Data Min Knowl Disc (2010) 21:472–508
DOI 10.1007/s10618-009-0162-1

Frequent subgraph mining in outerplanar graphs

Tamás Horváth · Jan Ramon · Stefan Wrobel

Received: 12 August 2008 / Accepted: 17 December 2009 / Published online: 26 January 2010

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 consider the class of *outerplanar graphs*, a strict generalization of trees, develop a frequent subgraph mining algorithm for outerplanar graphs, and show that it works in incremental polynomial time for the practically relevant subclass of *well-behaved* outerplanar graphs, i.e., which have only polynomially many simple cycles. We evaluate the algorithm empirically on chemo- and bioinformatics applications.

Responsible editor: M.J. Zaki.

A preliminary version of this paper appeared in: T. Eliassi-Rad, L. Ungar, M. Craven, and D. Gunopulos (Eds.), Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 197–206, ACM Press, New York, NY, 2006.

T. Horváth (✉) · S. Wrobel
Department of Computer Science III, University of Bonn, Bonn, Germany
e-mail: tamas.horvath@iais.fraunhofer.de

J. Ramon
Department of Computer Science, Katholieke Universiteit Leuven, Leuven, Belgium
e-mail: jan.ramon@cs.kuleuven.be

T. Horváth · S. Wrobel
Fraunhofer Institute IAIS, Schloss Birlinghoven, Sankt Augustin, Germany

S. Wrobel
e-mail: stefan.wrobel@iais.fraunhofer.de

Keywords Graph mining · Frequent pattern mining · Algorithms · Complexity · Applications

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 being interesting in their own right, frequent patterns can also be used as building blocks or features for predictive data mining tasks (see, e.g., Deshpande et al. 2005). For a long time, work on frequent pattern discovery has concentrated on relatively simple notions of patterns and elements (transactions) 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 is tractable (i.e., can be solved in *incremental polynomial time*; see Chi et al. 2005a for an overview on frequent subtree mining), for arbitrary graphs it becomes intractable (Horváth et al. 2007) (i.e., cannot even be solved in *output polynomial time*). Existing approaches to frequent pattern discovery for graphs have therefore resorted to various heuristic strategies and restrictions of the search space (see, e.g., Borgelt and Berthold 2002; Chi et al. 2005a; Cook and Holder 1994; Deshpande et al. 2005; Inokuchi et al. 2003; Kramer et al. 2001; Kuramochi and Karypis 2001; Maunz et al. 2009; Nijssen and Kok 2004; Yan and Han 2002), but have not identified a practically relevant tractable graph class beyond trees. The *main contributions* of this paper are the definition of a practically relevant graph class that is strictly more general than trees, the introduction of a meaningful non-standard embedding operator, and the development of an algorithm which allows frequent pattern discovery to be performed efficiently for this class with respect to this operator.

1.1 Well-behaved outerplanar graphs

Since trees, outerplanar graphs, and planar graphs form a natural hierarchy with respect to minors (Chartrand and Harary 1967), we consider the class of *outerplanar graphs* for the generalization of trees, which is the class of graphs that can be embedded in the plane in such a way that all of their vertices lie on the outer boundary, i.e., can be reached from the outside without crossing any edges. We develop a levelwise search algorithm (Mannila and Toivonen 1997) for listing frequent connected outerplanar graph patterns with respect to a constrained subgraph isomorphism discussed below. We show that if the transaction graphs are *well-behaved*, i.e., their number of cycles is bounded by some polynomial of their size, then this algorithm is guaranteed to work in *incremental polynomial time*. That is, the maximum delay between any two patterns consecutively outputted by the algorithm is bounded by a polynomial of the combined size of the input and the output computed so far.

In fact, for our levelwise search algorithm, the polynomial does not depend on the size of the output computed so far, but only on the size of frequent patterns found on the immediately preceding level of our search, and not on the number of patterns at levels before that. Interestingly, simply by changing the position of output statements in our algorithm, keeping its computation and total run time exactly the same, we can show that our algorithm actually can solve the problem with *polynomial delay* (in the parameters of the input problem) as well. However, when using the modified output sequencing, outputs are not shown to the user as soon as they have been computed, but are spread out to ensure polynomial delay between each of them. Therefore, we believe that in practice, most users would prefer to see computed outputs as soon as possible, and then rather wait a somewhat longer time whenever the next level of search is computed, resulting in incremental polynomial time between outputs. Either way, the total run time and storage requirements of the algorithms remain the same.

We also note that our listing algorithm is based on a *canonical form* of arbitrary outerplanar graphs which may be of some independent interest.

1.2 BBP subgraph isomorphism

To map a pattern to graphs in the database, we define a special embedding operator, the *block and bridge preserving* (BBP) subgraph isomorphism, which is motivated by complexity and application considerations. We show that, in contrast to ordinary subgraph isomorphism, it is decidable in polynomial time for arbitrary outerplanar graphs. Furthermore, the number of frequent patterns with respect to BBP subgraph isomorphism can be exponentially smaller than that with respect to ordinary subgraph isomorphism; a second advantageous property. Recent empirical studies in virtual screening using frequent patterns as predictive features (Schietgat et al. 2008, 2009) clearly indicate that features generated with respect to BBP subgraph isomorphism compare favorably to features generated with respect to ordinary subgraph isomorphism.

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 subgraph isomorphism may be a non-adequate matching operator in some applications (e.g., when pattern matching is required to preserve certain types of fragments in molecules), by considering BBP subgraph isomorphism we take a first step towards studying the frequent graph mining problem with respect to non-standard matching operators as well.

1.3 Applications

Our positive result on efficient pattern mining is not only of theoretical, but also of practical interest. Indeed, well-behaved outerplanar graphs appear in diverse practical applications, e.g., in *telecommunication*, *electrical circuits*, *computational drug design*, and *bioinformatics*. In telecommunication, for example, simple feeder networks can be represented by 3-*cactus graphs* (Koontz 1980) which form a further restricted class of outerplanar graphs; a k-cactus graph is an outerplanar graph with

blocks having at most k vertices and no diagonals. Thus, the number of simple cycles in a cactus graph is always linearly bounded by its size implying that cactus graphs are always well-behaved. In the theory of electrical circuits, well-behaved outerplanar graphs are used to characterize certain desirable properties of circuits; a circuit has a unique DC (direct current) solution if and only if it has a cactus graph representation (Nishi and Chua 1986). Next, regarding computational drug design, we observed in various datasets consisting of pharmacological compounds that most of the molecules have well-behaved outerplanar graphs. For example, in one of the popular graph mining data sets, the NCI data set,[1] 94.3% of all elements are well-behaved outerplanar graphs; with only 8.8% acyclic outerplanar graphs (i.e., trees). Finally, it is well-known that the contact structure of DNA and RNA molecules, called *secondary structures*, are always outerplanar graphs (see, e.g., Leydold and Stadler 1998). Our experiments conducted on mRNA secondary structures indicate that they are also well-behaved.

1.4 Outline

The rest of the paper is organized as follows. In Sect. 2, we first introduce the necessary definitions as the basis for our paper. In Sect. 3, we define the notion of subgraph isomorphism used in this paper, arriving at a definition of our frequent pattern mining problem. Section 4 is devoted to related work. Section 5 is the main part of the paper, and describes in detail our algorithm for mining outerplanar graphs. Beside complexity results, in Sect. 6 we also present empirical results which show that the favorable theoretical properties of the algorithm and pattern class also translate into efficient practical performance. We report experimental results on the standard benchmark data set from NCI and also on the mRNA secondary structure dataset used in Horváth et al. (2001). Section 7 concludes and discusses some open problems.

2 Preliminaries

In this section we recall the most important definitions and notions related to graphs (see, e.g., Diestel 2005; Harary 1971 for more details).

2.1 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 sequence $\{v_0, v_1\}, \{v_1, v_2\}, \ldots, \{v_k, v_0\}$ of edges of a graph forms a *simple cycle* if the v_i's are pairwise different. A *labeled undirected graph* is a triple (V, E, λ), where (V, E) is an undirected graph and $\lambda : V \cup E \to \mathbb{N}_0$ is a *labeling function*.[2] Unless otherwise stated,

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

[2] We denote by $\mathbb{N}$ the set $\{1, 2, 3, \ldots\}$ and $\mathbb{N}_0$ denotes $\mathbb{N} \cup \{0\}$.

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 neighbors of v (i.e., the set of vertices of G connected by an edge with v), and $N[v]$ is the set $N(v) \cup \{v\}$.

2.2 Trees

Throughout 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, $\mathrm{C}_{T^r}(v)$ 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 $v' \in \mathrm{C}_{T^r}(f_r(v))\backslash\{v\}$. Clearly, $T^r_{v,1}$ is defined if and only if $f_r(v)$ is defined.

Example 1 For an example of the above notions, consider the unlabeled free tree T given in Fig. 1. By choosing v_1 to be the root, we obtain the rooted tree T^{v_1}. For T^{v_1} we have that $\mathrm{C}_{T^{v_1}}(v_3) = \{v_4, v_5\}$ and $f_{v_1}(v_3) = v_1$. The (rooted) subtrees $T^{v_1}_{v_5,0}$ and $T^{v_1}_{v_5,1}$ of T^{v_1} corresponding to the vertex v_5 are also given in the figure. □

2.3 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 listed in time $O(|V_G| + |E_G|)$ (Tarjan 1972).

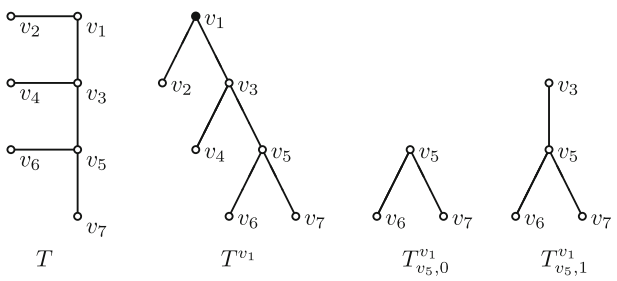


Fig. 1 A free tree T and the subtrees $T^{v_1}_{v_5,0}$ and $T^{v_1}_{v_5,1}$ of the rooted tree T^{v_1}

2.4 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) $\{u, v\} \in E_{G_1}$ if and only if $\{\varphi(u), \varphi(v)\} \in E_{G_2}$,
(ii) $\lambda_{G_1}(u) = \lambda_{G_2}(\varphi(u))$, and
(iii) $\lambda_{G_1}(\{u, v\}) = \lambda_{G_2}(\{\varphi(u), \varphi(v)\})$ holds for every $\{u, v\} \in E_{G_1}$.

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 list homomorphism introduced in Feder and Hell (1998), we define the following *constrained* subgraph isomorphism: Let G, H be graphs and $L_u \subseteq V_G$ for every $u \in V_H$. A *list subgraph isomorphism* from H to G satisfying $\{\langle u, L_u\rangle : u \in V_H\}$ is a subgraph isomorphism φ from H to G such that $\varphi(u) \in L_u$ for every $u \in V_H$. We denote by

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

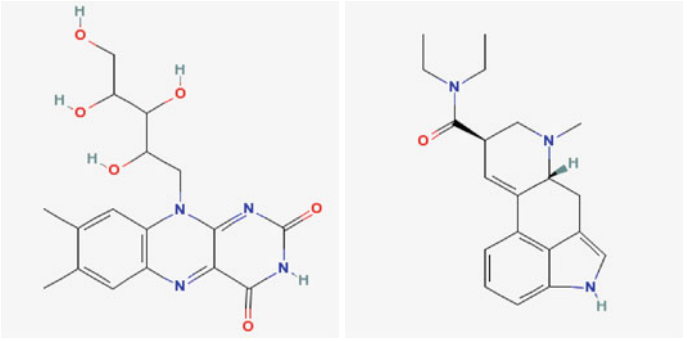
that there is a list subgraph isomorphism φ from H to G satisfying $\varphi(u) \in L_u$ for every $u \in V_H$. If $L_u = V_G$ for some particular vertex u, we will sometimes remove the pair $\langle u, L_u\rangle$ from the set below the arrow in the above notation.

