Mine 'Em All: A Note on Mining All Graphs

Ondřej Kuželka[1] and Jan Ramon[2]

[1] School of Computer Science & Informatics, Cardiff University, UK
KuzelkaO@cardiff.ac.uk
[2] Department of Computer Science, KU Leuven, Belgium
jan.ramon@cs.kuleuven.be

Abstract. We study the complexity of the problem of enumerating all graphs with frequency at least 1 and computing their support.

1 Introduction

We study graph mining problems from a nontraditional perspective. We are inspired by the question which properties of the problem make some graph mining problems solvable in incremental polynomial time or with polynomial delay[3]. Here, we do not require the discovered graph patterns to be frequent and we want to output all patterns occurring in at least one database graph. However, we still want to also output their occurrences. In addition, we constrain the order in which the patterns should be printed, e.g. from most frequent patterns to least frequent patterns, which allows us to connect our results to results on (in)frequent graph mining. Surprisingly, for several graph classes, we show that different orders lead to very different computational complexities. For instance mining planar graphs cannot be done *in incremental-polynomial time* when the output graphs should be ordered by frequency but it can be done with *polynomial delay* when they should be ordered from largest to smallest.

2 Preliminaries

Graphs. An *undirected graph* is a pair (V, E), where V is a finite set of *vertices* and $E \subseteq \{e \subseteq V : |e| = 2\}$ is a set of *edges*. A *labeled undirected graph* is a triple (V, E, λ), where (V, E) is an undirected graph and $\lambda : V \cup E \to \Sigma$ is a function assigning a label from an alphabet Σ to every element of $V \cup E$. A graph G' is a *subgraph* of a graph G, if $V(G') \subseteq V(G)$, $E(G') \subseteq E(G)$, and $\lambda_{G'}(x) = \lambda_G(x)$ for every $x \in V(G') \cup E(G')$; G' is an *induced subgraph* of G if it is a subgraph of G satisfying $\{u, v\} \in E(G')$ if and only if $\{u, v\} \in E(G)$ for every $u, v \in V(G')$. Graphs G and G' are *isomorphic* if there exists a bijection $\pi : V(G) \to V(G')$ such that $\{u, v\} \in E(G)$ if and only if $\{\pi(u), \pi(v)\} \in E(G')$. Graph canonization is a function from graphs to strings such that two graphs have the same canonization if and only if they are isomorphic. Graphs G and G'

[3] A longer version of this paper with proofs is available from https://goo.gl/NRrYAF.

are *isomorphic* if there exists a bijection $\pi : V(G) \rightarrow V(G')$ such that $\{u, v\} \in E(G)$ if and only if $\{\pi(u), \pi(v)\} \in E(G')$. Graph canonization is a function from graphs to strings such that two graphs have the same canonization if and only if they are isomorphic. We say that a graph G_1 is *(induced) subgraph isomorphic* to a graph G_2 if G_1 is isomorphic to an (induced) subgraph of G_2. There are graph classes, e.g. bounded-treewidth graphs or planar graphs, for which isomorphism can be decided in polynomial time but for which subgraph isomorphism is NP-complete [5]. We are mostly interested in such classes because for them it is not obvious whether *fast* graph mining algorithms exist.

3 Graph Mining Problems

A *transaction database* is a a multiset of graphs from a given class $\mathcal{G}$. Given a pattern matching operator $\preccurlyeq$ (subgraph isomorphism or induced subgraph isomorphism), the frequency of a graph G in a transaction database DB, denoted by $\mathsf{freq}(G, DB)$, is given as $\mathsf{freq}(G, DB) = |\{G' \in DB | G \preccurlyeq G'\}|$. Given a threshold t, G is said to be frequent if $\mathsf{freq}(G, DB) \geq t$. The elements of the multiset $\{G' \in DB | G \preccurlyeq G'\}$ are called *occurrences* of the graph G in the database DB.

