Towards Mining Generalized Patterns From RDF Data And A Domain Ontology

Tomas Martin[1], Victor Fuentes[1,2], Petko Valtchev[1], Abdoulaye Baniré Diallo[1,2], René Lacroix[3], Maxime Leduc[2], and Mounir Boukadoum[1]

[1] *CRIA*, Département d'informatique, UQÀM, Montréal, Canada
[2] *LACIM*, Département d'informatique, UQÀM, Montréal, Canada
[3] *Lactanet*, Sainte-Anne-de-Bellevue, Canada

Abstract. Nowadays, linked data (LD) are ubiquitous and mining them for knowledge, e.g. frequent patterns, needs not be argued for.
A domain ontology (DO) on top of a LD dataset enables the discovery of abstract patterns, a.k.a. *generalized*, capturing –rather than identical sub-structures– conceptual regularities in data. Yet with the resulting *ontologically-generalized graph patterns* (OGP), a miner faces the combined challenges of graph topology and a label hierarchy, which amplifies well-known difficulties with graphs such as support counting or non redundant pattern listing. As OGP mining is yet to be addressed in its generality, we propose a formalization and study two workaround methods that avoid tackling it head-on, i.e. deal with each aspect separately. Both perform pure graph mining with adapted label sets: *gSpan-OF* merely strips labels of hierarchical structure while *Tax-ON* first mines frequent graph topologies with only root classes as labels, then successively refines labels on each topology.

Keywords: Pattern mining · Ontologies · Graph data · Generalized patterns

1 Introduction

Linked data (LD) are nowadays produced and published in ever increased numbers, hence mining them for knowledge about the underlying domain has been recognized as an important research topic [19]. As a special case of such knowledge, structural patterns, either frequent [3] or rare [21], represent important (a)typical trends and regularities that might, for instance, reflect previously unknown phenomena or provide the explanation for observed behaviours.

The advantage of LD is they often come with, or might be fitted *a posteriori* to, a domain ontology (DO) [9] which is a prime source of descriptive domain knowledge [15]. A DO, through its class hierarchy, makes possible the discovery of more abstract patterns, a.k.a. *generalized patterns* (GP) [20] in the data mining (DM) field. GP refer to abstractions in places where data records refer to individual objects, or *items*. In this way, they go beyond the detection of identical sub-records (as in plain patterns) to capture the shared conceptual structure. GP are agnostic to data record topology, e.g. graph, sequence or flat set of items). We tackle pattern mining with a DO in a precision farming [6] context: DOs, typically designed in OWL, have gained significant popularity in

life sciences [8], in particular, in agriculture. In a project revolving around dairy production optimization [11], we are looking after interpretable patterns [16] that might help compare and contrast populations of dairy cows and entire herds within the Canadian dairy livestock. To that end, we designed a DO [10] in OWL. In these settings, the resulting *ontologically-generalized graph patterns* (OGP), as we named them, are labelled multi-digraphs with vertex/edge labels being DO classes/properties.

An OGP miner faces significant challenges due to interplay between graph topology and hierarchy on labels, e.g. in typical pattern mining concerns like support counting and non redundant traversal of the pattern space. Since the OGP mining problem has yet to be addressed in its generality, we propose here a formalization thereof. Then, as a means to assess the need for a dedicated OGP miner, we study two workaround methods that avoid tackling the problem head-on. Instead, our methods deal with a single facet of the problem at a time. Both perform pure graph mining using *gSpan* [22], a reference graph miner, with adapted ontological label sets: *gSpan-OF* merely strips DO classes of their hierarchical structure while *Tax-ON* uses the most generic classes to discover frequent graph topologies then puts each through a sequence of label specializations.

When assessing these methods on a sample of our dairy control data, we observed that both incur high computational costs due to the combinatorial nature of the underlying pattern spaces. We see this as an argument in favour of a more subtle approach that mixes topology extensions with label refinements within a unique mining step.

The remainder of the paper is structured as follows: Section 2 summarizes related prior work. Next, section 3 states the OGP mining problem and describes both workarounds. Then, section 4 reports on their respective performances and observed limitations. Finally, section 5 concludes the paper.

