, thus, is considered to be *trivial*. To find more interesting patterns, respective labels must be considered at arbitrary level combinations. By doing so, we can extract patterns as the one of Figure 1c. This one is expressing, that the specific employee Alice sent a quotation to a customer from Japan. Here, labels at different levels are combined, in particular, the top-level label *Japan.*.** and the bottom-level label *Employee.Sales.Alice*.

The naive approach to extract such patterns would be to mine bottom-levels first with a very low minimum support threshold and generalize patterns in a post processing step. In order to guarantee completeness, the minimum support threshold must be zero. However, since the frequent subgraph mining (FSM) contains the NP-complete subgraph isomorphism problem, a threshold close to zero would lead to an exploding result size as well as a dramatical increase of response time. For this reason, we developed two novel methods to GM-FSM which take the special characteristics of taxonomy paths into account. Both methods are used on the gSpan [36], an efficient FSM algorithm. As a problem relaxation we require vertices always to have a semantic meaning, i.e., we are not interested in structure-only patterns.

In the first method, we use a preprocessing step to integrate taxonomy paths into the graph structure as shown by Figure 1a. Additionally, we apply two modifications to find all but result and remove false-positives. In the second method, we decompose the problem into FSM and *generalized frequent vector mining (GFVM)*. Here, we decompose subgraphs and patterns into a top-level (most general) graph structure and a vector of lower-level tails. In particular, top-level labels are attached to the graph structure, lower level tails of taxonomy paths are stored in the vector and vertices are mapped to vector fields respectively. For example, we can derive a vector $\langle$ *Kyushu.Fukuoka, ACME, SalesPerson.Alice* $\rangle$ from the subgraph shown by Figure 1a where the 2nd vector field contains the tail of taxonomy path *Customer.ACME* of vertex 2. The extraction happens in two steps: First, we use a modified version of gSpan to identify frequent structural patterns based on only top-level labels including an additional set of lower-level vectors. Second, we apply GFVM to refine the result.

Depending on pattern size and the number of dimensional attribute combinations, GM-FSM may return a huge number of graph patterns. Thus, we use p-values to rank patterns by significance. More significant patterns are considered to be more interesting and presented first. P-values are a statistical measure reflecting the probability of a graph pattern to occur randomly under given graph statistics, such as label and degree distributions. In consequence, patterns with less frequent labels (e.g., bottom-levels) and extraordinary structure are more significant among the frequent ones. P-values for graph patterns are usually determined by generating a large number of random graphs and count how often a given pattern was created. However, since such methods show poor scalability, we use a fast analytical approximation instead.

Our contributions can be summarized as follows:

- We firstly study the problem of multi-dimensional generalized frequent subgraph mining and propose two efficient methods (Section II).
- We discuss the use of p-value approximation to rank result graph patterns (Section III) to present most significant results first. We use an efficient analytical model to achieve fast response times.
- We present results of performance evaluations in a business intelligence scenario where graphs represent business process executions (Section IV).

Additionally, we discuss related work in Section V and conclude with a preview on future work in Section VI.

## II. Generalized Multi-dimensional Frequent Subgraph Mining

### A. Problem statement

The basic problem definition is the same as the one of frequent subgraph mining (FSM) in the graph-transaction setting [16]: input is a collection of graphs $\mathcal{G} = \{G_1, \ldots, G_n\}$ and output is a set of frequent graph patterns $\mathcal{F} = \{P_1, \ldots, P_m\}$. For each $P \in \mathcal{F}$ there is a graph collection $\mathcal{G}_P \subseteq \mathcal{G}$ where each graph supports the pattern. Based on the size of both collections we can derive a pattern's support $sup(P) = {}^{|\mathcal{G}_P|}/_{|\mathcal{G}|}$. The algorithm's only configuration is a minimum support threshold $0 \leqslant sup_{min} \leqslant 1$. A pattern will be considered frequent if its support is above this threshold. The result will be complete if $sup(P) \geqslant sup_{min} \Leftrightarrow P \in \mathcal{F}$ holds.

The difference between FSM and GM-FSM is the definition of *pattern support*. In FSM, a graph will support a pattern if there is an isomorphic subgraph with equals labels for all mapped pairs of vertices and edges. In GM-FSM, we additionally consider patterns which are *generalizations* of a subgraph to be supported. In the next subsection, we will define graph generalization more precisely.

### B. Data Model and Graph Generalization

In our data model every dimension of a graph or pattern is represented by a vertex that can be linked to a taxonomy.

A **taxonomy** is defined as a balanced rooted tree:

$$T = \langle S_T, R_T, \ell_T \rangle \tag{1}$$

$S_T$ is a set of vertex labels, $R_T \subseteq (S_T)^2$ is a set of *isA* relationships and $\ell_T \in S$ is the root label.

Under a given taxonomy $T$ a **taxonomy path**[1] $\rho_T(\ell_i)$ is defined as an arbitrary path from root $\ell_T$ to $\ell_i$:

$$\rho_T(\ell_i) = \ell_T.\ell_2.\cdots.\ell_i \tag{2}$$

An important part of graph generalization is label generalization within a taxonomy. Therefore, we introduce an operator $\ell_1 \sqsubseteq_T \ell_2$ that will return *true*, iff $\ell_1$ is a generalization of $\ell_2$, so that $(S_T, \sqsubseteq_T)$ is a poset. Let $\ell_1, \ell_2 \in S_T$ be two labels of the same taxonomy and $\sqcap$ another operator returning the common path between paths $\rho_T(\ell_1)$ and $\rho_T(\ell_2)$, then **label generalization** is defined as follows:

$$\ell_1 \sqsubseteq_T \ell_2 \Leftrightarrow \rho_T(\ell_1) \sqcap \rho_T(\ell_2) = \rho_T(\ell_1) \tag{3}$$

At this point, we will be able to associate vertices of a graph to taxonomy labels. Given two global alphabets of vertex labels ($\Sigma_V$) and edge labels ($\Sigma_E$) as well as a global set of taxonomies $\mathcal{T} = \{T_1, \ldots, T_n\}$, we define a **graph** as the following tuple:

$$G = \langle V, E, s, t, \delta, \lambda, \tau \rangle \tag{4}$$

In this graph model $V = \{1, \ldots, v\}$ is the set of vertex identifiers and $E = \{1, \ldots, e\}$ is a set of edge identifiers. To support loops and parallel edges (*multigraph property*) the functions $s : E \to V$ / $t : E \to V$ map a *source* and a *target* vertex to a every edge. An edge $e \in E$ is *directed* from $s(e)$ to $t(e)$. The functions $\delta : V \to \Sigma_V$ / $\lambda : E \to \Sigma_E$ associate a label to every vertex and edge. For sake of simplicity, a function $\tau\colon V \to \mathcal{T}$ maps every vertex $v$ into its correspondent taxonomy $\tau(v)$. Vertices with simple labels are formally mapped to a dedicated taxonomy of depth 1, for example, to taxonomy with root *misc* in Figure 3. We demand consistency s.t. $\forall v \in V : \delta(v) \in S_{\tau(v)}$ holds.

