Frequent Subgraph Mining in Outerplanar Graphs*

Tamás Horváth[1,2], Jan Ramon[3], and Stefan Wrobel[2,1]

[1] Dept. of Computer Science III, University of Bonn, Germany
[2] Fraunhofer IAIS, Sankt Augustin, Germany
[3] Dept. of Computer Science, Katholieke Universiteit Leuven, Belgium

Abstract. In recent years there has been an increased interest in frequent pattern discovery in large databases of graph structured objects. While the frequent connected subgraph mining problem for tree datasets can be solved in incremental polynomial time, it becomes intractable for arbitrary graph databases. Existing approaches have therefore resorted to various heuristic strategies and restrictions of the search space, but have not identified a practically relevant tractable graph class beyond trees. In this paper, we define the class of so called *tenuous outerplanar graphs*, a strict generalization of trees, develop a frequent subgraph mining algorithm for tenuous outerplanar graphs that works in incremental polynomial time, and evaluate the algorithm empirically on the NCI molecular graph dataset.

1 Introduction

The discovery of *frequent patterns* in a database is one of the central tasks considered in data mining. In addition to be interesting in their own right, frequent patterns can also be used as features for predictive data mining tasks (see, e.g., [5]). For a long time, work on frequent pattern discovery has concentrated on relatively simple notions of patterns and elements in the database as they are typically used for the discovery of association rules (simple sets of atomic items). In recent years, however, due to the significance of application areas such as the analysis of chemical molecules or graph structures in the WWW, there has been an increased interest in algorithms that can perform frequent pattern discovery in databases of structured objects such as *trees* or arbitrary *graphs*.

While the frequent pattern problem for trees can be solved in *incremental polynomial time*, i.e., in time polynomial in the *combined size* of the input and the set of frequent tree patterns *so far* computed, the frequent pattern problem for graph structured databases in the general case cannot be solved in *output polynomial time*, i.e., in time polynomial in the *combined size* of the input and the set of *all* frequent patterns. Existing approaches to frequent pattern discovery for graphs have therefore resorted to various heuristic strategies and restrictions of the search space (see, e.g., [4, 5, 8, 17]), but have not identified a practically relevant tractable graph class beyond trees.

In this paper, we define the class of so called *tenuous outerplanar graphs*, which is the class of planar graphs that can be embedded in the plane in such a way that all of its vertices lie on the outer boundary, i.e. can be reached from the outside without crossing any edges, and which have a fixed limit on the number of inside diagonal edges. This

* A longer version of this paper has been accepted to ACM SIGKDD'06.

class of graphs is a strict generalization of trees, and is motivated by the kinds of graphs actually found in practical applications. In fact, in one of the popular graph mining data sets, the NCI data set[4], 94.3% of all elements are tenuous outerplanar graphs. We develop an incremental polynomial time algorithm for enumerating frequent tenuous outerplanar graph patterns.

Our approach is based on a canonical string representation of outerplanar graphs which may be of interest in itself, and further algorithmic components for mining frequent biconnected outerplanar graphs and candidate generation in an Apriori style algorithm. To map a pattern to graphs in the database, we define a special notion of *block and bridge preserving* (BBP) subgraph isomorphism, which is motivated by application and complexity considerations, and show that it is decidable in polynomial time for outerplanar graphs. We note that for trees, which form a special class of outerplanar graphs, BBP subgraph isomorphism is equivalent to subtree isomorphism. Thus, BBP subgraph isomorphism generalizes subtree isomorphism to graphs, but is at the same time more specific than subgraph isomorphism. Since in many applications, subgraph isomorphism is a non-adequate matching operator (e.g., when pattern matching is required to preserve certain type of fragments in molecules), by considering BBP subgraph isomorphism we take a first step towards studying the frequent graph mining problem w.r.t. non-standard matching operators as well. Beside complexity results, we present also empirical results which show that the favorable theoretical properties of the algorithm and pattern class also translate into efficient practical performance.

The paper is organized as follows. In Sections 2 and 3, we define the necessary notions and the problem setting for this work, respectively. Section 4 describes our algorithm for mining frequent tenuous outerplanar graphs. Section 5 contains our experimental evaluation and finally, Section 6 concludes and discusses some open problems. Due to space limitations, proofs are omitted in this short version.

2 Preliminaries

We recall some notions related to graphs [7]. An *undirected graph* is a pair (V, E), where $V \neq \emptyset$ 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 quadruple (V, E, Σ, λ), where (V, E) is an undirected graph, $\Sigma \neq \emptyset$ is a finite set of *labels* associated with some total order, and $\lambda : V \cup E \to \Sigma$ is a function assigning a label to each element of $V \cup E$. Unless otherwise stated, in this paper by graphs we always mean *labeled undirected* graphs and denote the set of vertices, the set of edges, and the labeling function of a graph G by $V(G)$, $E(G)$, and λ_G, respectively. Let G and G' be graphs. G' is a *subgraph* of G, if $V(G') \subseteq V(G)$, $E(G') \subseteq E(G)$, and $\lambda_{G'}(x) = \lambda_G(x)$ for every $x \in V(G') \cup E(G')$. For a vertex $v \in V(G)$, $N(v)$ denotes the set of vertices of G connected by an edge with v.