2 Related work

Graphs are among the most difficult data structures to mine as basic operations involved are akin to the costly (sub-)graph isomorphism. *gSpan* [22] is arguably the reference method: To mine collections of undirected labelled graphs, it moves down a spanning tree of the pattern space each time extending a parent with a new edge. It exploits a canonical form, the depth-first-search (DFS) encoding, to prune redundant tree branches, which may require extensive graph comparisons. *gSpan* is the basis for a number of RDF graph miners. *Gaston* [18] is another popular graph pattern miner.

Taxonomies have been used as a source of domain knowledge in DM from its onset [4] thus leading to the *generalized patterns* (GP) where categories from a domain taxonomy replace some of the individual items, e.g in sequential pattern mining [3]. Our own brand of patterns arise from labelled (multi-)graphs, i.e. named RDF graphs, as data records. Moreover, pattern nodes are labelled by OWL classes and edges by OWL properties at various abstraction levels (see definitions in section 3).

The generalized graph pattern mining was introduced in [12] which proposes adapting *AGM* [13] to vertex/edge label taxonomies. While the intended pattern generalizing/specializing operator(s) is unclear, by paper's admission, vertex taxonomies alone make the task way more challenging and, without effective pruning strategies, the output grows prohibitively large. In [1, 2], a framework for mining ontology-based pat-

terns from click-streams is presented: Their *xPMiner* method outputs sequences of ontology classes linked by object properties. While patterns are basically graphs of classes/properties, they have handy sequence backbones simplifying both pattern space traversal and support computing w.r.t. to our unrestricted settings. In [5], a *Gaston*-based method is proposed for mining abstract graph patterns from RDF. They proceed as follows: In the pre-processing step, the RDF graphs are transformed by replacing individual resource nodes by one of its abstract types from the ontology. However, since no generalization step is included, the method is unable to discover patterns involving classes on different abstraction levels than those explicitly assigned as vertex labels.

Taxogram [7] is arguably the first method to mine frequent generalized graph patterns (over a mere vertex label taxonomy, though). At step one, it runs *gSpan* [22] to discover all pattern topologies using the most general concept label for every vertex in data graphs. Then, each of the resulting most general patterns is gradually specialized: The method goes down the taxonomy for every vertex up till reaching an infrequent specialization. Since no order is assumed on vertices, duplicate patterns might be generated, hence the need for (expensive) isomorphism checks.

GP-Close [14] mines GP from RDF datasets with a schema: It splits graphs into triples and mines those as mere transactions of triple-shaped items. Yet resulting GPs might not constitute connected graphs. Later on, [23] adapted *gSpan* to RDF with a DO, yet with no label refinement step. Recently, the extraction of ontology-based *path-shaped* frequent patterns, i.e. sequences, was studied in [17]. Their method focuses on scalability issues related to blending graph combinatorics and DO hierarchy traversals.

3 Ontology-based graph pattern mining

Below, we state the problem and present two simple mining methods. In that, we use our dairy cattle performance ontology (DCPO) [10]: An excerpt thereof is given in Figure 1.

3.1 Problem statement

A pattern mining task [3] is defined by two languages (data records and patterns) and a quality criterion. Let $\Omega = \langle O, C, R, \leq_C, \leq_R, \in_C, \rho \rangle$ be an ontology where O, C and R are its sets of objects, classes, and object properties, respectively. Both classes and properties are organized into hierarchies $H_C = \langle C, \leq_C \rangle$ and $H_R = \langle R, \leq_R \rangle$ with $\leq_C$ denoting the `rdfs:subClassOf` relation and $\leq_R$ the `rdfs:subPropertyOf` one. The instantiation relation $\in_C \subseteq O \times C$ is the translation of `rdf:type`. The incidence relation $\rho \subseteq C \times R \times C$ is made of triples $c_1 \times r \times c_2$ denoting a property r between classes c_1 (*domain*) and c_2 (*range*). Observe that, from RDF/OWL point of view, we admit only object properties: Data ones are assumed encoded, prior to the analysis, into a suitable class hierarchy where classes model value ranges. For instance, in Figure 1, `HerdLeaveReason` and subclasses translate a data property whereby leaf classes model original values and the remainder expert-provided abstractions.