Definition 1 (The Frequent Connected Graph Mining (FCGM) Problem). Given *a class $\mathcal{G}$ of graphs, a transaction database DB of graphs from $\mathcal{G}$, a pattern matching operator $\preccurlyeq$, and frequency threshold,* list *the set of frequent connected graphs $G \in \mathcal{G}$ and their occurrences.*

In this paper, we are interested in another closely related type of problem which is to mine all graphs with frequency at least one *in certain order.*

Definition 2 (The Ordered Mining Problems). Given *a class $\mathcal{G}$ of graphs, a transaction database DB of graphs from $\mathcal{G}$ and a pattern matching operator $\preccurlyeq$,* list *the set of connected graphs $G \in \mathcal{G}$ with $\mathsf{freq}(G, DB) \geq 1$ and their occurrences in the transactions in the given order*[4]*: (i) from most frequent to least frequent ($\mathsf{ALL}_{F \rightarrow I}$ problem), (ii) from least frequent to most frequent ($\mathsf{ALL}_{I \rightarrow F}$ problem), (iii) from smallest size to largest size ($\mathsf{ALL}_{S \rightarrow L}$ problem), and (iv) from largest size to smallest size ($\mathsf{ALL}_{L \rightarrow S}$ problem). Here* size *of a graph G refers to $|V(G)| + |E(G)|$ when $\preccurlyeq$ is subgraph isomorphism and to $|V(G)|$ when $\preccurlyeq$ is induced subgraph isomorphism.*

The *parameter* of the above problems is the size of DB. For some input I, let O be the output set of some finite cardinality N. Then the elements of O, say $o_1, \ldots, o_N$, are listed with: a) *polynomial delay* if the time before printing o_1, the time between printing o_i and o_{i+1} for every $i = 1, \ldots, N-1$, and the time between printing o_N and the termination is bounded by a polynomial of the size of I, b) *incremental polynomial time* if o_1 is printed with polynomial delay, the time between printing o_i and o_{i+1} for every $i = 1, \ldots, N-1$ (resp. the time

[4] $\mathsf{ALL}_{F \rightarrow I}$ stands for 'frequent to infrequent', $\mathsf{ALL}_{I \rightarrow F}$ stands for 'infrequent to frequent', $\mathsf{ALL}_{S \rightarrow L}$ stands for 'small to large' and $\mathsf{ALL}_{L \rightarrow S}$ stands for 'large to small'.

between printing o_N and the termination) is bounded by a polynomial of the combined size of I and the set $\{o_1, \ldots, o_i\}$ (resp. O), c) *output polynomial time* (or *polynomial total time*) if O is printed in time polynomial in the combined size of I and the *entire* output O.

Remark 1. There is an incremental-polynomial-time algorithm for the FCGM (FCIGM) problem if and only if there is an incremental-polynomial time algorithm for $\mathsf{ALL}_{F \to I}$ with (induced) subgraph isomorphism as a pattern matching operator.

4 Mining All (Induced) Subgraphs

4.1 Negative Results

In this section, we provide several negative results regarding complexity of some of the enumeration problems considered in this paper. The first theorem connects the hardness of the frequent subgraph enumeration problem to fixed-parameter tractability of the pattern matching operator (subgraph isomorphism or induced subgraph isomorphism).

Theorem 1. *Let $\mathcal{G}$ be a class of graphs. Let $\preccurlyeq$ be either subgraph isomorphism or induced subgraph isomorphism. If deciding $H \preccurlyeq G$ where $G, H \in \mathcal{G}$ is not fixed-parameter tractable with the parameter $|H|$ then $\mathsf{ALL}_{F \to I}$ and $\mathsf{ALL}_{S \to L}$ cannot be solved in incremental polynomial time.*

However, there are also graph classes with FPT subgraph isomorphism, e.g. planar graphs [4], for which $\mathsf{ALL}_{F \to I}$ cannot be solved in incr.-poly. time.

Theorem 2. *The problem $\mathsf{ALL}_{F \to I}$ cannot be solved in incremental polynomial time for the class $\mathcal{G}$ of planar graphs.*

This theorem is interesting because in Section 4.2, we will see that the problem $\mathsf{ALL}_{L \to S}$ can be solved with polynomial delay for planar graphs.

Even stronger result can be obtained for the problem $\mathsf{ALL}_{I \to F}$.

Theorem 3. *Let $\mathcal{G}$ be a class of graphs. Let $\preccurlyeq$ be either subgraph isomorphism or induced subgraph isomorphism. If deciding $H \preccurlyeq G$ where $G, H \in \mathcal{G}$ is NP-hard then $\mathsf{ALL}_{I \to F}$ cannot be solved in incremental polynomial time (unless $\mathsf{P} = \mathsf{NP}$).*

Using the fact that (induced) subgraph isomorphism is NP-complete even for bounded-treewidth graphs [5] and planar graphs [4], we can obtain the following.

