Frequent Subgraph Mining in Outerplanar Graphs

Tamás Horváth
Dept. of Computer Science III
University of Bonn and
Fraunhofer Institute IAIS
Sankt Augustin, Germany
tamas.horvath@iais.fraunhofer.de

Jan Ramon
Dept. of Computer Science
Katholieke Universiteit
Leuven, Belgium
janr@cs.kuleuven.be

Stefan Wrobel
Fraunhofer Institute IAIS and
Dept. of Computer Science III
University of Bonn
Sankt Augustin, Germany
stefan.wrobel@iais.fraunhofer.de

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.

Categories and Subject Descriptors

F.2.2 [**Nonnumerical Algorithms and Problems**]: Pattern matching; H.2.8 [**Database Applications**]: Data Mining; J.2 [**Physical Sciences and Engineering**]: Chemistry

General Terms

Algorithms, Experimentation

Keywords

graph mining, frequent pattern discovery, computational chemistry

1. INTRODUCTION

The discovery of *frequent patterns* in a database, i.e., patterns that occur in at least a certain specified number of elements of the 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 building blocks or 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 World Wide Web, 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 mining problem for trees can be solved in *incremental polynomial time* (see [2] for an overview on frequent subtree mining), 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 mining problem for graph structured databases in the general case cannot be solved in *output polynomial time* [10], 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, 11, 20]), 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 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 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[1], 94.3% of all elements are tenuous outerplanar graphs. We develop an algorithm for enumerating frequent tenuous outerplanar graph patterns, and prove that this algorithm is guaranteed to work in incremental polynomial time.

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

[1] `http://cactus.nci.nih.gov/`

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
KDD'06, August 20–23, 2006, Philadelphia, Pennsylvania, USA.
Copyright 2006 ACM 1-59593-339-5/06/0008 ...$5.00.

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 also present 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 Section 2, we first introduce the necessary definitions as the basis for our paper. In Section 3, we define the class of tenuous outerplanar graphs and the notion of subgraph isomorphism used in this paper, arriving at a definition of our frequent pattern mining problem. Section 4 is the main part of the paper, and describes in detail our algorithm for mining tenuous outerplanar graphs. Section 5 contains our experimental evaluation on the standard benchmark data set from NCI. Section 6 concludes and discusses some open problems. Due to space limitations, proofs are only sketched or even omitted in this short version.

2. PRELIMINARIES

In this section we recall the most important definitions and notions related to graphs (see, e.g., [8] for more details).

Graphs 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*. If, in addition, parallel edges (i.e., multiple edges connecting the same pair of vertices) and loops (i.e., edges joining a vertex to itself) are also allowed, we speak of *undirected multigraphs*. 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, and $N[v]$ is the set $N(v) \cup \{v\}$.

Trees In this paper, unless otherwise stated, by trees we mean labeled free trees (i.e., unrooted and unordered labeled trees). For a tree T and vertices $r, v \in V(T)$, T^r denotes the rooted tree obtained from T by choosing r to be its root, $\mathrm{C}_r(v)$ denotes the set of children of v in T^r, and $f_r(v)$ denotes the parent of v in T^r if $v \neq r$; otherwise $f_r(v)$ is undefined. We denote by $T^r_{v,0}$ the largest subtree of T^r rooted at v, and $T^r_{v,1}$ denotes the tree obtained from $T^r_{f_r(v),0}$ by removing $T^r_{v',0}$ for every $v' \in \mathrm{C}_r(f_r(v)) \setminus \{v\}$. Clearly, $T^r_{v,1}$ is defined iff $f_r(v)$ is defined.

Blocks and Bridges 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 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 set of trees of 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 $O(|V(G)| + |E(G)|)$ [19].

Isomorphism and Subgraph Isomorphism 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.

Analogously to the list homomorphism problem defined in [6], we introduce the *list subgraph isomorphism problem* defined as follows: *Given* graphs G and H, and sets $L_u \subseteq V(G)$ for every $u \in V(H)$, *decide* whether there exists a subgraph isomorphism φ from H to G such that $\varphi(u) \in L_u$ for every $u \in V(H)$. We denote by

$$H \xrightarrow[\{\langle u, L_u\rangle : u \in V(H)\}]{} G$$

that there is a list subgraph isomorphism φ from H to G satisfying $\varphi(u) \in L_u$ for every $u \in V(H)$. Notice that the list subgraph isomorphism problem is a generalization of the ordinary subgraph isomorphism where $L_u = V(G)$ for every $u \in V(H)$. If $L_u = V(G)$ for some particular vertex u, we will sometimes remove the pair $\langle u, L_u\rangle$ from the set below the arrow in the above notation.

Planar 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. To give a topologically rigorous definition of planar graphs, we need some auxiliary notions. A *simple curve* is the image of an injective continuous function $\gamma : [0, 1] \to \Re^2$; its endpoints are $\gamma(0)$ and $\gamma(1)$. Notice that by definition, simple curves are non self-intersecting. Let G be a graph. An *embedding* of G in the plane is a function s mapping each vertex of G to a distinct point of the plane and each edge $\{u, v\}$ of G to a simple curve of the plane connecting $s(u)$ and $s(v)$. If, in addition, it holds that any two distinct curves representing edges do not intersect except possibly at their endpoints then s is a *planar embedding* of G. A graph is *planar* iff it has a planar embedding.

