Efficient homomorphism-free enumeration of conjunctive queries

Jan Ramon[1], Samrat Roy[1], and Jonny Daenen[2]

[1] K.U.Leuven, Belgium,
Jan.Ramon@cs.kuleuven.be, Samrat.Roy@cs.kuleuven.be
[2] University of Hasselt, Belgium,
Jonny.Daenen@uhasselt.be

Abstract. Many algorithms in the field of inductive logic programming rely on a refinement operator satisfying certain desirable properties. Unfortunately, for the space of conjunctive queries under θ-subsumption, no optimal refinement operator exists. In this paper, we argue that this does not imply that frequent pattern mining in this setting can not be efficient. As an example, we consider the problem of efficiently enumerating all conjunctive queries of bounded treewidth and show that it can be achieved with polynomial delay.

1 Introduction

Many algorithms in the fields of machine learning and data mining consist of a component enumerating all hypotheses or patterns of a search space in a certain order, and a component further investigating these hypotheses. For example, a frequent pattern mining algorithm typically has a component generating candidate patterns in the pattern language and a component checking whether the candidates are frequent.

The computational cost of both components depends on amongst other the matching operator chosen. The majority of Inductive Logic Programming algorithms uses theta subsumption, which is equivalent to homomorphism between graphs for datalog. Some algorithms also employ the so-called object identity matching operator, which is equivalent to subgraph isomorphism for datalog.

While the complexity of deciding subgraph homomorphism is lower than deciding subgraph isomorphism, enumerating patterns under isomorphism is easier and better understood. Indeed, deciding whether a graph H is subgraph isomorphic to a graph G is NP-complete, even if both G and H belong to the restricted class of graphs with bounded treewidth. By contrast, if H has bounded treewidth, one can decide in polynomial time whether a graph H is subgraph homomorphic to a graph G, without imposing any restrictions to G [9]. As far as enumeration is concerned, isomorphism-free enumeration of patterns has been studied extensively [4, 13, 12], while homomorphism-free enumeration of graphs is more difficult because larger graphs may be subgraph homomorphic to smaller graphs. For example, for any $n > 1$, the query

$p(X_1, X_2), p(X_2, X_3), \ldots, p(X_{n-1}, X_n)$ is more general than the short query $p(X, X)$. One can even show that an optimal refinement operator does not exist, even for simple classes of patterns [10]. These difficulties were one of the reasons why after the initial homomorphism-based pattern miners (e.g. the Warmr system [2]), attention shifted to isomorphism-based graph mining [14, 8, 11, 7].

However, there have been proposed an increasing number of applications featuring large networks of millions of nodes, such as social networks, economic networks, traffic networks, chemical interaction networks and concept networks. For such applications, subgraph isomorphism is intractable outside a very restricted class of patterns. For this reasons, we argue that it may be important to revisit homomorphism based pattern mining, since in that setting complexity only depends polynomially on the size of the network and only moderately on the size of the mined patterns.

In this paper, we show that despite the negative theoretical results about the non-existence of certain types of refinement operators, it is still possible to enumerate equivalence classes of conjunctive queries under θ-subsumption (homomorphism) with polynomial delay. Our contribution is twofold. First, we point out a theoretical argument indicating that most of the solutions of an algorithm for isomorphism-free enumeration of graphs will also be solutions for the problem of homomorphism-free enumeration. Second, we provide a polynomial delay algorithm listing all conjunctive queries of bounded treewidth.

The remainder of this paper is structured as follows. After Section 2, where we review some basic concepts, definitions and earlier results. We first consider the general case and discuss in Section 3 an algorithm for homomorphism-free listing of patterns. Next, in Section 4 we consider the special case of bounded treewidth patterns and propose a deterministic algorithm. Finally, Section 5 provides some further perspectives and concludes.

2 Preliminaries

2.1 Basic graph theory concepts

In the technical part of this paper, we will consistently use graph terminology. In this subsection, we will first review a number of basic concepts.

A (vertex-labeled) **graph** is a triple $G = (V, E, \lambda)$ where V is a set of **vertices**, $E \subseteq \big\{\{u, v\} \mid u, v \in V\big\}$ is a set of **edges**, and λ is a **labeling** function $\lambda : V \to \Sigma$ on some alphabet Σ.

