Mining of Frequent Block Preserving Outerplanar Graph Structured Patterns

Yosuke Sasaki[1], Hitoshi Yamasaki[1], Takayoshi Shoudai[1], and Tomoyuki Uchida[2]

[1] Department of Informatics, Kyushu University
Fukuoka 819-0395, Japan
e-mail: {yosuke.sasaki,h-yama,shoudai}@i.kyushu-u.ac.jp
[2] Faculty of Information Sciences, Hiroshima City University
Hiroshima 731-3194, Japan
e-mail: uchida@cs.hiroshima-cu.ac.jp

Abstract. An outerplanar graph is a planar graph which can be embedded in the plane in such a way that all of vertices lie on the outer boundary. Many semi-structured data like the NCI dataset having about 250,000 chemical compounds can be expressed by outerplanar graphs. In this paper, we consider a data mining problem of extracting structural features from semi-structured data like the NCI dataset. For this data mining problem, first of all, we define a new graph pattern, called a block preserving outerplanar graph pattern, as an outerplanar graph having structured variables. Then, we present an effective Apriori-like algorithm for enumerating frequent block preserving outerplanar graph patterns from semi-structured data in incremental polynomial time. Lastly, by reporting some preliminary experimental results on a subset of the NCI dataset, we evaluate the performance of our algorithms.
Keywords. pattern discovery, graph mining, graph structured pattern, outerplanar graph.

1 Introduction

Large amount of data having graph structures, called semi-structured data, such as map data, CAD, biomolecular, chemical molecules, the World Wide Web are stored in databases. In the fields of Web mining and graph mining, many Web documents and many chemical compounds can be represented by ordered trees and outerplanar graphs, respectively. Outerplanar graphs are planar graphs which can be embedded in the plane in such a way that all of vertices lie on the outer boundary. In Fig. 1, we give four outerplanar graphs G, g_1, g_2, g_3 as examples of outerplanar graphs. For example, in the NCI dataset [4], which is one of popular graph mining datasets, 94.3% of all elements are expressed by outerplanar graphs. In analyzing semi-structured data, we must often solve a subgraph isomorphism problem, which is known to be NP-complete in general. However, subgraph isomorphism problems on some classes of graphs, including trees and biconnected outerplanar graphs, can be solved in polynomial time

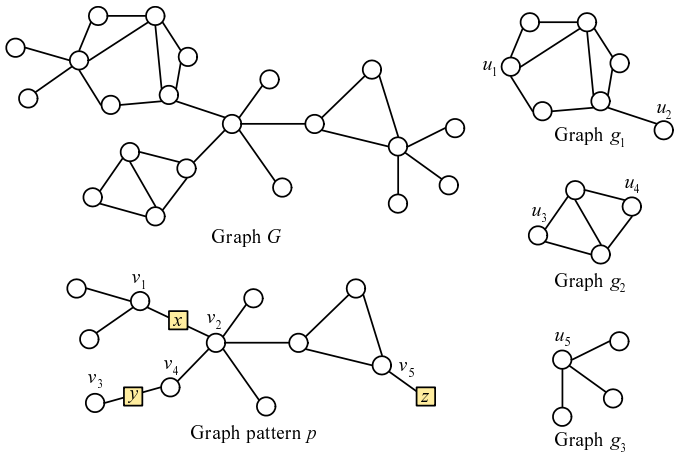

Fig. 1. Outerplanar graphs G, g_1, g_2, g_3 and a block preserving outerplanar graph pattern p. A variable is drawn by a box with lines to its elements. The label inside a box represents the variable label of the variable.

[2, 5]. Moreover, graph isomorphism problems tend to be easier than subgraph isomorphism problems and are computed in polynomial time for the classes of interval graphs, circular-arc graphs, and planar graphs.

Based on the viewpoints of these facts, in this paper, we consider a graph mining problem of extracting structural features from semi-structured data having outerplanar graph structures. In order to solve this graph mining problem, first of all, we define a new graph pattern, called a *block preserving outerplanar graph pattern* (*bpo-graph pattern* for short) as a connected outerplanar graph having structured variables. A variable is a list of at most 2 vertices and can be replaced with an arbitrary connected outerplanar graph.

We say that a bpo-graph pattern p *matches* an outerplanar graph G if G is obtained from p by replacing all variables with arbitrary connected outerplanar graphs. In Fig. 1, as an example of bpo-graph patterns, we give a bpo-graph pattern p having variables (v_1, v_2), (v_3, v_4), (v_5) labeled with variable labels x, y, z, respectively. The bpo-graph pattern p matches the outerplanar graph G in Fig. 1 obtained by replacing variables (v_1, v_2), (v_3, v_4), (v_5) with outerplanar graphs g_1, g_2, g_3, respectively.