A graph G is *connected* if there is a path between any pair of its vertices; it is *biconnected* if for any two vertices u and v of G, there is a simple cycle containing u and v. A *block* (or *biconnected component*) of a graph is a maximal subgraph that is biconnected. Edges not belonging to blocks are called *bridges*. The definitions imply

[4] http://cactus.nci.nih.gov/

that the blocks of a graph are pairwise edge disjoint and that the set of bridges forms a forest. For the set of blocks and the forest formed by the bridges of a graph G it holds that their cardinalities are bounded by $|V(G)|$ and they can be enumerated in time $O(|V(G)| + |E(G)|)$ [16].

Let G_1 and G_2 be graphs. G_1 and G_2 are *isomorphic*, denoted $G_1 \simeq G_2$, if there is a *bijection* $\varphi : V(G_1) \to V(G_2)$ such that (i) $\{u, v\} \in E(G_1)$ iff $\{\varphi(u), \varphi(v)\} \in E(G_2)$, (ii) $\lambda_{G_1}(u) = \lambda_{G_2}(\varphi(u))$, (iii) and if $\{u, v\} \in E(G_1)$ then $\lambda_{G_1}(\{u, v\}) = \lambda_{G_2}(\{\varphi(u), \varphi(v)\})$ hold for every $u, v \in V(G_1)$. In this paper, two graphs are considered to be the same if they are isomorphic. G_1 is *subgraph isomorphic* to G_2 if G_1 is isomorphic to a subgraph of G_2. Deciding whether a graph is subgraph isomorphic to another graph is NP-complete, as it generalizes e.g. the Hamiltonian path problem.

Outerplanar Graphs Informally, a graph is *planar* if it can be drawn in the plane in such a way that no two edges intersect except at a vertex in common. An *outerplanar graph* is a planar graph which can be embedded in the plane in such a way that all of its vertices lie on the boundary of the outer face. Throughout this work we consider connected outerplanar graphs and denote the set of connected outerplanar graphs over an alphabet Σ by $\mathcal{O}_\Sigma$. Clearly, trees are outerplanar graphs and hence, a graph is outerplanar iff each of its blocks is outerplanar [7]. Furthermore, as the blocks of a graph can be computed in linear time [16] and outerplanarity of a block can be decided also in linear time [10, 12], one can decide in linear time whether a graph is outerplanar.

A biconnected outerplanar graph G with n vertices contains at most $2n - 3$ edges and has a unique Hamiltonian cycle which bounds the outer face of a planar embedding of G [7]. This unique Hamiltonian cycle can be computed efficiently [10]. Thus, G can be considered as an n-polygon with at most $n - 3$ non-crossing diagonals. Below we state a bound for the number of cycles of G. Due to space limitation, we omit the proof.

Proposition 1 *A biconnected outerplanar graph with d diagonals has at most 2^{d+1} cycles.*

Given outerplanar graphs G and H, deciding whether H is subgraph isomorphic to G is an NP-complete problem. This follows from the fact that outerplanar graphs generalize forests and deciding whether a forest is subgraph isomorphic to a tree is NP-complete [6]. The following stronger negative result is shown in [15].

Theorem 2 *Deciding whether a connected outerplanar graph H is subgraph isomorphic to a biconnected outerplanar graph G is NP-complete.*

If, however, H is also biconnected, the following positive result holds [10].

Theorem 3 *Let G, H be biconnected outerplanar graphs. Then one can decide in time $O\left(|V(H)| \cdot |V(G)|^2\right)$ whether H is subgraph isomorphic to G.*

For the special case of trees, the following positive result holds [11].[5]

Theorem 4 *The problem whether a tree H is subgraph isomorphic to a tree G can be decided in time $O\left(|V(H)|^{1.5} \cdot |V(G)|\right)$.*

[5] The bound in Theorem 4 is improved by a log factor in [14]. For the sake of simplicity, we generalize the algorithm in [11] to outerplanar graphs in the long version of this paper. We note that the complexity of our algorithm can also be improved using the idea of [14].

3 The Problem Setting

In this section we define the frequent subgraph mining problem for a practically relevant class of outerplanar graphs with respect to a matching operator that preserves the pattern graph's bridge and block structure. To define the mining problem, we need the notions of tenuous outerplanar graphs and BBP subgraph isomorphism.

Tenuous Outerplanar Graphs Let $d \geq 0$ be some integer. A *d-tenuous* outerplanar graph G is an outerplanar graph such that each block of G has at most d diagonals. For an alphabet Σ and integer $d \geq 0$, $\mathcal{O}_\Sigma^d$ denotes the set of connected d-tenuous outerplanar graphs labeled by the elements of Σ. The class of d-tenuous outerplanar graphs forms a practically relevant graph class e.g. in chemoinformatics. As an example, out of the 250251 pharmacological molecules in the NCI dataset, 236180 (i.e., 94.3%) compounds have an outerplanar molecular graph. Furthermore, among the outerplanar compounds, there is no molecular graph having a block with more than 11 diagonals. In fact, there is only one compound containing a block with 11 diagonals; 236083 (i.e., 99.99%) compounds among the outerplanar graphs have at most 5 diagonals per block.