We also denote the vertices of a graph G with $V(G)$, the edges with $E(G)$, and the labeling function of G with λ_G. A vertex $v \in V$ is **incident** with an edge $e \in E$ if $v \in e$. The **degree** $d(v)$ of a vertex v is the number of edges incident with v. For a graph G, the (maximum) degree of G is $\Delta(G) = \max_{v \in V(G)} d(v)$.

G' is a **subgraph** of G, denoted $G' \subseteq G$, if $V(G') \subseteq V(G)$ and $E(G') \subseteq E(G)$. If $G' \subseteq G$ and $E(G') = \big\{\{x, y\} \in E(G) \mid x, y \in V(G')\big\}$, then G' is an **induced** subgraph of G.

A **simple** graph G is a graph which does not have self-loops, i.e. $\forall\{x,y\} \in E(G), x \neq y$ or parallel edges. For this paper, unless explicitly stated otherwise, we assume all graphs to be simple.

A **path** from a vertex v_0 to a vertex v_n in a graph G is a sequence of pairwise distinct vertices $(v_i)_{i=1}^n$ such that $\forall i, 0 \leq i < n, \{v_i, v_{i+1}\} \in E(G)$. A **tree** is a graph where there is exactly one path between each pair of vertices.

For two graphs G and H, G is called **isomorphic** to H, written as $G \cong_i H$, if there exists a bijective mapping $\pi : V(G) \to V(H)$, such that for any $u, v \in V(G)$, $\{u,v\} \in E(G) \Leftrightarrow \{\pi(u), \pi(v)\} \in E(H)$ and $\lambda(u) = \lambda(\pi(u))$. π is called an isomorphism from G to H. For two graphs G and H, G is called **homomorphic** to H, written as $G \cong_h H$, if there is a mapping $\varphi : V(G) \to V(H)$ such that $\forall u, v \in V(G), (u,v) \in E(G) \Rightarrow (\varphi(u), \varphi(v)) \in E(H)$ and $\forall v \in V(G), \lambda(u) = \lambda(\varphi(u))$. φ is called an homomorphism from G and H. An isomorphism $\pi : G \to G$, is called an **automorphism** of G. A homomorphism, $\varphi : G \to G$ is called an **endomorphism** of G. A graph G is asymmetric if it does not have any automorphism except for the identity. A graph G is called a **core** if every endomorphism of G is also an automorphism of G.

If a graph H is isomorphic to a subgraph of a graph G then H is called **subgraph isomorphic** to G, denoted $H \preceq_i G$. Similarly, if H is homomorphic to a subgraph of G then H is called **subgraph homomorphic** to G and it is denoted by $H \preceq_h G$.

A **tree-decomposition** of a graph G is a pair (X, T) where $T = (I, F)$ is a tree and $X = \{X_i | i \in I\}$ is a family of subsets of $V(G)$, called **bags**, one for each node of T, such that

- $\cup_{i \in I} X_i = V$.
- for all edges $(v, w) \in E$, there exists an $i \in I$ with $v \in X_i$ and $w \in X_i$.
- for all $i, j, k \in I$: if j is on the path from i to k in T, then $X_i \cap X_k \subseteq X_j$.

The **width** of a tree decomposition (X, T) is defined as: $\max_{i \in I} |X_i| - 1$. The **treewidth** of a graph G is the minimum width over all tree decompositions of G.

2.2 Refinement operators

Let $\leq$ be a partial order on a set L of patterns, and let $\equiv$ be the corresponding equivalence relation. For the class of graphs patterns, examples of such partial orders are $\preceq_i$ and $\preceq_h$, the corresponding equivalence relations being $\cong_i$ and $\cong_h$. Let $\top = \min(L)$ be the top element of L. A refinement operator ρ is a function mapping every element $x \in L$ on a set $\rho(x) \subseteq L$ such that $\forall y \in \rho(x), x \leq y$. We denote $\rho^1 = \rho$, $\rho^n = \rho \circ \rho^{n-1}$ and $\rho^* = \cup_i \rho^i$ A refinement operator ρ is called **finite** iff for every $A \in L$, $\rho(A)$ is finite. ρ is called **globally complete** iff for every $A \in L$, there is an $A' \in \rho^*(\top)$ with $A' \equiv A$. ρ is called **optimal** if ρ is locally finite, globally complete and for all $A_1, A_2 \in \rho^*(\top)$, there exists exactly one A' such that $A_1 \in \rho^*(A') \wedge A_2 \in \rho^*(A')$;

2.3 Complexity notions

An algorithm which lists solutions $s_1, ..., s_n$ of some problem is said to run with **polynomial delay** if the time before printing s_1, the time between printing s_i and s_{i+1} and the termination time after printing s_n is bounded by a polynomial in the size of the input.