Our data language $\mathscr{L}_d$ is akin to RDF named graphs: A graph data record $g_d \in \mathscr{L}_d$ (see Figure 2 on the bottom) represents a doubly-labelled directed multi-graph. Formally, it is a tuple $g_d = \langle V_d, E_d, \lambda_o, \lambda_r \rangle$ where V_d is a set of vertices, E_d is a bag

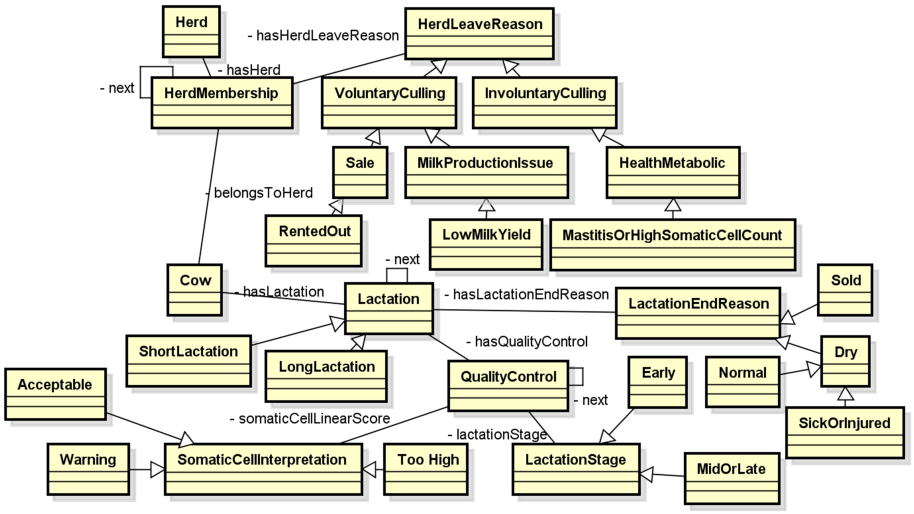

Fig. 1: Excerpt of our DCPO [10].

of pairs of vertices and λ_o, λ_r are two labelling functions based on Ω. Moreover, $\lambda_o : V_d \to O$ maps each vertex to an object while $\lambda_r : E_d \to R$ maps an edge to a property. Intuitively, a pair of adjacent vertices in g_d exists iff the corresponding RDF triple exists in the triple store. Next, a pattern $g_p \in \mathscr{L}_p$ is also a doubly-labelled directed multi-graph $g_p = \langle V_p, E_p, \lambda_c, \lambda_r \rangle$. $\lambda_c : V_p \to C$ sends vertices into ontology classes while λ_r works the same way as above. We call such patterns *ontologically-generalized graphs patterns* (OGP), yet the term will only be used whenever ambiguity can arise.

Figure 2 shows an OGP and a matching data graph (labels from Figure 1). Albeit of similar composition, $\mathscr{L}_p$ is not easily mapped to RDFS. It is rather akin to an RDF format where resources are *exemplars* of the corresponding DO entities.

Next, a relation $\dashv_\Omega \subseteq \mathscr{L}_d \times \mathscr{L}_p$ reflects the fact that a data graph $g_d = \langle V_d, E_d, \lambda_o, \lambda_r \rangle$ matches a pattern $g_p = \langle V_p, E_p, \lambda_c, \lambda_r \rangle$. Formally speaking, it is akin to a sub-graph isomorphism extended by *is-a* links. Thus, we note $g_d \dashv_\Omega g_p$ whenever an injective graph morphism $\mu : g_p \to g_d$ exists s.t. $\forall v_p \in V_p$, $\lambda_o(\mu(v_p)) \in_C \lambda_c(v_p)$ and $\forall e_p \in E_p$, $\lambda_r(\mu(e_p)) \leq_R \lambda_r(e_p)$. As an example, in Figure 2, consider the μ mapping (dashed line) of the edge labelled *&hasLactation*[4] (top) to `hasFirstLactation` (bottom). In DCPO, there is a `rdfs:subPropertyOf` link between both. Similarly, *&InvoluntaryCulling* refers to a super-class of `MastitisOrHighSCC`, the most specific type of `Mastitis_1`.