Our goal of this paper is to present an effective algorithm of enumerating all frequent bpo-graph patterns from a given finite set D of outerplanar graphs. It is natural that for a given finite set D of outerplanar graphs, bpo-graph patterns, which are frequent with respect to the number of outerplanar graphs in D which are matched by p, are characteristic in D. As our approaches, in a similar way to a *block and bridge tree* given in [1], we introduce a *block tree* $t(G)$ and a *block tree pattern* $t(p)$ according to an outerplanar graph G and a bpo-graph pattern p, respectively. Then we reduce a problem of deciding whether or not a

bpo-graph pattern p matches an outerplanar graph G to a problem of deciding whether or not the tree pattern $t(p)$ corresponding to p matches the tree $t(G)$ corresponding to G. By giving an Apriori-like algorithm of generating candidate block tree patterns, we enumerate all frequent bpo-graph patterns in incremental polynomial time.

Horváth et al. [1] proposed an Apriori-like algorithm, which works in incremental polynomial time, for enumerating frequent subgraphs appearing in a restricted class of outerplanar graphs, called d-tenuous outerplanar graphs. Then by applying their algorithm to the NCI dataset [4], they found typical subgraph structures of chemical compounds. Yamasaki and Shoudai [9] proposed an interval graph pattern and presented a polynomial time algorithm for finding a minimally generalized interval graph pattern explaining a given finite set of interval graphs. As other related works, in the framework of inductive inference, by Suzuki et al. [6] and Takami et al. [7] gave polynomial time learning algorithms for tree patterns with internal structured variables and two-terminal series parallel graph patterns, respectively.

This paper is organized as follows. In Section 2, we introduce a graph pattern based on [8] and, in Section 3, a bpo-graph pattern as an outerplanar graph having structured variables. In Section 4, we propose a polynomial time algorithm which solves a matching problem between bpo-graph patterns and outerplanar graphs. In Section 5, we propose an Apriori-like algorithm which enumerates all frequent bpo-graph patterns from a finite set of outerplanar graphs. Lastly, by reporting an experimental result of applying our algorithms to a subset of the NCI dataset, we evaluate the performance of our algorithm.

2 Graph Pattern

Let Λ and Δ be two alphabets. Each symbol in Λ and Δ is called a *vertex label* and an *edge label*, respectively. Let G be an undirected graph. In this paper, G is called a (Λ, Δ)-*labeled graph* if all vertices and edges in G are labeled with symbols in Λ and Δ, respectively. We denote by $V(G)$ the set of vertices in G and by $E(G)$ the set of edges in G. A *graph pattern* is defined as a graph-structured pattern with internal variables, which represents characteristic common structures in graph-structured data. In [8], we introduced a general graph-structured pattern, called a term graph, in order to design efficient algorithms for computational problems on graphs. We define a class of graph patterns as a special class of term graphs as follows. Let X be an infinite alphabet where $X \cap (\Lambda \cup \Delta) = \emptyset$, whose elements are called *variable labels*.

Definition 1 (Graph pattern). Let G be a (Λ, Δ)-labeled graph. A *variable* of G is a list of different vertices of $V(G)$, which is denoted by $(v_1, \ldots, v_\ell)$ $(\ell \geq 1)$, where $v_i \neq v_j$ if $i \neq j$ $(1 \leq i, j \leq \ell)$. All vertices in a variable are called *ports* of the variable. All variables are labeled with a variable label in X. If a variable has only one port, it is called a *terminal variable*, otherwise called an *internal variable*. A triplet $p = (V, E, H)$ is called a (Λ, Δ)-*graph pattern* if (V, E) is a (Λ, Δ)-labeled graph and H is a set of variables of (V, E).

Below (Λ, Δ)-labeled graphs and (Λ, Δ)-graph patterns are simply called labeled graphs and graph patterns, respectively, when the label sets are clear from the context.

Let p be a graph pattern. We denote by $V(p)$, $E(p)$ and $H(p)$ the sets of all vertices, edges and variables of p, respectively. And we denote by $\lambda_p(v)$ the label of $v \in V(p)$ of p and by $\delta_p(e)$ the label of $e \in E(p)$ of p. The *degree* of v, denoted by $d_p(v)$, is the total sum of edges adjacent to v and internal variables including v, that is $d_p(v) = |\{\{u, v\} \mid \{u, v\} \in E(p)\} \cup \{h \mid h \in H(p),\ |h| \geq 2,\ h \text{ contains } v\}|$.

A *graph pattern* p is called a *linear* (*or regular*) *graph pattern* if all variables in $H(p)$ have mutually distinct variable labels in X. Let p and q be linear graph patterns. We say that p is *isomorphic* to q if there exists a bijection $\psi : V(p) \to V(q)$ such that (1) for any $v \in V(p)$, $\lambda_p(v) = \lambda_q(\psi(v))$, (2) $\{u, v\} \in E(p)$ if and only if $\{\psi(u), \psi(v)\} \in E(q)$, (3) for any $\{u, v\} \in E(p)$, $\delta_p(\{u, v\}) = \delta_q(\{\psi(u), \psi(v)\})$, and (4) for $\ell \geq 1$, $(v_1, v_2, \ldots, v_\ell) \in H(p)$ if and only if $(\psi(v_1), \psi(v_2), \ldots, \psi(v_\ell)) \in H(q)$.