Using the above notion, the *list subgraph isomorphism problem* can be defined as follows: *Given* graphs G and H, and sets $L_u \subseteq V_G$ for every $u \in V_H$, *decide* whether $H \xrightarrow[\{\langle u, L_u\rangle : u \in V_H\}]{} G$ holds. Notice that the list subgraph isomorphism problem is a generalization of the ordinary subgraph isomorphism where $L_u = V_G$ for every $u \in V_H$.

2.5 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. In Fig. 2, we give two planar molecular graphs. 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 $\gamma : [0, 1] \to \mathbb{R}^2$; 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* if and only if 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



(a) *Riboflavin* (Vitamin B_2) **(b)** *Lysergic Acid Diethylamide* (LSD)

Fig. 2 An outerplanar (**a**) and a non-outerplanar (**b**) molecular graph

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.

2.6 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.[3] The molecular graph on the left-hand side of Fig. 2 is outerplanar; the graph on the right-hand side is not outerplanar.

Throughout this work we consider connected outerplanar graphs and denote the set of connected outerplanar graphs by $\mathcal{O}$. For technical reasons we assume without loss of generality that each graph in $\mathcal{O}$ is labeled by the elements of $\mathbb{N}$ (i.e., 0 is not used as a label). Clearly, trees are outerplanar and hence, a graph is outerplanar if and only if each of its blocks is outerplanar (Harary 1971). Furthermore, as the blocks of a graph can be computed in linear time (Tarjan 1972) and outerplanarity of a block can be decided also in linear time (Lingas 1989; Mitchell 1979),[4] one can decide in linear time whether a graph is outerplanar.

[3] The class of outerplanar graphs was introduced and characterized in terms of minors in Chartrand and Harary (1967); a graph is outerplanar if and only if it contains neither K_4 nor $K_{2,3}$ as a minor, where K_n denotes the complete graph with n vertices and K_{n_1,n_2} denotes the complete bipartite graph with n_1 vertices on one side and with n_2 vertices on the other one. We also note that outerplanar graphs have *treewidth* 2 (see, e.g., Bodlaender 1998) implying that several NP-hard problems on general graphs can be solved efficiently for outerplanar graphs (e.g., the Hamiltonian cycle problem).

[4] We note that both outerplanarity testing algorithms in Lingas (1989) and Mitchell (1979) must be extended by an additional step checking condition (iii) of Theorem 2 in Mitchell (1979), as otherwise a class of non-outerplanar graphs will be misclassified by both algorithms.

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 (Harary 1971). This unique Hamiltonian cycle can be computed efficiently (see, e.g., Lingas 1989). Thus, G can be considered as an n-polygon with at most $n - 3$ non-crossing diagonals. In the proposition below we give an upper bound on the number of simple cycles of a biconnected outerplanar graph.

Proposition 2 *Let G be a biconnected outerplanar graph with d diagonals. Then G has at most 2^{d+1} cycles.*

Proof Since G is a biconnected outerplanar graph with d diagonals, the definition of weak dual graphs implies that G^* is a tree with $d + 1$ vertices. It also holds that there is a bijection between the set of biconnected subgraphs of G and the set of subtrees of G^*. Hence, the number of biconnected subgraphs of G is at most 2^{d+1}. The statement then follows from the fact that for every simple cycle C of G there is a biconnected subgraph G' of G such that the Hamiltonian cycle of G' is C. □

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 (Garey and Johnson 1979). The following stronger negative result is shown in Syslo (1982).

Theorem 3 *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 (Lingas 1989).[5]

Theorem 4 *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 Matula (1978) on subgraph isomorphism for the special case of trees.

Theorem 5 *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

$$O\left((|V_H|^{1.5}/\log|V_H|) \cdot |V_G|\right)$$

[5] Although this positive result has been shown for unlabeled graphs, the algorithm in Lingas (1989) can be generalized to labeled graphs without changing its complexity.

(see Shamir and Tsur 1999). For the sake of simplicity, in Sect. 5.4 we will generalize the algorithm in Matula (1978) to outerplanar graphs. We note that the practical runtime of our algorithm can also be improved using the idea for trees in Shamir and Tsur (1999). However, this will not improve the worst case complexity of our algorithm, as it is dominated by the complexity of deciding subgraph isomorphism between biconnected outerplanar graphs.

3 Problem setting

In this section we define the frequent subgraph mining problem for outerplanar graphs with respect to a constrained subgraph isomorphism that preserves the pattern graph's bridge and block structure. We start the problem description by introducing this embedding operator between outerplanar graphs.

3.1 BBP subgraph isomorphism

Let $G, H \in \mathcal{O}$. 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.

Example 6 Consider the outerplanar graphs given in Fig. 3. The vertices of the graphs are labeled by the elements of $\{1, 2, \ldots, 5\}$, and each edge by the same integer, say 1, not shown in the figure. For every $i = 1, 2, 3$, there is a unique ordinary subgraph isomorphism from H_i to G. It is a BBP subgraph isomorphism only for H_1; for H_2, it maps the bridge of H_2 to an edge belonging to a block of G, and for H_3, the two blocks of H_3 to the same block of G. Since BBP subgraph isomorphism is a constrained subgraph isomorphism, there is no BBP subgraph isomorphism from H_2 and H_3 to G. □

Notice that for trees, which are special outerplanar graphs (i.e., block-free), BBP subgraph isomorphism is equivalent to 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.

Regarding the motivation of using BBP subgraph isomorphism as embedding operator, we first note that by Theorem 3, the ordinary subgraph isomorphism problem

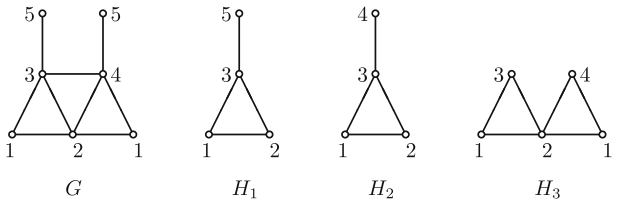


Fig. 3 BBP subgraph isomorphism: $H_1 \preccurlyeq_{BBP} G$, $H_2 \not\preccurlyeq_{BBP} G$, $H_3 \not\preccurlyeq_{BBP} G$

Fig. 4 A graph G with $2n+1$ vertices having more than 2^n connected graphs that are subgraph isomorphic to G, but only n connected graphs that are BBP subgraph isomorphic to G if all slanted edges are labeled by the same symbol, say 1, and all horizontal edges by another symbol, say 2 (not shown in the figure)

between outerplanar graphs is NP-complete. It follows from the reduction used to show this negative result in Syslo (1982) that subgraph isomorphism remains NP-complete even for connected well-behaved outerplanar graphs. In contrast, as we show in Sect. 5.4, BBP subgraph isomorphism between outerplanar graphs can be decided in polynomial time if the pattern graphs are connected. Beside this complexity reason, the use of BBP subgraph isomorphism may result in an exponentially smaller set of frequent patterns. As an example of such a case, consider the database consisting of the single well-behaved outerplanar graph G with $2n + 1$ vertices given in Fig. 4 and let the frequency threshold be 1. If all slanted edges of G are labeled by the same symbol, say 1, and all horizontal edges by another symbol, say 2, then there are more than 2^n frequent connected graphs that are subgraph isomorphic to G, but only n frequent connected graphs that are BBP subgraph isomorphic to G. Recent empirical studies in virtual screening (Schietgat et al. 2008, 2009) show that instead of predictive features based on frequent patterns with respect to ordinary subgraph isomorphism one can also use frequent patterns with respect to BBP subgraph isomorphism without drop in the predictive performance. These empirical studies also indicate that powerful predictors might be obtained by other non-standard embedding operators, motivating the theoretical and empirical study of such operators.

3.2 The FOSM problem

Using the above notions, we define the *frequent outerplanar subgraph mining problem* (FOSM) as follows: *Given*

(i) a finite set $\mathcal{D} \subseteq \mathcal{O}$ of *transaction graphs* and
(ii) an integer threshold $t > 0$,

list the set of all connected outerplanar graphs in $\mathcal{O}$ that match at least t graphs in $\mathcal{D}$ with respect to BBP subgraph isomorphism, i.e., the set

$$\mathcal{F}(\mathcal{D}, t) = \{H \in \mathcal{O} : |\{G \in \mathcal{D} : H \preccurlyeq_{BBP} G\}| \geq t\}. \quad (1)$$

Notice that by definition, $\mathcal{F}(\mathcal{D}, t)$ does not contain isomorphic graphs. Furthermore, it is closed downwards with respect to BBP subgraph isomorphism, i.e., $G_1 \in \mathcal{F}(\mathcal{D}, t)$ whenever $G_2 \in \mathcal{F}(\mathcal{D}, t)$ and $G_1 \preccurlyeq_{BBP} G_2$. Given $\mathcal{D}$ and t, the graphs belonging to $\mathcal{F}(\mathcal{D}, t)$ are called *t-frequent*.

The *parameters* of the FOSM problem are the cardinality N of the transaction dataset (i.e., $N = |\mathcal{D}|$) and the size M of the largest graph in $\mathcal{D}$ (i.e., $M = \max\{|V_G| : G \in \mathcal{D}\}$). Note that the cardinality of $\mathcal{F}(\mathcal{D}, t)$ can be exponential in the above parameters of $\mathcal{D}$. Clearly, in such cases it is impossible to list $\mathcal{F}(\mathcal{D}, t)$ in time polynomial

in the parameters of $\mathcal{D}$. To overcome this problem, the following classes are usually considered in the literature (see, e.g., Johnson et al. 1988): Let S be a set of cardinality N. Then its elements, say $s_1, \ldots, s_N$, are listed with

polynomial delay if the time until printing s_1, the time between printing s_i and s_{i+1} for every $i = 1, \ldots, N-1$, and the termination time after printing s_N is bounded by a polynomial of the size of the input,
incremental polynomial time if the time between printing s_i and s_{i+1} for every $i = 1, \ldots, N-1$ (resp. the termination time after printing s_N) is bounded by a polynomial of the combined sizes of the input and $\{s_1, \ldots, s_i\}$ (resp. $\{s_1, \ldots, s_N\}$),
output polynomial time (or *polynomial total time*) if S is printed in the combined sizes of the input and the *entire* set S.

Clearly, polynomial delay implies incremental polynomial time, which, in turn, implies output polynomial time. We also note 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 ith element for any $i \geq 1$.

Although several algorithms mining frequent connected subgraphs from arbitrary graphs with respect to subgraph isomorphism have demonstrated their performance empirically (see, also, Sect. 4 below), we note that, unless P = NP, this general problem cannot be solved in output polynomial time (Horváth et al. 2007; i.e., in the most liberal class in the above hierarchy). 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 Chi et al. (2005a). Since outerplanar graphs form a practically relevant graph class that naturally generalizes trees, by considering the FOSM problem we take a first step towards going beyond trees in frequent graph mining. Notice that the FOSM problem generalizes also the frequent itemset mining problem which is solved in incremental polynomial time by the Apriori algorithm (Agrawal et al. 1996).

