Mining Frequent Most Informative Subgraphs

Frédéric Pennerath[1] and Amedeo Napoli[2]

[1] Supélec, Campus de Metz, 2 rue Edouard Belin 57070 Metz, France
frederic.pennerath@supelec.fr
[2] Orpailleur team, LORIA, BP 239, 54506 Vandoeuvre-lès-Nancy Cedex, France
amedeo.napoli@loria.fr

Abstract. The main practical problem encountered with frequent subgraph search methods is the tens of thousands of returned graph patterns that make their visual analysis impossible. In order to face this problem, are introduced a very restricted family of relevant graph patterns called the *most informative patterns* along with an algorithm to mine them and associated experimental results.

In *graph-based data mining problems*, mined patterns are connected labelled graphs isomorphically distinct. Several algorithms have been proposed [1–4] to mine frequent graph patterns in graph databases by analogy with the *frequent itemset search problem*. Given a graph database, the *frequency* of a graph pattern is the number of graphs in the database containing at least one subgraph isomorphic to the pattern. The *frequent subgraph search problem* consists in determining the set of frequent patterns whose frequency is higher than a minimum threshold along with their frequency. The main practical problem encountered with frequent patterns in general and frequent subgraphs in particular is their huge number, resulting in the impossibility for the expert to analyse visually every returned pattern. Increasing the minimal frequency threshold and mining only the few most frequent graphs does not help since most frequent graphs, following the example of the empty graph, are generally the least interesting as well. A better solution consists in extracting a restricted but informative subset of frequent patterns, such as the subset of frequent *closed patterns* [5]. However even dense and relatively small graph data sets still contain several thousands of frequent closed patterns [5]. Information theory better defines what informative means: informative patterns are simultaneously discriminating (i.e. subsuming a small subset of all possible object descriptions) and representative (i.e. having a high frequency). Since the frequency decreases when the pattern description grows, the extraction of most informative patterns is necessarily an optimization problem that selects patterns providing the best optimum between their frequency and the information contained in their description. In a very general setting, the quality of this compromise can be assessed by an arbitrary *scoring function* associating a score to every pattern. Higher is the score, better is the compromise. Given a graph database and a scoring function, the proposed method implemented in the `Forage` software platform searches within the set of frequent graph patterns, the most informative patterns defined as the local maxima of the scoring function in the patterns order.

The Most Informative Pattern Extraction Problem

Frequent subgraph mining considers a set $\mathcal{G}$ of **isomorphically distinct** graph patterns whose vertices and edges are labelled in a set $\mathcal{L}$. Because two isomorphic patterns of $\mathcal{G}$ are necessarily equal, the isomorphic subgraph relation $\leq_{\mathcal{G}}$ is an ordering relation over $\mathcal{G}$. Given a database $\mathcal{D}$ of graphs labelled in $\mathcal{L}$, the *frequency* $\text{freq}(p)$ of a pattern $p \in \mathcal{G}$ is the number of elements of $\mathcal{D}$ which contain at least one subgraph isomorphic to p. A pattern is frequent if its frequency is greater or equal to a given threshold. A *most informative pattern* is then defined relatively to a *scoring order* and an *informative scoring function*:

Definition. *Given a database $\mathcal{D}$ and a total or partial ordered set $(\mathbb{S}, \leq_{\mathbb{S}})$ called the* scoring order*, a* scoring function *is a function $s : \mathcal{G} \times \mathbb{N} \to \mathbb{S}$ mapping the pair (p, f) of a pattern p and its frequency f in $\mathcal{D}$ to the* score *$s(p, f)$ of p relatively to $\mathcal{D}$. A scoring function s is* informative *if every function $s_f : p \mapsto s(p, f)$ for any $f \in \mathbb{N}$ (resp. every function $s_p : f \mapsto s(p, f)$ for any $p \in \mathcal{G}$) is an increasing function of p (resp. of f).*

The *frequent most informative pattern extraction problem* then consists in finding every frequent pattern whose the score is a local maximum in $(\mathcal{G}, \leq_{\mathcal{G}})$.