Assumption. All graph patterns in this paper are linear. Then we call linear graph patterns graph patterns simply.

Definition 2 (Binding). Let p and q be graph patterns and x a variable label in X. Let $\sigma = (u_1, \ldots, u_k)$ be a list of k distinct vertices in q. The form $x := [q, \sigma]$ is called a *binding* for x. We can apply a binding $x := [q, \sigma]$ to a variable $h = (v_1, \ldots, v_\ell)$ in p which is labeled with x if the binding $x := [q, \sigma]$ satisfies that (1) $\ell = k$ and (2) $\lambda_q(u_i) = \lambda_p(v_i)$ for all i $(1 \leq i \leq \ell = k)$. A new graph pattern $p\{x := [q, \sigma]\}$ is obtained by applying the binding $x := [q, \sigma]$ to p in the following way. Let $h = (v_1, \ldots, v_\ell)$ be a variable in p with the variable label x. Let q' be one copy of q and $u'_1, \ldots, u'_k$ the vertices of q' corresponding to $u_1, \ldots, u_k$ of q, respectively. For the variable $h = (v_1, \ldots, v_\ell)$, we attach q' to p by removing the variable h from $H(p)$ and by identifying the vertices $v_1, \ldots, v_\ell$ with the vertices $u'_1, \ldots, u'_k$ of q', respectively.

A *substitution* θ is a finite collection of bindings $\{x_1 := [q_1, \sigma_1], \cdots, x_m := [q_m, \sigma_m]\}$, where $x_1, \ldots, x_m$ are mutually distinct variable labels in X. A graph pattern $p\theta$ is obtained by applying all the bindings $x_1 := [q_1, \sigma_1], \ldots, x_m := [q_m, \sigma_m]$ on p simultaneously.

Below we regard all labeled graphs as graph patterns without variables. As an example of graph patterns, in Fig. 1, we give a graph pattern p having variables (v_1, v_2), (v_3, v_4) and (v_5) labeled with $x,\ y,\ z$, respectively, so that the graph pattern $p\{x := [g_1, (u_1, u_2)], y := [g_2, (u_3, u_4)], z := [g_3, (u_5)]\}$ is isomorphic to the graph G in Fig. 1 where $g_1,\ g_2,\ g_3$ are labeled graphs in Fig. 1.

3 Block Preserving Outerplanar Graph Patterns and Block Tree Patterns

Let G be a connected labeled graph. G is said to be *biconnected* if for any two vertices in V, there exists a cycle which contains the two vertices. For a

subset U of V, the *induced subgraph* by U, denoted by $G[U]$, is a subgraph $G[U] = (U, \{\{u, v\} \in E(G) \mid \text{both } u \text{ and } v \text{ are in } U\})$. An induced subgraph $G[U]$ is said to be a *block* if it is biconnected and there is no proper superset U' of U such that $G[U']$ is biconnected. An edge which does not belong to any block is called a *bridge*. For a vertex v of a connected labeled graph G, v is called a *cutpoint* if $G[V(G) - \{v\}]$ is unconnected. An *outerplanar labeled graph* is a planar labeled graph which can be drawn in the plane in such a way that all vertices have a border with the outer face. We denote by $\mathcal{O}$ the set of all outerplanar labeled graphs. Since any block B of an outerplanar labeled graph has a unique cycle which contains all vertices of the block, we call the unique cycle of B the Hamiltonian cycle of the block. A *diagonal* is an edge which is contained in B but not on a Hamiltonian cycle of B.

In order to make our discussion simpler, we assume that all outerplanar labeled graphs are connected. We can easily extend our result of this paper to the case without the assumption.

Definition 3 (Block preserving outerplanar graph patterns). A graph pattern p is a *block preserving outerplanar graph pattern*, *bpo-graph pattern* for short, if p satisfies the following three conditions.

1. Any internal variable has exactly 2 ports.
2. A labeled graph $G_p = (V(p), E(p) \cup \{\{u, v\} \mid (u, v) \in H(p)\})$ is outerplanar.
3. Each port of any internal variable is either a cutpoint or a vertex of degree 1 in G_p.

Since all internal variables in a bpo-graph pattern are bridges in G_p, we call an internal variable in a bpo-graph pattern a *bridge variable*. For example, a graph pattern p in Fig. 1 is a bpo-graph pattern and two variables (v_1, v_2) and (v_3, v_4) of p are bridge variables. We denote by $\mathcal{OP}$ the set of all bpo-graph patterns.

In [1], a special data structure, called a block-bridge tree, was proposed for representing all connections among blocks of an outerplanar labeled graph. In a similar way to a block-bridge tree, for a bpo-graph pattern p, we introduce a new tree-structured pattern, called a block tree pattern of p.