The main contribution of this work is a levelwise algorithm solving the FOSM problem. The algorithm generates frequent patterns in incremental polynomial time if the transaction graphs are *well-behaved*, that is, their number of simple cycles is bounded by a polynomial of their size. As an example, the molecular graph of Riboflavin (see Fig. 2) is a well-behaved outerplanar graph because it contains only 6 simple cycles. Our positive result on well-behaved outerplanar graphs is of both theoretical and practical importance. From a theoretical viewpoint, outerplanar graphs form the first level beyond trees in the hierarchy defined in terms of minors.[6] Thus, to generalize the positive complexity result on mining frequent trees, it is natural to consider outerplanar graphs. From a practical viewpoint, as already discussed in Sect. 1, well-behaved outerplanar graphs form a practically relevant graph class because they have applications in various fields including telecommunication, electrical circuits, computational drug design, and bioinformatics.

[6] Trees, outerplanar graphs, and planar graphs form a natural hierarchy with respect to minors (Hedetniemi et al. 1971): a graph G is a tree if neither K_3 nor $K_{2,2}$ is a minor of G, it is outerplanar if neither K_4 nor $K_{2,3}$ is a minor of G, and it is planar if neither K_5 nor $K_{3,3}$ is a minor of G.

4 Related work

In the field of frequent pattern mining from graph structured data, two main settings can be distinguished: the *single network* and the *transactional* settings. In the single network setting (see, e.g., Calders et al. 2008; He and Singh 2007; Tong et al. 2007 for some recent results) the input consists of a single graph, which is usually large and does not have some special graph structure that could be exploited by the mining algorithms (e.g., the web graph or metabolic pathways).

The frequent pattern mining problem considered in this work belongs to the transactional setting where the input consists of a set of, usually small, disjoint graphs. There is a huge literature on this problem setting; an exhaustive overview of all related results on this problem goes beyond the scope of this paper. Since most of the related approaches allow arbitrary transaction graphs and use ordinary subgraph isomorphism as embedding operator, they deal with a computationally intractable task (Horváth et al. 2007), and resort therefore to various heuristics. Below we briefly overview three types of heuristics appearing in the algorithms.

4.1 Candidate generation heuristics

Several papers on mining frequent subgraphs devote a lot of attention to heuristics speeding up the candidate generation phase (e.g., Inokuchi et al. 2003 describing the AGM system). The particular challenge of these approaches is to avoid or at least to reduce the expensive isomorphism test needed to generate non-redundant patterns. The system FSG (Kuramochi and Karypis 2001) for example employs a canonical form for this purpose, but also includes further optimizations such as sparse graph representation and iterative single edge extension of the patterns. However, all the approaches relying on canonical forms for arbitrary graphs fail to guarantee polynomial delay enumeration as long as the complexity of graph isomorphism remains an open problem; an efficiently computable canonical form would imply the graph isomorphism problem to belong to P. For restricted graph classes, polynomial delay generation of candidate patterns is possible; for monotone graph classes e.g. such a polynomial delay algorithm based on automorphism group analysis has recently been shown in Ramon and Nijssen (2009). The system gSpan (Yan and Han 2002) avoids isomorphism tests by using a depth-first code for graphs and by constructing only pairwise non-isomorphic candidate patterns. Gaston (Nijssen and Kok 2004), one of the fastest frequent subgraph mining algorithms (see, also, Sect. 6), employs a similar strategy, but searches first path and tree patterns.

4.2 Support counting heuristics

Besides candidate generation, several heuristics have been developed for counting the support of candidate patterns. A common approach to this problem (see, e.g., again Gaston in Nijssen and Kok 2004) is to store information about embeddings of the frequent patterns that were found in earlier passes over the database. On the one hand, this strategy has the clear advantage that finding embeddings of larger patterns can

be speeded up in this way. On the other hand, however, it has the drawback that it takes a significant amount of memory and does not therefore meet scalability requirements. We note that our FOG system, the implementation of our algorithm described in Sect. 5, has a similar feature; the user can toggle whether or not to remember the embeddings between passes over the database. The optimization speeded up FOG with a factor up to five on small datasets, but neither FOG nor Gaston were able to store all such information for a transaction dataset consisting of more than 200,000 graphs.

4.3 Restricted pattern and/or transaction graph classes

Another approach applied by several systems is to control the complexity by considering restricted pattern languages and/or specific applications. Examples of this approach include MolFea (Kramer et al. 2001), which finds frequent linear fragments in databases of molecules, the recent algorithm in Maunz et al. (2009), which is restricted to mining tree-shaped patterns, and the system described in Borgelt and Berthold (2002), which also considers molecules and gains the speed-up by embedding fragments in parallel and by applying an incomplete pruning strategy. Our algorithm is not restricted to applications related to virtual screening because well-behaved outerplanar graph appear in other real-world applications as well (see Sect. 1 for some examples). Furthermore, though the pattern language in the problem setting we consider is not restricted to linear or tree-shaped patterns, the output is guaranteed to be complete with respect to BBP subgraph isomorphism.

5 The mining algorithm

In this section we present an Apriori-like (Agrawal et al. 1996) algorithm that solves the FOSM problem in incremental polynomial time for well-behaved outerplanar graphs. In Sect. 6 we report empirical results demonstrating the practical applicability of our algorithm on large graph datasets. In Sect. 7 we show that the problem can in fact be solved with polynomial delay.

The main steps of the algorithm are sketched in Algorithm 1. It takes as input a set $\mathcal{D} \subseteq \mathcal{O}$ of outerplanar graphs and an integer 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 $G \in \mathcal{O}$ such that the sum of the number of blocks of G and the number of bridges of G is k. (Isolated vertices are regarded as blocks.) 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 (1) a single vertex or (2) a single block or (3) a single edge. The first (resp. third) set, denoted $\mathcal{F}_v$ (resp. $\mathcal{F}_e$) in step 1, can be computed in linear time. The second set, denoted $\mathcal{F}_b$, can be computed in incremental polynomial time if the graphs in $\mathcal{D}$ are in addition all *well-behaved*; an efficient Apriori-based algorithm for this problem is presented in Sect. 5.2. We note that this is the only step in the algorithm making use of well-behavedness; the complexity of all other steps is independent of this property.

In step 2 of the algorithm, we compute the set of t-frequent 2-patterns, i.e., the set of graphs in $\mathcal{O}$ consisting of either (1) a path of length 2 or (2) two blocks having a

Algorithm 1 FREQUENTOUTERPLANARGRAPHS

Input: set $\mathcal{D}$ of outerplanar graphs and integer $t > 0$
Output: $\mathcal{F}(\mathcal{D}, t)$ defined in Eq. (1)

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

$$\mathcal{F}_v = \{H \in \mathcal{O} : |V_H| = 1 \text{ and } H \text{ is } t\text{-frequent}\}$$
$$\mathcal{F}_b = \{H \in \mathcal{O} : H \text{ is biconnected and } t\text{-frequent}\}$$
$$\mathcal{F}_e = \{H \in \mathcal{O} : |E_H| = 1 \text{ and } H \text{ is } t\text{-frequent}\}$$

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

$$\mathcal{F}_{ee} = \{H \in \mathcal{O} : H \text{ is a } t\text{-frequent path of length 2}\}$$
$$\mathcal{F}_{bb} = \{H \in G_1 \bowtie G_2 : G_1, G_2 \in \mathcal{F}_b \text{ and } H \text{ is } t\text{-frequent}\}$$
$$\mathcal{F}_{be} = \{H \in G_1 \bowtie G_2 : G_1 \in \mathcal{F}_b \text{ and } G_2 \in \mathcal{F}_e \text{ and } H \text{ is } t\text{-frequent}\}$$

3: **print** $\mathcal{L}_1 \cup \mathcal{L}_2$
4: $k = 2$
5: **while** $\mathcal{L}_k \neq \emptyset$ **do**
6: $k = k + 1$
7: $\mathcal{L}_k = \text{COMPUTEFREQUENTPATTERNS}(\mathcal{L}_{k-1})$
8: **endwhile**

common vertex or (3) a block and a bridge edge having a common vertex. We denote the corresponding three sets in step 2 by $\mathcal{F}_{ee}$, $\mathcal{F}_{bb}$, 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 G_1 and G_2 by contracting[7] a vertex from G_1 with a vertex from G_2 that have the same label. Clearly, $G_1 \bowtie G_2 \subseteq \mathcal{O}$ for every $G_1, G_2 \in \mathcal{O}$. The set $\mathcal{F}_{ee}$ of t-frequent paths of length *2* can be computed in polynomial 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 Sect. 5.4 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 loop 5–8, we compute the set of t-frequent k-patterns for every $k \geq 3$ in a way similar to the Apriori algorithm (Agrawal et al. 1996). In particular, we generate the set of frequent k-patterns from that of frequent $(k-1)$-patterns (step 7). The corresponding function COMPUTEFREQUENTPATTERNS is composed of (i) the generation of candidate k-patterns from the set of t-frequent $(k-1)$-patterns and (ii) the decision of t-frequency of the candidate patterns. We note that the set of frequent patterns will be printed in the function called in step 7. In Sects. 5.3 and 5.4 below we describe these steps in detail.

Putting together the results given in Theorems 14–16 stated in Sects. 5.2–5.4, respectively, we can formulate the main result of this paper:

[7] 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.

Theorem 7 *Algorithm 1 is correct and solves the FOSM problem in incremental polynomial time for well-behaved outerplanar graphs.*

Before going into the technical details in Sects. 5.2–5.4, in the next section we first describe a transformation and a canonical form for outerplanar graphs that will be used in different steps of the mining algorithm.

5.1 Canonical form

One time consuming step of mining frequent outerplanar graphs is to test whether a particular graph $H \in \mathcal{O}$ belongs to some set $S \subseteq \mathcal{O}$. To apply some advanced data structure (e.g., hash tables, B-trees, etc.) that allows fast search in large sets of outerplanar graphs, we define a total order on $\mathcal{O}$. Similarly to many other frequent graph mining algorithms, we solve this problem by assigning a list of integers, called *canonical form*, to each element of $\mathcal{O}$ such that

(i) two graphs have the same canonical form if and only if they are isomorphic and
(ii) for every $G \in \mathcal{O}$, the canonical form of G can be computed in time polynomial in $|V_G|$.

Using some canonical form satisfying the above properties, a total order on $\mathcal{O}$ can be defined by some total order (e.g. lexicographic) on the set of integer sequences assigned to the elements of $\mathcal{O}$. Furthermore, property (i) allows one to decide isomorphism between outerplanar graphs by comparing their canonical forms. Although isomorphism can be decided efficiently even for planar graphs (Hopcroft and Wong 1974), the canonical form for outerplanar graphs described in this section may be of some independent interest as well.

5.1.1 BB-trees

We first define a natural transformation on outerplanar graphs by means of contracting the non-cut vertices[8] of the blocks. More precisely, for a graph $G \in \mathcal{O}$, let $\widetilde{G}$ denote the graph derived from G by the following transformation: For every block B in G,

(i) introduce a new vertex v_B and label[9] it by 0,
(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 0, if v is adjacent to a bridge or to another block of G; otherwise remove v.

For a graph $G \in \mathcal{O}$ 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 $\lambda_{\widetilde{G}}(v) = 0$; otherwise it is the subgraph of G consisting of the single vertex corresponding to v. Since $\widetilde{G}$ is a tree by Proposition 9 below, we call it the *block and bridge tree (BB-tree)* of G.

[8] A *cut* or *articulation* vertex of a graph G is a vertex whose removal increases the number of connected components of G.

[9] We recall that by definition, 0 is not used as a label in any of the graphs in the input database.