At this point, we need to define an enforced notion of **isomorphism** taking into account that besides structure, vertex taxonomies (dimensions) and edge labels must match:

$$G_1 \cong G_2 \Leftrightarrow \begin{array}{l} \exists \alpha : V_{G_1} \leftrightarrow V_{G_2}.\exists \beta : E_{G_1} \leftrightarrow E_{G_2}. \\ \Big(\forall v \in V_{G_1}.\tau(v) = \tau(\alpha(v))\Big) \wedge \\ \Big(\forall e \in E_{G_1}.\lambda(e) = \lambda(\beta(e)) \wedge \\ \alpha(s(e)) = s(\beta(e)) \wedge \alpha(t(e)) = t(\beta(e))\Big) \end{array} \tag{5}$$

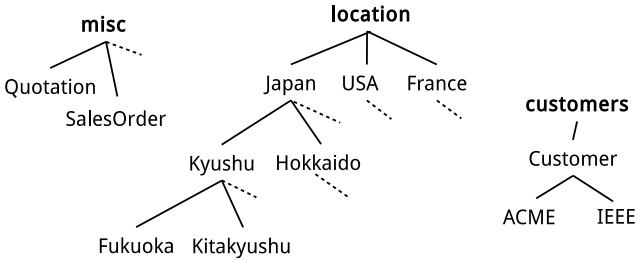
Finally, we can define **graph generalization** as follows:

$$G_1 \sqsubseteq G_2 \Leftrightarrow G_1 \cong G_2 \wedge \forall v \in V_{G_1}.\delta(v) \sqsubseteq_{\tau(v)} \delta(\alpha(v)) \tag{6}$$

In the remainder of this paper we will use the notations of $\ell_1 \sqsubset_T \ell_2$ and respectively $G_1 \sqsubset G_2$ to explicitly exclude equality from generalization, i.e., sharing equal vertex labels.

[1]Since the length $n$ of taxonomy paths is bounded by the depth $m$ of the associated taxonomy s.t. $0 \leqslant n \leqslant m$ we use sequences of $*$ symbols with length $m-n$ in illustrations to express that a taxonomy path of length $n < m$ is head of all longer paths containing it, i.e., $\ell_T.\ell_2.*.*$ means a label $\ell_T.\ell_2$ with $m = 4$

Fig. 3: Example taxonomies of Figures 1 and 2.




### C. Terminology

Before we can explain GM-FSM in more detail, we need to further define important terms used throughout this paper. First, the input graph's vertices must have taxonomy leafs as labels. Thus, their subgraphs can be described by patterns of only taxonomy leafs, too. Thus, **input graphs** $G$ as well as the **bottom-level patterns** $P^{bot}$ contained in $G$ must satisfy the following property:

$$\forall v \in V_G \backslash V_{P^{bot}}.\ \nexists \ell \in S_{\tau(v)}.\ \delta(v) \sqsubset_{\tau(v)} \ell \tag{7}$$

Second, the graph generalization definition in Equation 6 could lead to meaningless patterns due to a schema overgeneralization [13] (e.g., a *customer* that has a *location* which occurs as frequent as only a *customer*). For this reason, we generalize graphs by generalizing vertex labels only up to the second level of taxonomies. Thus, it is an application-specific data modeling decision whether the second level (*top-level*) will be an identifier for the taxonomy (e.g., taxonomy with root *customers* in Figure 3) or just provide specializations (e.g., taxonomy with root *location* in Figure 3). Formally, a **top-level pattern** $P^{top}$ must satisfy the following property:

$$\forall v \in V_P.\exists \left\langle \ell, \ell_{\tau(v)} \right\rangle \in R_{\tau(v)}.\ell = \delta(v) \tag{8}$$

Please note that the term "top-level pattern" must be distinguished from the "most generalized graph" according to Equation 6, since the notion of "most generalized graph" is only referred to isomorphic graphs as in Equation 5.

### D. Frequent Subgraph Mining

Both of our methods are based on the gSpan algorithm [36], in particular an extension supporting directed multigraphs [24]. The most fundamental component of gSpan is the use of DFS codes to represent graph patterns in a canonical form. The algorithm is efficient since DFS codes are directly generated during a constrained depth first search in the graph with only few duplicate detections of the same pattern. In a normalized[2] way a **DFS code** of a pattern $P$ can be represented by the following pair:

$$C_P = \langle \langle \delta(v_1), .., \delta(v_n) \rangle, \langle x_1, .., x_k \rangle \rangle \tag{9}$$

In this representation $\langle \delta(v_i) \rangle_{1 \leqslant i \leqslant n}$ is a tuple of labels of $n$ visited vertices and $\langle x_j \rangle_{1 \leqslant j \leqslant k}$ a tuple of $k$ edge extensions.

[2]Typically (e.g., in [36]) vertex labels are shown redundantly within extensions. Thus, we use the attribute *normalized*.

Since the latter are not relevant for the remainder of this paper, we refer to [37] and omit further details about their representation. The indices $i, j$ correspond to the discovery order, for example, $v_1$ was is the first extension's start vertex.

To explain our methods, we briefly explain gSpan and its pseudocode shown by Algorithm 1. The algorithm's input are besides a graph collection and a minimum support also a maximum edge count $k_{max}$. The latter is important to limit response times for scenarios where graphs can be very large and contain many frequent schema-induced automorphisms, e.g., business process executions [25].

In a preprocessing step (line 1) frequently supported vertex and edge labels are determined to create label dictionaries. The dictionaries are used to replace strings by integer translations (line 2) and to remove vertices and edges showing infrequent labels. gSpan is using a lexicographical order among DFS codes and discovers them in a pattern-growth manner, i.e., children are generated from parents by 1-edge extensions and there is a DFS code tree of parent-child relationships. $\mathcal{Q}$ represents a queue of unextended parents and initially contains only an empty root-pattern $P_{root}$ (line 3). The parents in queue are processed subsequently (line 6). For each parent children are generated (line 7) and evaluated (line 8). In the method *countAndPrune* we not only filter by minimum support but also verify the few duplicates generated by gSpan (non canonical DFS codes [37]). Finally, all frequent patterns are added to the output $\mathcal{F}$ (lines 4,9,12) and those whose childrens' size will be less than or equal to $k_{max}$ are added to the queue (line 10).