3 The general case

An important problem when enumerating patterns is that we only want to list one representative of every equivalence class of patterns. In the case of isomorphism-free enumeration, for every equivalence class of isomorphic graphs we only want to list exactly one. A typical strategy is to use a canonical form or canonical way to generate the graph.

For homomorphism-free enumeration, the problem is harder, since two homomorphic graphs do not necessarily have the same size. One strategy is to just perform isomorphism-free enumeration of graphs, and then filter out those which are not cores. Two questions arise here. First, what fraction of the graphs will be rejected because they are non-cores, i.e. what part of the work will be unproductive in terms of generating representatives of new classes of graphs under homomorphism? Second, what is the complexity of checking whether they are cores?

Let us start with the first question. For many properties, either almost all graphs satisfy the property or almost no graph satisfies the property. This does not hold for all properties expressible in first order logic [3] but also for many other properties. In [5], a method is proposed to transform any property which holds for all graphs into an algorithm listing all graphs satisfying the property with polynomial delay. It is well-known that almost all graphs are cores [6]. Therefore, we can use the algorithm proposed in [5] to list all core graphs. The listing itself is possible with polynomial delay, the only additional cost is checking whether the graphs are cores.

Unfortunately, in general, deciding whether a graph is a core is NP-hard. Therefore, this filtering strategy would only result in a polynomial delay algorithm for a class of graphs which is both sufficiently large so that almost all graphs are member of it, but which is at the same time sufficiently restricted to make it possible to decide whether graphs are cores in polynomial time. We do not know of such class. Therefore, in the next section, we will consider a more restricted class and revert to a more specialized algorithm.

4 Bounded treewidth

In this section we will show an algorithm for homomorphism-free listing of graphs of bounded treewidth. Our method is based on the following construction. First, we present an algorithm that enumerates with polynomial delay graphs of bounded treewidth which are not guaranteed to be cores. Second, we prove that in a consecutive interval of a polynomial amount of enumerated graphs, at least one will be a core. From these two elements, we can then conclude that the delay between outputting two core graphs is bounded by a polynomial.

For the efficient isomorphism-free enumeration of bounded treewidth graphs, we rely on a canonical form for bounded treewidth graphs which is described in more detail in [1]. Using an extension of the standard tree enumeration methods, we can achieve an algorithm for isomorphism-free listing of all bounded treewidth graphs with delay $O(n^{w+3})$ where n is the number of vertices of the graphs and w is the maximal treewidth. In particular, we extend graphs from the rightmost path of their canonical tree decomposition. We only start a new branch in the tree decomposition of core graphs. If a graph is extended such that it is not a core anymore, then this branch is continued until the graph is a core again.

Checking whether a graph of bounded treewidth is a core is possible in polynomial time. Therefore, we now turn towards the remaining question whether a sufficient fraction of the enumerated patterns will be cores. Therefore, we prove the following lemma:

Lemma 1. *Let G be a connected graph with vertex labels from an alphabet Σ with $|\Sigma| \geq 3$, and let (T, B) be a tree decomposition for G of width at most $w \geq 2$. Let z be a node of T and let $Z = B(z)$ be the bag of z. Assume that all endomorphisms of G mapping all elements of Z on themselves are automorphisms of G. Let $\rho_z^m(G)$ be the set of all connected supergraphs P of G such that $|V(P)| - |V(G)| \leq m$, P has treewidth w and $\forall \{v, w\} \in V(P), v \in V(P) \setminus V(G) \Rightarrow w \in (V(P) \setminus V(G)) \cup Z$. Then, if $m \geq |V(P)|^2 / \log(|\Sigma| - 1)$ at least one element of $\rho_z^m(G)$ is a core.*

Proof (sketch). Let $Z = \{v_1, v_2, \ldots v_{|Z|}\}$ for some arbitrary ordering of the vertices in Z. Let $V_+ = v_{|Z|+1}, v_{|Z|+2}, \ldots v_{|Z|+m}$ be new vertices. Let $E_+ = \big\{\{v_i, v_j\} \mid (v_i, v_j \in V_+ \cup E) \wedge |i - j| \leq w\big\}$. Now consider the set of all graphs P with $V(P) = V(G) \cup V+$ and $E(P) = E(G) \cup E_+$ where the vertices in V_+ get assigned labels from Σ such that for all $1 \leq i \leq |Z| + m - (w + 1)$, $\lambda(v_i) \neq \lambda(v_{i+w+1})$. This is a strict subset of $\rho_z^m(G)$.