Definition 4 (Block tree patterns). Let p be a bpo-graph pattern in $\mathcal{OP}$. A *block tree pattern of* p, denoted by $t(p)$, is a graph pattern defined as follows. Let "#" be a symbol which is not contained in $\Lambda \cup \Delta$. Let $V_{\mathcal{B}}(p) = \{v_B \mid B \text{ is a block of } p\}$, whose elements are called *block vertices*. For all blocks B of p, let $E_{\mathcal{B}}(B) = \{\{v_B, v\} \mid v \in V(B)\}$. Then

$$
\begin{aligned}
V(t(p)) &= V(p) \cup V_{\mathcal{B}}(p), \\
H(t(p)) &= H(p), \\
E(t(p)) &= (E(p) - \bigcup_{\text{all blocks } B \text{ of } p} E(B)) \cup \bigcup_{\text{all blocks } B \text{ of } p} E_{\mathcal{B}}(B).
\end{aligned}
$$

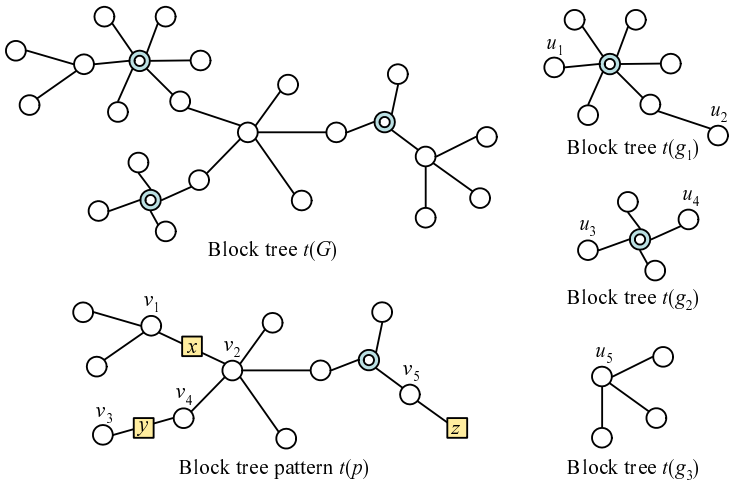

Fig. 2. Block trees $t(G)$, $t(g_1)$, $t(g_2)$, $t(g_3)$ and the block tree pattern $t(p)$ obtained from outerplanar labeled graphs G, g_1, g_2, g_3 and the bpo-graph pattern p in Fig. 1, respectively. A block vertex is drawn by a double circle.

We define vertex labels of $t(p)$ as follows: $\lambda_{t(p)}(v) = \lambda_p(v)$ if $v \in V(p)$, $\lambda_{t(p)}(v) = \#$ otherwise. In a similar way, we define edge labels of $t(p)$ as follows: $\delta_{t(p)}(e) = \delta_p(e)$ if $e \in E(p)$, $\delta_{t(p)}(e) = \#$ otherwise.

Next, for each block vertex $v_B \in V_{\mathcal{B}}(p)$, we assign a circular list, denoted by $\mu_p(v_B)$, to v_B. It is called a *block label* of v_B. We use a notation $[a_1, \ldots, a_\ell]$ to represent a circular list consisting of elements $a_1, \ldots, a_\ell$. We assume that every circular list has a pointer which points to the first element a_1 in $[a_1, \ldots, a_\ell]$. Let ℓ be the number of vertices in B. We note that any Hamiltonian cycle of a block is unique and represented in two ways, that is, clockwise and anticlockwise rotations, if we specify a start vertex of the cycle. First we give a numbering from 1 to ℓ to the vertices of B along one of rotation orders of the Hamiltonian cycle of B, and identify the numbers with the vertices themselves. For any i $(1 \leq i \leq \ell)$, we suppose that the i-th vertex is adjacent to k_i diagonals $\{i, j_1\}, \{i, j_2\}, \ldots, \{i, j_{k_i}\}$ in B, where $0 \leq k_i \leq \ell - 3$ and $1 \leq j_1 < j_2 < \cdots < j_{k_i} \leq \ell$. Then

$$\mu_p(v_B) = [(\phi_p(1), \delta_p(\{1,2\})), (\phi_p(2), \delta_p(\{2,3\})), \ldots, (\phi_p(\ell), \delta_p(\{\ell,1\}))],$$

where $\phi_p(i) = [(j_1, \delta_p(\{i, j_1\})), (j_2, \delta_p(\{i, j_2\})), \ldots, (j_{k_i}, \delta_p(\{i, j_{k_i}\}))]$ $(1 \leq i \leq \ell)$.

In this way, specifying a start vertex and a rotation direction of the Hamiltonian cycle of B, we construct a block label $\mu_G(v_B)$ of a block vertex v_B. From it we can easily compute another block label with another start vertex or the other rotation direction of the Hamiltonian cycle. We call a block tree pattern without variables a *block tree*, simply. As examples of block tree patterns and block trees, in Fig. 2, we give the block tree pattern $t(p)$ of the bpo-graph pat-

tern p in Fig. 1 and block trees $t(G),\ t(g_1),\ t(g_2),\ t(g_3)$ of outerplanar labeled graphs $G,\ g_1,\ g_2,\ g_3$ in Fig. 1, respectively.