BBP Subgraph Isomorphism We continue our problem definition by introducing a matching operator between outerplanar graphs. Let $G, H \in \mathcal{O}_\Sigma$. A *bridge and block preserving* (BBP) subgraph isomorphism from H to G, denoted $H \preccurlyeq_{BBP} G$, is a subgraph isomorphism from H to G mapping (i) the set of bridges of H to the set of bridges of G and (ii) different blocks of H to different blocks of G. Notice that for trees, which are special outerplanar graphs (i.e., block-free), BBP subgraph isomorphism is equivalent to the ordinary subtree isomorphism. Thus, BBP subgraph isomorphism can be considered as a generalization of subtree isomorphism to outerplanar graphs which is more specific than ordinary subgraph isomorphism.

Besides complexity reasons raised by Theorem 2, the use of BBP subgraph isomorphism as matching operator is motivated by recent results in chemoinformatics which indicate that more powerful predictors can be obtained by considering matching operators that map certain fragments of the pattern molecule to certain fragments of the target molecule. One natural step towards this direction is to require that only ring structures (i.e., blocks) can be mapped to ring structures and that edge disjoint ring structures are mapped to edge disjoint ring structures.

The FTOSM Problem Using the above notions, we define the *frequent d-tenuous outerplanar subgraph mining problem* (FTOSM) as follows: *Given* (i) an alphabet Σ, (ii) a finite set $\mathcal{D} \subseteq \mathcal{O}_\Sigma^d$ of *transactions* for some integer $d \geq 0$, and (iii) an integer threshold $t > 0$, *enumerate* the set of all connected d-tenuous outerplanar graphs in $\mathcal{O}_\Sigma^d$ that match at least t graphs in $\mathcal{D}$ w.r.t. BBP subgraph isomorphism, i.e., enumerate the set

$$\mathcal{F}_{\Sigma,d}^t(\mathcal{D}) = \{H \in \mathcal{O}_\Sigma^d : \Pi_t(\mathcal{D}, H)\} \ , \tag{1}$$

where $\Pi_t(\mathcal{D}, H)$ is the *frequency property* defined by

$$\Pi_t(\mathcal{D}, H) = |\{G \in \mathcal{D} : H \preccurlyeq_{BBP} G\}| \geq t \ . \tag{2}$$

By definition, $\mathcal{F}^t_{\Sigma,d}(\mathcal{D})$ does not contain isomorphic graphs. Furthermore, it is closed downwards w.r.t. BBP subgraph isomorphism, i.e., $G_1 \in \mathcal{F}^t_{\Sigma,d}(\mathcal{D})$ whenever $G_2 \in \mathcal{F}^t_{\Sigma,d}(\mathcal{D})$ and $G_1 \preccurlyeq_{BBP} G_2$. Given $\mathcal{D}$ and t, we call a graph H satisfying (2) *t-frequent*.

The *parameters* of the FTOSM problem are the cardinality of the transaction dataset (i.e., $|\mathcal{D}|$) and the size of the largest graph in $\mathcal{D}$ (i.e., $\max\{|V(G)| : G \in \mathcal{D}\}$). Since d is usually small, it is assumed to be a *constant*. Note that the cardinality of $\mathcal{F}^t_{\Sigma,d}(\mathcal{D})$ can be exponential in the above parameters of $\mathcal{D}$. Clearly, in such cases it is impossible to enumerate $\mathcal{F}^t_{\Sigma,d}(\mathcal{D})$ in time polynomial in the parameters of $\mathcal{D}$. We therefore ask whether the FTOSM problem can be solved in *incremental polynomial time* (see, e.g., [9]), that is, whether there exists an enumeration algorithm listing the first k elements of $\mathcal{F}^t_{\Sigma,d}(\mathcal{D})$ in time polynomial in the *combined size* of $\mathcal{D}$ and the set of these k elements for every $k = 1, \ldots, |\mathcal{F}^t_{\Sigma,d}(\mathcal{D})|$.

We note that in the literature (see, e.g., [9]) one usually considers also the notion of *output polynomial time* (or *polynomial total time*) complexity for enumeration algorithms. Algorithms in this more liberal class are required to enumerate a set S in the combined size of the input and the *entire* set S. Thus, in contrast to incremental polynomial time, an output polynomial time algorithm may have in worst-case a delay time exponential in the size of the input before printing the kth element for some $k \geq 1$.

Although several algorithms mining frequent connected subgraphs from datasets of arbitrary graphs w.r.t. subgraph isomorphism have demonstrated their performance empirically, we note that this general problem cannot be solved in output polynomial time, unless P = NP. On the other hand, the frequent graph mining problem is solvable in incremental polynomial time when the graphs in the dataset are restricted to forests and the patterns to trees. This follows e.g. from the results in [3]. Since tenuous outerplanar graphs form a practically relevant graph class that naturally generalizes trees, by considering the FTOSM problem we take a step towards going beyond trees in frequent subgraph mining.