Definition. *A pattern p is a* most informative pattern *relatively to a scoring order $(\mathbb{S}, \leq_{\mathbb{S}})$ and an informative scoring function s if and only if no immediate predecessor[3] or immediate successor p' of p in $(\mathcal{G}, \leq_{\mathcal{G}})$ verifies $s(p', \text{freq}(p')) >_{\mathbb{S}} s(p, \text{freq}(p))$.*

Closed patterns appear as the most informative patterns relatively to the scoring order $(\mathcal{G}, \leq_{\mathcal{G}}) \times (\mathbb{N}, \leq)$[4] and the informative scoring function $s_c : (p, f) \mapsto (p, f)$. The closed pattern extraction has actually the most conservative filtering scoring function among all possible ones. A more selective scoring function defines the said *most synthetic patterns*.

Definition. The most synthetic patterns *are the most informative patterns relatively to the scoring order of positive real numbers $(\mathbb{R}^+, \leq)$ and the informative scoring function $s_p : (p, f) \mapsto |p| \cdot f$, where $|p|$ denotes the size of pattern p (i.e. for a graph pattern, the size is the number of vertices and edges).*

The function s_p is a rough compression gain estimator that approximately estimates the space saved when every occurrence of a pattern p in $\mathcal{D}$ is replaced by a single vertex identified by a special label. This information criterion follows the minimum description length principle found in `Subdue` [6]. However `Subdue` is an incomplete graph beam search algorithm that uses the information criterion to converge towards some interesting patterns whereas `Forage` is a complete graph mining algorithm returning all frequent most informative patterns and only them. Test results have shown frequent most synthetic patterns are typically few tens where frequent closed patterns are several thousands.

[3] p' is an immediate predecessor (resp. immediate successor) of p if $p' < p$ and $p' \leq p'' < p \Rightarrow p'' = p'$ (resp. $p' > p$ and $p' \geq p'' > p \Rightarrow p'' = p'$).

[4] Given two orders $(E_1, \leq_1)$ and $(E_2, \leq_2)$, the *product order* $(E_1 \times E_2, \leq_{12})$ is defined by: $(x_1, x_2) \leq_{12} (y_1, y_2)$ if and only if $x_1 \leq_1 y_1$ and $x_2 \leq_2 y_2$.

A Most Informative Pattern Extraction Method

The `Forage` algorithm has been developed to extract from a set of labelled graphs the frequent most informative graph patterns relatively to a parameterizable scoring function. It can therefore be used to extract frequent closed subgraphs and frequent most synthetic subgraphs. Because every frequent pattern must have its score compared with the ones of all its immediate predecessors and successors, every immediate successor p' of the currently mined pattern p must be generated when p is frequent. The canonical graph c of p' is then computed using an algorithm similar to Nauty [7] and is used as a key to retrieve from a trie the entry of pattern p'. Each entry stores the pattern score and a boolean flag. This flag is initialized to true if and only if p' is frequent and later gets false if any immediate predecessor or successor of p' has a greater score than p'. If an entry for p' already exists, a pattern isomorphic to p' was previously generated and the algorithm thus backtracks. Otherwise the frequency of p' is computed using embedding list [4], the score of p' is estimated and is used to initialize a new entry added to the trie for the key c. If p' is found frequent, the algorithm is recursively called for all successors of p' so that all frequent patterns are processed in a depth first search order. In every case, the boolean flags of p and p' are updated by comparing their respective scores. At the end of the recursion, patterns with a true flag are extracted from the trie and returned as the frequent most informative patterns. Whereas algorithms such as `gSpan` [3] or `Gaston` [4] use efficient optimizations to reduce redondant generations of distinct but isomorphic patterns, `Forage` has to generate the whole order diagram (or Hasse diagram) to filter the most informative patterns without any possible form of optimization. This substantially increases the algorithm complexity and the required processing time as the price to pay for an efficient pattern selection.