In a similar way, we define the generality between two patterns, $\sqsubseteq_\Omega \subseteq \mathscr{L}_p \times \mathscr{L}_p$, i.e. via a subgraph isomorphism extended by *subclass* links from the ontology (the major difference w.r.t. $\dashv_\Omega$ is `rdf:type` is replaced by `rdfs:subClassOf`*). Thus, we note $g_p \sqsubseteq_\Omega g'_p$ whenever an injective graph morphism $\eta : g'_p \to g_p$ exists s.t. $\forall v'_p \in V'_p$, $\lambda_c(\eta(v_p)) \leq_C \lambda_c(v'_p)$ and $\forall e'_p \in E'_p$, $\lambda_r(\eta(e_p)) \leq_R \lambda_r(e'_p)$. A proper illustration of η is not in our figures: The pattern in Figure 2 trivially generalizes the one in Figure 3 as

[4] Pattern labels in $\mathscr{L}_p$ will be prefixed by & to differentiate them from ontology entities.

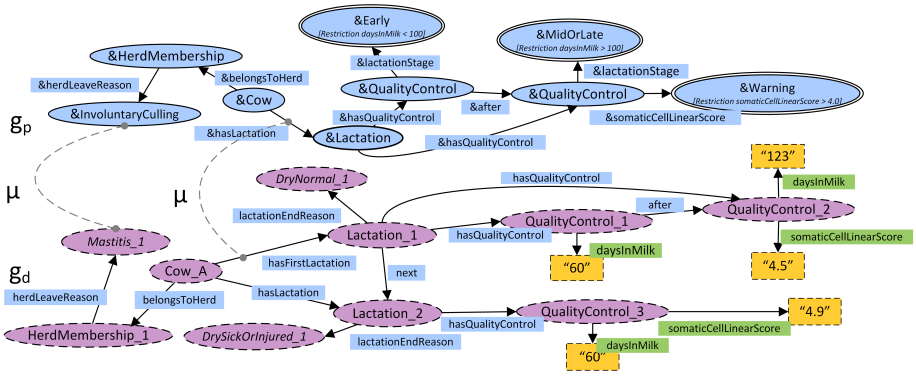

Fig. 2: Sample pattern (top), supporting data graph (bottom) and some μ-mappings. Labels are provided within the ovals for vertices and next to the line segment for edges.

a subgraph thereof. For a non trivial example one could replace *&InvoluntaryCulling* in Figure 3 by *&HealthMetabolic*. $\sqsubseteq_\Omega$, induces a hierarchy on $\mathscr{L}_p$, $\langle \mathscr{L}_p, \sqsubseteq_\Omega \rangle$. OGP miners have to traverse it while using an interestingness criterion, e.g. *support*.

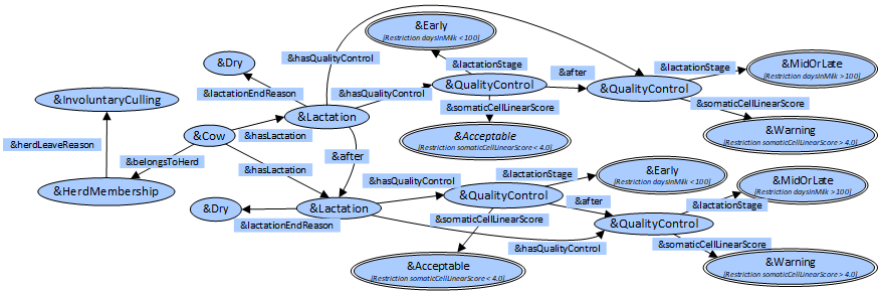

Fig. 3: An interesting pattern too far down the pattern space ($\sigma = 138$, $|D| = 7425$).