4 The Mining Algorithm

In this section we present Algorithm 1, an Apriori-like [1] algorithm, that solves the FTOSM problem in incremental polynomial time. For a set $\mathcal{D} \subseteq \mathcal{O}^d_\Sigma$ and integer $t \geq 0$, the algorithm computes iteratively the set of t-frequent k-patterns from the set of t-frequent $(k-1)$-patterns. A *k-pattern* is a graph $G \in \mathcal{O}^d_\Sigma$ such that the sum of the number of blocks of G and the number of vertices of G not belonging to any block is k.

In step 1 of the algorithm, we first compute the set of t-frequent 1-patterns, that is, the set of t-frequent graphs consisting of either a single vertex or a single block. The first set, denoted by $\mathcal{F}_v$ in step 1, can be computed in linear time. The second set, denoted $\mathcal{F}_b$, can be computed in time polynomial in the parameters of $\mathcal{D}$; an efficient Apriori-based algorithm for this problem is presented in Section 4.2.

In step 2 of the algorithm, we then compute the set of t-frequent 2-patterns, i.e., the set of graphs in $\mathcal{O}^d_\Sigma$ consisting of either (1) a single edge or (2) two blocks having a common vertex or (3) a block and a bridge edge having a common vertex. We denote the corresponding three sets in step 2 by $\mathcal{F}_e$, $\mathcal{F}_{bb}$, and $\mathcal{F}_{be}$, respectively. In the definitions of $\mathcal{F}_{bb}$ and $\mathcal{F}_{be}$, $G_1 \bowtie G_2$ denotes the set of graphs that can be obtained from the

Algorithm 1 FREQUENTOUTERPLANARGRAPHS

Require: $\mathcal{D} \subseteq \mathcal{O}_\Sigma^d$ for some alphabet Σ and integer $d \geq 0$, and integer $t > 0$
Ensure: $\mathcal{F}_{\Sigma,d}^t(\mathcal{D})$ defined in Eq. (1)

1: $\mathcal{L}_1 = \mathcal{F}_v \cup \mathcal{F}_b$, where

$$\mathcal{F}_v = \{H \in \mathcal{O}_\Sigma : |V(H)| = 1 \wedge \Pi_t(\mathcal{D}, H)\}$$
$$\mathcal{F}_b = \{H \in \mathcal{O}_\Sigma^d : H \text{ is biconnected} \wedge \Pi_t(\mathcal{D}, H)\}$$

2: $\mathcal{L}_2 = \mathcal{F}_e \cup \mathcal{F}_{bb} \cup \mathcal{F}_{be}$, where

$$\mathcal{F}_e = \{H \in \mathcal{O}_\Sigma : |E(H)| = 1 \wedge \Pi_t(\mathcal{D}, H)\}$$
$$\mathcal{F}_{bb} = \{H \in G_1 \bowtie G_2 : G_1, G_2 \in \mathcal{F}_b \wedge \Pi_t(\mathcal{D}, H)\}$$
$$\mathcal{F}_{be} = \{H \in G_1 \bowtie G_2 : G_1 \in \mathcal{F}_b \wedge G_2 \in \mathcal{F}_e \wedge \Pi_t(\mathcal{D}, H)\}$$

3: $k = 2$
4: **while** $\mathcal{L}_k \neq \emptyset$ **do**
5: $\quad k = k + 1$
6: $\quad \mathcal{C}_k = \text{GENERATECANDIDATES}(\mathcal{L}_{k-1})$
7: $\quad \mathcal{L}_k = \{H \in \mathcal{C}_k : \Pi_t(\mathcal{D}, H)\}$
8: **endwhile**
9: **return** $\cup_{i=1}^k \mathcal{L}_i$

union of G_1 and G_2 by contracting[6] a vertex from G_1 with a vertex from G_2 that have the same label. Clearly, $G_1 \bowtie G_2 \subseteq \mathcal{O}_\Sigma^d$ for every $G_1, G_2 \in \mathcal{O}_\Sigma^d$. The set $\mathcal{F}_e$ of t-frequent edges can be computed in linear time. Since the cardinalities of both $\mathcal{F}_{bb}$ and $\mathcal{F}_{be}$ are polynomial in the parameters of $\mathcal{D}$, and BBP subgraph isomorphism between outerplanar graphs can be decided in polynomial time by the result of Section 4.4 below, it follows that both $\mathcal{F}_{bb}$ and $\mathcal{F}_{be}$ and hence, the set $\mathcal{L}_2$ of t-frequent 2-patterns can be computed in time polynomial in the parameters of $\mathcal{D}$.

In the loop 4–8, we compute the set of t-frequent k-patterns for $k \geq 3$ in a way similar to the Apriori algorithm [1]. The crucial steps of the loop are the generation of candidate k-patterns from the set of t-frequent $(k-1)$-patterns (step 6) and the decision of t-frequency of the candidate patterns (step 7). In Sections 4.3 and 4.4 below we describe these steps in detail.