The algorithm has been tested on various data sets containing few hundreds to few thousands molecular graphs or reaction graphs derived from chemical reaction equations. One of these tests consisted for instance in mining frequent reaction patterns in a set of 200 reaction graphs. This set is actually composed of two subsets of 100 similar graphs representing two distinct well known families of reactions (i.e acetoacetic ester and Sonogashira synthesis methods). `Gaston` is used as a reference to mine the frequent reaction patterns whereas `Forage` mines both sets of frequent closed patterns and frequent most synthetic patterns. As expected, processing times on Fig. 1 (b) show `Forage` is about 100 hundreds times slower than `Gaston` in searching frequent patterns. However the quality of returned most synthetic patterns makes up for `Forage` 's slowness: figure 1 (a) shows the number of frequent most synthetic patterns slowly grows ranging from 5 to 21 whereas at the same time the numbers of frequent patterns and frequent closed patterns exponentially grow, respectively from 2000 to 20000 and from 70 to 1000. A qualitative analysis shows that among the 21 found frequent most synthetic patterns, 7 of them are characteristic of the two synthesis methods, 3 others represent the three resonance forms of aromatic cycles and only the 11 remaining ones are apparently uninteresting reaction patterns. These figures are to compare with the hundreds of irrelevant frequent closed patterns.

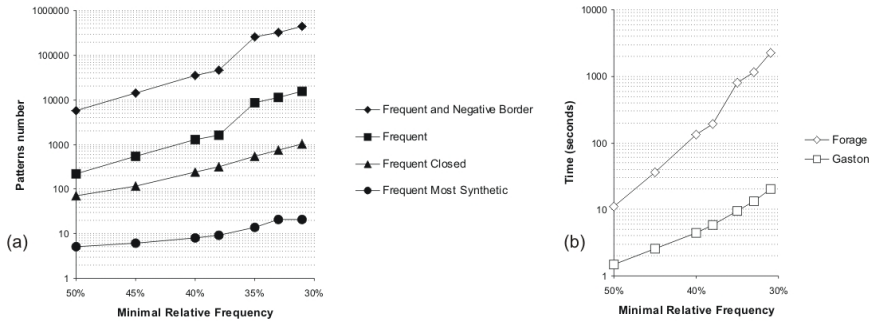

Fig. 1. Patterns distribution (a) and processing time (b)

In conclusion, the first experimental tests have emphasized the quality of the frequent most synthetic patterns as patterns being simultaneously very representative of the database content and very rare: they are typically only few tens where frequent closed patterns are thousands. In order to further validate the interest of such patterns and to better understand the computation limits, tests must now be generalized to non homogeneous graph databases (i.e. such as general chemical reaction databases with lower density) and the influence of various scoring functions on the quality of patterns must be evaluated.

References

1. Inokuchi, A., Washio, T., Motoda, H.: An apriori-based algorithm for mining frequent substructures from graph data. In: PKDD '00: Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery, London, UK, Springer-Verlag (2000) 13–23
2. Kuramochi, M., Karypis, G.: Frequent subgraph discovery. In: ICDM '01: Proceedings of the 2001 IEEE International Conference on Data Mining. (2001) 313–320
3. Yan, X., Han, J.: gspan: Graph-based substructure pattern mining. In: ICDM '02: Proceedings of the 2002 IEEE International Conference on Data Mining, Washington, DC, USA, IEEE Computer Society (2002) 721
4. Nijssen, S., Kok, J.N.: A quickstart in frequent structure mining can make a difference. In: KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, New York, NY, USA, ACM Press (2004) 647–652
5. Yan, X., Han, J.: Closegraph: mining closed frequent graph patterns. In: KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, New York, NY, USA, ACM Press (2003) 286–295
6. Cook, D.J., Holder, L.B.: Substructure discovery using minimum description length and background knowledge. Journal of Artificial Intelligence Research **1** (1994) 231–255
7. McKay, B.D.: Practical graph isomorphism. Congressus Numerantium **30** (1981) 45–87