Let G be a planar graph and s be some planar embedding of G. Removing from the plane all points and curves corresponding to the vertices and edges of G, respectively, we obtain a set of connected pieces of the plane, called *faces*. Since the number of vertices of G is finite, exactly one of the faces, called the *outer face*, is unbounded. For G and s, one can construct an undirected multigraph G^*, called the *dual graph* of G, as follows: G^* has a distinct vertex for each face, and for every edge e of G we connect the two vertices representing the faces having the boundary simple curve $s(e)$ in common. The *weak dual graph* of G is the multigraph obtained from G^* by removing the vertex representing the outer face and each edge containing this vertex.

Outerplanar Graphs 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 and hence, a graph is outerplanar iff each of its blocks is outerplanar [8]. Furthermore, as the blocks of a graph can be computed in linear time [19] and outerplanarity of a block can be decided also in linear time [13, 15][2], 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 [8]. This unique Hamiltonian cycle can be computed efficiently (see, e.g., [13]). Thus, G can be considered as an n-polygon with at most $n-3$ non-crossing diagonals. It also holds that the weak dual graph of G is always a tree with at most $n-2$ vertices corresponding to the interior faces of G's planar embedding. This property implies the following bound for the number of cycles of G. Due to space limitation, we omit the proof of the next proposition.

PROPOSITION 1. *Let G be a biconnected outerplanar graph with d diagonals. Then G 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 an NP-complete problem [7]. The following stronger negative result is shown in [18].

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 [13].[3]

THEOREM 3. *The problem whether a biconnected outerplanar graph H is subgraph isomorphic to a biconnected outerplanar graph G can be decided in time*

$$O\left(|V(H)| \cdot |V(G)|^2\right) \ .$$

Finally we cite another positive result from [14] on subgraph isomorphism for the special case of trees.

THEOREM 4. *The problem whether a tree H is subgraph isomorphic to a tree G can be decided in time*

$$O\left(|V(H)|^{1.5} \cdot |V(G)|\right) \ .$$

The subtree isomorphism problem can be solved in fact in time $O\left((|V(H)|^{1.5}/\log|V(H)|) \cdot |V(G)|\right)$ [17]. For the sake of simplicity, in Section 4.4 we will generalize the algorithm in [14] to outerplanar graphs by noting that the complexity of our algorithm can also be improved using the idea given in [17] for trees.

3. 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. We start the problem description with the definition of the graph class to be mined.

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 main contribution of this work is an algorithm mining d-tenuous outerplanar graphs. This problem is motivated by the observation that d-tenuous outerplanar graphs form a practically relevant graph class e.g. in computational chemistry. Consider for example the NCI dataset which is one of the largest chemical graph databases used as a benchmark domain by the data mining community. Out of the 250251 pharmacological molecules in this 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 $G, H \in \mathcal{O}_\Sigma^d$ for some $d \geq 0$. 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., the set

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

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

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

By definition, $\mathcal{F}_{\Sigma,d}^t(\mathcal{D})$ does not contain isomorphic graphs. Furthermore, it is closed downwards w.r.t. BBP subgraph

[2]We note that both outerplanarity testing algorithms in [13, 15] must be extended by an additional step checking condition (iii) of Theorem 2 in [15], as otherwise a class of non-outerplanar graphs will be misclassified by both algorithms.
[3]Although this positive result has been shown for unlabeled graphs, the algorithm in [18] can be generalized to labeled graphs as well.

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., [12]), 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 $k = 1, \ldots, |\mathcal{F}^t_{\Sigma,d}(\mathcal{D})|$. Notice that the FTOSM problem generalizes the frequent itemset mining problem solved by the Apriori algorithm [1] in incremental polynomial time.

We note that in the literature (see, e.g., [12]) one usually considers also the notion of *output polynomial time* (or *polynomial total time*) complexity for enumeration algorithms. Algorithms belonging to this more liberal class are required to enumerate a set S in the combined size of the input and the *entire* set S. This implies that, 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 [10].

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 given in [2]. Since tenuous outerplanar graphs form a practically relevant graph class that naturally generalizes trees, by considering the FTOSM problem we take a first step towards going beyond trees in frequent graph mining.

4. THE MINING ALGORITHM

In this section we present an Apriori-like [1] algorithm, that solves the FTOSM problem in incremental polynomial time. The main steps of the algorithm are sketched in Algorithm 1. It takes as input a set $\mathcal{D} \subseteq \mathcal{O}^d_\Sigma$ of d-tenuous outerplanar graphs and a frequency threshold $t \geq 0$, and computes iteratively the set of t-frequent k-patterns from the set of t-frequent $(k-1)$-patterns, where 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 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.