3.2 An OGP from the dairy dataset

The pattern in Figure 3 –found via a SPARQL query and deemed useful by our experts– reflects the fact that 138 cows culled for reasons beyond farmer's control (`Involuntary` class) had prior health issues. Thus, for at least two lactations, they recorded a period of acceptable somatic cell levels followed by another one with more worrisome values. Such scenarios are plausible as higher scores are major signals for *mastitis* (udder inflammation). Consequently, recurrent health issues late in the lactation period could very well be the trigger for the involuntary culling and hence deserve closer monitoring.

Noteworthily, OGPs capture shared structure in data plain patterns would miss as the matching elements in data graphs might differ. For instance, the cows whose graphs match our OGP might have been culled for a variety of reasons, e.g. udder breakdown. Yet by referring to the common `Involuntary` class, the OGP helps factor out higher-order commonalities only available through the DO. This effect of label hierarchy, known from GP, spreads over edge labels also in our DO-based settings. Computationally, though, the OGP proved hardly reachable since located 32-deep within $\mathscr{L}_p$.

3.3 Two workaround approaches

As a first approach, we designed two workaround solutions that avoid mixing concept hierarchy- and graph topology-related aspects during $\langle \mathscr{L}_p, \sqsubseteq_\Omega \rangle$ traversal. Here, both methods produce $\mathscr{F}^\Omega$, the set of all frequent OGPs. The goal was to assess the alternatives to a fully-blown OGP miner going down $\langle \mathscr{L}_p, \sqsubseteq_\Omega \rangle$ while using a yet-to-define canonical form and a dedicated refinement operator (e.g. as in [2]).

First, *gSpan-OF* (for *gSpan* with Ontology Flattened), runs *gSpan* with a flat set of ontological labels, i.e. ignoring the relations $\leq_C$ and $\leq_R$ at pattern generation time. Only during support check for a pattern ($\dashv_\Omega$), do $\leq_C$, $\leq_R$, and $\in_C$ interfere in mining. Precisely speaking, the traversal follows a subset of $\sqsubseteq_\Omega$ corresponding to pure subgraph isomorphism, thus ignoring links in $\langle \mathscr{L}_p, \sqsubseteq_\Omega \rangle$ representing label specializations. *gSpan-OF* benefits from *gSpan*'s parsimonious traversal: Its canonical form checks help avoid duplicate generations. An overview of *gSpan-OF* is provided by Algorithm 1.

Algorithm 1: *gSpan-OF*

Input: Ontology Ω, graph database D, minimal support threshold ς
Output: Set of frequent OGPs $\mathscr{F}^\Omega$
1 **Mine** frequent triples $t \in C \times \mathscr{R} \times C$ into $\mathscr{F}_1^\Omega$, the set of frequent *single-edge* OGPs
2 **Apply** *gSpan* to **discover** all frequent OGPs, **recursively** extending OGPs in $\mathscr{F}_1^\Omega$
3 | **Extend** a frequent OGPs g_p using a frequent $t \in \mathscr{F}_1^\Omega$ to produce $g_{p'}$
4 | For each $g_{p'}$, **test** its DFS encoding for canonicity
5 | **Compute** $\sigma(g_{p'})$ (find data graphs $g_d \dashv_\Omega g_{p'}$)
6 | **If** $\sigma(g_{p'}) > \varsigma$, $g_{p'}$ is a frequent OGP, then **recursively** extend $g_{p'}$
7 **end**

Second, *Tax-ON* (for *Taxogram* with Ontology and Non-redundant output) untangles topology extensions and label specialization differently (Algorithm 2). Loosely following [7], it tackles them separately, in subsequent steps: (1) graph mining on suitably re-labelled data graphs and (2) label refinements on each of the patterns found at step (1). Thus, *Tax-ON* first mines $\mathscr{F}^G$, the most generic frequent graph patterns, using plain *gSpan*. To that end, data graphs are cloned and then vertex/edge labels replaced by the most generic super-entities in H_C and H_R, respectively. In step two, an extension of *Taxogram* refines vertex labels. Thus, each OGP undergoes a sequence of individual label specializations, i.e. substitutions of a class by a direct subclass, to go down H_C.