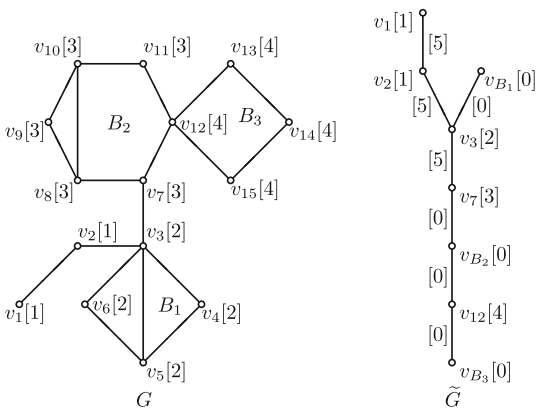


Fig. 5 An outerplanar graph G containing the blocks B_1, B_2, and B_3, and its BB-tree $\widetilde{G}$. Vertex and edge labels are given in *brackets*. Each edge of G (*left-hand* side) is labeled by 5 which is not denoted in the figure

Example 8 Consider the outerplanar graph $G \in \mathcal{O}_1$ given in Fig. 5. Each edge of G is labeled by the same symbol 5 which is not denoted in the figure (left-hand side). G contains three blocks: B_1, B_2, and B_3. To construct the BB-tree $\widetilde{G}$ of G, we introduce the vertices v_{B_1}, v_{B_2}, and v_{B_3} for the blocks B_1, B_2, and B_3, respectively, and label them by 0. Since the vertex v_3 in $\tau(v_{B_1}) = B_1$ belongs also to a bridge of G, we connect it with v_{B_1} and label the new edge by 0. The other three vertices of B_1 (i.e., v_4, v_5, and v_6) are removed, as they do not belong to any other block or bridge of G. Applying the same transformation to B_2 and B_3, we obtain the BB-tree of G given on the right-hand side of the figure. □

We now state some basic properties of $\widetilde{G}$ that will be used many times in what follows.

Proposition 9 *Let $G \in \mathcal{O}$. Then*

(i) $|V_{\widetilde{G}}| = 1$ *if and only if* $|V_G| = 1$ *or G is biconnected,*
(ii) *for every $e \in E_{\widetilde{G}}$, at most one vertex of e is labeled by* 0*, and*
(iii) *$\widetilde{G}$ is a free tree.*

Proof The proof of (i) and (ii) follows directly from the construction of $\widetilde{G}$. To see (iii), suppose that $\widetilde{G}$ has a cycle C. Then C must contain a vertex v labeled by 0, as otherwise C would be a cycle of G. But this implies that the biconnected subgraph $\tau(v)$ of G corresponding to v is not maximal contradicting the definition of $\widetilde{G}$. □

5.1.2 The Canonical form

Using the above notions and notations, we define the canonical form of G by means of $\widetilde{G}$. By (iii) of Proposition 9, $\widetilde{G}$ is a free tree. We utilize this property and generalize the depth-first canonical string representation for free trees to outerplanar graphs (see Chi et al. 2005a for an overview on canonical string representations for trees). Given some distinguished vertex $r \in V_{\widetilde{G}}$, we assign recursively a list[10] $\rho_r(v)$ of integers to

[10] In the definition of canonical form, we apply the standard Prolog notation for lists. The concatenation of two lists X and Y is denoted by $X \cdot Y$.

every vertex v of $\widetilde{G}^r$, and define an encoding $\rho_r(G)$ of G with respect to r by the list $\rho_r(r)$ associated with r. To define $\rho_r(v)$ for a vertex $v \in V_{\widetilde{G}}$, we distinguish two cases:

(i) Suppose $\lambda_{\widetilde{G}}(v) \neq 0$, i.e., $\tau(v) \in V_G$. Then we define $\rho_r(v)$ by

$$\rho_r(v) = [l_0] \cdot \left[l_{i_1} | \rho_r\left(v_{i_1}\right)\right] \cdot \ldots \cdot \left[l_{i_k} | \rho_r\left(v_{i_k}\right)\right] \cdot [-1], \tag{2}$$

where

- l_0 is the label of v,
- $\left\{v_{i_1}, \ldots, v_{i_k}\right\}$ is the set of children of v in $\widetilde{G}^r$,
- for every $q = 1, \ldots, k$, l_{i_q} is the label of the edge connecting v and v_{i_q} if $\lambda_{\widetilde{G}}(v_{i_q}) \neq 0$; otherwise $l_{i_q} = -2$, and
- $\left[l_{i_p} | \rho_r\left(v_{i_p}\right)\right] \preceq \left[l_{i_q} | \rho_r\left(v_{i_q}\right)\right]$ for every $1 \leq p < q \leq k$, where $\preceq$ denotes the lexicographic order on the set of integer lists.

(ii) Suppose $\lambda_{\widetilde{G}}(v) = 0$. Then, by definition, $\tau(v)$ is a block of G and hence, as G is outerplanar, it has a unique Hamiltonian cycle H. Let ℓ denote the length of H. Clearly, there are 2ℓ sequences of vertices defining H. Let $s = v_1, \ldots, v_\ell$ be such a sequence. For s, we define the list $\rho_{r,s}(v)$ of integers by

$$\rho_{r,s}(v) = [-3, \ell] \cdot S_V \cdot S_E \cdot [\delta] \cdot S_D \cdot \left[i_1 | \rho_r\left(v_{i_1}\right)\right] \cdot \ldots \cdot \left[i_k | \rho_r\left(v_{i_k}\right)\right] \cdot [-1], \tag{3}$$

where

- $S_V = [\lambda_G(v_1), \ldots, \lambda_G(v_\ell)]$,
- $S_E = [\lambda_G(\{v_1, v_2\}), \ldots, \lambda_G(\{v_{\ell-1}, v_\ell\}), \lambda_G(\{v_\ell, v_1\})]$,
- δ is the number of diagonals in $\tau(v)$,
- S_D is the unique list $[i_1, j_1, l_1, \ldots, i_\delta, j_\delta, l_\delta]$ 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) $\left\{v_{i_q}, v_{j_q}\right\}$ is a diagonal with label l_{i_q} in G for every $q = 1, \ldots, \delta$,
- $1 \leq i_1 < \ldots < i_k \leq \ell$, and
- $\left\{v_{i_1}, \ldots, v_{i_k}\right\}$ 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 the sequences defining H whose first element is $\tau(f_r(v))$. For the second case, notice that $\lambda_{\widetilde{G}}(f_r(v)) \neq 0$ by (ii) of Proposition 9, i.e., $\tau(f_r(v))$ is a vertex in G. Using the above definitions of $\rho_{r,s}$ and $\mathcal{S}$, we define $\rho_r(v)$ by

$$\rho_r(v) = \min_{s \in \mathcal{S}} \rho_{r,s}(v). \tag{4}$$

Given G and $r \in V_{\widetilde{G}}$, the lists of integers assigned to the vertices of $\widetilde{G}^r$ can be computed efficiently by traversing $\widetilde{G}^r$ in postorder.

Using the above definition of ρ_r, the canonical form of an outerplanar graph can be defined by the *center-based* canonical form of free trees (see, e.g., Chi et al. 2005a). 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 form of G by

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

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

$$\rho(G) = \min\{\rho_{r_1}(G_1) \cdot [-4, l, -4] \cdot \rho_{r_2}(G_2), \rho_{r_2}(G_2) \cdot [-4, l, -4] \cdot \rho_{r_1}(G_1)\} \ , (5)$$

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.

Example 10 Consider the outerplanar graph G and its BB-tree $\widetilde{G}$ given in Fig. 5. As a first step, we have to compute the set C of centers of the free tree $\widetilde{G}$ given by

$$\begin{aligned} C &= \{v \in V_{\widetilde{G}} : \max_{u \in V_{\widetilde{G}}} \mathrm{dist}(u, v) \leq \max_{u \in V_{\widetilde{G}}} \mathrm{dist}(u, w) \quad \text{for every } w \in V_{\widetilde{G}}\} \\ &= \{v_7\}. \end{aligned}$$

Thus, to compute the canonical form of G, we have to traverse the rooted tree $\widetilde{G}_{v_7}$ in postorder. Applying the corresponding Eqs. from (2) and (4), we get the following definitions for $\rho_{v_7}(v)$ for every $v \in V_{\widetilde{G}}$:

$$\begin{aligned}
\rho_{v_7}(v_1) &= [1, -1], \\
\rho_{v_7}(v_2) &= [1] \cdot [5|\rho_{v_7}(v_1)] \cdot [-1] \\
&= [1, 5, 1, -1, -1], \\
\rho_{v_7}(v_{B_1}) &= [-3, 4] \cdot [2, 2, 2, 2] \cdot [5, 5, 5, 5] \cdot [1] \cdot [1, 3, 5] \cdot [-1], \\
\rho_{v_7}(v_3) &= [2] \cdot [-2|\rho_{v_7}(v_{B_1})] \cdot [5|\rho_{v_7}(v_2)] \cdot [-1] \\
&= [2, -2, -3, 4, 2, 2, 2, 2, 5, 5, 5, 5, 1, 1, 3, 5, -1, 5, 1, 5, 1, -1, -1, -1], \\
\rho_{v_7}(v_{B_3}) &= [-3, 4] \cdot [4, 4, 4, 4] \cdot [5, 5, 5, 5] \cdot [0] \cdot [-1], \\
\rho_{v_7}(v_{12}) &= [4] \cdot [-2|\rho_{v_7}(v_{B_3})] \cdot [-1] \\
&= [4, -2, -3, 4, 4, 4, 4, 4, 5, 5, 5, 5, 0, -1, -1], \\
\rho_{v_7}(v_{B_2}) &= [-3, 6] \cdot [3, 3, 3, 3, 3, 4] \cdot [5, 5, 5, 5, 5, 5] \cdot [1] \cdot [2, 4, 5] \\
&\quad \cdot [6|\rho_{v_7}(v_{12})] \cdot [-1] \\
&= [-3, 6, 3, 3, 3, 3, 3, 4, 5, 5, 5, 5, 5, 5, 1, 2, 4, 5, 6, 4, -2, \\
&\quad -3, 4, 4, 4, 4, 4, 5, 5, 5, 5, 0, -1, -1, -1], \\
\rho_{v_7}(v_7) &= [3] \cdot [-2|\rho_{v_7}(v_{B_2})] \cdot [5|\rho_{v_7}(v_3)] \cdot [-1] \\
&= [3, -2, -3, 6, 3, 3, 3, 3, 3, 4, 5, 5, 5, 5, 5, 5, 1, 2, 4, 5, 6, 4, -2, \\
&\quad -3, 4, 4, 4, 4, 4, 5, 5, 5, 5, 0, -1, -1, -1, 5, 2, -2, \\
&\quad -3, 4, 2, 2, 2, 2, 5, 5, 5, 5, 1, 1, 3, 5, -1, 5, 1, 5, 1, -1, -1, -1, -1].
\end{aligned}$$

By definition, the canonical form $\rho(G)$ of G is given by $\rho_{v_7}(v_7)$. □

In Theorem 13 below we show that ρ is indeed a canonical form for outerplanar graphs. To prove the theorem, we need some lemmas.

Lemma 11 *For every $G_1, G_2 \in \mathcal{O}$ it holds that $G_1 \simeq G_2$ if and only if for every $r_1 \in V_{\widetilde{G}_1}$ there is a $r_2 \in V_{\widetilde{G}_2}$ such that $\rho_{r_1}(G_1) = \rho_{r_2}(G_2)$.*

Proof The necessity is automatic because for every $G \in \mathcal{O}$ and $r \in V_{\widetilde{G}}$, the BB-tree $\widetilde{G}$ and the encoding $\rho_r(G)$ are unique. The sufficiency follows directly by induction on the number of occurrences of -1 in $\rho_{r_1}(G_1)$. □

Lemma 12 *Let $G \in \mathcal{O}$ with $|V_G| = n$. Then $\rho(G)$ can be computed in time $O\left(n^2 \log n\right)$.*

Proof Since $|E_G|$ is bounded by $2n - 3$ (Mitchell 1979) and the sets of blocks and bridges of a graph can be listed with linear delay (Tarjan 1972), the set of blocks and the set of bridges of G can be computed in time $O(n)$. This implies that $\widetilde{G}$ can be constructed in time $O(n)$. The center (or centers) of $\widetilde{G}$ can also be determined in time $O(n)$. Thus, to prove the statement, it is sufficient to show that $\rho_r(v)$ can be computed in time $O\left(n^2 \log n\right)$ for every v of $\widetilde{G}^r$.