Algorithm 1 FREQUENTOUTERPLANARGRAPHS

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

1: $\mathcal{L}_1 = \mathcal{F}_v \cup \mathcal{F}_b$, where
$\mathcal{F}_v = \{H \in \mathcal{O}_\Sigma : |V(H)| = 1 \wedge \Pi_t(\mathcal{D}, H)\}$
$\mathcal{F}_b = \{H \in \mathcal{O}^d_\Sigma : H \text{ is biconnected} \wedge \Pi_t(\mathcal{D}, H)\}$
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: $k = k + 1$
6: $\mathcal{C}_k = \text{GENERATECANDIDATES}(\mathcal{L}_{k-1})$
7: $\mathcal{L}_k = \{H \in \mathcal{C}_k : \Pi_t(\mathcal{D}, H)\}$
8: **endwhile**
9: **return** $\cup_{i=1}^k \mathcal{L}_i$

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 union of G_1 and G_2 by contracting[4] a vertex from G_1 with a vertex from G_2 that have the same label. Clearly, $G_1 \bowtie G_2 \subseteq \mathcal{O}^d_\Sigma$ for every $G_1, G_2 \in \mathcal{O}^d_\Sigma$. 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 8 – 10 stated in Sections 4.2 – 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 4.2 – 4.4, in the next section we first describe a transformation and a canonical string representation of outerplanar graphs that will be used in different steps of the mining algorithm.

4.1 Canonical String Representation

One time consuming step of mining frequent d-tenuous outerplanar graphs is to test whether a particular graph

[4]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.

$H \in \mathcal{O}_\Sigma^d$ belongs to some subset S of $\mathcal{O}_\Sigma^d$. To apply some advanced data structure (e.g., hash tables, B-trees, etc.) that allows 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 in time polynomial in $|V(G)|$. 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 isomorphism can be decided efficiently even for planar graphs [9], the canonical string representation for outerplanar graphs described in this section may be of some interest in itself.

We first define a transformation on outerplanar graphs by means of contraction of blocks into new vertices. More precisely, for a graph $G \in \mathcal{O}_\Sigma$, let $\widetilde{G}$ denote the 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.

PROOF SKETCH. The proof of (i) and (ii) is trivial. To see (iii), suppose that $\widetilde{G}$ has a cycle C. Then C must contain a vertex v labeled by $\#$. But this implies that the biconnected subgraph of G corresponding to v is not maximal contradicting the definition of $\widetilde{G}$. □

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

For a graph $G \in \mathcal{O}_\Sigma$ and $v \in V(\widetilde{G})$, let $\tau(v)$ denote the subgraph of G corresponding to v, i.e., it denotes the block of G represented by v if $\lambda_G(v) = \#$; otherwise it is the subgraph of G consisting of the single vertex corresponding to v.

Using the above notions and notations, we define the canonical representation of G by means of $\widetilde{G}$. By (iii) of Proposition 6, $\widetilde{G}$ is a free tree. We utilize this property and generalize the depth-first canonical representation for free trees to outerplanar graphs (see [2] for an overview on canonical string representations for trees). Given some distinguished vertex $r \in V(\widetilde{G})$, we assign recursively a string $\rho_r(v) \in (\Sigma \cup \mathbb{N} \cup \{\$, \#, @\})^*$ to every vertex v of $\widetilde{G}^r$, and define a string encoding of G w.r.t. r by the string $\rho_r(r)$ associated with r. To define $\rho_r(v)$ for a vertex $v \in V(\widetilde{G})$, we distinguish two cases:

(i) Suppose $\lambda_{\widetilde{G}}(v) \neq \#$. Then we define $\rho_r(v)$ by

$$\rho_r(v) = l_0 l_{i_1} \rho_r(v_{i_1}) \dots l_{i_k} \rho_r(v_{i_k}) \$ \ ,$$

where

- l_0 is the label of v,
- $\{v_{i_1}, \dots, v_{i_k}\}$ is the set of children of v in $\widetilde{G}^r$,
- for every $q = 1, \dots, k$, l_{i_q} is the label of the edge connecting v and v_{i_q} if $\lambda_{\widetilde{G}}(v_{i_q}) \neq \#$; otherwise $l_{i_q} = 1$, and
- $l_{i_p} \rho_r(v_{i_p}) \leq l_{i_q} \rho_r(v_{i_q})$ for every $1 \leq p < q \leq k$.[5]

(ii) Suppose $\lambda_{\widetilde{G}}(v) = \#$. Then $\tau(v)$ must be a block of G and has thus a unique Hamiltonian cycle H. Let ℓ denote the length of H. Clearly, there are 2ℓ sequences of vertices defining H. Let $s = v_1, \dots, v_\ell$ be such a sequence. For s, we define the string $\rho_{r,s}(v)$ by

$$\rho_{r,s}(v) = @ S_V S_E \delta S_D i_1 \rho_r(v_{i_1}) \dots i_k \rho_r(v_{i_k}) \$ \ ,$$

where

- $S_V = \lambda_G(v_1) \dots \lambda_G(v_\ell)$,
- $S_E = \lambda_G(\{v_1, v_2\}) \dots \lambda_G(\{v_{\ell-1}, v_\ell\}) \lambda_G(\{v_\ell, v_1\})$,
- δ is the number of diagonals in $\tau(v)$,
- S_D is the unique sequence $i_1 j_1 l_1 \dots i_\delta j_\delta l_\delta$ of integers satisfying (1) $i_q < i_r$ or ($i_q = i_r$ and $j_q < j_r$) for every $1 \leq q < r \leq \delta$ and (2) $\{v_{i_q}, v_{j_q}\}$ is a diagonal with label l_{i_q} in G for every $q = 1, \dots, \delta$,
- $1 \leq i_1 < \dots < i_k \leq \ell$, and
- $\{v_{i_1}, \dots, v_{i_k}\}$ is the set of children of v in $\widetilde{G}^r$.