Let $P[V_+]$ be the subgraph of P induced by V_+. From the construction, it follows that any endomorphism of P will map $P[V_+]$ on an isomorphic copy of $P[V_+]$. There are at most $|V(P)|^2$ different possible images of $P[V_+]$ under such endomorphism. If $(|\Sigma| - 1)^m \geq |V(P)|^2$, then for at least one of the $(|\Sigma| - 1)^m$ considered graphs, all endomorphisms must map V_+ on itself, and hence Z on itself. This graph will therefore be a core. □

Several of the assumptions in the lemma are only here to keep the proof simple. We can conclude that at least one of every $O(|V(G)|^2)$ refinements of a pattern will be a minimum size representative of its homomorphism class. Note that although it is possible that not all of these refinements are in canonical form, the pruning of these does not affect the polynomial delay result.

5 Conclusions

In this paper we presented methods for the homomorphism-free enumeration of graph patterns. Such methods are important as a first step towards the efficient mining of frequent conjunctive queries in large networks. We argue that despite the existing negative results, homomorphism-free enumeration may still be possible efficiently for broad classes of graphs such as the class of all bounded treewidth graphs.

In future work, we want to improve our results. In particular, we are interested in the question whether one can mine with polynomial delay all conjunctive queries which are frequent with respect to some given data set. This problem is harder than the one tackled in the current paper, as it is possible that a fraction of the enumerated patterns is not frequent and hence must be discarded.

References

1. J. Daenen. Isomorfvrije generatie van een contextvrije, geconnecteerde graaftaal met begrensde graad. Technical report, Universiteit Hasselt, 2009. In Dutch, available from the author.
2. L. Dehaspe and H. Toivonen. Discovery of frequent datalog patterns. *Data Mining and Knowledge Discovery*, 3(1):7–36, 1999.
3. R. Fagin. Probabilities on finite models. *J. of Symbolic Logic*, 41(1):50–58, 1976.
4. L.A. Goldberg. Efficient algorithms for listing unlabeled graphs. *Journal of Algorithms*, 13(1):128–143, 1992.
5. L.A. Goldberg. Polynomial space polynomial delay algorithms for listing families of graphs. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing (STOC'93)*, pages 218–225, New York, NY, USA, 1993. ACM Press.
6. P. Hell and J. Nestril. *Graphs and homomorphisms.* Oxford University Press, 2004.
7. Tamás Horváth and Jan Ramon. Efficient frequent connected subgraph mining in graphs of bounded tree-width. *Theoretical Computer Science*, 411:2784–2797, 2010.
8. A. Inokuchi, T. Washio, and H. Motoda. Complete mining of frequent patterns from graphs: Mining graph data. *Machine Learning*, 50(3):321–354, 2003.
9. Jiff Matousek and Robin Thomas. On the complexity of finding iso- and other morphisms for partial k-trees. *Discrete Mathematics*, 108:343–364, 1992.
10. Siegfried Nijssen and Joost Kok. There is no optimal, theta-subsumption based refinement operator. Personal communication.
11. Siegfried Nijssen and Joost N. Kok. A quickstart in frequent structure mining can make a difference. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 647–652, 2004.
12. Christophe Paul, Andrzej Proskurowski, and Jan Arne Telle. Generating graphs of bounded branchwidth. In *Proceedings of the 32nd International Workshop on Graph Theoretical Concepts in Computer Science*, volume 4271 of *Lecture Notes in Computer Science*, pages 206–216, 2006.
13. J. Ramon and S. Nijssen. Polynomial delay enumeration of monotonic graph classes. *Journal of Machine Learning Research*, 10:907–929, 2009.
14. Xifeng Yan and Jiawei Han. gSpan: Graph-based substructure pattern mining. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002)*, pages 721–724, Japan, 2002. IEEE Computer Society.