Corollary 1. *The problem $\mathsf{ALL}_{I \to F}$ cannot be solved in incremental polynomial time for the class of planar graphs and for the class of bounded-treewidth graphs.*

Note that Theorem 1 cannot be made as strong as Theorem 3 (i.e. showing that $\mathsf{ALL}_{F \to I}$ cannot be solved in incremental-polynomial time if the pattern matching operator is NP-hard) because the results of Horváth and Ramon from [2] demonstrate that even if the pattern matching operator is NP-hard there can be an incremental-polynomial-time algorithm for mining frequent subgraphs. Theorem 3 shows that we cannot expect such a result for mining *infrequent* subgraphs (i.e. subgraphs with frequency below a threshold).

4.2 Positive Results for $\mathsf{ALL}_{F \to I}$ and $\mathsf{ALL}_{S \to L}$

Before presenting our new results for $\mathsf{ALL}_{L \to S}$ in the next section, we note that there exists the following positive result for frequent graph mining from bounded-treewidth graphs, which was presented in [2].

Theorem 4 (Horváth and Ramon [2], Horváth, Otaki and Ramon [1]). *The FCGM and FCIGM problems can be solved in incremental-polynomial time for the class of bounded-treewidth graphs.*

This result directly translates to a positive result for the problem $\mathsf{ALL}_{F \to I}$ summarized in the following corollary (recall that we have shown in the previous section that $\mathsf{ALL}_{I \to F}$ cannot be solved in incremental-polynomial time for bounded-tree-width graphs) and to a result for the problem $\mathsf{ALL}_{S \to L}$ (this other result follows from the fact that the respective algorithms are level-wise).

Corollary 2. *The problems* $\mathsf{ALL}_{F \to I}$ *and* $\mathsf{ALL}_{S \to L}$ *can be solved in incremental-polynomial time for the class of bounded-treewidth graphs.*

4.3 Positive Results for $\mathsf{ALL}_{L \to S}$

In this section, we describe an algorithm called LargerToSmaller (Algorithm 1) which, when given a class of graphs $\mathcal{G}$ in which isomorphism can be decided in polynomial time, solves the problem $\mathsf{ALL}_{L \to S}$ in incremental-polynomial time. The main employed trick is the observation that for the problem $\mathsf{ALL}_{L \to S}$ it is not necessary to use subgraph isomorphism for computing occurrences of graphs in a graph database.

The algorithm maintains a data structure ALL storing key-value pairs where keys are graphs and values are sets of IDs[5] of graphs in which the given key graph is contained either as a subgraph or as an induced subgraph (depending on whether we are mining subgraphs or induced subgraphs). The data structure provides four functions: $\mathsf{ADD}(K, OCC, ALL)$, $\mathsf{GET}(K, ALL)$, $\mathsf{KEYS}(n, ALL)$, and $\mathsf{DELETE}(n, ALL)$.

The function $\mathsf{ADD}(K, OCC, ALL)$ adds the IDs contained in OCC to the set associated with a key contained in ALL which is isomorphic to K or, if no such key is contained in ALL, the function stores K in ALL and associates OCC with it. If we restrict attention to graphs from a class $\mathcal{G}$ for which a polynomial-time canonization function running in $O(p(|H|))$ exists (where p is a polynomial) then the function $\mathsf{ADD}(K, OCC, ALL)$ can be implemented to run in time $O(p(|K|))$ (we can just store the key graphs as canonical strings, therefore a hashtable with constant-time methods for finding and adding values by their keys can be used). If a polynomial-time canonization function does not exist but graph isomorphism can be decided in time $O(p_{iso}(|K|))$ where p_{iso} is a polynomial then the function $\mathsf{ADD}(K, OCC, ALL)$ can be implemented to run in time $O(|\{K' \in KEYS(ALL) \colon |V(K')| = |V(K)| \text{ and } |E(K')| = |E(K)|\}| \cdot p_{iso}(|K|))$.

[5] Here, IDs are just some identifiers given to the database graphs.

Algorithm 1 LargerToSmaller

Require: database DB of transaction graphs
Ensure: all connected (induced) subgraphs and their occurrences

```
1: let $ALL$ be a data structure for storing graphs and their occurrences (as described in the main
   text).
2: for $G \in DB$ do
3:   ADD($G, \{\mathsf{ID}(G)\}, ALL$)
4: endfor
5: let $m$ be the maximum order[8] of a graph in $DB$.
6: for ($l := m; l > 0;\ l := l - 1$) do
7:   for $H \in \mathsf{KEYS}(l, ALL)$ do
8:     $OCC \leftarrow \mathsf{GET}(H, ALL)$
9:     PRINT($H, OCC$)
10:    for $H' \in \mathsf{REFINE}(H)$ do
11:      if $H'$ is connected then
12:        ADD($H', OCC, ALL$)
13:      endif
14:    endfor
15:  endfor
16:  DELETE($l, ALL$)
17: endfor
```

The function $\mathsf{GET}(K, ALL)$ returns all IDs associated with a key isomorphic to K. The exactly same consideration as for the function ADD apply also for this function.