Putting together the results given in Theorems 7 – 10 stated in Sections 4.1 – 4.4, respectively, we can formulate the main result of this paper:

Theorem 5 *Algorithm 1 is correct and solves the FTOSM problem in incremental polynomial time.*

Before going into the technical details in Sections 4.1 – 4.4, we first describe a transformation on outerplanar graphs by means of block contraction that is used in different steps of the mining algorithm. More precisely, for a graph $G \in \mathcal{O}_\Sigma$, let $\widetilde{G}$ denote the

[6] The contraction of the vertices u and v of a graph G is the graph obtained from G by introducing a new vertex w, connecting w with every vertex in $N(u) \cup N(v)$, and removing u and v, as well as the edges adjacent to them.

graph over the alphabet $\Sigma \cup \{\#\}$ derived from G by the following transformation: For each block B in G, (i) introduce a new vertex v_B and label it by $\#$, (ii) remove each edge belonging to B, and (iii) for every vertex v of B, connect v with v_B by an edge labeled by $\#$, if v is adjacent to a bridge or to another block of G; otherwise remove v. In the following proposition we state some basic properties of $\widetilde{G}$.

Proposition 6 *Let $G \in \mathcal{O}_\Sigma$. Then*

(i) $|V(\widetilde{G})| = 1$ *iff* $|V(G)| = 1$ *or G is biconnected,*
(ii) for every $e \in E(\widetilde{G})$, at most one vertex of e is labeled by $\#$, and
(iii) $\widetilde{G}$ is a free tree.

Since $\widetilde{G}$ is a tree, we call it the *block and bridge tree (BB-tree)* of G.

4.1 Canonical String Representation

One time consuming step of mining frequent d-tenuous outerplanar graphs is to test whether a particular graph $H \in \mathcal{O}_\Sigma^d$ belongs to some subset S of $\mathcal{O}_\Sigma^d$. To apply advanced data structures that allow fast search in large subsets of $\mathcal{O}_\Sigma^d$, we need to define a total order on $\mathcal{O}_\Sigma^d$. Similarly to many other frequent graph mining algorithms, we solve this problem by assigning a *canonical string* to each element of $\mathcal{O}_\Sigma$ such that (i) two graphs have the same canonical string iff they are isomorphic and (ii) for every $G \in \mathcal{O}_\Sigma$, the canonical string of G can be computed efficiently. Using some canonical string representation satisfying the above properties, a total order on $\mathcal{O}_\Sigma$ and thus, on $\mathcal{O}_\Sigma^d$ as well, can be defined by some total order (e.g. lexicographic) on the set of strings assigned to the elements of $\mathcal{O}_\Sigma$. Furthermore, property (i) allows one to decide isomorphism between two outerplanar graphs by comparing their canonical strings.

Although the canonical string representation for outerplanar graphs may be of some interest in itself, due to space limitations we omit its definition which is based on the BB-tree $\widetilde{G}$ of G. By (iii) of Proposition 6, $\widetilde{G}$ is a free tree. Utilizing this property, we can generalize the depth-first canonical representation for free trees (see, e.g., [2]) to outerplanar graphs, and state the following result:

Theorem 7 *A canonical string representation of a graph in $\mathcal{O}_\Sigma$ with n vertices can be computed in time $O\left(n^2 \log n\right)$.*

4.2 Mining Frequent Biconnected Graphs

In this section we present Algorithm 2, an Apriori-like algorithm, that computes the set $\mathcal{F}_b$ of t-frequent d-tenuous biconnected graphs used in step 1 of Algorithm 1. Since d is constant, Algorithm 2 runs in time polynomial in the parameters of $\mathcal{D}$.

In step 1 of Algorithm 2, we first compute the set $\mathcal{L}_0$ of t-frequent cycles as follows: We list the cycles of G for every $G \in \mathcal{D}$ and count their frequencies. Proposition 1 in Section 2 implies that the number of cycles of a d-tenuous outerplanar graph G is bounded by $O\left(|V(G)|\right)$ if d is assumed to be constant. Furthermore, from [13, 16] it follows that the cycles of a graph can be listed with linear delay. Since isomorphism

Algorithm 2 FREQUENTBICONNECTEDGRAPHS

Require: $\mathcal{D} \subseteq \mathcal{O}_\Sigma^d$ for some alphabet Σ and integer $d \geq 0$, and integer $t > 0$
Ensure: $\mathcal{F}_b$ defined in step 1 of Algorithm 1

1: **let** $\mathcal{L}_0 \subseteq \mathcal{O}_\Sigma^0$ be the set of t-frequent cycles in $\mathcal{D}$
2: **for** $k = 1$ **to** d **do**
3: **let** $\mathcal{C}_k \subseteq \mathcal{O}_\Sigma^k \setminus \mathcal{O}_\Sigma^{k-1}$ be the set of biconnected graphs H such that $H \ominus \Delta \in \mathcal{L}_{k-1}$ for every diagonal Δ of H
4: $\mathcal{L}_k = \{H \in \mathcal{C}_k : \Pi_t(\mathcal{D}, H)\}$
5: **endfor**
6: **return** $\bigcup_{k=0}^{d} \mathcal{L}_k$

between cycles can be decided efficiently, these results together imply that $\mathcal{L}_0$ can be computed in time polynomial in the parameters of $\mathcal{D}$.