Let $\mathcal{S}$ be the set of 2ℓ sequences defining H, if v has no parent in $\widetilde{G}$ (i.e., $v = r$); otherwise let $\mathcal{S} = \{s_1, s_2\}$, where s_1 and s_2 are vertex sequences defining H and are of the form $\tau(f_r(v))s_1'$ and $\tau(f_r(v))s_2'$, respectively. Using the above definitions of $\rho_{r,s}$ and $\mathcal{S}$, we define $\rho_r(v)$ by

$$\rho_r(v) = \min_{s \in \mathcal{S}} \rho_{r,s}(v) \ .$$

Given G and $r \in V(\widetilde{G})$, the strings assigned to the vertices of $\widetilde{G}^r$ can be computed efficiently by traversing $\widetilde{G}$ in postorder.

Using the above definition of ρ_r, the canonical string representation of an outerplanar graph can be defined by the *center-based* canonical string representation of free trees (see, e.g., [2]). A *center* of a free tree is a vertex minimizing the maximum distance to any other vertex in the tree. Clearly, a tree has at most two centers. Consider first the case when $\widetilde{G}$ has one center, say r. Then we define the canonical string representation of G by

$$\rho(G) = \rho_r(r) \ .$$

Otherwise, when $\widetilde{G}$ has two centers, say r_1 and r_2, we consider the trees $\widetilde{G}_1$ and $\widetilde{G}_2$ obtained from $\widetilde{G}$ by removing the edge connecting r_1 and r_2, and define $\rho(G)$ by

$$\rho(G) = \rho_{r_1}(G_1) \# l \# \rho_{r_2}(G_2) \ ,$$

where l is the label of the edge connecting r_1 and r_2, and G_1 and G_2 are the subgraphs of G corresponding to $\widetilde{G}_1$ and $\widetilde{G}_2$, respectively. For $\rho(G)$, the following result can be shown:

[5]We assume some total order on the set of symbols and define the total order $\leq$ on strings by the lexicographic order.

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 s.t. $H \ominus \Delta \in \mathcal{L}_{k-1}$ for every diagonal Δ of H
4: $\mathcal{L}_k = \{H \in \mathcal{C}_k : \Pi_t(\mathcal{D}, H)\}$
5: **endfor**
6: **return** $\bigcup_{k=0}^{d} \mathcal{L}_k$

THEOREM 7. *Let $G \in \mathcal{O}_\Sigma$ with n vertices. Then $\rho(G)$ is a canonical string representation of G and it 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 that computes the set $\mathcal{F}_b$ of t-frequent d-tenuous biconnected graphs used in step 1 of Algorithm 1. Since d is assumed to be 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 [16, 19] it follows that the cycles of a graph can be listed with linear delay. Since isomorphism between cycles can be decided efficiently, these results together imply that $\mathcal{L}_0$ can be computed in time polynomial in the parameters of $\mathcal{D}$. As a cycle may be compared to many other cycles, to decide isomorphism, we use the canonical string representation described in Section 4.1.

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.

For $k = 1$ in particular, $\mathcal{C}_1$ can be computed as follows: For every cycle $H \in \mathcal{L}_0$ and for every t-frequent edge symbol $l \in \Sigma$, we connect each pair of non-adjacent vertices of H by an edge, label this new diagonal by l, and add the obtained graph H' to $\mathcal{C}_1$ if $H' \notin \mathcal{C}_1$. To decide the membership in $\mathcal{C}_1$, here we use again the graphs' canonical string representations.

For $k > 1$, we compute $\mathcal{C}_k$ by the following algorithm: For every pair $H_1, H_2 \in \mathcal{L}_{k-1}$ with the same Hamiltonian cycle, and for every pair d_1 and d_2 of diagonals of H_1 and H_2, respectively, for which

$$\lambda_{H_i}(d_i) \leq \min\{\lambda_{H_j}(\Delta) : j = 1, 2, \Delta \neq d_j \text{ is a diagonal of } H_j\}$$

holds for $i = 1, 2$, consider the graphs H_1' and H_2' obtained from H_1 and H_2 by removing d_1 and d_2, respectively. If $H_1' \simeq H_2'$ then for every isomorphism φ from H_1' to H_2', we take the graph H obtained from H_2 by connecting the images of the endpoints of d_1 by an edge labeled by the symbol assigned to d_1 in H_1. If H remains outerplanar, i.e., the new diagonal doesn't cross any other diagonal, then for every diagonal Δ of H except the new one and d_2, we remove Δ from H and check whether the graph obtained belongs to $\mathcal{L}_{k-1}$. If this is the case for every Δ, we add H to $\mathcal{C}_k$ if $H \notin \mathcal{C}_k$. Notice that this step corresponds to the pruning technique applied in the candidate generation step of the Apriori algorithm. We note that the number of isomorphisms between H_1 and H_2 is at most $2|V(H_1)|$. Furthermore, in the selection of H_1, H_2 and d_1, d_2 above, H_1 and H_2, or even d_1 and d_2 can be identical because automorphisms (i.e., isomorphisms from a graph to itself) must also be considered.