4 Matching Algorithm for Block Preserving Outerplanar Graph Patterns

Let p and p' be bpo-graph patterns in $\mathcal{OP}$, and r and r' vertices of p and p', respectively. Let $t(p, r)$ be an unordered tree obtained from $t(p)$ by specifying r as a root, and for all block labels $\mu_p(v_B)$, regarding the nearest vertex from the root in B as a start vertex of the Hamiltonian cycle of B to reconstruct a block label, denoted by $\mu_p(v_B, r)$. We denote by $\bar{\mu}_p(v_B, r)$ the block label obtained from $\mu_p(v_B, r)$ by changing the rotation direction of the Hamiltonian cycle. We say that $t(p, r)$ is *equivalent* to $t(p', r')$ if there exists an isomorphism ψ from $t(p, r)$ to $t(p', r')$ such that for all block vertices v_B in $t(p, r)$, either $\mu_p(v_B, r) = \mu_{p'}(\psi(v_B), r')$ or $\mu_p(v_B, r) = \bar{\mu}_{p'}(\psi(v_B), r')$ holds. We say that $t(p)$ is *equivalent* to $t(p')$ if there exists two vertices r in p and r' in p' such that $t(p, r)$ is equivalent to $t(p', r')$.

Let r be a vertex in p and v a vertex of $t(p, r)$. A block tree pattern rooted at v of $t(p, r)$, denoted by $t(p, r)[v]$, is a block tree pattern induced by v and all descendants of v.

We call a block tree pattern with no variable a *block tree*. For a block tree pattern t and a block tree T, we say that t *matches* T if there exists a substitution θ such that all graph patterns appearing in θ are block trees and $t\theta$ is equivalent to T. We show the next lemmas for bpo-graph patterns and block tree patterns. We omit the proof.

Lemma 1. *Let p and p' be bpo-graph patterns in $\mathcal{OP}$ and r a vertex in p. Let $x := [p', \sigma]$ be a binding for p. Then there exists a vertex r' in σ such that $t(p, r)\{x := [t(p', r'), \sigma]\}$ is equivalent to $t(p\{x := [q, \sigma]\}, r)$.*

Lemma 2. *Let p and q be bpo-graph patterns in $\mathcal{OP}$. Then, p is isomorphic to q if and only if $t(p)$ is equivalent to $t(q)$.*

We give a polynomial time algorithm for computing the following problem.

Matching Problem for $\mathcal{OP}$
Input: An outerplanar graph $G \in \mathcal{O}$ and a bpo-graph pattern $p \in \mathcal{OP}$.
Problem: Decide whether or not p matches G.

From Lemma 1 and Lemma 2, we can show that Matching Problem for $\mathcal{OP}$ is reduced to the matching problem for deciding, given a block tree T and a block tree pattern t, whether or not t matches T.

Let r be a vertex in $G \in \mathcal{O}$ and r' a vertex in $p \in \mathcal{OP}$. For all vertices u in $t(G, r)$, we compute a subset of $V(t(p, r'))$, which is called a *correspondence-set* (C-set for short) of u and denoted by $CS(u)$, in the following way. Let N and n be the numbers of vertices in G and p, respectively. We assume that each C-set is stored by a simple array of length $O(n)$. We compute C-sets of all vertices in

$t(G,r)$ in postorder depending on a kind of each vertex. For a vertex u in $t(G,r)$, let $CS_{\mathcal{P}}(u) = \{c' \in CS(c) \mid c \text{ is a child of } u \text{ and } c' \text{ is a port of a variable in } p\}$.

Leaf: For all leaves u of $t(G,r)$, $CS(u)$ is the set of all vertices u' of $t(p,r')$ such that $d_{t(p,r')}(u') = 1$ and $\lambda_p(u') = \lambda_G(u)$.

Block vertex: Let u be a block vertex of $t(G,r)$ which represents a block B of G. Let $c_1, \ldots, c_\ell$ be the children of u which appear on a Hamiltonian cycle of B in this order. Let $CS_{\mathcal{B}}(u)$ be the set of all block vertices u' in $t(p,r)$ satisfying the following conditions: u' has just ℓ children $c'_1, \ldots, c'_\ell$ such that they appear on a Hamiltonian cycle of a block represented by u' in this order and either $(\mu_G(u,r) = \mu_p(u',r')$ and $c'_i \in CS(c_i))$ or $(\mu_G(u,r) = \bar{\mu}_p(u',r')$ and $c'_i \in CS(c_{\ell-i+1}))$ holds for $1 \leq i \leq \ell$. This work consumes $O(n\ell)$ time. Finally $CS(u) = CS_{\mathcal{B}}(u) \cup CS_{\mathcal{P}}(u)$.