For each vertex v of $\widetilde{G}^r$, the canonical strings computed for v's children can be sorted by $O(k_v \log k_v)$ string comparisons, where k_v is the number of children of v. Since the length of $\rho_r(v)$ is bounded by $O(n)$, each string comparison can be performed in $O(n)$ time. Thus, the total time of computing $\rho_r(v)$ is bounded by

$$\sum_{v \in V_{\widetilde{G}^r}} n k_v \log k_v \leq n \log n \sum_{v \in V_{\widetilde{G}^r}} k_v = O\left(n^2 \log n\right) ,$$

as $k_v \leq n$ and $\sum_{v \in V_{\widetilde{G}^r}} k_v = O(n)$. □

Combining Lemmas 11 and 12 above, we have the main result of this section:

Theorem 13 *For every $G \in \mathcal{O}$, $\rho(G)$ is an efficiently computable canonical form of G.*

5.2 Mining frequent biconnected graphs

In this section we present Algorithm 2 that computes the set $\mathcal{F}_b$ of t-frequent biconnected graphs used in step 1 of Algorithm 1 and show that it runs in incremental polynomial time for well-behaved outerplanar graphs.

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. From Read and Tarjan (1975) and Tarjan (1972) 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 $\mathcal{D}$ if the graphs in $\mathcal{D}$ are all well-behaved. We note that well-behavedness is exploited only in this step in the entire mining algorithm; the efficiency results of all other steps

Algorithm 2 FREQUENTBICONNECTEDGRAPHS

Input: set $\mathcal{D}$ of outerplanar graphs and integer $t > 0$
Output: $\mathcal{F}_b$ defined in step 1 of Algorithm 1

1: **let** $\mathcal{L}_0$ be the set of t-frequent cycles in $\mathcal{D}$
2: $k = 0$
3: **while** $\mathcal{L}_k \neq \emptyset$ **do**
4: **print** $\mathcal{L}_k$
5: $k = k + 1$
6: **let** $\mathcal{C}_k$ be the set of biconnected outerplanar graphs H such that
H has exactly k diagonals and $H \ominus \Delta \in \mathcal{L}_{k-1}$ for every diagonal Δ of H
7: $\mathcal{L}_k = \{H \in \mathcal{C}_k : H \text{ is } t\text{-frequent}\}$
8: **endwhile**

hold (and are formulated) for arbitrary (i.e., not necessarily well-behaved) outerplanar graphs. As a cycle may be compared to many other cycles, to decide isomorphism, we use the canonical form described in Sect. 5.1.

In loop 3–8 of Algorithm 2, we compute the sets of t-frequent biconnected graphs containing k diagonals for every $k > 0$ integer. In particular, in step 6 we compute the set $\mathcal{C}_k$ of candidate biconnected outerplanar graphs H satisfying the following conditions: H has exactly k diagonals and the removal of any diagonal from H, denoted by $\ominus$ in step 6, 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 label l, 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 forms.
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 $i = 1, 2$, we 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 does not 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 $2 \cdot |V_{H_1}|$. 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.

One can show that the method described above is complete, i.e., for every $k > 0$, $\mathcal{C}_k$ contains the set of t-frequent biconnected outerplanar graphs with exactly k diagonals.

Finally, in order to compute the set of t-frequent biconnected graphs from the $\mathcal{C}_k$'s (step 7), 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 (Tarjan 1972), it is therefore sufficient to consider the efficiency of subgraph isomorphism between biconnected outerplanar graphs. Theorem 17 in Sect. 5.4 below deals with a more general case implying that this problem can be solved in cubic time. Putting the above results together, we have the following theorem.

Theorem 14 *Algorithm 2 is correct and computes the set of t-frequent biconnected outerplanar graphs in incremental polynomial time if the input dataset $\mathcal{D}$ consists of well-behaved outerplanar graphs.*

Proof The proof of the correctness is similar to that of the Apriori algorithm and therefore we only show the statement's second part concerning the complexity. In step 1, we compute the set $\mathcal{L}_0$ of t-frequent cycles by listing all simple cycles occurring in the graphs in $\mathcal{D}$ and calculating their support counts. Since simple cycles of a graph can be listed with delay linear in the number of edges (Tarjan 1972), the set of all simple cycles can be computed in time $O(NKM)$, where K is an upper bound on the number of simple cycles of the transaction graphs. For a simple cycle C, we can compute its canonical form in time $O\left(M^2\right)$. Storing the canonical form of cycles in a prefix tree, we need $O(M)$ time either for inserting C or incrementing its frequency counter. Thus, the total time to find and process a cycle is bounded by $O\left(M^2\right)$. Since the total number of cycles in $\mathcal{D}$ is at most $O(NK)$, step 1 can be performed in time $O\left(NKM^2\right)$, which in turn is bounded by a polynomial of the size of $\mathcal{D}$ if it contains only well-behaved outerplanar graphs.

Regarding the complexity of loop 3–8, we first note that $|\mathcal{C}_k|$ is bounded by $O(M|\mathcal{L}_{k-1}|)$ for every $k > 0$ and that the conditions of step 6 for H can be decided in time $O\left(M^3\right)$. Thus $\mathcal{C}_k$ can be computed in time $O\left(M^4|\mathcal{L}_{k-1}|\right)$. As subgraph isomorphism between biconnected outerplanar graphs can be decided in cubic time by Theorem 4, from $\mathcal{C}_k$ we can compute $\mathcal{L}_k$ in time $O\left(NM^3|\mathcal{C}_k|\right)$ (step 7). Putting all these together we get that for every $k > 0$, $\mathcal{L}_k$ can be computed from $\mathcal{L}_{k-1}$ in time $O\left(NM^4|\mathcal{L}_{k-1}|\right)$, i.e., in incremental polynomial time. □

5.3 Candidate generation

In step 7 of Algorithm 1, we generate the set of frequent k-patterns. The pseudocode of this function is given in Algorithm 3. Using a generalization of the candidate generation algorithm for free trees described in Chi et al. (2005b), the algorithm 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 bridges is k.) Applying the candidate generation principle of the Apriori

Algorithm 3 COMPUTEFREQUENTPATTERNS

Input: set $\mathcal{L}_{k-1}$ of frequent $(k-1)$-patterns for some $k \geq 2$
Output: set $\mathcal{L}_k$ of candidate k-patterns

1: $\mathcal{C}_k = \mathcal{L}_k = \emptyset$
2: **forall** $G_1, G_2 \in \mathcal{L}_{k-1}$ s.t. G_1 and G_2 have the same core and $G_1 \leq G_2$ **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: **let** C be the graph obtained by attaching g_2 in G_1 to g_1' consistently with G_2
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 $g \in \text{Leaf}(C)$ **then**
9: add C to $\mathcal{C}_k$
10: **if** C is t-frequent **then**
11: **print** C and add it to $\mathcal{L}_k$
12: **endfor**
13: **endfor**
14: **endfor**
15: **return** $\mathcal{L}_k$

algorithm (Agrawal et al. 1996), each candidate is obtained by joining two frequent $(k-1)$-patterns that have an isomorphic $(k-2)$-pattern core.

In the outer loop 2–14 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 3–13, 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 that g_1 and g_2 are isomorphic leaf subgraphs. For an outerplanar graph G, Leaf(G) denotes the set of nodes of $\widetilde{G}$ that have degree 1. 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 5–12) 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 with respect to a center of $\widetilde{C}$. By definition, a vertex representing

a leaf subgraph of C is always a leaf in $\widetilde{C}$. If this condition holds then we add C to the set $\mathcal{C}_k$ 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). For every candidate pattern C added to $\mathcal{C}_k$, we check in step 10 whether or not it is frequent. If yes, we print and add it to the set $\mathcal{L}_k$ of frequent k-patterns. We will discuss in the next section the algorithmic issues of deciding frequency in step 10.

Theorem 15 *The set $\mathcal{C}_k$ computed by Algorithm 3 is a superset of the set of t-frequent k-patterns. Furthermore, $|\mathcal{C}_k|$ is polynomial in $|\mathcal{L}_{k-1}|$ and Algorithm 3 computes $\mathcal{C}_k$ in time polynomial in the size of $\mathcal{L}_{k-1}$, not counting the complexity of frequency testing in step 10.*

Proof An element of $\mathcal{L}_k$ is determined completely by a choice of $G_1, G_2 \in \mathcal{L}_{k-1}, g_1, g_1' \in Leaf(G_1)$ and $g_2 \in Leaf(G_2)$. Clearly, this is polynomial in the sizes of $\mathcal{L}_{k-1}$ and M. It is straightforward to show that the time complexity is polynomial in these parameters. □

5.4 BBP subgraph isomorphism

Algorithms 2 and 3 contain the steps of deciding whether a candidate pattern $H \in \mathcal{O}$ 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 3), Theorem 16, 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 well-behavedness. The connectivity is necessary, as otherwise the problem would generalize the NP-complete subforest isomorphism problem (Garey and Johnson 1979). We note that the result of Theorem 16 generalizes the positive result on subtree isomorphism given in Theorem 5 and may thus be of some interest in itself.

Theorem 16 *For every $G \in \mathcal{O}$ and connected graph $H \in \mathcal{O}$, $H \preccurlyeq_{BBP} G$ can be decided in polynomial time.*

The proof of the above theorem is based on a combination of the ideas given in Matula (1978) for unlabeled subtree isomorphism and Lingas (1989) for subgraph isomorphism between unlabeled biconnected outerplanar graphs. We first define some further notations and assertions. We start with a theorem generalizing Theorem 4, the positive result from Lingas (1989) on ordinary subgraph isomorphism between *unlabeled* biconnected outerplanar graphs to list subgraph isomorphism between *labeled* biconnected outerplanar graphs.

Theorem 17 *Let $G, H \in \mathcal{O}$ 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 4 in Lingas (1989). 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 (see Lingas 1989 for the details). This constraint can easily be generalized to list subgraph isomorphism between labeled biconnected outerplanar graphs without changing the asymptotic time complexity of the special case considered in Theorem 4 in (Lingas 1989). □

We need some further notations. Let $G \in \mathcal{O}$ and $T^v_{\widetilde{G}}$ be a subtree of $\widetilde{G}$ rooted at v. Then $G\left[T^v_{\widetilde{G}}\right]$ denotes the subgraph of G induced by $\bigcup_{v' \in V_{T^v_{\widetilde{G}}}} V_{\tau(v')}$. Furthermore, for a graph $H \in \mathcal{O}$ and subtree $T^u_{\widetilde{H}}$ of $\widetilde{H}$ rooted at u, $H\left[T^u_{\widetilde{H}}\right] \subseteq_{BBP} G\left[T^v_{\widetilde{G}}\right]$ denotes that there is a BBP subgraph isomorphism from $H\left[T^u_{\widetilde{H}}\right]$ to $G\left[T^v_{\widetilde{G}}\right]$ mapping $V_{\tau(u)}$ to $V_{\tau(v)}$. The proof of the following proposition is immediate from the definitions.