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, $C \notin \mathcal{C}_k$, 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** $\mathcal{C}_k$

One can show that the method described above is complete, i.e., for every $k = 1, \ldots, d$, $\mathcal{C}_k$ contains the set of t-frequent biconnected graphs in $\mathcal{O}_\Sigma^k \setminus \mathcal{O}_\Sigma^{k-1}$.

Finally, in order to compute the set of t-frequent biconnected graphs from the $\mathcal{C}_k$'s (step 4), we have to decide the existence of BBP subgraph isomorphisms from biconnected outerplanar graphs to outerplanar graphs. For this case, BBP subgraph isomorphism is equivalent to subgraph isomorphism because the pattern graph consists of a single block (and no bridge), and blocks are mapped to blocks by any subgraph isomorphism. Since the blocks of a graph can be computed in linear time [19], it is therefore sufficient to consider the efficiency of subgraph isomorphism between biconnected outerplanar graphs. Theorem 11 in Section 4.4 below deals with a more general case implying that this problem can be solved in cubic time.

Putting the above results together, one can show 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. (Recall that a k-pattern is a graph such that the sum of the number of its blocks and the number of its vertices not belonging to any block is k.) 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. For completeness, we have to allow G_1 and G_2 to be the same. In loop 3–11, we consider 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 graph $\tau(v)$ for some leaf v of the BB-tree $\widetilde{H}$, i.e., a leaf subgraph is either a vertex of H not belonging to a block and adjacent to exactly one other vertex or it is a block which has exactly one common vertex with a bridge or with another block. 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. From G_1 and G_2, we remove g_1 and g_2, respectively, denoted this operation 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 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 statement.

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, shows that this constrained subgraph isomorphism can be decided efficiently between outerplanar graphs if the pattern graph H is connected. This result holds for arbitrary outerplanar graphs, i.e., we do not assume any bound on the number of diagonals in a block. The connectivity is necessary, as otherwise the problem would generalize the NP-complete subforest isomorphism problem [7]. 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. For the rest of this section, we fix an alphabet Σ and a distinguished symbol, say $\#$, satisfying $\# \notin \Sigma$.

THEOREM 10. *For every $G \in \mathcal{O}_\Sigma$ and connected graph $H \in \mathcal{O}_\Sigma$, $H \preccurlyeq_{BBP} G$ can be decided in polynomial time.*

To sketch the proof of Theorem 10, we need some further notations and assertions. We start with a theorem generalizing Theorem 3, the positive result from [13] on ordinary subgraph isomorphism between unlabeled biconnected outerplanar graphs to list subgraph isomorphism between labeled biconnected outerplanar graphs. Due to space limitation, we only sketch the proof.

THEOREM 11. *Let $G, H \in \mathcal{O}_\Sigma$ be biconnected outerplanar graphs and $L_u \subseteq V(G)$ for every $u \in V(H)$. Then one can decide whether there exists a list subgraph isomorphism*

$$H \xrightarrow[\{\langle u, L_u\rangle : u \in V(H)\}]{} G$$

in time $O\left(|V(H)| \cdot |V(G)|^2\right)$.

PROOF SKETCH. The proof is similar to that of Theorem 3 (see [13] for the proof). It is based on constructing first a directed graph D with a distinguished source and target vertex s and t, respectively, and then determining whether there is a directed path from s to t. The edges in D represent whether there is a subgraph isomorphism from a certain subgraph of the pattern graph containing an edge $\{u_i, u_{i+1}\}$ to a certain subgraph of the text graph containing an edge $\{v_i, v_{i+1}\}$ such that u_i and u_{i+1} are mapped to v_i and v_{i+1}, respectively. This constraint can be generalized to list subgraph isomorphism between labeled biconnected outerplanar graphs without changing the asymptotic time complexity of the special case considered in Theorem 3. □

We need some further notations. Let $G \in \mathcal{O}_\Sigma$ and $T_{\widetilde{G}}^v$ be a subtree of $\widetilde{G}$ rooted at v. Then $G[T_{\widetilde{G}}^v]$ denotes the subgraph of G induced by $\bigcup_{v' \in V(T_{\widetilde{G}}^v)} V(\tau(v'))$. Furthermore, for a graph $H \in \mathcal{O}_\Sigma$ and subtree $T_{\widetilde{H}}^u$ of $\widetilde{H}$ rooted at u, $H[T_{\widetilde{H}}^u] \subseteq_{BBP} G[T_{\widetilde{G}}^v]$ denotes that there is a BBP subgraph isomorphism from $H[T_{\widetilde{H}}^u]$ to $G[T_{\widetilde{G}}^v]$ mapping $V(\tau(u))$ to $V(\tau(v))$. The proof of the following proposition is immediate from the definitions.