To provide additional context, the pattern in Figure 3 is a 21-pattern within the *gSpan-OF*'s pattern space (same as *gSpan*), while it can be obtained by refining a more realistic 10-pattern belonging to $\mathscr{F}^G$ with *Tax-ON* (striping out all white vertices).

Algorithm 2: *Tax-ON*

Input: Ontology Ω, graph database D, minimal support threshold ς
Output: Set of frequent OGPs $\mathscr{F}^{\Omega}$
Mine $\mathscr{F}^{G}$ the set of frequent most-generic OGPs
  Build D' a copy of D, where vertex labels are replaced by their most generic class in $C_0 = roots(H_C)$
  Mine frequent triples $t \in C_0 \times \mathscr{R} \times C_0$ into $\mathscr{F}^{G}_1$, the single-edge subset of $\mathscr{F}^{G}$ over D'
  Apply *gSpan* to **discover** all OGPs in $\mathscr{F}^{G}$, from all OGPs in $\mathscr{F}^{G}_1$
end
Mine all other frequent OGPs, from $\mathscr{F}^{G}$ over D
  Apply *Taxogram* to **recursively** label-refine OGPs in $\mathscr{F}^{G}$
    Replace one vertex label by one of its direct descendants in H_C to generate $g_{p'}$
    Compute $\sigma(g_{p'})$ (same as in *gSpan-OF*), **if** $g_{p'}$ is a frequent OGP, then **recursively** label-refine $g_{p'}$
  end
  Eliminate duplicate OGPs using graph isomorphism checks
end

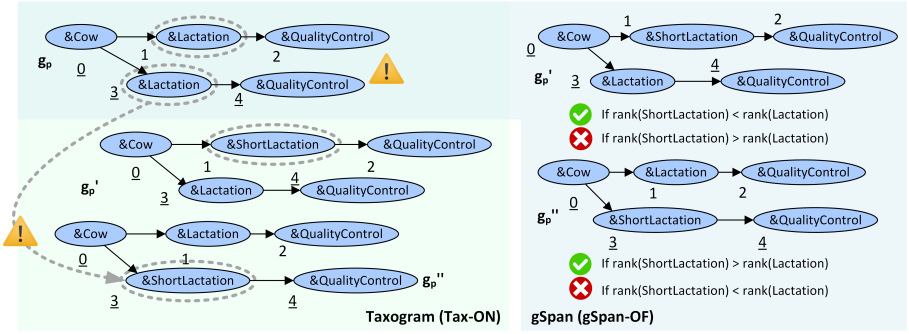

Fig. 4: Hierarchy-centered exploration vs Vertex-centered exploration.

4 Evaluation of the DO-powered pattern miners

We put both methods through some preliminary experiments. In practice, the main limiting factor, shared by both, is the number of automorphisms in data graphs and patterns. In *gSpan*, this slows down candidate generation, canonical check and support-computing steps. Also, it creates a prohibitive number of embeddings of a candidate pattern to the data graphs (up to 10^6 for one data graph). This prevents, in turn, the efficient pruning during candidate generation based on explicit embedding management. That same issue prevents *Taxogram* from building a vertical database, which forces expensive support counting over the graph database (via $\dashv_{\Omega}$).

Conceptually, the key point is that both methods forgo part of the available structure: *gSpan-OF* ignores the hierarchies in the DO, while *Tax-ON* strips a pattern from its graph information bringing it down to vertex sequence. Simply put, neither flattening the conceptual hierarchy nor ignoring the graph topology is efficient enough in practice to allow for an in-depth exploration of $\langle \mathscr{L}_p, \sqsubseteq_{\Omega} \rangle$. On one hand, *gSpan*'s non-redundant exploration is burdened by a large number of candidates induced by flat label sets since. On the other, *Taxogram*'s loose refinement approach outputs lots of duplicate patterns.