Proposition 18 *Let $G, H \in \mathcal{O}$ and $r \in V_{\widetilde{G}}$. Then $H \preccurlyeq_{BBP} G$ if and only if there are $u \in V_{\widetilde{H}}$ and $v \in V_{\widetilde{G}}$ such that*

$$H\left[\widetilde{H}^u\right] \subseteq_{BBP} G\left[\widetilde{G}^r_{v,0}\right]. \tag{6}$$

Applying the above proposition, $H \preccurlyeq_{BBP} G$ can be decided as follows: Select an arbitrary vertex $r \in V_{\widetilde{G}}$ and decide for every $u \in V_{\widetilde{H}}$ and for every $v \in V_{\widetilde{G}}$, whether (6) holds. Below we show that for given u and v, (6) can be decided by determining first for every $w \in N[u]$, whether

$$H\left[\widetilde{H}^w_{u,\overline{\delta}_{u,w}}\right] \subseteq_{BBP} G\left[\widetilde{G}^r_{v,\overline{\delta}_{u,w}}\right] \tag{7}$$

holds, where $\overline{\delta}$ is the complement of the Kronecker delta function (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

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

Notice that by definition, $\widetilde{H}^u = \widetilde{H}^u_{u,0}$ and hence, by Proposition 18 and Eq. (6) we have that

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

Depending on the label of u, we distinguish two cases and characterize (7) accordingly by one of the following two lemmas. We recall that $N(v)$, $N[v]$, and $\mathrm{C}_{T^r}(v)$ denote the

set of neighbors of v, the set $N(v) \cup \{v\}$, and the set of children of v in T^r, respectively. Furthermore, a vertex of a BBP tree of an outerplanar graph G is labeled by 0 if and only if it represents a block of G. We first consider the case when u represents a block of H.

Lemma 19 *Let $G, H \in \mathcal{O}$, $r, v \in V_{\widetilde{G}}$, $u \in V_{\widetilde{H}}$, $w \in N[u]$, and suppose $\lambda_{\widetilde{H}}(u) = 0$ (i.e., $\tau(u)$ is a block in H). Then*

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

if and only if there is an injection $\varphi : \mathrm{C}_{\widetilde{H}^w}(u) \to \mathrm{C}_{\widetilde{G}^r}(v)$ such that

(i) $H\left[\widetilde{H}^{u}_{u',1}\right] \subseteq_{BBP} G\left[\widetilde{G}^{r}_{\varphi(u'),1}\right]$ *for all* $u' \in \mathrm{C}_{\widetilde{H}^w}(u)$,
(ii) $\tau(u) \xrightarrow[\{\langle \tau(u'), L_{u'} \rangle : u' \in N(u)\}]{} \tau(v)$, *where*

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

Proof sketch Since $\lambda_{\widetilde{H}}(u) = 0$, $\tau(u)$ is biconnected. By (ii) of Proposition 9, for every $u' \in N(u)$ we have that $\lambda_{\widetilde{H}}(u') \neq 0$ and hence, $\tau(u')$ must be a vertex of the block $\tau(u)$. The proof can be shown by considering the cases $u = w$ and $u \neq w$ separately. □

We now consider the case when u represents a vertex of H.

Lemma 20 *Let the graphs G, H and the vertices r, v, u, and w be defined as in Lemma 19, and suppose that $\lambda_{\widetilde{H}}(u) \neq 0$ (i.e., $\tau(u)$ is a vertex in H). Then*

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

if and only if there is an injection $\varphi : \mathrm{C}_{\widetilde{H}^w}(u) \to \mathrm{C}_{\widetilde{G}^r}(v)$ such that

(i) $H\left[\widetilde{H}^{u}_{u',1}\right] \subseteq_{BBP} G\left[\widetilde{G}^{r}_{\varphi(u'),1}\right]$ *for all* $u' \in \mathrm{C}_{\widetilde{H}^w}(u)$,
(ii) *and if $w \neq u$ then*
- $r \neq v$ *(i.e., $f_{\widetilde{G}^r}(v)$ is defined),*
- $\lambda_{\widetilde{H}}(w) = \lambda_{\widetilde{G}}\left(f_{\widetilde{G}^r}(v)\right)$,
- $\lambda_{\widetilde{H}}(\{u, w\}) = \lambda_{\widetilde{G}}\left(\left\{v, f_{\widetilde{G}^r}(v)\right\}\right)$, *and*
- $\tau(w) \xrightarrow[\{\langle \tau(u), \{\tau(v)\} \rangle\}]{} \tau\left(f_{\widetilde{G}^r}(v)\right)$ *if* $\lambda_{\widetilde{H}}(w) = 0$ *(i.e., $\tau(w)$ is a block).*

Proof sketch The proof can be shown by considering the following cases separately: (1) $u = w$, (2) $u \neq w$ and $\lambda_{\widetilde{H}}(w) = 0$, and (3) $u \neq w$ and $\lambda_{\widetilde{H}}(w) \neq 0$. □

Proof (Proof of Theorem 16) Using the above notions and statements, we are now ready to sketch a bottom-up algorithm deciding $H \preccurlyeq_{BBP} G$ for some $G, H \in \mathcal{O}$ in polynomial time.

We first note that it is sufficient to consider the case when G and H satisfy

$$1 < |V_{\widetilde{H}}| \leq |V_{\widetilde{G}}|. \tag{8}$$

Indeed, if $|V_{\widetilde{H}}| = 1$ then, by (i) of Proposition 9, 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 17 can be applied. Furthermore, if $|V_{\widetilde{H}}| > |V_{\widetilde{G}}|$ then there is no BBP subgraph isomorphism from H to G.

Given $G, H \in \mathcal{O}$ satisfying (8), 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 (8) implies that $r \neq v$ and thus $v' = f_{\widetilde{G}^r}(v)$ is defined. For this case we have that $S(v, u) = N(u)$ if and only if (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 $\lambda_{\widetilde{H}}(u) = 0$ (resp. $\lambda_{\widetilde{H}}(u') = 0$) then there is a list subgraph isomorphism from $\tau(u)$ to $\tau(v)$ (resp. $\tau(u')$ to $\tau(v')$) mapping the vertex $\tau(u')$ to $\tau(v')$ (resp. $\tau(u)$ to $\tau(v)$). Otherwise, i.e., when conditions (i) and (ii) 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 (7).
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 that $\lambda_{\widetilde{H}}(u) = 0$, 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 9. By Lemma 19 we have that

- $u \in S(v, u)$ if and only if 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 $i = 1, \ldots, s$, $u_i \in S(v, u)$ if and only if $r \neq v$ and there is a list subgraph isomorphism from $\tau(u)$ to $\tau(v)$ mapping $\tau(u_i)$ to $\tau\big(f_{\widetilde{G}^r}(v)\big)$ 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 17. For the second case, i.e., when $\lambda_{\widetilde{H}}(u) \neq 0$, 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 20 we have that

- $u \in S(v,u)$ if and only if B_0 has a maximum matching of size s and
- for every $i = 1, \ldots, s$, $u_i \in S(v,u)$ if and only if 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 $\lambda_{\widetilde{H}}(u_i) = 0$ then there is a list subgraph isomorphism from $\tau(u_i)$ to $\tau\left(f_{\widetilde{G}^r}(v)\right)$ which maps the vertex $\tau(u)$ to $\tau(v)$.[11]

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 of Theorem 16. □

6 Experimental evaluation

In Sect. 1 we listed some examples of real-world applications of well-behaved outerplanar graphs to demonstrate that they form a practically relevant graph class. For the telecommunication and electrical circuit applications mentioned in Sect. 1, the underlying outerplanar graphs are always cactus graphs (also known as *Husimi trees*). Since cactus graphs have no diagonals, the number of their simple cycles is bounded by the number of their vertices. Thus, cactus graphs are well-behaved. As cactus graphs form a very restricted subclass of well-behaved outerplanar graphs, in our experiments we consider the other two, much more challenging, applications discussed in Sect. 1 because they involve more complex well-behaved outerplanar graphs.

In particular, in Sect. 6.1 we first report experimental results with the subset of the NCI dataset formed by the outerplanar graphs; the outerplanar graphs in this dataset turn out to be well-behaved and dominate the dataset consisting of 250,251 chemical compounds. One of the goals of these experiments is to compare the performance of our algorithm with other state-of-the-art frequent pattern mining algorithms on a relatively large graph dataset. In Sect. 6.2 we then consider another well-behaved outerplanar graph dataset consisting of 269 mRNA secondary structures. In contrast to the NCI dataset, RNA secondary structures are usually larger and more complex graphs than the molecular graphs of pharmacological compounds. Though this is a relatively small dataset, our experiments clearly demonstrate the relative compactness of the frequent pattern class defined by BBP subgraph isomorphism against that defined by ordinary subgraph isomorphism.

For the purpose of these experimental studies, we implemented our algorithm in the FOG (Frequent Outerplanar subGraphs) system.[12]

6.1 The NCI dataset

Proposition 2 implies that outerplanar graphs having at most d diagonals per block, where $d \geq 0$ is some constant, are always well-behaved. This class of outerplanar graphs will be referred to as *d-tenuous* outerplanar graphs. To demonstrate the practical

[11] 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 Shamir and Tsur 1999 for the details).

[12] For the implementation of the FOG system and for the two datasets used in our experiments see http://www.cs.kuleuven.be/~janr/fog/.

applicability of our algorithm on a relatively large dataset, we used the NCI molecular graph dataset, a popular graph database in the frequent pattern mining community, that consists of 250,251 chemical compounds.

For our work, it was important to recognize that 236,180 (i.e., 94.3%) of these compounds have an outerplanar molecular graph. In the experiments, we have removed the non-outerplanar graphs from the dataset. We note that the resulting dataset is still much larger than the other subsets of the NCI dataset usually considered for empirical evaluations of frequent pattern mining algorithms; the largest such popular subset is the NCI-HIV dataset consisting of only 42,689 compounds (see, e.g., Deshpande et al. 2005). Altogether, the outerplanar molecules contain 423,378 blocks, with up to 11 diagonals per block. However, 236,083 (i.e., 99.99%) of the outerplanar molecular graphs have only at most five diagonals per block. Thus, the dataset obtained by removing the non-outerplanar molecules is 11-tenuous.

Our experimental results on the 236,180 outerplanar graphs are given in Table 1. It shows the number of candidate and frequent k-patterns (columns #C and #FP, respectively) discovered for $k = 1, \ldots, 29$, as well as the runtime (column T) in seconds for the computation and evaluation of the candidates using frequency thresholds 10, 5, and 2%. As expected, the number and the size of the discovered patterns is much larger when the frequency threshold is lower. One of our observations is that the database contains a wide variety of structures, as a low relative frequency threshold is needed to mine a significant number of patterns. For example, though there are 15,426 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 a relative frequency of 66%. Another observation is that the numbers of frequent patterns with growing k may have several local maxima. As an example, for relative frequency 2%, after the number of frequent k-patterns drops a bit when k gets larger than 7, this number again increases when k exceeds 11. Furthermore, from $k = 12$, 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[13] (such as the HIV active substance dataset).

Regarding the runtime of our algorithm, we note that matching blocks is more expensive than matching bridges. Since, however, the number of bridges is much larger, roughly half of the time was used for each of the two types of matchings in all our experiments. 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. We also note that the time needed for candidate generation is always smaller than 1% of the total time (T). The time needed for coverage testing per pattern depends on how many structures these patterns share. If the number of patterns is large, the time needed per pattern is usually lower.

[13] Having several maxima when plotting the number of patterns against the size of the patterns is not uncommon. It can even happen in a database containing only one transaction graph. Consider e.g. the graph G with $V_G = \{v_1, v_2, v_3, v_4, v_5\}$, $E_G = \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_1, v_4\}, \{v_1, v_5\}\}$, $\lambda_G(v_1) = 1$, $\lambda_G(v_2) = 2$, $\lambda_G(v_3) = 2$, $\lambda_G(v_4) = 3$, $\lambda_G(v_5) = 3$ and $\forall e \in E_G : \lambda_G(e) = 4$. One can easily check that there are three 0-patterns, two 1-patterns, three 2-patterns, two 3-patterns and one 4-pattern.