PROPOSITION 12. *Let $G, H \in \mathcal{O}_\Sigma$ and $r \in V(\widetilde{G})$. Then $H \preccurlyeq_{BBP} G$ iff there are $u \in V(\widetilde{H})$ and $v \in V(\widetilde{G})$ such that*

$$H[\widetilde{H}^u] \subseteq_{BBP} G[\widetilde{G}_{v,0}^r] \ . \tag{3}$$

Applying the above proposition, $H \preccurlyeq_{BBP} G$ can be decided as follows: Select an arbitrary vertex $r \in V(\widetilde{G})$ and compute for every $u \in V(\widetilde{H})$ and for every $v \in V(\widetilde{G})$, whether (3) holds. Below we show that for given u and v, (3) can be computed by deciding first for every $w \in N[u]$, whether

$$H[\widetilde{H}_{u,\overline{\delta}_{u,w}}^w] \subseteq_{BBP} G[\widetilde{G}_{v,\overline{\delta}_{u,w}}^r] \tag{4}$$

holds, where $\overline{\delta}$ is the complement of Kronecker delta (i.e., $\overline{\delta}_{u,w} = 0$ if $u = w$; otherwise it is 1). For $v \in V(\widetilde{G})$ and $u \in V(\widetilde{H})$, we define the set

$$S(v,u) = \{w \in N[u] : H[\widetilde{H}^w_{u,\overline{\delta}_{u,w}}] \subseteq_{BBP} G[\widetilde{G}^r_{v,\overline{\delta}_{u,w}}]\} \ .$$

Notice that by definition, $\widetilde{H}^u = \widetilde{H}^u_{u,0}$ and hence, by Proposition 12 and Eq. (3) we have that

$$H \preccurlyeq_{BBP} G \iff \exists u \in V(\widetilde{H}) \wedge \exists v \in V(\widetilde{G}) \text{ such that } u \in S(v,u) \ .$$

Depending on the label of u, we distinguish two cases and characterize (4) accordingly by one of the following two lemmas.

LEMMA 13. *Let $G, H \in \mathcal{O}_\Sigma$. Let $r, v \in V(\widetilde{G})$, $u \in V(\widetilde{H})$, and $w \in N[u]$. If $\lambda_{\widetilde{H}}(u) = \#$ then*

$H[\widetilde{H}^w_{u,\overline{\delta}_{u,w}}] \subseteq_{BBP} G[\widetilde{G}^r_{v,\overline{\delta}_{u,w}}]$ *iff there is an injection*

$\varphi : C_{\widetilde{H}^w}(u) \to C_{\widetilde{G}^r}(v)$ *such that*

(i) $H[\widetilde{H}^u_{u',1}] \subseteq_{BBP} G[\widetilde{G}^r_{\varphi(u'),1}]$ *for all* $u' \in C_{\widetilde{H}^w}(u)$,

(ii) $\tau(u) \xrightarrow[\{\langle \tau(u'), L_{u'} \rangle : u' \in N(u)\}]{} \tau(v)$, *where*

$$L_{u'} = \begin{cases} \{\tau(\varphi(u'))\} & \text{if } u' \in N(u) \setminus \{w\} \\ \{\tau(f_{\widetilde{G}^r}(v))\} & \text{o/w (i.e., } u' = w \neq u) \ . \end{cases}$$

PROOF SKETCH. Since $\lambda_{\widetilde{H}}(u) = \#$, $\tau(u)$ is biconnected. By (ii) of Proposition 6, for every $u' \in N(u)$ we have that $\lambda_{\widetilde{H}}(u') \neq \#$ and hence, $\tau(u')$ must be a vertex belonging to the block $\tau(u)$. The proof can be shown by considering the cases $u = w$ and $u \neq w$ separately. □

LEMMA 14. *Let the graphs G, H and vertices r, v, u, and w be defined as in Lemma 13. If $\lambda_{\widetilde{H}}(u) \neq \#$ then*

$H[\widetilde{H}^w_{u,\overline{\delta}_{u,w}}] \subseteq_{BBP} G[\widetilde{G}^r_{v,\overline{\delta}_{u,w}}]$ *iff there is an injection*

$\varphi : C_{\widetilde{H}^w}(u) \to C_{\widetilde{G}^r}(v)$ *such that*

(i) $H[\widetilde{H}^u_{u',1}] \subseteq_{BBP} G[\widetilde{G}^r_{\varphi(u'),1}]$ *for all* $u' \in C_{\widetilde{H}^w}(u)$,

(ii) and if $w \neq u$ then

- $r \neq v$ *(i.e., $f_{\widetilde{G}^r}(v)$ is defined),*
- $\lambda_{\widetilde{H}}(w) = \lambda_{\widetilde{G}}(f_{\widetilde{G}^r}(v))$,
- $\lambda_{\widetilde{H}}(\{u,w\}) = \lambda_{\widetilde{G}}(\{v, f_{\widetilde{G}^r}(v)\})$, *and*
- $\tau(w) \xrightarrow[\{\langle \tau(u), \{\tau(v)\}\rangle\}]{} \tau(f_{\widetilde{G}^r}(v))$ *if w represents a block, i.e., $\lambda_{\widetilde{H}}(w) = \#$.*

PROOF SKETCH. The proof can be shown by considering the following cases separately: (1) $u = w$, (2) $u \neq w$ and $\lambda_{\widetilde{H}}(w) = \#$, and (3) $u \neq w$ and $\lambda_{\widetilde{H}}(w) \neq \#$. □

PROOF SKETCH OF THEOREM 10. Using the above notions and statements, we are now ready to sketch a bottom-up algorithm deciding $H \preccurlyeq_{BBP} G$ for some $G, H \in \mathcal{O}_\Sigma$ in polynomial time. Due to space limitation, we do not provide the pseudo code of the algorithm. Notice that it is sufficient to consider the case when G and H satisfy

$$1 < |V(\widetilde{H})| \leq |V(\widetilde{G})| \ . \tag{5}$$

Indeed, if $|V(\widetilde{H})| = 1$ then, by (i) of Proposition 6, H consists of either a single vertex or a block. Deciding BBP subgraph isomorphism is obvious for the first case; for the second case the problem can be considered as a special instance of the list subgraph isomorphism problem between biconnected outerplanar graphs and thus, Theorem 11 can be applied. Furthermore, if $|V(\widetilde{H})| > |V(\widetilde{G})|$ then there is no BBP subgraph isomorphism from H to G.