### E. Path-Substitution Method

In the path-substitution method we replace vertices whose labels are not the top-level of their associated taxonomy by the respective taxonomy path. For example, Figure 2 shows Figure 1a after path-substitution. In a typical semantic annotation one would just add the taxonomy path to the vertex, e.g.:
*Quotation* $\xrightarrow{sentBy}$ *Alice* $\xleftarrow{isA}$ *SalesPerson* $\xleftarrow{isA}$ *Employee*

However, for the pattern mining process it is important that the bottom-level (e.g. vertex 3, *Alice*) is replaced by the top-level (e.g. vertex 9, *Employee*). Otherwise a semantic generalization like *Quotation* $\xrightarrow{sentBy}$ *SalesPerson* cannot be found if *Alice* is infrequent. In contrast, although *Alice* is infrequent, path-substitution allows to extract equivalent patterns like:
*Quotation* $\xrightarrow{sentBy}$ *Employee* $\xrightarrow{isA}$ *SalesPerson*

With regard to Algorithm 1 our modification affects multiple parts. First, there must be a preprocessing step before line 1 to perform the actual path-substitution. Thus, all generalized labels as well as the *isA* edge label will pass the dictionary coding and infrequent labels will already be pruned. Second, the edge count restriction of line 10 must consider only logical edges, i.e., those showing not the label *isA*. In consequence, structurally extracted patterns may be much larger that $k_{max}$. Third, gSpan considers taxonomy paths, i.e., paths whose edges all show label *isA*, as normal subgraphs. Thus they must not be added to the result in line 9.

**Algorithm 1** gSpan algorithm

**Require:** $\mathcal{G}, f_{min}, k_{max}$
1: $D_v, D_e \Leftarrow frequentLabelDictionaries(\mathcal{G})$
2: $\mathcal{G} \Leftarrow dictionaryCoding(\mathcal{G}, D_v, D_e)$
3: $\mathcal{Q} = \{P_{root}\}$
4: $\mathcal{F} = \varnothing$
5: **while** $\mathcal{Q} \neq \varnothing$ **do**
6: $\quad P_p^k \Leftarrow \mathcal{Q}.poll()$; // step $k$
7: $\quad \mathcal{P}_c^{k+1} \Leftarrow growChildren(\mathcal{G}, P_p^k)$
8: $\quad \mathcal{F}_c^{k+1} \Leftarrow countAndPrune(\mathcal{P}_c^{k+1}, f_{min})$
9: $\quad \mathcal{F} \Leftarrow \mathcal{F} \cup \mathcal{F}_c^{k+1}$
10: $\quad$ **if** $k + 2 \leqslant k_{max}$ **then**
11: $\quad\quad \mathcal{Q} \Leftarrow \mathcal{Q} \cup \mathcal{F}_c^{k+1}$ // if children of $k+1$ reach $k_{max}$
12: $\quad$ **end if**
13: **end while**
14: **return** $\mathcal{F}$

### F. Pattern-Decomposition Method

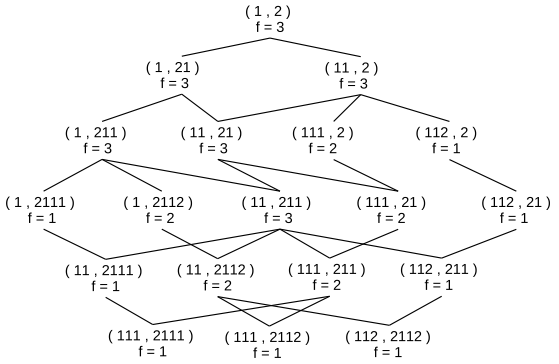
The theoretical disadvantage of the Path-substitution method are additional isomorphism resolutions caused by the inserted edges. Thus, we investigated a second method based on pattern-decomposition. Here, we exploit that all common generalizations of two patterns are isomorphic (see Definitions 5 and 6). Further on, since our taxonomies are trees, any top-level pattern is at least as frequent as any of its specializations. In return, a pattern can only be frequent if its top-level pattern is, too. Thus, we first mine frequent top-level patterns and frequent specialization vectors (see next section) in a subsequent step. In this way the number of isomorphism resolutions is bounded by the number of top-level patterns.

Therefore, Algorithm 1 is changed as follows: When deriving patterns from subgraphs (line 7), we decompose every pattern into a triple $\langle C_{P^{top}}, \vec{\ell}, \iota \rangle$ containing top-level DFS code $C_{P^{top}}$, a bottom-level label vector $\vec{\ell}$ and a mapping $\iota$. In order to avoid redundant information, we will store the bottom level only if it differs from the top-level: therefore, we additionally need a mapping $\iota : \mathbb{N} \to \mathbb{N}$ from vector fields to the vertex order of the DFS code. Based on the decomposition, we are able to first mine frequent top-level patterns (line 8) and identify frequent specializations afterwards by mining all frequent generalized vectors attached to the same top-level pattern. Finally, we derive a specialized DFS code for each frequent vector by replacing vertex labels according to the mapped fields from $\iota$ and add them to the output (line 9).

### G. Generalized Frequent Vector Mining

The problem of *Generalized frequent vector mining* aims to identify a set of frequent generalizations from a given set of vectors which fields occur in one or more taxonomy. Input for each execution is a triple $\langle C_P, L, \iota \rangle$ of DFS code $C_P$, a list of vectors $L = \langle \vec{\ell}_1, \ldots, \vec{\ell}_m \rangle$ and a mapping $\iota$. Since DFS codes provide an order $v_1, \ldots, v_n$ for the vertices of a pattern $P$, where each of them is mapped to just one taxonomy, the $i$-th vector dimension shall correspond to the taxonomy of

Fig. 4: Generalization search lattice for a 2-dimensional vector set $L = \{(111, 2111), (111, 2112), (112, 2112)\}$. $f$ indicates frequency (absolute support). Common prefixes indicate label generalizations (e.g., $11 \sqsubset_{T_1} 112$). Edges represent vector generalization from bottom to top.




the vertex $v_{\iota(i)} \in C_P$. Hereby, the **vector space** is defined as follows:

$$\mathbb{S}_P = S_{\tau(v_1)} \times \cdots \times S_{\tau(v_n)} \tag{10}$$

Thus, we can define **vector generalization** as follows:

$$\forall \vec{\ell}, \vec{\ell}' \in \mathbb{S}_P. \quad \vec{\ell} \sqsubseteq_{\mathbb{S}_P} \vec{\ell}' \;\Leftrightarrow\; \ell_i \sqsubseteq_{\tau(v_i)} \ell'_i \mid 1 \leqslant i \leqslant n \tag{11}$$

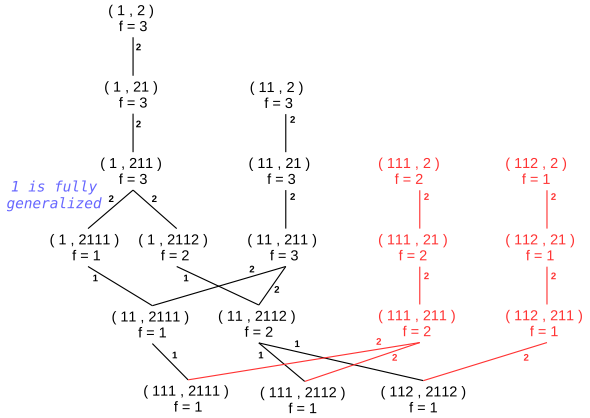
Based on this definition, the search space of all possible generalizations of vector set $L$ can described by a lattice. Figure 4 shows a respective lattice for an example vector set, where each label $\ell_i$ refers to a vertex $v_i$. To indicate taxonomy paths labels are represented by a canonical labeling for trees [20], e.g., label Fukuoka in Figure 3 can be expressed by 1111, while IEEE is now 212. To efficiently extract all frequent generalizations we must avoid visiting lattice nodes multiple times. We investigated two algorithms satisfying this requirement:

*1) Bottom-up search:* The first approach is to level-wise generalize the input vector list. The search is shown by Algorithm 2 and an example search graph by Figure 5. To avoid multiple node visits, we only *generalize to the right*, i.e., a field will only be generalized if itself, none or a smaller field was generalized in the previous iteration. Thus, a minimum generalization index $i_{min}$ is passed among lattice nodes. Lines 1 and 2 of Algorithm 2 show the initialization of the bottom level nodes, i.e., bottom-level vectors, their frequency and an initial $i_{min} = 1$. In lines 3 to 5 we instantiate the three important collections of our algorithm: $N_p/N_c$ are the current level's *parent/child nodes* and $N$ are the result nodes.