Table 1 Results of our algorithm on the outerplanar subset of the NCI dataset

Level (k)	10%			5%			2%		
	$\#C$	#FP	T	$\#C$	#FP	T	$\#C$	#FP	T
1	86	7	107	144	11	169	582	25	380
2	74	16	446	216	24	570	1,332	61	1,118
3	139	41	1,133	234	74	1,393	510	170	2,123
4	133	77	1,232	266	154	2,038	642	356	4,079
5	139	91	1,071	319	222	2,268	909	644	5,603
6	107	72	754	332	252	1,847	1,212	918	6,105
7	61	41	472	295	195	1,168	1,266	990	4,964
8	37	25	354	182	137	741	1,086	893	3,384
9	20	13	205	137	116	602	956	803	2,282
10	8	5	130	131	119	594	828	700	1,635
11	0	0	0	131	117	565	697	604	1,360
12	0	0	0	115	107	536	707	665	1,483
13	0	0	0	78	64	412	1,027	1,022	2,017
14	0	0	0	27	21	250	1,702	1,700	2,858
15	0	0	0	4	3	89	2,725	2,715	3,957
16	0	0	0	0	0	0	4,079	4,072	5,578
17	0	0	0	0	0	0	5,518	5,487	6,898
18	0	0	0	0	0	0	6,729	6,711	8,175
19	0	0	0	0	0	0	7,326	7,311	8,813
20	0	0	0	0	0	0	7,114	7,079	8,703
21	0	0	0	0	0	0	6,000	5,947	7,627
22	0	0	0	0	0	0	4,435	4,407	5,954
23	0	0	0	0	0	0	2,857	2,855	4,129
24	0	0	0	0	0	0	1,633	1,633	2,609
25	0	0	0	0	0	0	787	786	1,444
26	0	0	0	0	0	0	325	310	741
27	0	0	0	0	0	0	80	73	286
28	0	0	0	0	0	0	10	9	113
29	0	0	0	0	0	0	0	0	0
Total	804	388	5,904	2,611	1,616	13,242	63,074	58,946	104,418

Number of candidate k-patterns ($\#C$), number of frequent k-patterns (#FP), and runtime in seconds for candidate generation and evaluation (T) are reported for $k = 1, \ldots, 29$ and for frequency thresholds 10, 5, and 2%

One of the conclusions we can draw from Table 1 is that 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 600 Mb), 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 this would bring e.g. an overhead of about 15 s per pass over the database. Another conclusion is that the runtime of the coverage

testing scales well as the pattern size grows, as indicate the complexity considerations given in the previous section. 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.

6.1.1 BBP vs. ordinary subgraph isomorphism

Using the outerplanar subset of the NCI dataset, we now report some empirical results comparing BBP and ordinary subgraph isomorphisms. The results of this paragraph should be interpreted carefully; the embedding operators have different complexities, result in completely different sets of frequent patterns, and the authors of the respective algorithms may have had different implementation foci. For the comparison we selected some state-of-the-art systems using ordinary subgraph isomorphism as embedding operator, namely gSpan (Yan and Han 2002), FSG (Kuramochi and Karypis 2001), and Gaston (Nijssen and Kok 2004).[14] Due to various reasons, it was not easy to evaluate all of these algorithms on the outerplanar NCI dataset. In particular, gSpan (available only in binary form) accepts transaction graphs with at most 254 edges only. For this reason, we removed all the graphs with more than 254 edges.

Table 2 shows the number of patterns of different sizes and the cumulative runtimes of Gaston and FSG in seconds at frequency threshold 2%. Since gSpan does not report runtimes for individual levels, it is not included in the table. For Gaston and FSG we report only the first 10 levels; FSG ran out of memory at level 10, Gaston and GSpan were shut down after several days before completing the task. As can be seen, the number of patterns increases much faster with the size than what we observed for BBP subgraph isomorphism.

For the 5 and 10% thresholds, all subgraphs could completely be mined by each of the systems. Table 3 shows the runtimes in seconds and the number of frequent patterns for ordinary and BBP subgraph isomorphism for the four systems. It can be observed that even though our algorithm on average consumes more time per pattern, this is largely compensated by the much smaller number of patterns. One interesting effect is that our algorithm succeeds in finding all pattern with respect to BBP subgraph isomorphism for frequency threshold 2%, while this is not practically feasible in reasonable time with the other systems using ordinary subgraph isomorphism.

It is of course important to compare not only the quantity, but also the *quality* of the patterns. A common technique of such comparisons is to empirically investigate the *predictive performance* by regarding frequent patterns as binary features: A feature has value 1 for an example graph if the corresponding pattern can be embedded into the example; it is 0 otherwise. Some recent studies compare the predictive performance of feature sets based on patterns with respect to ordinary subgraphs and those with respect to BBP subgraph isomorphism. In particular, Schietgat et al. (2008) generates sets of features exhaustively, while Schietgat et al. (2009) applies a sampling method based on maximal common subgraphs to select the features based on patterns with respect to BBP subgraph isomorphism. The empirical results clearly indicate for both methods

[14] We have tried other systems as well, but they were not able either to read in the dataset or to produce any output within several days even for high relative frequencies.

Table 2 Results with Gaston and FSG on the outerplanar subset of the NCI dataset using ordinary subgraph isomorphism

Size	2%		
	#FP	Gaston	FSG
2	57	32	232
3	183	94	440
4	516	190	750
5	1,419	408	1,980
6	3,434	780	3,570
7	7,573	1,708	7,396
8	14,955	3,324	14,576
9	26,177	7,354	27,716
10	39,647	11,866	o.o.m

Number of frequent patterns consisting of k vertices and cumulative runtimes in seconds are reported for both systems at 2% frequency thresholds and for $k = 2, \ldots, 10$ (o.o.m indicates that the system ran out of memory before completing the level). Note the difference between k in this table and that in Table 1

Table 3 Number of patterns and runtimes in seconds at 5 and 10% thresholds for BBP and ordinary subgraph isomorphism on the outerplanar subset of the NCI dataset (o.o.t. indicates that the system could not complete the task within several days)

	2%	5%	10%
#FP for ordinary subgraph isomorphism	o.o.t.	49,415	7,107
Gaston's runtime	o.o.t.	28,212	2,571
FSG's runtime	o.o.t.	60,821	10,464
gSpan's runtime	o.o.t.	74,136	4,649
#FP for BPP subgraph isomorphism	58,864	1,616	388
Runtime of our algorithm (FOG)	100,494	13,242	5,905

that frequent patterns with respect to BBP subgraph isomorphism compare favorably to those with respect to ordinary subgraph isomorphism in predictive performance. Thus, at least on the virtual screening problem, BBP subgraph isomorphism results in a more compact set of frequent patterns without drop in the predictive performance.

6.2 The mRNA dataset

As another demonstrative example of the relative compactness of BBP frequent patterns, in this section we report some experimental results comparing BBP frequent subgraphs with ordinary ones on mRNA *secondary structures*. RNAs are single-stranded chains of nucleotides. Their contact structures, called *secondary structures*, are formed by bindings of non-adjacent nucleotides (C–G and A–U pairs) by hydrogen bonds, usually over longer stretches of the sequence forming stems. These additional bonds together with those in the backbone chain always result in an outerplanar graph (see, e.g., Leydold and Stadler 1998); the graph can be drawn in the plane without edges crossing in interior points such that all edges corresponding to the

Table 4 Results with our algorithm (FOG) on the mRNA dataset

Level (k)	10%		5%		2%	
	#FP	T	#FP	T	#FP	T
1	120	54	241	121	12,264	1,160
2	82	8	118	8	193	21
3	120	5	168	5	279	14
4	118	3	166	3	274	13
5	78	2	110	2	182	9
6	39	2	55	2	91	5
7	0	0	0	0	0	0
Total	557	74	858	141	13,283	1,222

Number of frequent patterns with respect to BBP subgraph isomorphism (#FP) and runtimes in seconds (T) are reported for 10, 5, and 2% frequency thresholds

backbone chain (i.e., those connecting consecutive nucleotides) lie on the outer face, while almost all the other edges (i.e., those connecting non-consecutive nucleotides) are diagonals. Though, in contrast to the application in the previous section, we cannot assume some reasonable small constant on the number of diagonals, the number of simple cycles in RNA secondary structures still remains small as the "branching" of the stems is usually quite limited. Thus, in this application we again deal with well-behaved outerplanar graphs.

In our experiments we have used a relatively small dataset consisting of 269 mRNA molecules adopted from Horváth et al. (2001), as even this small dataset clearly validates the application of BBP subgraph isomorphism. Table 4 shows the results with our algorithm. As can be seen, most frequent patterns consist of a single block, especially for the lowest frequency threshold 2% where the number of 1-patterns is very high relative to that of k-patterns for $k > 1$. While our algorithm generated only 557, 885, and 13,283 frequent BBP patterns for the frequency thresholds 10, 5, and 2%, and completed its work within about 20 min even for 2%, Gaston, as shown in Table 5, respectively produced 864,862, 1,490,982, and 184,342,582 frequent subgraphs for the first 20 levels with a very impressive speed. We have shut down Gaston from level 21 because these numbers clearly demonstrate the compactness of BBP subgraph isomorphism against the ordinary one; in particular, the exponential grow of the number of frequent patterns for ordinary subgraph isomorphism at frequency threshold 2%.

7 Concluding remarks

We have presented an algorithm listing frequent patterns from outerplanar graphs with respect to a constrained subgraph isomorphism, called BBP subgraph isomorphism, and showed that it generates frequent patterns in incremental polynomial time if the graphs are also well-behaved, i.e., have a polynomial upper bound on their number of simple cycles. To the best of our knowledge, no efficient fragment of the frequent subgraph mining problem *beyond trees* has been identified before this positive result. The importance of positive, as well as negative theoretical results on efficient pattern mining in graph structured data is that they contribute to a better understanding of the border between tractable and intractable problem classes. A deeper understanding of

Table 5 Gaston's results on the mRNA dataset

Size (k)	10%		5%		2%	
	#FP	T	#FP	T	#FP	T
1	13	1	13	1	13	1
2	62	1	64	1	64	1
3	253	1	344	1	361	1
4	607	1	1,241	1	1,988	1
5	917	1	2,858	1	7,389	1
6	1,223	1	4,429	1	17,953	1
7	1,928	1	5,407	1	32,323	2
8	3,627	1	6,733	1	50,984	3
9	7,014	1	10,114	2	80,411	5
10	13,320	4	18,149	3	135,728	8
11	24,220	5	33,624	6	243,553	15
12	41,151	9	59,543	10	448,801	26
13	64,400	16	97,918	19	838,610	52
14	91,630	27	146,488	33	1,586,313	104
15	117,337	42	196,618	53	3,046,462	218
16	132,905	62	231,917	80	5,936,914	480
17	130,972	83	236,355	114	11,710,025	1,105
18	110,393	104	204,703	141	23,254,130	2,569
19	78,009	116	147,858	162	46,154,499	6,676
20	44,881	124	86,606	173	90,796,061	17,013
Total	864,862	124	1,490,982	173	184,342,582	17,013

Number of patterns with respect to ordinary subgraph isomorphism (#FP) and cumulative running time excluding the time for printing the patterns are reported in seconds for frequency thresholds 10, 5, and 2% for the first 20 levels

In contrast to our results in Table 4, Gaston couldn't complete the task within several days (Note again the difference between the meaning of k in this table and that in Table 4)

the border could provide, among others, useful insights into the problem class, which could then be exploited in the design of practical algorithms.

Our algorithm is based on a canonical form of outerplanar graphs which may be of some independent 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 showed that it is decidable in polynomial time for arbitrary outerplanar graphs.