Given $G, H \in \mathcal{O}_\Sigma$ satisfying (5), we select an arbitrary vertex $r \in V(\widetilde{G})$, traverse $\widetilde{G}^r$ in postorder, and, for each visited vertex $v \in V(\widetilde{G}^r)$, compute the set $S(v,u)$ for every $u \in V(\widetilde{H})$ in the following way:

Suppose v is a leaf of $\widetilde{G}^r$. Then (5) implies that $r \neq v$ and thus $v' = f_{\widetilde{G}^r}(v)$ is defined. For this case we have that $S(v,u) = N(u)$ iff (i) $N(u) = \{u'\}$ for some $u' \in \widetilde{H}$ (i.e., u is also a leaf in $\widetilde{H}$) and (ii) the vertices u, u' and v, v' satisfy the following condition: $\lambda_{\widetilde{H}}(u) = \lambda_{\widetilde{G}}(v)$, $\lambda_{\widetilde{H}}(u') = \lambda_{\widetilde{G}}(v')$, $\lambda_{\widetilde{H}}(\{u,u'\}) = \lambda_{\widetilde{G}}(\{v,v'\})$, and if $\lambda_{\widetilde{H}}(u) = \#$ (resp. $\lambda_{\widetilde{H}}(u') = \#$) then there is a list subgraph isomorphism from $\tau(u)$ to $\tau(v)$ (resp. $\tau(u')$ to $\tau(v')$) which maps the vertex $\tau(u')$ to $\tau(v')$ (resp. $\tau(u)$ to $\tau(v)$). Otherwise, i.e., when conditions (i) and (ii) above do not hold we set $S(v,u) = \emptyset$.

Suppose v is an internal vertex of $\widetilde{G}^r$. Then let $N(u)$ and $N(v)$ be $\{u_1, \ldots, u_s\}$ and $\{v_1, \ldots, v_t\}$, respectively. If $s > t+1$ or $\lambda_{\widetilde{H}}(u) \neq \lambda_{\widetilde{G}}(v)$ then $S(v,u) = \emptyset$, as in this case there is no $w \in N[u]$ satisfying (4).

Otherwise, i.e., when $\lambda_{\widetilde{H}}(u) = \lambda_{\widetilde{G}}(v)$ and $s \leq t+1$, we distinguish two cases depending on $\lambda_{\widetilde{H}}(u)$. Consider first the case when $\lambda_{\widetilde{H}}(u) = \#$, i.e., $\tau(u)$ is biconnected. Then, $\tau(v)$ is also biconnected, as $\lambda_{\widetilde{H}}(u) = \lambda_{\widetilde{G}}(v)$, and hence, $\tau(z)$ is a vertex for every $z \in N(u) \cup N(v)$ by (ii) of Proposition 6. By Lemma 13 we have that

- $u \in S(v,u)$ iff there is a list subgraph isomorphism from $\tau(u)$ to $\tau(v)$ mapping $\tau(u_i)$ to one of the elements of $\{\tau(v_k) : 1 \leq k \leq t, u \in S(v_k, u_i)\}$ for every $i = 1, \ldots, s$ and
- for every $i = 1, \ldots, s$, $u_i \in S(v,u)$ iff $r \neq v$ and there is a list subgraph isomorphism from $\tau(u)$ to $\tau(v)$ mapping $\tau(u_i)$ to $\tau(f_{\widetilde{G}^r}(v))$ and $\tau(u_j)$ into $\{\tau(v_k) : 1 \leq k \leq t, u \in S(v_k, u_j)\}$ for every j, $1 \leq j \neq i \leq s$.

In both cases, the existence of the corresponding list subgraph isomorphism between biconnected outerplanar graphs can be decided in polynomial time by Theorem 11.

For the second case, i.e., when $\lambda_{\widetilde{H}}(u) \neq \#$, we construct a bipartite graph B_0 defined as follows:

$$V(B_0) = N(u) \cup N(v)$$
$$E(B_0) = \{\{x,y\} : x \in N(u), y \in N(v), u \in S(x,y)\}.$$

Let B_i denote the subgraph of B_0 obtained by removing u_i and the edges adjacent to u_i. Using these notations, by Lemma 14 we have that

- $u \in S(v,u)$ iff B_0 has a maximum matching of size s and
- for every $i = 1, \ldots, s$, $u_i \in S(v,u)$ iff B_i has a maximum matching of size $s-1$, $v' = f_{\widetilde{G}^r}(v)$ is defined (i.e., $r \neq v$), $\lambda_{\widetilde{H}}(u_i) = \lambda_{\widetilde{G}}(v')$, $\lambda_{\widetilde{H}}(\{u,u_i\}) =$

$\lambda_{\widetilde{G}}(\{v, v'\})$, and if $\lambda_{\widetilde{H}}(u_i) = \#$ then there is a list subgraph isomorphism from $\tau(u_i)$ to $\tau(f_{\widetilde{G}^r}(v))$ which maps the vertex $\tau(u)$ to $\tau(v)$.[6]

Thus, $S(v, u)$ can be computed in polynomial time for every $u \in V(\widetilde{H})$ and $v \in V(\widetilde{G}^r)$ completing the proof sketch of Theorem 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%.