In loop 2–5 of Algorithm 2, we compute the sets of t-frequent biconnected graphs containing k diagonals for every $k = 1, \ldots, d$. In particular, in step 3 we compute the set $\mathcal{C}_k$ of candidate biconnected graphs $H \in \mathcal{O}_\Sigma^k$ satisfying the following conditions: H has exactly k diagonals and the removal of any diagonal from H, denoted by $\ominus$ in step 3, results in a t-frequent biconnected graph. Putting the above results together, we can state the following theorem. (We omit the proof in this short version.)

Theorem 8 *Algorithm 2 is correct and computes the set of t-frequent d-tenuous biconnected outerplanar graphs in time polynomial in the parameters of $\mathcal{D}$.*

4.3 Candidate Generation

In step 6 of Algorithm 1, we generate the set of candidate k-patterns. In this section we give Algorithm 3, a generalization of the candidate generation algorithm for free trees described in [3], that computes the set of candidate k-patterns from the set of frequent $(k-1)$-patterns. Applying the candidate generation principle of the Apriori algorithm [1], each candidate is obtained by joining two frequent $(k-1)$-patterns that have an isomorphic $(k-2)$-pattern core.

In the outer loop 2–12 of the algorithm, we consider each possible pair G_1, G_2 of frequent $(k-1)$-patterns, and in loop 3–11, each pair g_1 and g_2 of *leaf* subgraphs of G_1 and G_2, respectively. By a leaf subgraph of a k-pattern H for $k \geq 2$ we mean the subgraph of H represented by a leaf of the BB-tree $\widetilde{H}$. If G_1 and G_2 are the same graphs then, for completeness, we consider also the case when g_1 and g_2 are isomorphic leaf subgraphs. We remove g_1 and g_2 from G_1 and G_2, respectively, denoted by $\ominus$ in the algorithm, and check whether the obtained graphs G_1' and G_2' are isomorphic (step 4). The removal of a biconnected component means the deletion of each of its edges and vertices except the distinguished vertex which is adjacent to a bridge or to another block.

If G_1' and G_2' are isomorphic then we consider every leaf subgraph g_1' of G_1' (loop 5–10) and check whether g_2 can be attached to g_1' in G_1 consistently with G_2 (step 6). More precisely, let g_2' be a block or a vertex not belonging to a block in G_2 such that g_2 is hanging from g_2', i.e., the only edge adjacent to g_2 is adjacent also to g_2'. We say that g_2 can be attached to g_1' in G_1 consistently with G_2 if g_1' is isomorphic to g_2'. Thus, if

Algorithm 3 GENERATECANDIDATES

Require: set $\mathcal{L}_{k-1}$ of frequent $k-1$-patterns for some $k > 2$
Ensure: set $\mathcal{C}_k$ of candidate k-patterns

1: $\mathcal{C}_k = \emptyset$
2: **forall** $G_1, G_2 \in \mathcal{L}_{k-1}$ **do**
3: **forall** $g_1 \in \text{Leaf}(G_1)$ and $g_2 \in \text{Leaf}(G_2)$ **do**
4: **if** $G_1 \ominus g_1 \simeq G_2 \ominus g_2$ **then**
5: **forall** $g_1' \in \text{Leaf}(G_1 \ominus g_1)$ **do**
6: **if** g_2 is attachable to g_1' consistently with G_2 **then**
7: attach g_2 in G_1 to g_1' consistently with G_2 and denote the obtained graph by C
8: **if** g_1, g_2 have the top two string encodings in C, $C \notin C_k$, and
$C \ominus g \in \mathcal{L}_{k-1}$ for every $g \in \text{Leaf}(C)$
9: **then** add C to $\mathcal{C}_k$
10: **endfor**
11: **endfor**
12: **endfor**
13: **return** $\mathcal{C}_k$

the condition in step 6 holds then we attach g_2 to g_1' consistently with G_2 and denote the obtained graph by C (step 7).

Notice that C can be generated in many different ways, depending on the particular choice of g_1 and g_2. To reduce the amount of unnecessary computation, we consider only those pairs which are among the top leaf subgraphs of C, i.e., which have the top two string encodings w.r.t. a center of $\tilde{C}$. By definition, a vertex representing a leaf subgraph of C is always a leaf in $\tilde{C}$. If this condition holds then we add C to the set of candidates in step 9 if for every leaf subgraph g of C, the $(k-1)$-pattern obtained from C by removing g is frequent (see step 8). We omit the proof of the following theorem.