Figure 4 compares *gSpan*'s vertex-centered exploration to *Taxogram*'s hierarchy-centered one on their way to deal with an OGP g_p and its two (isomorphic) specializations g'_p and g''_p. On the left, *Taxogram* produces both g'_p and g''_p from g_p by refining

&Lactation into *&ShortLactation* on positions 1 and 3, respectively. This exemplifies a key shortage: By specializing all positions unrestrictedly, it allows for duplicates to arise. In contrast, *gSpan* –through its canonical form-driven exploration– avoids either g'_p or g''_p since exactly one of them will comply to that form constraints. This capacity is independent from the order on labels that underlies the canonical form, as shown on the right of the figure (for additional details, see Figure 6 in Appendix).

Conversely, by traversing every subspace of $\langle \mathscr{L}_p, \sqsubseteq_\Omega \rangle$ induced by a OGP topology while following $\sqsubseteq_\Omega$, *Taxogram* benefits from support anti-monotony. Thus, g'_p and g''_p will be tested only if g_p proves frequent. For *gSpan*, though, g_p, g'_p, and g''_p are incomparable since located at the same (size-induced) level within its flattened pattern space. Therefore, the status of any of the three OGPs is immaterial while testing the other two.

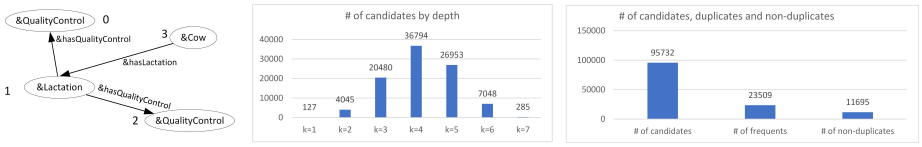

Fig. 5: *Taxogram*: number of patterns, duplicates and candidates for a 3-pattern.

Figure 5 clarifies the number of candidates *Taxogram* examines while testing all possible specializations of a specific 3-pattern (left). Here, up to seven specializations are required to reach a most specific pattern while the peak number of candidates is generated at depths four and five below. The worrying aspect is among the ca. 100k specializations tested, some 50% were duplicates (ratio increases with the pattern size).

A suitable canonical form is essential for the efficient exploration of $\langle \mathscr{L}_p, \sqsubseteq_\Omega \rangle$. Yet current forms such as *gSpan*'s are not designed to work with label specializations. A critical property to guarantee for a refinement operator is the anti-monotonicity of the canonical form: A canonical OGP code must only have canonical ancestors. The highly irregular structure of $\langle \mathscr{L}_p, \sqsubseteq_\Omega \rangle$ is a major challenge faced by such operators.

5 Conclusion

We presented here a first attempt at efficiently mining frequent generalized patterns from an RDF dataset while using a DO as a generalization source. As the corresponding DM task is beyond the reach of state-of-the-art methods, we designed two workaround solutions: (1) pure graph pattern mining with a flattened set of labels and (2) graph mining with only root class/property labels followed by recursive pattern specialization.

Having put them through experimental evaluation, we observed that both approaches suffer on high computational overhead, most likely due to the highly combinatorial nature of the resulting pattern spaces. This clearly warrants a more subtle blend of topology extension and label refinement in a uniform descend in $\langle \mathscr{L}_p, \sqsubseteq_\Omega \rangle$. Therefore, we are currently investigating the design of a dedicated OGP miner exploiting a tailor-made canonical form and support computing mechanisms.