The function $\mathsf{KEYS}(n, ALL)$ returns a pointer to a linked list containing all key graphs stored in ALL which have size n. Note that since the data structure ALL does not allow deletion of keys, it is easy to maintain such a linked list[6].

Finally, the function $\mathsf{DELETE}(n, ALL)$ removes the pointer[7] to the linked list containing all key graphs of order n stored in ALL.

The algorithm LargerToSmaller fills in the data structure ALL, starting with the largest graphs and proceeding to the smaller ones. When it processes a graph H, it first prints it and the IDs of the graphs associated to it in the data structure ALL, and then it calls the function REFINE which returns all connected subgraphs H' of H which can be obtained from H by removing an edge or an edge and its incedent vertex of degree one (in the case of subgraph mining), or just by removing a vertex and all its incident vertices (in the case of induced subgraph mining). It then associates all occurrences of the graph H with the graphs H' in the datastructure ALL using the function ADD. Since the same graph H' may be produced from different graphs H, the occurrences of H' accumulate and we can prove that when a graph H is printed, the data structure ALL already contains all IDs of graphs in which H is contained.

Theorem 5. *Let $\mathcal{G}$ be a hereditary class of graphs with isomorphism decidable in polynomial time. Given a database DB of graphs from $\mathcal{G}$, the algorithm* Larg-

[6] The reason why the function KEYS does not just return all the key graphs but rather a pointer to the linked list is that if it did otherwise, Algorithm 1 could never run with polynomial delay

[7] Note that we just remove the pointer and do not actually "free" the memory occupied by the graphs. For the practical implementation, we used a programming language with a garbage collector.

erToSmaller *solves the problem* $\mathsf{ALL}_{L\to S}$ *in incremental polynomial time. If the graphs from* $\mathcal{G}$ *also admit a poly-time canonization then the algorithm* LargerToSmaller *solves the problem* $\mathsf{ALL}_{L\to S}$ *with polynomial delay.*

Using the results on complexity of graph canonization for planar [6] and bounded-treewidth graphs [3], we can get the following corollary.

Corollary 3. *The problem* $\mathsf{ALL}_{L\to S}$ *can be solved with polynomial delay for the classes of planar and bounded-treewidth graphs.*

The following theorem asserts that the results for the problem $\mathsf{ALL}_{L\to S}$ are essentially optimal w.r.t. incremental polynomial time.

Theorem 6. *The problem* $\mathsf{ALL}_{L\to S}$ *can be solved in incremental-polynomial time for graphs from a hereditary class* $\mathcal{G}$ *if and only if graph isomorphism can be decided in polynomial-time for graphs from* $\mathcal{G}$*.*

5 Concluding Remarks

Most of the theorems presented here can be generalized for mining under (induced) homeomorphism and minor embedding. Even with a simple implementation of the algorithm for the $\mathsf{ALL}_{L\to S}$ problem, we were able to mine completely about 70% molecules from NCI GI dataset.

Acknowledgement. This work has been supported by ERC Starting Grant 240186 "MiGraNT: Mining Graphs and Networks, a Theory-based approach". The first author is supported by a grant from the Leverhulme Trust (RPG-2014-164).

References

1. T. Horváth, K. Otaki, and J. Ramon. Efficient frequent connected induced subgraph mining in graphs of bounded tree-width. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2013*, pages 622–637, 2013.
2. T. Horváth and J. Ramon. Efficient frequent connected subgraph mining in graphs of bounded tree-width. *Theor. Comput. Sci.*, 411(31-33):2784–2797, 2010.
3. D. Lokshtanov, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. Fixed-parameter tractable canonization and isomorphism test for graphs of bounded treewidth. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014*, pages 186–195, 2014.
4. D. Marx and M. Pilipczuk. Everything you always wanted to know about the parameterized complexity of subgraph isomorphism (but were afraid to ask). In *31st International Symposium on Theoretical Aspects of Computer Science (STACS 2014), STACS 2014*, pages 542–553, 2014.
5. J. Matoušek and R. Thomas. On the complexity of finding iso- and other morphisms for partial k-trees. *Discrete Mathematics*, 108(1-3):343–364, 1992.
6. J. Torán and F. Wagner. The complexity of planar graph isomorphism. *Bulletin of the EATCS*, 97:60–82, 2009.