Otherwise: Let u be a non-block vertex which is not a leaf in $t(G,r)$ and $c_1, \ldots, c_\ell$ the children of u. Let $CS_{\mathcal{NB}}(u)$ be the set of all non-block vertices u' in $t(p,r)$ satisfying the following condition: $\lambda_G(u) = \lambda_p(u')$, u' has at most ℓ children $c'_1, \ldots, c'_{\ell'}$ $(\ell' \leq \ell)$ and there are ℓ' vertices $c_{k_1}, \ldots, c_{k_{\ell'}}$ among $c_1, \ldots, c_\ell$ such that $c'_i \in CS(c_{k_i})$ and $\delta_p(\{u', c'_i\}) = \delta_G(\{u, c_{k_i}\})$ for all $i = 1, \ldots, \ell'$. We can decide whether or not this condition are satisfied for u and u' in the following way. First we construct a bipartite graph (U, V, E) as follows: $U = \{CS(c_1), \ldots, CS(c_\ell)\}$, $V = \{c'_1, \ldots, c'_{\ell'}\}$ and $E = \{(CS(c_i), c'_j) \mid c'_j \in CS(c_i)$ and $\delta_p(\{u', c'_j\}) = \delta_G(\{u, c_i\})$ $(1 \leq i \leq \ell,\ 1 \leq j \leq \ell')\}$. Next we compute a maximum bipartite graph matching problem for (U, V, E). If u' is a port of a variable and the bipartite graph has a matching of size ℓ', or u' is not a port of any variable and the bipartite graph has a matching of size exactly ℓ, we conclude that u and u' satisfy the above condition. We need $O(\ell\ell'\sqrt{\ell + \ell'})$ time to find a maximum matching for the bipartite graph (U, V, E) by Dinic's algorithm. Then we need $O(n\ell\sqrt{\ell})$ time for all non-block vertices in $t(p,r)$. Finally $CS(u) = CS_{\mathcal{NB}}(u) \cup CS_{\mathcal{P}}(u)$.

The correctness of the above algorithm is shown from the following lemmas.

Lemma 3. *Let G and p be an outerplanar graph in $\mathcal{O}$ and a bpo-graph pattern in $\mathcal{OP}$, respectively. Moreover let r and r' be vertices of G and p, respectively, and u and u' vertices in $t(G,r)$ and $t(p,r')$, respectively. Then,*

1. *$u' \in CS_{\mathcal{B}}(u)$ or $u' \in CS_{\mathcal{NB}}(u) - CS_{\mathcal{P}}(u)$ if and only if $t(p,r')[u']$ matches $t(G,r)[u]$, and*
2. *$u' \in CS_{\mathcal{P}}(u)$ if and only if there is a descendant d of u such that $t(p,r')[u']$ matches $t(G,r)[d]$.*

Lemma 4. *Let G and p be an outerplanar graph in $\mathcal{O}$ and a bpo-graph pattern in $\mathcal{OP}$, respectively. And let r be a vertex in G. Then there exists a vertex r' in p such that $r' \in CS(r)$ if and only if $t(p,r')$ matches $t(G,r)$.*

For each of n block tree patterns, we need $O(nN\sqrt{d})$ time for computing all C-sets for vertices in $t(G,r)$, where d is the maximum degree of cutpoints in p. Then we have the following theorem.

Theorem 1. *Matching Problem for $\mathcal{OP}$ is computable in $O(n^2N\sqrt{d})$ time.*

5 Pattern Enumeration Algorithm for Frequent BPO Graph Pattern Problem

In this section, we give an Apriori-like algorithm for enumerating all frequent bpo-graph patterns from a given finite set of outerplanar labeled graphs.

Let $\mathcal{OP}$ be the set of all bpo-graph patterns. Let p and q be bpo-graph patterns in $\mathcal{OP}$ and σ a list of vertices of length one or two in q. We easily show that for a variable h in p labeled with x, a graph pattern $p\{x := [q, \sigma]\}$ is also a bpo-graph pattern in $\mathcal{OP}$. For $p \in \mathcal{OP}$, let $L(p) = \{p\theta \in \mathcal{O} \mid$ all graph patterns appearing in θ are in $\mathcal{O}\}$. It is easy to see that if Λ and Δ have infinitely many symbols, for p and $p' \in \mathcal{OP}$, $L(p) \subseteq L(p')$ if and only if there is a substitution θ such that all graph patterns appearing in θ are in $\mathcal{OP}$ and p is isomorphic to $p'\theta$. For an outerplanar labeled graph $G \in \mathcal{O}$ and a bpo-graph pattern $p \in \mathcal{OP}$, we say that p *matches* G if $G \in L(p)$.

Let D be a finite set in $\mathcal{O}$. We denote by $match_D(p)$ the number of outerplanar labeled graphs in D which are matched by p. The *frequency* of p with respect to D, denoted by $supp_D(p)$, is defined as $supp_D(p) = match_D(p)/|D|$. Let t be a real number where $0 < t \leq 1$. A bpo-graph pattern $p \in \mathcal{OP}$ is *t-frequent* with respect to D if $supp_D(p) \geq t$.

In this paper, we give an effective algorithm for computing the next problem.

Frequent Block Preserving Outerplanar Graph Pattern Problem

Input: A finite set of outerplanar labeled graphs $D \subset \mathcal{O}$ and a real number t $(0 < t \leq 1)$.

Output: The set of all t-frequent bpo-graph patterns in $\mathcal{OP}$ with respect to D.