References

1. Adda, M., et al.: On the discovery of semantically enhanced sequential patterns. In: 4th ICMLA. pp. 8–pp. IEEE (2005)
2. Adda, M., et al.: A framework for mining meaningful usage patterns within a semantically enhanced web portal. In: 3rd C* Conf. on Comp. Sci. and Soft. Eng. pp. 138–147 (2010)
3. Aggarwal, C., et al.: Frequent Pattern Mining. Springer, 2014 edn. (Aug 2014)
4. Anand, S., et al.: The role of domain knowledge in data mining. In: CIKM. pp. 37–43 (1995)
5. Berendt, B.: Using and learning semantics in frequent subgraph mining. In: International Workshop on Knowledge Discovery on the Web, 2005. pp. 18–38. Springer (2005)
6. Brett, D., et al.: A survey of semantic web technology for agriculture. Information Processing in Agriculture (2019)
7. Cakmak, A., Ozsoyoglu, G.: Taxonomy-superimposed graph mining. In: 11th EDBT. pp. 217–228. ACM (2008)
8. Cannataro, M., Santos, R.D., et al.: Biomedical and bioinformatics challenges to computer science. Proc. Comp. Science **1**(1), 931–933 (2010)
9. Dou, D., et al.: Semantic data mining: A survey of ontology-based approaches. In: IEEE ICSC. pp. 244–251 (2015)
10. Fuentes, V., et al.: Dairy ontology to support precision farming. In: 12th ICBO (2021)
11. Gonçalves Frasco, C., et al.: Towards an Effective Decision-making System based on Cow Profitability using Deep Learning. In: 12th ICAART. pp. 949–958 (2020)
12. Inokuchi, A.: Mining generalized substructures from a set of labeled graphs. In: Fourth IEEE International Conference on Data Mining (ICDM'04). p. 415–418. IEEE (2004)
13. Inokuchi, A., et al.: An apriori-based algorithm for mining frequent substructures from graph data. In: European Conf. on Principles of DM & KD. pp. 13–23. Springer (2000)
14. Jiang, T., et al.: Mining generalized associations of semantic relations from textual web content. IEEE transactions on knowledge and data engineering **19**(2), 164–179 (2007)
15. Kramer, F., Beißbarth, T.: Working with ontologies. In: Bioinformatics, pp. 123–135 (2017)
16. Martin, T., et al.: Leveraging a domain ontology in (neural) learning from heterogeneous data. In: CIKM (Workshops) (2020)
17. Monnin, P.: Matching and mining in knowledge graphs of the Web of data-Applications in pharmacogenomics. Ph.D. thesis, Université de Lorraine (2020)
18. Nijssen, S., Kok, J.: A quickstart in frequent structure mining can make a difference. In: 10th ACM KDD. pp. 647–652 (2004)
19. Rettinger, A., et al.: Mining the semantic web. DMKD **24**(3), 613–662 (2012)
20. Srikant, R., Agrawal, R.: Mining generalized association rules. Future Generation Computer Systems **13**(2–3), 161–180 (1997)
21. Szathmary, L., et al.: Towards Rare Itemset Mining. In: 19th IEEE ICTAI. vol. 1, pp. 305–312 (Oct 2007)
22. Yan, X., Han, J.: gspan: Graph-based substructure pattern mining. In: IEEE ICDM. pp. 721–724 (2002)
23. Zhang, X., et al.: Mining link patterns in linked data. In: 13th WAIM. pp. 83–94 (2012)

Appendix

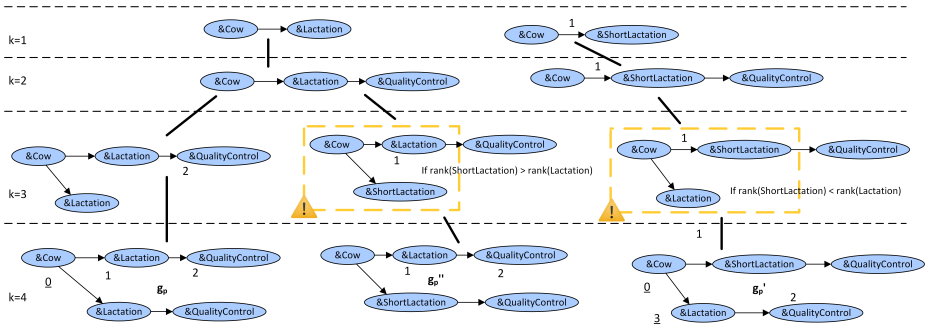

Fig. 6: *gSpan*'s flattened exploration of $\mathscr{L}_p$