Until the is nothing more to generalize (line 7) we generate all valid parents of each child (lines 8 to 15). Since a parent must be only one generalized field ahead of its child exactly one parent is generated for every valid field $i$ (lines 11 to 15). Before, the current $i_{min}$ is checked for being the top-level (line 8). If so, our search pruning is applied and $i_{min}$ is increased, i.e., only fields $i \geqslant i_{min}$ may generalized for all future parents. This happens once in the example of Figure 5 and is indicated by blue text. After all level parents are generated their frequencies are aggregated (line 17) and they are added to the result collection (line 18). Current parents are set to be next level's children (line 19).

The bottom-up search visits every lattice node only once and aggregates frequencies level-wise. However, frequency pruning (line 21) can first happen after all nodes are discoved and, thus, unnecessary paths in the lattice may be traversed. In particular these are the ones which may never lead to a frequent generalization (see red paths in Figure 5).

Fig. 5: Bottom-up search in the example lattice of Figure 4. Edge labels correspond to $i_{min}$ of Algorithm 2. Red lines indicate unnecessarily traversed paths which at $f_{min} = 3$.




**Algorithm 2** Bottom-up search

**Require:** $\mathbb{S} = (S_{T_i})^n$, $L = \langle \vec{\ell}_m \rangle \mid \vec{\ell} = (\ell_i)_{1 \leqslant i \leqslant n}, f_{min} \in \mathbb{N}$

1: $N_{bot} = L.\textbf{map}(\vec{\ell} \mapsto \langle \vec{\ell}, f : 1, i_{min} : 1 \rangle)$
2: $aggregateFrequencies(N_{bot})$
3: $N_c \Leftarrow N_{bot}$
4: $N_p \Leftarrow \langle\rangle$
5: $N \Leftarrow N_c$
6: **while** $N_c \neq \langle\rangle$ **do**
7: **for all** $\langle \vec{\ell}_c, f_c, i_{min} \rangle \in N_c$ **do**
8: **if** $\vec{\ell}_c.getField(i_{min}) = \ell^{top}_{T_{min}}$ **then**
9: $i_{min}$++
10: **end if**
11: **for** $i = i_{min}; i \leqslant n; i$++ **do**
12: $\vec{\ell}_p \Leftarrow \vec{\ell}_c$
13: $generalize(\vec{\ell}_p, i)$
14: $N_p.add(\langle \vec{\ell}_p, f_c, i \rangle)$
15: **end for**
16: **end for**
17: $aggregateFrequencies(N_p)$
18: $N.addAll(N_p)$
19: $N_c \Leftarrow N_p$
20: **end while**
21: $filter(N, f \geqslant f_{min})$
22: **return** $N$

*2) Top-down search:* The second approach is to start from the top-level vector and to specialize level-wise. Here, we can stop generating children as soon as the parent is not frequent anymore. To avoid multiple node visits, this time we *specialize to the right*, i.e., a field will only be specialized if itself, none or a smaller field was specialized in the previous iteration. The according search is shown by Algorithm 3 and an example search graph by Figure 6. Lines 1 and 2 correspond to those of the bottom-up search.

Until the is nothing more to specialize (line 6) we generate all valid children of each parent (lines 6 to 18). To generate children we need to find all input vectors matching a pattern (line 11). Base on these, we collect respective specializations for every valid field $i$ (lines 12-16). In contrast to bottom-up, the current $i_{min}$ is checked for being the bottom-level (line 8). If so, our search pruning is applied and $i_{min}$ is increased, i.e., only fields $i \geqslant i_{min}$ may specialized for all future children. This happens twice in the example of Figure 6 and is indicated by blue text.

As the major advantage over the bottom-up approach only frequent parents (line 20) are added to the result collection (line 21) and passed among iterations (line 22). Consequently, children of infrequent parents will never be processed. For example, in Figure 6 the red crossed nodes are filtered out and the gray paths will not be traversed. However, the there is the additional cost of the pattern matching (line 11). We use an index between $idx : N_p \rightarrow \mathcal{P}(N_{bot})$ to avoid enumerating $N_{bot}$ fully for each call.

## III. Result Ranking

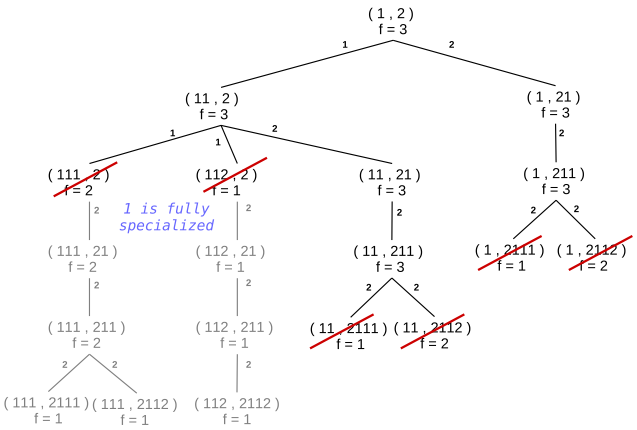
Since GM-FSM potentially returns a large number of results, they are ranked according to their statistical significance. The significance is established through the computation of a p-value, which measures the degree of over-representation of the pattern in the union graph of a collection of graph transactions with respect to a null model. The p-value is calculated by using a fast analytical model, starting from the mean number of occurrences of the pattern with respect to the Expected Degree Distribution (EDD) model [5] (also knonwn as Chung-Lu model), which is chosen as reference null model.

In what follows we briefly describe the EDD model for multigraphs and show how to compute the expected number of occurrences of a pattern according to the EDD model.

Given a graph $G = \langle V, E, s, t, \delta, \lambda, \tau \rangle$, we call **multiplicity** of an edge $e \in E$, and we denote it as $\Theta(e)$, the number of parallel edges between $s(e)$ and $t(e)$. The multiplicity $\mathcal{M}$ of a graph is defined as the maximum multiplicity of its edges. The **projection** of $G$ with respect to $\theta$ is the subgraph $G_\theta$ of $G$ composed only by all the edges of $G$ with multiplicity $\theta$.