We note that our result is not only theoretical but also practical, as well-behaved outerplanar graphs form a practically relevant graph class. As an example, the molecular graphs of 236,180 molecules in the NCI dataset consisting of 250,251 compounds are well-behaved outerplanar graphs. Our empirical results with this outerplanar subset of the NCI dataset clearly show the effective performance of our algorithm in the

Algorithm 4 COMPUTEFREQUENTPATTERNS (POLYNOMIAL DELAY)

Input: set $\mathcal{L}_{k-1}$ of frequent $(k-1)$-patterns for some $k \geq 2$
Output: set $\mathcal{L}_k$ of candidate k-patterns

1: $\mathcal{C}_k = \mathcal{L}_k = \emptyset$
2: **forall** $G_1 \in \mathcal{L}_{k-1}$ **do**
3: **forall** $G_2 \in \mathcal{L}_{k-1}$ s.t. G_1 and G_2 have the same core and $G_1 \leq G_2$ **do**
4: **forall** $g_1 \in \text{Leaf}(G_1)$ and $g_2 \in \text{Leaf}(G_2)$ **do**
5: **if** $G_1 \ominus g_1 \simeq G_2 \ominus g_2$ **then**
6: **forall** $g_1' \in \text{Leaf}(G_1 \ominus g_1)$ **do**
7: **if** g_2 is attachable to g_1' consistently with G_2 **then**
8: **let** C be the graph obtained by attaching g_2 in G_1 to g_1' consistently with G_2
9: **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 $g \in \text{Leaf}(C)$ **then**
10: add C to $\mathcal{C}_k$
11: **if** C is t-frequent **then**
12: add it to $\mathcal{L}_k$
13: **endfor**
14: **endfor**
15: **endfor**
16: **print** G_1 if $k > 3$
17: **endfor**
18: **return** $\mathcal{L}_k$

practice on large graph databases. We have also conducted empirical experiments on a dataset of mRNA secondary structures consisting of non-tenuous well-behaved outerplanar graphs demonstrating the relative compactness of BBP frequent subgraphs with respect to ordinary ones.

We note that by changing the position of the print statement in Algorithm 3, we can turn our algorithm into a polynomial delay algorithm, keeping its computation and total run time exactly the same. Such a modified algorithm is given in Algorithm 4. In the algorithm we make use of the fact that for each frequent k-pattern G, the set of frequent k-patterns having the same core as G can be listed with polynomial delay (e.g., sort the set of frequent k-patterns by the canonical string of their cores). Notice that the number of frequent k-patterns having the same core as G is bounded by a polynomial of the size of $\mathcal{D}$. Thus, the inner loop 3–15 runs in time polynomial in the size of $\mathcal{D}$ for every $G_1 \in \mathcal{L}_{k-1}$. These together imply that the delay between printing two consecutive frequent patterns in step 16 is bounded by a polynomial of the size of $\mathcal{D}$. Clearly, the modified algorithm works in the same total time as the original one. Though the modified algorithm lists frequent patterns with polynomial delay, it is obtained in an unnatural way: outputs are not shown to the user as soon as they have been computed, but are spread out to ensure polynomial delay between each of them. Therefore, we believe that in practice, most users would prefer to see computed outputs as soon as possible, and then rather wait a somewhat longer time whenever the next level of search is computed, resulting in incremental polynomial time between outputs. We also note that applying the same technique to Algorithm 2, which also works in incremental polynomial time, we get that the set $\mathcal{F}_b$ of frequent biconnected outerplanar graphs (see step 1 of Algorithm 1) can also be listed with polynomial delay.

Before closing the paper with an interesting problem, we recall that our algorithm exploits well-behavedness only in the computation of the set $\mathcal{F}_b$ of frequent biconnected graphs in step 1 of Algorithm 1. In fact, the positive result formulated in Theorem 7 also holds for the broader class of outerplanar graphs having at most polynomially many pairwise non-isomorphic simple cycles. This follows from the fact that outerplanar graphs have treewidth 2 (see, e.g., Bodlaender 1998) and from the result that the set of pairwise non-isomorphic cycles in a (labeled) graph of bounded treewidth can be listed in output polynomial time (Horváth 2005). Thus, $\mathcal{F}_b$ can be computed in polynomial time even for this more general case. Clearly, this class includes the class of well-behaved outerplanar subgraphs. One can easily construct outerplanar graphs with exponentially many simple cycles, but with only polynomially many, pairwise non-isomorphic cycles.[15]

It is natural to ask whether the positive result of this paper can be generalized to arbitrary outerplanar graphs. This question raises the following problem: *Given* a finite set $\mathcal{D}$ of, not necessarily well-behaved, biconnected outerplanar graphs and a non-negative integer frequency threshold t, *compute* the set of t-frequent patterns in $\mathcal{D}$ with respect to 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.

Acknowledgements This work was partially supported by the Fund for Scientific Research Flanders (FWO), the K.U. Leuven and ERC Starting Grant 240186 'MiGraNT', and by the German Science Foundation (DFG) under the reference number 'GA 1615/1-1'.

References

Agrawal R, Mannila H, Srikant R, Toivonen H, Verkamo AI (1996) Fast discovery of association rules. In: Fayyad UM, Piatetsky-Shapiro G, Smyth P, Uthurusamy R (eds) Advances in knowledge discovery and data mining. AAAI Press/The MIT Press, Menlo Park, CA, pp 307–328

Bodlaender HL (1998) A partial k-arboretum of graphs with bounded treewidth. Theor Comput Sci 209 (1–2):1–45

Borgelt C, Berthold M (2002) Mining molecular fragments: finding relevant substructures of molecules. In: Proceedings of the 2002 IEEE international conference on data mining (ICDM). IEEE Computer Society, pp 51–58

Calders T, Ramon J, Van Dyck D (2008) Anti-monotonic overlap-graph support measures. In: Proceedings of the 2008 IEEE international conference on data mining (ICDM). IEEE Computer Society, pp 73–82

Chartrand G, Harary F (1967) Planar permutation graphs. Annales de l'institut Henri Poincaré, (Sec. B) Probabilités et Statistiques 3(4):433–438

Chi Y, Nijssen S, Muntz RR, Kok JN (2005) Frequent subtree mining–an overview. Fundam Inform 66:161–198

Chi Y, Yang Y, Muntz RR (2005) Canonical forms for labelled trees and their applications in frequent subtree mining. Knowl Inf Syst 8(2):203–234

Cook D, Holder L (1994) Substructure discovery using minimum description length and background knowledge. J Artif Intell Res 1:231–255

[15] For example, consider the unlabeled graph consisting of the cycles $v_1, v_2, \ldots, v_{2n}$ and v_1, $v_3, \ldots, v_{2n-1}$.

Deshpande M, Kuramochi M, Wale N, Karypis G (2005) Frequent substructure-based approaches for classifying chemical compounds. IEEE Trans Knowl Data Eng 17(8):1036–1050
Diestel R (2005) Graph theory. 3. Springer, Heidelberg
Feder T, Hell P (1998) List homomorphisms to reflexive graphs. J Comb Theory B 72(2):236–250
Garey MR, Johnson DS (1979) Computers and intractability: a guide to the theory of NP-completeness. Freeman, San Francisco
Harary F (1971) Graph theory. Addison-Wesley, Reading
He H, Singh AK (2007) Efficient algorithms for mining significant substructures in graphs with quality guarantees. In: Proceedings of the 2007 IEEE international conference on data mining (ICDM). IEEE Computer Society, pp 163–172
Hedetniemi S, Chartrand G, Geller D (1971) Graphs with forbidden suhgraphs. J Comb Theory 10:12–41
Hopcroft JE, Wong JK (1974) Linear time algorithm for isomorphism of planar graphs. In: Proceedings of the sixth annual ACM symposium on theory of Computing (STOC). ACM Press, New York, pp 172–184
Horváth T (2005) Cyclic pattern kernels revisited. In: Proceedings of the 9th Pacific-Asia conference on advances in knowledge discovery and data mining (PAKDD), vol 3518 of LNAI. Springer, Heidelberg, pp 791–801
Horváth T, Bringmann B, Raedt LD (2007) Frequent hypergraph mining. In: Proceedings of the 16th international conference on inductive logic programming (ILP), vol 4455 of LNAI. Springer, Heidelberg, pp 244–259
Horváth T, Wrobel S, Bohnebeck U (2001) Relational instance-based learning with lists and terms. Mach Learn 43(1/2):53–80
Inokuchi A, Washio T, Motoda H (2003) Complete mining of frequent patterns from graphs: mining graph data. Mach Learn 50(3):321–354
Johnson DS, Papadimitriou CH, Yannakakis M (1988) On generating all maximal independent sets. Inform Process Lett 27(3):119–123
Koontz W (1980) Economic evaluation of loop feeder relief alternatives. Bell Syst Tech J 59:277–281
Kramer S, De Raedt L, Helma C (2001) Molecular feature mining in HIV data. In: Proceedings of the seventh ACM SIGKDD international conference on knowledge discovery and data mining. ACM Press, New York, pp 136–143
Kuramochi M, Karypis G (2001) Frequent subgraph discovery. In: Proceedings of the 2001 international conference on data mining (ICDM). IEEE Computer Society, pp 313–320
Leydold J, Stadler PF (1998) Minimal cycle bases of outerplanar graphs. Electron J Comb 5
Lingas A (1989) Subgraph isomorphism for biconnected outerplanar graphs in cubic time. Theor Comput Sci 63(3):295–302
Mannila H, Toivonen H (1997) Levelwise search and borders of theories in knowledge discovery. Data Mining Knowl Discover 1(3):241–258
Matula DW (1978) Subtree isomorphism in $O(n^{5/2})$. Ann Discrete Math 2:91–106
Maunz A, Helma C, Kramer S (2009) Large-scale graph mining using backbone refinement classes. In: Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining. ACM Press, New York, pp 617–626
Mitchell SL (1979) Linear algorithms to recognize outerplanar and maximal outerplanar graphs. Inform Process Lett 9(5):229–232
Nijssen S, Kok JN (2004) A quickstart in frequent structure mining can make a difference. In: Proceedings of the tenth ACM SIGKDD international conference on knowledge discovery and data mining. ACM Press, New York, pp 647–652
Nishi T, Chua LO (1986) Uniqueness of solution for nonlinear resistive circuits containing CCCS's or VCVS's whose controlling coefficients are finite. IEEE Trans Circuits Syst 33(4):381–397
Ramon J, Nijssen S (2009) Polynomial-delay enumeration of monotonic graph classes. J Mach Learn Res 10:907–929
Read RC, Tarjan RE (1975) Bounds on backtrack algorithms for listing cycles, paths, and spanning trees. Networks 5(3):237–252
Schietgat L, Costa F, Ramon J, De Raedt L (2009) Maximum common subgraph mining: a fast and effective approach towards feature generation. In: Proceedings of the 7th international workshop on mining and learning with graphs (MLG). Leuven, Belgium, pp 1–3

Schietgat L, Ramon J, Bruynooghe M, Blockeel H (2008) An efficiently computable graph-based metric for the classification of small molecules. In: Proceedings of the 11th international conference on discovery science (DS), vol 5255 of LNAI. Springer, Heidelberg, pp 197–209

Shamir R, Tsur D (1999) Faster subtree isomorphism. J Algorithms 33(2):267–280

Syslo MM (1982) The subgraph isomorphism problem for outerplanar graphs. Theor Comput Sci 17:91–97

Tarjan RE (1972) Depth-first search and linear graph algorithms. SIAM J Comput 1(2):146–160

Tong H, Faloutsos C, Gallagher B, Eliassi-Rad T (2007) Fast best-effort pattern matching in large attributed graphs. In: Proceedings of the 13th ACM SIGKDD international conference on knowledge discovery and data mining. ACM Press, New York, pp 737–746

Yan X, Han J (2002) gSpan: graph-based substructure pattern mining. In: Proceedings of the 2002 IEEE international conference on data mining. IEEE Computer Society, pp 721–724