Theorem 9 *Let $\mathcal{C}_k$ be the output of Algorithm 3 and $\mathcal{L}_k$ the set of frequent k-patterns for any $k > 2$. Then $\mathcal{L}_k \subseteq \mathcal{C}_k$, the cardinality of $\mathcal{C}_k$ is polynomial in the cardinality of $\mathcal{L}_{k-1}$, and $\mathcal{C}_k$ can be computed in time polynomial in the size of $\mathcal{L}_{k-1}$.*

4.4 BBP Subgraph Isomorphism

Algorithms 1 and 2 contain the steps of deciding whether a candidate pattern $H \in \mathcal{O}_\Sigma^d$ is t-frequent, i.e., whether it is BBP subgraph isomorphic to at least t graphs in $\mathcal{D}$. While subgraph isomorphism between outerplanar graphs is NP-complete even for very restricted cases (see Theorem 2), Theorem 10, the main result of this section, states that BBP subgraph isomorphism can be decided efficiently between outerplanar graphs if the pattern graph H is connected. The connectivity is necessary, as otherwise the problem would generalize the NP-complete subforest isomorphism problem [6]. We note that the result of Theorem 10 generalizes the positive result on subtree isomorphism given in Theorem 4 and may thus be of some interest in itself.

Theorem 10 *Let $G, H \in \mathcal{O}_\Sigma$ such that H is connected. Then $H \preccurlyeq_{BBP} G$ can be decided in polynomial time.*

Table 1. Number of patterns (#C), number of frequent patterns (#FP), and runtime in seconds for candidate generation and evaluation (T) with frequency thresholds 10%, 5%, 2%, and 1%

size	10%			5%			2%			1%		
(k)	#C	#FP	T	#C	#FP	T	#C	#FP	T	#C	#FP	T
1	86	7	107	144	11	169	582	25	380	2196	55	824
2	74	16	446	216	24	570	1332	61	1118	6208	174	2554
3	139	41	1133	234	74	1393	510	170	2123	1516	659	5653
4	133	77	1232	266	154	2038	642	356	4079	2554	1776	11899
5	139	91	1071	319	222	2268	909	644	5603	4550	3886	20411
6	107	72	754	332	252	1847	1212	918	6105	7314	6490	28811
7	61	41	472	295	195	1168	1266	990	4964	10165	9058	34967
8	37	25	354	182	137	741	1086	893	3384	11479	10396	36391
9	20	13	205	137	116	602	956	803	2282	11129	10194	31721
10	8	5	130	131	119	594	828	700	1635	9370	8623	23412
11	0	0	0	131	117	565	697	604	1360	7276	6818	15530
12	0	0	0	115	107	536	707	665	1483	5533	5184	9345
13	0	0	0	78	64	412	1027	1022	2017	4395	4145	5252
14	0	0	0	27	21	250	1702	1700	2858	4303	4194	3707
15	0	0	0	4	3	89	2725	2715	3957	5422	5376	4089

Due to space limitations, we omit the proof of the above theorem. We only note that the algorithm first computes the BB-trees of the input graphs G and H, and then combines the subgraph isomorphism algorithms between labeled trees (generalization of [11]) and labeled biconnected outerplanar graphs (generalization of [10]).

5 Experimental Evaluation

In our experiments, we used the NCI dataset consisting of 250251 chemical compounds. For our work, it was important to recognize that 236180 (i.e., 94.3%) of these compounds have outerplanar molecular graph. Thus, outerplanar graphs form a practically relevant class of graphs. Among the outerplanar molecular graphs, there are 21963 trees (i.e., 8.8% of the outerplanar subset). In the experiments, we have removed the non-outerplanar graphs from the dataset. Altogether, the outerplanar molecules contain 423378 blocks, with up to 11 diagonals per block. However, 236083 (i.e., 99.99%) of the outerplanar molecular graphs have at most 5 diagonals per block. This empirical observation validates our approach to assume the number of diagonals to be constant.

The database contains a wide variety of structures, and a low relative frequency threshold is needed to mine a significant number of patterns. E.g. though there are 15426 pairwise non-isomorphic cycles in the database, only a few of them are really frequent; the only one above 10% is the benzene ring with frequency 66%.

Our results are given in Table 1. It shows the number of candidate (#C) and frequent (#FP) k-patterns discovered for $k = 1, \ldots, 15$, as well as the runtime (T) in seconds for the computation and evaluation of the candidates using the frequency thresholds 10%, 5%, 2% and 1%. As expected, the number and the size of the discovered patterns is much larger when the frequency threshold is lower. Even though the embeddings of

$(k-1)$ patterns are computed (again) in level k, the time needed to complete one level does not necessarily increase with k. It is interesting to note that after the number of frequent k-patterns drops a bit when k gets larger then 8, this number again increases when k exceeds 12, and the number of frequent patterns gets close to the number of candidate patterns. This is because this particular dataset contains large subsets with molecules sharing large biconnected structures (such as the HIV active substance dataset). The time needed for candidate generation is always smaller than 1% of the total time. The time needed for coverage testing per pattern depends on how much structure these patterns share. If the number of patterns is large, the time needed per pattern is usually lower.