We next define $\mathcal{M} \times |\Sigma_V|$ random variables, one for each edge multiplicity $\theta$ conditioned to each vertex label $\ell \in \Sigma_V$. $Deg_\theta|\ell$ is the degree distribution of the projected graph $G_\theta$ conditioned by $\ell$, and thus considering only the degrees of vertices with label $\ell$. Specifically, $\mathbb{P}((Deg_\theta \mid \ell) = d)$ is the probability that given a vertex with label $\ell$, it has degree $d$ in the projection $G_\theta$. The degree of a vertex $v$ given the vertex label $\ell$ in $G_\theta$, $D_{v,\theta}$, is sampled according to the $Deg_\theta|\ell$ distribution.

Fig. 6: Top-down search in the example lattice of Figure 4. Edge labels correspond to $i_{min}$ of Algorithm 3. Gray lines indicate pruned paths at $f_{min} = 3$.




**Algorithm 3** Top-down search

**Require:** $\mathbb{S} = (S_{T_i})^n$, $L = \langle \vec{\ell}_m \rangle \mid \vec{\ell} = (\ell_i)_{1 \leqslant i \leqslant n}, f_{min} \in \mathbb{N}$

1: $N_{bot} = L.\textbf{map}(\vec{\ell} \mapsto \langle \vec{\ell}, f : 1, i_{min} : 1 \rangle)$
2: $aggregateFrequencies(N_{bot})$
3: $N_c \Leftarrow \langle\rangle$
4: $N_p \Leftarrow \langle \langle \vec{\ell} : (\ell^{top}_{T_1}, \ell^{top}_{T_2}), f : |L|, i_{min} : 1 \rangle \rangle$
5: $N \Leftarrow N_p$
6: **while** $N_p \neq \langle\rangle$ **do**
7: **for all** $\langle \vec{\ell}_p, f_p, i_{min} \rangle \in N_p$ **do**
8: **if** $\vec{\ell}_p.getField(i_{min}) = \ell^{bot}_{T_{min}}$ **then**
9: $i_{min}$++
10: **end if**
11: **for** $\vec{\ell}_{bot}, f_{bot} \in match(N_{bot}, \vec{\ell}_p)$ **do**
12: **for** $i = i_{min}; i \leqslant n; i$++ **do**
13: $\vec{\ell}_c \Leftarrow \vec{\ell}_p$
14: $specialize(\vec{\ell}_c, i, \vec{\ell}_{bot})$
15: $N_c.add(\langle \vec{\ell}_c, f_{bot}, i \rangle)$
16: **end for**
17: **end for**
18: **end for**
19: $aggregateFrequencies(N_c)$
20: $filter(N_c, f \geqslant f_{min})$
21: $N.addAll(N_c)$
22: $N_p \Leftarrow N_c$
23: **end while**
24: **return** $N$

The probability of adding an edge with multiplicity $\theta$ between two vertices $u$ and $v$ according to the EDD model is defined as:

$$\mathbb{P}(u, v, \theta | D_{u,\theta}, D_{v,\theta}) = \min(1, \gamma_\theta \times D_{u,\theta} \times D_{v,\theta}) \qquad (12)$$

where $\gamma_\theta = 1/\left[(|V| - 1) \times \mathbb{E}\left[Deg_\theta\right]\right]$. $Deg_\theta$ is the degree distribution of the projected graph $G_\theta$, considering all vertices

and is given by $\sum_{\ell} Deg_{\theta,\ell} \cdot \mathbb{P}(\delta_P(v) = \ell)$, where $\mathbb{P}(\delta_P(v) = \ell)$ is the probability of having a vertex with label $\ell$.

Given a pattern $P = \langle V_p, E_p, s_p, t_p, \delta_p, \lambda_p, \tau_p \rangle$ with $|V_p| = k$, the occurrence probability of $P$ in $G$, given a label assignment $\delta_p$ to the vertices of $P$ and a label assignment $\lambda_p$ to the edges of P, is obtained by summing across all the possible degree assignments to each vertex for that motif.

$$\mu(P|\delta_p, \lambda_p) = \prod_{e \in E_p} \gamma_{\Theta(e)} \times \prod_{u=1}^{k} \prod_{\theta=1}^{\mathcal{M}} \mathbb{E}\left[Deg_{\theta}^{p_{s(\star)=u,\theta}} | \delta_p(u)\right] \tag{13}$$

where $p_{s(\star)=u,\theta}$ is the number of edges with multiplicity $\theta$ and $u$ as source vertex, while $\mathbb{E}\left[Deg_{\theta}^{p_{s(\star)=u,\theta}} | \delta_p(u)\right]$ is the moment of degree $p_{s(\star)=u,\theta}$ of $Deg_{\theta}|\delta_p(u)$ distribution. In general, the moment of degree $d$ of a distribution $X$ which can assume values in a set $R(X)$ is defined as $\sum_{x \in R(X)}[x \times \mathbb{P}(X = x)]$.

The probability of the motif is then defined as:

$$\mu(P) = \mu(P|\delta_p, \lambda_p) \times \sigma(\delta_p) \times \eta(\lambda_p) \tag{14}$$

where:

$$\sigma(\delta_p) = \prod_{u=1}^{k} \mathbb{P}(\delta_p(u)) \tag{15}$$

$$\eta(\lambda_p) = \prod_{e \in E_p} \mathbb{P}(\lambda_p(e)|\delta_p(s(e)), \delta_p(t(e)) \tag{16}$$

$\mathbb{P}(\delta_p(u))$ is the probability of nodes with label $\delta_p(u)$ in $G$, while $\mathbb{P}(\lambda_p(e)|\delta_p(s(e)), \delta_p(t(e))$ is the probability of observing edges in $G$ with label $\lambda_p(e)$ having source and target vertices labeled $\delta_p(s(e))$ and $\delta_p(t(e))$, respectively.

To compute the mean of the number $N(P)$ of occurrences of P, we have to consider all possible locations of the pattern in $G$. if $P$ has $k$ vertices, a location che be defined as a set of $k$ vertex identifiers in $G$, representing the vertices of $G$ that match the vertices of $P$. The number of all possible locations is then obtained as $\binom{|V|}{k}$.

Furtheremore, a pattern $P$ in a specific location can occur in different configurations, where each configuration correspond to a permutation of vertex identifiers. We call Non-Redundant Permutations (NPRs) of $P$, a set NPRS($P$), consisting of all the distinct subgraphs resulting from all possible configurations of the current set of vertices in the location. Two subgraphs are considered distinct if they differ in at least one of the following quantities: i) adjacency matrix, ii) ordered list of vertex labels according to vertex identifiers, iii) ordered list of edge labels according to edge identifiers. We therefore define $\pi(P) = |NRPS(P)|$.

An important property of the EDD model is exchangeabilty, which means that the probability of observing $P$ does not depend on its specific location. Thanks to such a property, we can compute the expectation of $N(P)$ as:

$$\mathbb{E}\left[N(P)\right] = \binom{|V|}{k} \times \pi(P) \times \mu(P) \tag{17}$$

The variance of the number of occurrences of P is given by $\mathbb{V}\left[N(P)\right] = \mathbb{E}\left[N^2(P)\right] - \mathbb{E}\left[N(P)^2\right]$.