Matching blocks is more expensive than matching bridges, but the number of bridges is much larger. In all our experiments, roughly half of the time is used for each of the two types of matching.

5.1 Results

Our results are given in Table 1. It shows the number of candidate (#C) and frequent (#FP) k-patterns discovered for $k = 1, \ldots, 25$, 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]We note that it is sufficient to compute a maximum bipartite matching M in B_0 because we can compute the size of a maximum matching in $B_1, \ldots, B_s$ from M in time $O(st)$ (see [17] for the details).

6. CONCLUSION AND OPEN PROBLEMS

We have presented an incremental-polynomial time algorithm for the enumeration of the frequent patterns from a set of d-tenuous outerplanar graphs w.r.t. BBP subgraph isomorphism. 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 result is not only theoretical, but also practical; most of the graphs in the NCI dataset consisting of more than 250000 compounds are 5-tenuous outerplanar graphs.

Our algorithm 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. 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 gave a proof sketch that it 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.

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, we first 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 40/2-2) *Hybride Methoden und Systemarchitekturen für heterogene Informationsräume.* Jan Ramon is a post-doctoral fellow of the Fund for Scientific Research (FWO) of Flanders.

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
16	0	0	0	0	0	0	4079	4072	5578	7976	7957	5828
17	0	0	0	0	0	0	5518	5487	6898	12742	12729	9077
18	0	0	0	0	0	0	6729	6711	8175	21606	21600	14752
19	0	0	0	0	0	0	7326	7311	8813	37386	37383	24017
20	0	0	0	0	0	0	7114	7079	8703	64241	64238	38821
21	0	0	0	0	0	0	6000	5947	7627	106645	106621	61238
22	0	0	0	0	0	0	4435	4407	5954	168821	168805	93109
23	0	0	0	0	0	0	2857	2855	4129	251406	251278	135048
24	0	0	0	0	0	0	1633	1633	2609	349306	348683	184429
25	0	0	0	0	0	0	787	786	1444	448014	447305	234828

7. 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. *Journal 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 Trans. Knowl. Data Eng*, 17(8):1036–1050, 2005.

[6] T. Feder and P. Hell. List homomorphisms to reflexive graphs. *J. Comb. Theory, Ser. B*, 72(2):236–250, 1998.

[7] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to NP-Completeness*. Freeman, San Francisco, CA, 1979.

[8] F. Harary. *Graph Theory*. Addison–Wesley, Reading, Massachusetts, 1971.

[9] J. E. Hopcroft and J. K. Wong. Linear time algorithm for isomorphism of planar graphs. In *Proc. of the Sixth Annual ACM Symposium on Theory of Computing*, pp. 172–184. ACM Press, New York, NY, 1974.

[10] T. Horváth, B. Bringmann, and L. D. Raedt. Frequent hypergraph mining. *Unpublished manuscript*, 2004.

[11] A. Inokuchi, T. Washio, and H. Motoda. Complete mining of frequent patterns from graphs: Mining graph data. *Machine Learning*, 50(3):321–354, 2003.

[12] D. S. Johnson, M. Yannakakis, and C. H. Papadimitriou. On generating all maximal independent sets. *Information Processing Letters*, 27(3):119–123, 1988.

[13] A. Lingas. Subgraph isomorphism for biconnected outerplanar graphs in cubic time. *Theoretical Computer Science*, 63:295–302, 1989.

[14] D. W. Matula. Subtree isomorphism in $O(n^{\frac{5}{2}})$. *Annals of Discrete Mathematics*, 2:91–106, 1978.

[15] S. L. Mitchell. Linear algorithms to recognize outerplanar and maximal outerplanar graphs. *Information Processing Letters*, 9(5):229–232, 1979.

[16] R. C. Read and R. E. Tarjan. Bounds on backtrack algorithms for listing cycles, paths, and spanning trees. *Networks*, 5(3):237–252, 1975.

[17] R. Shamir and D. Tsur. Faster subtree isomorphism. *Journal of Algorithms*, 33(2):267–280, 1999.

[18] M. M. Sysło. The subgraph isomorphism problem for outerplanar graphs. *Theoretical Computer Science*, 17:91–97, 1982.

[19] R. E. Tarjan. Depth first search and linear graph algorithms. *SIAM J. Comput.*, 1(2):146–160, 1972.

[20] X. Yan and J. Han. gspan: Graph-based substructure pattern mining. In *Proc. of the IEEE Int. Conf. on Data Mining*, pp. 721–724, IEEE Comp. Society, 2002.