One can make several conclusions. First, our algorithm can mine an expressive class of molecular patterns from a relatively large database. Although the presented experiments happened entirely in memory (taking about 600Mb), our approach does not depend on storing intermediate results in memory between the different passes over the database. This means that we could also perform this algorithm with a database on disk. In our application e.g., this would bring an overhead of about 15 seconds per pass over the database. Second, we can conclude that the complexity of the coverage testing scales well as the pattern size grows, as predicted by theory. In this application, due to the implementation exploiting shared structure among patterns, the time needed for evaluation per pattern does not even depend in a clear systematic way on the pattern size.

6 Conclusion and Open Problems

We have defined the FTOSM problem motivated by chemical datasets and presented an Apriori-based algorithm solving this enumeration problem in incremental-polynomial time. To the best of our knowledge, no fragment of the frequent subgraph mining problem *beyond trees* has so far been identified, for which the problem can be solved in incremental polynomial time. Our algorithm is based on a canonical string representation of outerplanar graphs and further algorithmic components for mining frequent biconnected outerplanar graphs and candidate generation in an Apriori style algorithm. Motivated by application and complexity considerations, we introduced a special kind of subgraph isomorphism which generalizes subtree isomorphism but is at the same time more specific than ordinary subgraph isomorphism, and which is decidable in polynomial time for outerplanar graphs. We presented also empirical results with a large dataset indicating the effective practical performance of our algorithm. We believe that the identification of tractable practical fragments of the frequent subgraph mining problem is an important challenge for the data mining community.

Besides working on optimization of the algorithm, e.g., on improving the time complexity of the coverage testing, it is natural to ask whether the positive result of this paper can be generalized to arbitrary outerplanar graphs. Notice that our algorithm exploits the constant bound on the number of diagonals only in the computation of the set $\mathcal{F}_b$ of frequent biconnected graphs in step 1 of Algorithm 1. Therefore, to generalize the result of this paper to arbitrary outerplanar graphs, it is sufficient to consider the following special problem: *Given* a finite set $\mathcal{D} \subseteq \mathcal{O}_\Sigma$ of biconnected outerplanar graphs and a

non-negative integer t, compute the set of t-frequent patterns in $\mathcal{D}$ w.r.t. BBP subgraph isomorphism. Notice that this problem definition implicitly requires t-frequent patterns to be biconnected because by definition, there is no BBP subgraph isomorphism from a non-biconnected graph to a biconnected outerplanar graph. We do not know whether this special problem can be solved in incremental or at least in output polynomial time.

Acknowledgments

Tamás Horváth and Stefan Wrobel were partially supported by the DFG project (WR 40/2-1) *Hybride Methoden und Systemarchitekturen für heterogene Informationsräume*. Jan Ramon is a post-doctoral fellow of the Fund for Scientific Research (FWO) of Flanders.

References

1. R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, pp. 307–328. AAAI/MIT Press, 1996.
2. Y. Chi, R. R. Muntz, S. Nijssen, and J. N. Kok. Frequent subtree mining - An overview. *Fundamenta Informaticae*, 66(1-2):161–198, 2005.
3. Y. Chi, Y. Yang, and R. R. Muntz. Canonical forms for labelled trees and their applications in frequent subtree mining. *Knowledge and Information Systems*, 8(2):203–234, 2005.
4. D. J. Cook and L. B. Holder. Substructure discovery using minimum description length and background knowledge. *J. of Artificial Intelligence Research*, 1:231–255, 1994.
5. M. Deshpande, M. Kuramochi, N. Wale, and G. Karypis. Frequent substructure-based approaches for classifying chemical compounds. *IEEE Transactions on Knowledge and Data Engineering*, 17(8):1036–1050, 2005.
6. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to NP-Completeness*. Freeman, San Francisco, CA, 1979.
7. F. Harary. *Graph Theory*. Addison–Wesley, Reading, Massachusetts, 1971.
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. D. S. Johnson, M. Yannakakis, and C. H. Papadimitriou. On generating all maximal independent sets. *Information Processing Letters*, 27(3):119–123, 1988.
10. A. Lingas. Subgraph isomorphism for biconnected outerplanar graphs in cubic time. *Theoretical Computer Science*, 63:295–302, 1989.
11. D. W. Matula. Subtree isomorphism in $O(n^{\frac{5}{2}})$. *Annals of Discrete Mathematics*, 2:91–106, 1978.
12. S. L. Mitchell. Linear algorithms to recognize outerplanar and maximal outerplanar graphs. *Information Processing Letters*, 9(5):229–232, 1979.
13. R. C. Read and R. E. Tarjan. Bounds on backtrack algorithms for listing cycles, paths, and spanning trees. *Networks*, 5(3):237–252, 1975.
14. R. Shamir and D. Tsur. Faster subtree isomorphism. *J. of Algorithms*, 33(2):267–280, 1999.
15. M. M. Sysło. The subgraph isomorphism problem for outerplanar graphs. *Theoretical Computer Science*, 17:91–97, 1982.
16. R. E. Tarjan. Depth first search and linear graph algorithms. *SIAM J. on Computing*, 1(2):146–160, 1972.
17. X. Yan and J. Han. gSpan: Graph-based substructure pattern mining. In *Proc. of the 2002 IEEE Int. Conference on Data Mining (ICDM)*, pp. 721–724, IEEE Computer Society, 2002.