To compute $\mathbb{E}[N^2(P)]$, we have to take into account that two occurrences of a pattern may overlap. Two occurrences overlap if they share at least one vertex. This causes the generation of super-pattern, i.e. a subgraph $P$ composed by two NRPs of overlapping occurrences which have the same node and vertex labels in the overlapping region. Given two NRPs of $P$, say $P$' and $P$", we define the overlapping operation with $s$ common nodes as $P^{'}\Omega_s P^{''}$. Therefore the computation of expectation of squared count of $P$ involves the computation of such super-patterns.

The expectation of the squared count of $P$ is given by the contribution of two terms, one is related to pairs of disjoint occurrences and one is related to pairs of overlapping occurrences (with different amounts of overlap). In both cases we have to consider: (i) all possible locations of the two occurrences of pattern $P$ in the graph and (ii) all possible NRPs of $P$.

The expectation of the squared count is then given by the following equation:

$$\mathbb{E}\left[N^2(P)\right] = \binom{N}{N-2k, k, k} \pi^2(P)\mu^2(P) + \sum_{s=1}^{k} \binom{N}{k-s, s, k-s, N-2k+s} \sum_{P', P'' \in NRPS(P)} \mu(P^{'}\Omega_s P^{''}) \tag{18}$$

where $\binom{N}{N-2k,k,k}$ is the number of all possible combinations of locations of two non-redundant permutations of $m$ with no overlap and $\binom{N}{k-s,s,k-s,N-2k+s}$ is the number of all possible combinations of locations of two non-redundant permutations of $m$ with overlap $s$.

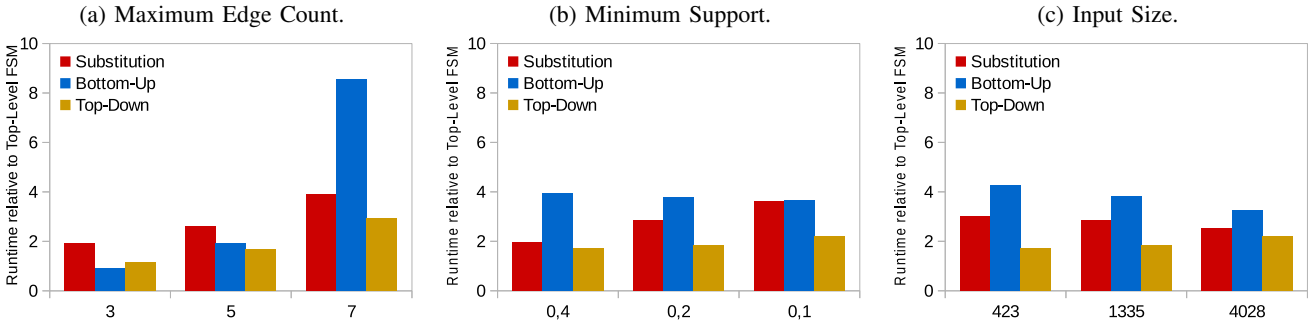
Finally, to establish whether pattern $P$ is over-represented in $G$, one needs to calculate the probability (i.e. p-value) $\mathbb{P}(N(P) \geqslant F(P))$, where $F(P)$ is the frequency of $P$ in $G$ and $N(P)$ is a random variable representing the number of occurrences of P in a random graph generated according to the chosen reference model.

Following [27], we model the random variable $N(P)$ using the Pólya-Aeppli (denoted by PA) distribution [17]. Mean and variance of the number of occurrences of $P$ are used to compute the two parameters of the PA distributiojn: $a = \frac{\mathbb{V}[N(P)] - \mathbb{E}[N(P)]}{\mathbb{V}[N(P)] + \mathbb{E}[N(P)]}$ and $\lambda = (1-a)\mathbb{E}\left[N(P)\right]$.

Dealing with very large patterns may be still problematic even with the proposed analytical model. The main drawback is related to the computation of the variance which is quite time consuming due to the generation of super-patterns and the computation of their occurrence probabilities.

To overcome such a limitation we can observe that the variance is needed to estimate the parameter $a$ in the PA distribution. Indeed, by varying the variance we can estimate several p-values. In such a distribution of p-values we extract the maximum and return it as the p-value of the pattern. Such an approximation allows us to establish the significance of very large patterns. The accuracy and performance of p-value approximation method have been tested and full results will follow in a dedicated publication.

Fig. 7: GM-FSM Evaluation Results.




## IV. Experimental Evaluation

We implemented a prototype in the Java programming language. As the core of both methods we reimplemented gSpan [36] including the extension for directed multigraphs described in [24]. The current version is sequential but adding thread parallelization is straight forward. The source code can be found online[3] under GPL licence. All experiments were run on a machine equipped with an Intel i7-4770 CPU, 16GB RAM, SSD and running Ubuntu 14.04.

### A. Dataset

Since we are targeting business intelligence we choose the XXXXX data generator [25] to create synthetic data with real-world inspired properties. XXXXX is based on business process simulation. Each simulation results into a graph of transactional (e.g. quotation, sales orders and invoices) and master data (e.g. employees, products and customers). Multiple graphs may contain the same master data objects, e.g., the same employee sending a quotation. Each process execution can be evaluated by financial result. There are implanted correlations, e.g., if certain employees are involved or master data objects interact the probability of financial loss is increased. Thus, we did not use the full data set but only a subset of lossy graphs for our evaluation. Thus, we know that frequent patterns on low taxonomy levels are contained.

We modified the configuration such that all graphs have 19 vertices and 22 edges. Additionally all input graphs are fully isomorphic to each other according to Definition 5. Thus, the data set is very challenging for frequent subgraph mining despite despite the small number of only 1335 graphs. However, in this way we have achieved our goal to generate a dataset with many results on low taxonomy levels as required to effectively benchmark the generalization aspect of our methods. For example, at $sup_{min} = 0.1$ and $k_{max} = 8$ we can extract nearly 1M frequent patterns.

### B. MG-FSM method comparison

TODO

[3]https://github.com/anonymized-for-submission

### C. P-value calculation

To evaluate the scalability of the P-Value calculation we extracted frequent patterns and choose 1000 random patterns for the edge counts $4 \leqslant k \leqslant 10$. Figure 8 shows the average computation time in milliseconds for each $k$. We see that the processing time is increasing super-linear over pattern size. However, we consider patterns of more than 10 edges either irrelevant for analytical scenarios. Thus, we consider a computation time of less than 10 milliseconds per p-value in relation to the gained accuracy of significance as a notable result.

## V. Related Work

GM-FSM uses techniques from different fields of research. Thus, we discuss related work about frequent subgraph, generalized and multi-dimensional pattern mining as well as significance of graph patterns.