For $k \geq 0$, a *k-block tree pattern* is defined to be a block tree pattern such that the total sum of the numbers of block vertices, bridge variables, and edges not adjacent to any block vertex is equal to k. Let D be a set of outerplanar labeled graphs in $\mathcal{O}$ and t a real number where $0 < t \leq 1$. Let L_k^t be the set of all t-frequent k-block tree patterns with respect to D and C_k^t a set of candidate k-block tree patterns, which contains L_k^t. Let Λ_D (resp. Δ_D) be the set of all vertex (resp. edge) labels appearing in D. We compute C_k^t and L_k^t $(k \geq 0)$ in the following way.

0-block tree patterns. For all $a \in \Lambda_D$, we make a new vertex v labeled with a to construct a new block tree pattern $p_a = (\{v\}, \emptyset, \{(v)\})$. C_0^t is the set of all block tree patterns p_a obtained from all $a \in \Lambda_D$ in such a way. Let L_0^t be the set of all block tree patterns in C_0^t which are t-frequent.

1-block tree patterns. C_1^t is the set of all block tree patterns obtained from L_0^t in the following three ways. Initially let $C_1^t = \emptyset$.

1. For two 0-block tree patterns $p = (\{v\}, \emptyset, \{(v)\})$ and $p' = (\{v'\}, \emptyset, \{(v')\})$ in L_0^t, two copies $q = (\{w\}, \emptyset, \{(w)\})$ of p and $q' = (\{w'\}, \emptyset, \{(w')\})$ of p' are made. Then,
   (a) for all $s \in \Delta_D$, a new block tree pattern $q_s = (\{w, w'\}, \{\{w, w'\}\}, \{(w), (w')\})$ with an edge $\{w, w'\}$ labeled with s is added to C_1^t, and

(b) a new block tree pattern $q_x = (\{w, w'\}, \emptyset, \{(w), (w'), (w, w')\})$ is added to C_1^t.

2. For every block B appearing in all outerplanar labeled graphs in D, a new block tree pattern $q_B = (V(t(B)), E(t(B)), \{(w) \mid w \text{ is a non-block vertex in } V(t(B))\})$ is added to C_1^t.

Let L_1^t be the set of all block tree patterns in C_1^t which are t-frequent.

Let p be a block tree pattern. We say that p' is a *block tree subpattern* of p if p' is a block tree pattern and $V(p') \subseteq V(p)$, $E(p') \subseteq E(p)$, and $H(p') \subseteq H(p)$. Moreover we say that p' is a *terminal block tree subpattern* if p' is a block tree subpattern of p and either of the following forms:

1. $p' = (\{u, v\}, \{\{u, v\}\}, \{(v)\})$ or $p' = (\{u, v\}, \emptyset, \{(u, v), (v)\})$, where v is a non-block vertex adjacent to only u in p.
2. $p' = (\{u, v_B, v_1, \ldots, v_\ell\}, \{\{u, v_B\}, \{v_1, v_B\}, \ldots, \{v_\ell, v_B\}\}, \{(v_1), \ldots, (v_\ell)\})$ for some $\ell \geq 2$, where v_B is a block vertex adjacent to only $u, v_1, \ldots, v_\ell$ in p, while all $v_1, \ldots, v_\ell$ are adjacent only to v_B in p.

The vertex u appearing in (1) and (2) is called a *connected point* of p'. For a block tree pattern p and a terminal block tree subpattern p', we denote by $p \ominus p'$ the block tree subpattern obtained from p by removing all vertices in p' except for the connected point of p' and all edges and variables in p'.

k-block tree patterns ($k \geq 2$). Initially let $C_k^t = \emptyset$. For two $(k-1)$-block tree patterns p and q in L_{k-1}^t, let p' and q' be two terminal block tree subpatterns of p and q, respectively. If $p \ominus p'$ is equivalent to $q \ominus q'$ then a new block tree pattern r is constructed in such a way that a copy of p' and a copy of q' are connected to a copy of $p \ominus p'$ through the connected points of p' and q', respectively. The block tree pattern r is added to C_k^t. Let L_k^t be the set of all block tree patterns in C_k^t which are t-frequent.

In Fig. 3, we give examples of 0-block tree patterns, 1-block tree patterns and k-block tree patterns constructed by the above algorithm.

Lemma 5. *One generation of all candidates of k-block tree patterns from two t-frequent $(k-1)$-block tree patterns p_1 and p_2 is computed in $O(n_1^2 n_2^2 (n_1 + n_2))$ time, where n_1 and n_2 are the numbers of vertices of p_1 and p_2, respectively.*

Lemma 6. *For any $k \geq 2$, L_k^t is correctly computed from L_{k-1}^t in polynomial time with respect to the size of L_{k-1}^t by the above algorithm.*

Theorem 2. *The algorithm described above correctly computes Frequent Block Preserving Outerplanar Graph Pattern Problem.*