### A. Frequent Subgraph Mining

The origin of frequent pattern mining [1] is its role as a primitive operation at the extraction of association rules from itemsets [2]. Frequent subgraph mining (FSM) is a variant of frequent pattern mining where patterns are represented by graphs that are isomorphic to subgraphs of either a single large graph (*single-graph setting*) or graphs in a collection *(graph-transaction setting*) [16]. Since GM-FSM is based on the graph-transaction setting, we omit a further discussion of the single-graph setting [7], [34]. The first transactional FSM algorithms, e.g., AGM [14] and FSG [18], followed an *a priori* approach. These algorithms first generate candidate patterns and count their frequency by graph pattern matching. Their disadvantages are the expensive isomorphism resolutions during candidate generation and support counting as well as the generation of many candidates which might not even appear. Thus, the next generation of *pattern-growth* based FSM algorithms appeared and outperformed the a priori ones. Popular representatives of this category are MOFA [4], gSpan [36], FFSM [12] and Gaston [23]. These algorithms systematically derive patterns form actually occurring subgraphs and represent them by canonical labels. Existing algorithms to transactional FSM on shared nothing clusters [11], [19], [21], [3] follow pattern-growth approaches.

TABLE I: GM-FSM Benchmark Measurements.

| Parameter | | Result Size $\|\mathcal{F}\|$ | | Runtime in seconds | | | |
|---|---|---|---|---|---|---|---|
| $ms$ | $km$ | Top | All | Top | PS | BU | TD |
| 0.4 | 3 | 213 | 420 | 1 | 3 | 2 | 2 |
| | 4 | 646 | 1465 | 2 | 11 | 6 | 6 |
| | 5 | 1827 | 4692 | 7 | 36 | 29 | 21 |
| | 6 | 4690 | 13433 | 18 | 99 | 136 | 61 |
| | 7 | 10727 | 33793 | 45 | 244 | 654 | 165 |
| | 8 | 21669 | 74112 | 96 | 517 | 3124 | 358 |
| 0.2 | 3 | 233 | 1045 | 1 | 4 | 2 | 2 |
| | 4 | 737 | 4067 | 2 | 15 | 6 | 7 |
| | 5 | 2181 | 14419 | 7 | 52 | 29 | 25 |
| | 6 | 5870 | 45132 | 18 | 158 | 137 | 76 |
| | 7 | 14101 | 122539 | 47 | 416 | 673 | 214 |
| | 8 | 29966 | 286702 | 100 | 926 | 3209 | 506 |
| 0.1 | 3 | 313 | 2284 | 1 | 5 | 2 | 2 |
| | 4 | 1099 | 9662 | 2 | 19 | 7 | 8 |
| | 5 | 3580 | 37193 | 7 | 69 | 30 | 30 |
| | 6 | 10496 | 125765 | 20 | 217 | 143 | 94 |
| | 7 | 27203 | 365926 | 50 | 609 | 698 | 283 |
| | 8 | 61837 | 909163 | 111 | 1422 | 3437 | 695 |

km / ms : maximum edge count / minium support
Top / All : top-level only / all-levels
PS : path-substitution
TD : pattern-decomposition with top-down GFVM
BU : pattern-decomposition with bottom-up GFVM

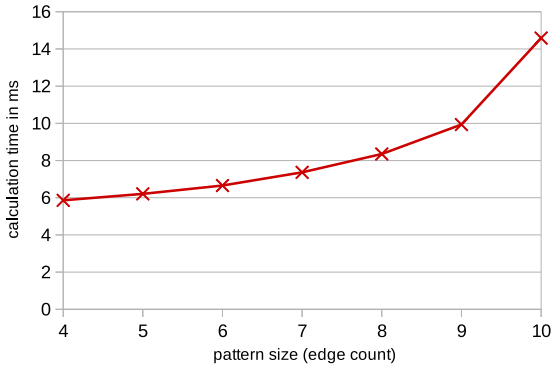
### B. Generalized and Multi-dimensional Pattern Mining

In [32] first taxonomies were used to find frequent generalizations of potentially infrequent patterns. This concept was applied to sequential patterns in [33]. The first approach to mine itemsets at multiple levels was proposed in [9]: However, this approach is only able to extract patterns where all values are at the same level of the same hierarchy. The approach of [38] provides support for generalized and multi-level frequent pattern mining and uses a graph model at the extraction of association rules. In more recent work, algorithms to cross-level frequent itemset mining were proposed in [6] and [15]. All of the work discussed so far assumed that all items are of the same type, i.e., belong to a single dimension. This is also the case for the only other approach to generalized frequent subgraph mining [13] based on an a priori method where a weighted support is used to prune over-generalizations during the mining process. Approaches to multi-dimensional pattern mining were proposed in the context of sequential patterns [28], [39]. However, dimensions are not part of patterns but attached to them. In contrast, the approach of [29] is supporting sequences of multi-dimensional items as well as mining them at multiple levels.

### C. Significance of Graph Patterns

In certain scenarios patterns are not interesting just because of their bare frequency. Therefore, research attention has been spent to identify statistically significant graph patterns. In [10] and [30] authors depict fast methods to convert graphs into a feature space and evaluate the significance of the graph patterns in such a space. In [35] a general mining framework, is capable to exploit the correlation between structural similarity and significance similarity. Through this strategy the mining of the most significant graph patterns measured by different kinds of non-monotonic objective functions is achieved. In [8] authors introduced an indexing method to extract Top-K subgraphs with surprising and rare associations.

Fig. 8: Time for p-value estimation by pattern size $k$.




Focusing on the methods that rank patterns through statistics, the aim is to mine subgraphs having a p-value below a user specified threshold (usually the default critical value for the p-value is 0.05). The common approach to deal with this problem relies on simulation which usually generate databases of random graphs (namely the null model) sharing some characteristics of the input one and look for the same patterns. The p-value is then estimated as the number of times the pattern appears more frequently than in the input database.

Over the last decades, researchers have worked on replacing simulation by analytical methods. Picard et al. [27] proposed a model to exactly compute the mean and variance of the count of a given graph pattern under any exchangeable random graph model. The authors make use of the Pólya-Aeppli distribution (also known as the Poisson Geometric distribution which is a special case of the Poisson-Compound distribution) [17]. Schbath et al [31] extends the model to deal with topology-free graph pattens which are defined as any connected topology of $k$ vertices having a given multiset of labels. In [22] authors further extended the analytical model to establish the significance of labeled (induced or not induced) patterns on directed and undirected graphs, under the Chung-Lu random model [5], and in which labels are either independent or dependent on the degrees of vertices.

## VI. Conclusions & Future Work

We presented the first study about multi-dimensional frequent subgraph mining. To represent different dimensions, vertex labels can be assigned to multiple taxonomies. Besides our conceptual contribution, we proposed two efficient methods to extract frequent generalizations without determining the frequency of all bottom-level patterns. Among the proposed algorithms, the decomposition into frequent subgraph and top-down frequent vector mining has shown the best performance. Additionally, we presented a fast method to calculate p-values for the resulting graph patterns which are an accurate and expressive measure to rank them by significance.

In our future work, we will investigate the integration of p-values directly in the mining process for significant pattern mining. Further on we plan to create parallel versions of GM-FSM to mine larger data sets in reasonable time. Here

## REFERENCES


[1] C. C. Aggarwal and J. Han. *Frequent pattern mining*. Springer, 2014.

[2] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. In *Acm sigmod record*, volume 22, pages 207–216. ACM, 1993.

[3] Anomymized For Submission. A horizontally scalable frequent subgraph miner.

[4] C. Borgelt and M. R. Berthold. Mining molecular fragments: Finding relevant substructures of molecules. In *IEEE International Conference on Data Mining (ICDM)*, pages 51–58, 2002.

[5] F. Chung and L. Lu. The average distances in random graphs with given expected degrees. *Proceedings of the National Academy of Sciences*, 99(25):15879–15882, 2002.

[6] T. Eavis and X. Zheng. Multi-level frequent pattern mining. In *International Conference on Database Systems for Advanced Applications*, pages 369–383. Springer, 2009.

[7] M. Elseidy, E. Abdelhamid, S. Skiadopoulos, and P. Kalnis. Grami: Frequent subgraph and pattern mining in a single large graph. *Proceedings of the VLDB Endowment*, 7(7):517–528, 2014.

[8] M. Gupta, J. Gao, X. Yan, H. Cam, and J. Han. Top-k interesting subgraph discovery in information networks. In *Data Engineering (ICDE), 2014 IEEE 30th International Conference on*, pages 820–831. IEEE, 2014.

[9] J. Han and Y. Fu. Mining multiple-level association rules in large databases. *IEEE Transactions on knowledge and data engineering*, 11(5):798–805, 1999.

[10] H. He and A. K. Singh. Graphrank: Statistical modeling and mining of significant subgraphs in the feature space. In *Data Mining, 2006. ICDM'06. Sixth International Conference on*, pages 885–890. IEEE, 2006.

[11] S. Hill, B. Srichandan, and R. Sunderraman. An iterative mapreduce approach to frequent subgraph mining in biological datasets. In *Proc. ACM Conf. on Bioinformatics, Computational Biology and Biomedicine*, pages 661–666, 2012.

[12] J. Huan, W. Wang, and J. Prins. Efficient mining of frequent subgraphs in the presence of isomorphism. In *IEEE Int. Conf. on Data Mining (ICDM)*, pages 549–552, 2003.

[13] A. Inokuchi. Mining generalized substructures from a set of labeled graphs. In *Data Mining, 2004. ICDM'04. Fourth IEEE International Conference on*, pages 415–418. IEEE, 2004.

[14] A. Inokuchi, T. Washio, and H. Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 13–23. Springer, 2000.

[15] B. Jayanthi, K. Duraiswamy, et al. A novel algorithm for cross level frequent pattern mining in multidatasets. *International Journal of Computer Applications (0975–8887) Vol*, 37, 2012.

[16] C. Jiang, F. Coenen, and M. Zito. A survey of frequent subgraph mining algorithms. *The Knowledge Eng. Review*, 28(01):75–105, 2013.

[17] N. L. Johnson, S. Kotz, and A. W. Kemp. Univariate discrete distributions. *Biometrical Journal*, 36(2):164–164, 1994.

[18] M. Kuramochi and G. Karypis. Frequent subgraph discovery. In *IEEE Int. Conf. on Data Mining (ICDM)*, pages 313–320, 2001.

[19] W. Lin, X. Xiao, and G. Ghinita. Large-scale frequent subgraph mining in mapreduce. In *International Conference on Data Engineering (ICDE)*, pages 844–855. IEEE, 2014.

[20] J. Liu and X. Zhang. Dynamic labeling scheme for xml updates. *Know.-Based Syst.*, 106(C):135–149, Aug. 2016.

[21] W. Lu, G. Chen, A. Tung, and F. Zhao. Efficiently extracting frequent subgraphs using mapreduce. In *IEEE Int. Conf. on Big Data*, pages 639–647, 2013.

[22] G. Micale, R. Giugno, A. Ferro, M. Mongioví, D. Shasha, and A. Pulvirenti. Fast analytical methods for finding signicant colored graph motifs. *To appear on Data Mining and Knowledge Discovery*, 2017.

[23] S. Nijssen and J. N. Kok. The gaston tool for frequent subgraph mining. *Electronic Notes in Theoretical Computer Science*, 127(1):77–87, 2005.

[24] A. Petermann, M. Junghanns, S. Kemper, K. Gómez, N. Teichmann, and E. Rahm. Graph mining for complex data analytics. In *IEEE Int. Conf. on Data Mining Workshops (ICDMW)*, pages 1316–1319, 2016.

[25] A. Petermann, M. Junghanns, R. Müller, and E. Rahm. Foodbroker-generating synthetic datasets for graph-based business analytics. In *Workshop on Big Data Benchmarks*, pages 145–155. Springer, 2014.

[26] A. Petermann et al. Graph-based Data Integration and Business Intelligence with BIIIG. *PVLDB*, 7(13), 2014.

[27] F. Picard, J. J. Daudin, M. Koskas, S. Schbath, and S. Robin. Assessing the exceptionality of network motifs. *Journal of Computational Biology*, 15(1):1–20, 2008.

[28] H. Pinto, J. Han, J. Pei, K. Wang, Q. Chen, and U. Dayal. Multi-dimensional sequential pattern mining. In *Proceedings of the tenth international conference on Information and knowledge management*, pages 81–88. ACM, 2001.

[29] M. Plantevit, A. Laurent, D. Laurent, M. Teisseire, and Y. W. Choong. Mining multidimensional and multilevel sequential patterns. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 4(1):4, 2010.

[30] S. Ranu and A. K. Singh. Graphsig: A scalable approach to mining significant subgraphs in large graph databases. In *IEEE Int. Conf. on Data Engineering (ICDE)*, pages 844–855. IEEE, 2009.

[31] S. Schbath, V. Lacroix, and M.-F. Sagot. Assessing the exceptionality of coloured motifs in networks. *EURASIP J. Bioinformatics Syst. Biol.*, 2009:3:1–3:9, Jan. 2009.

[32] R. Srikant and R. Agrawal. Mining generalized association rules. pages 407–419, 1995.

[33] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. *Advances in Database TechnologyEDBT'96*, pages 1–17, 1996.

[34] C. H. Teixeira et al. Arabesque: a system for distributed graph mining. In *Proc. of the 25th Symposium on Operating Systems Principles*, pages 425–440. ACM, 2015.

[35] X. Yan, H. Cheng, J. Han, and P. S. Yu. Mining significant graph patterns by leap search. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 433–444. ACM, 2008.

[36] X. Yan and J. Han. gspan: Graph-based substructure pattern mining. In *IEEE International Conference on Data Mining (ICDM)*, pages 721–724, 2002.

[37] X. Yan and J. Han. gspan: Graph-based substructure pattern mining. In *Technical Report UIUCDCS-R-2002.2296*, 2002.

[38] S.-J. Yen and A. L. P. Chen. A graph-based approach for discovering various types of association rules. *IEEE Transactions on Knowledge and data Engineering*, 13(5):839–845, 2001.

[39] C.-C. Yu and Y.-L. Chen. Mining sequential patterns from multidimensional sequence data. *IEEE Transactions on Knowledge and Data Engineering*, 17(1):136–140, 2005.