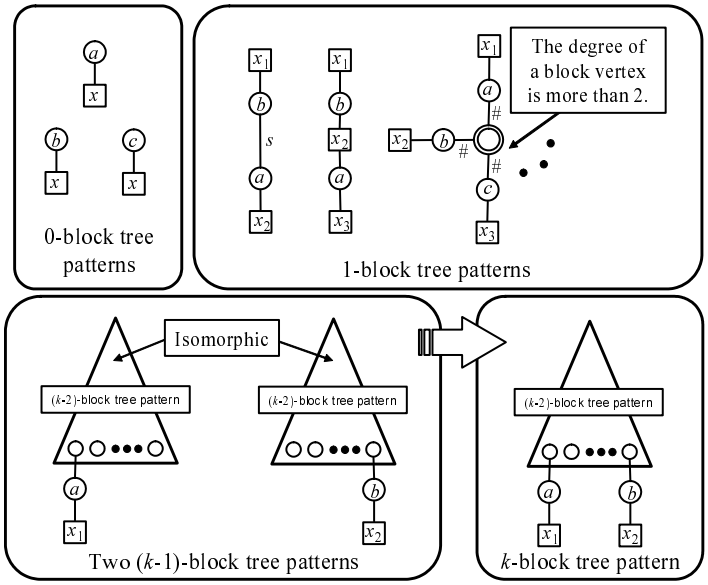

Fig. 3. Examples of 0-block tree patterns and 1-block tree patterns, and a generation of a k-block tree pattern from two $(k-1)$-block tree patterns.

6 Experimental Result

We have implemented our graph mining algorithm and tested on a chemical dataset. In our experiments, we used a dataset consisting of 100 outerplanar molecular graphs from the NCI database. The results are given in Fig. 4. We set frequency thresholds to be 0.5, 0.3 and 0.1, and tested on the dataset with respect to the frequencies. The table shows the numbers of candidate and frequent k-patterns, and the runtime in seconds for the generation of frequent patterns. In the table, we only show experimental results obtained by experiments which finished in 5 days for frequencies 0.1 and 0.3, and in 1 day for frequency 0.1.

For frequency 0.3, the number of generated frequent 4-block tree patterns is 12 per second, but the number of generated frequent 9-block tree patterns becomes 0.1 per second. Such a generation rate decreases polynomially as the total amount of sizes of generated candidate block tree patterns increases. Fig. 6 shows some of graph patterns generated by our algorithm.

Our algorithm generated a huge amount of frequent bpo-graph patterns, comparing with an experiment of enumerating all frequent bpo-graph patterns with no bridge variable (Fig. 5). The generated patterns certainly contain useless or unimportant patterns from theoretical point of view, because if a frequent bpo-graph pattern has a labeled edge, a bpo-graph pattern which is obtained from the frequent bpo-graph pattern by replacing the labeled edge with a bridge variable is also frequent. For a given set D of outerplanar labeled graphs in $\mathcal{O}$,

	$0.5^{\dagger}$			$0.3^{\dagger}$			$0.1^{\ddagger}$		
k	C_k	L_k	time(sec)	C_k	L_k	time(sec)	C_k	L_k	time(sec)
0	10	4	0.05	10	4	0.05	10	5	0.03
1	53	10	0.2	53	15	0.2	63	24	0.1
2	49	23	1	101	39	3	223	99	6
3	202	96	10	331	146	12	784	296	22
4	852	280	84	1599	555	115	3359	1041	168
5	2329	713	404	5931	1798	593	12414	4128	847
6	5142	1332	1280	16423	4996	2571	45101	15010	7293
7	8956	1696	2894	40952	12390	16050	158068	49379	55087
8	11231	1367	4606	98925	26330	78523	–	–	–
9	9521	606	4535	221875	46677	406132	–	–	–
10	4431	107	2064	–	–	–	–	–	–
11	816	0	339	–	–	–	–	–	–

Fig. 4. This table shows enumeration results of 0.5-, 0.3-, and 0.1-frequent bpo-graph patterns. (†Pentium D 2.80GHz, 2.00GB RAM. ‡Core2 6300 1.86GHz, 1.00GB RAM.

	$0.5^{\dagger\dagger}$		
k	C_k	L_k	time(sec)
0	10	4	0.06
1	37	3	0.3
2	7	5	1
3	28	11	6
4	47	14	16
5	82	14	41
6	76	5	48
7	34	0	26

Fig. 5. This table shows an enumeration result of 0.5-frequent bpo-graph patterns with no bridge variable. (††Pentium 4 1.90GHz, 256MB RAM).

a bpo-graph pattern $p \in \mathcal{OP}$ is minimally generalized with respect to D if there is no bpo-graph pattern $p' \in \mathcal{OP}$ such that $D \subseteq L(p') \subsetneq L(p)$. If we want more refined frequent bpo-graph patterns, we need to consider such a minimality of bpo-graph patterns.

7 Conclusion and Future Works

In this paper, we have considered a data mining problem of extracting structural features from semi-structured data whose data can be expressed by outerplanar graphs. Firstly, we have defined a block preserving outerplanar graph pattern as a new graph pattern having an outerplanar graph structure and structured variables. Secondly, we have presented a polynomial time Apriori-like algorithm for enumerating all frequent bpo-graph patterns w.r.t. a given finite set of out-

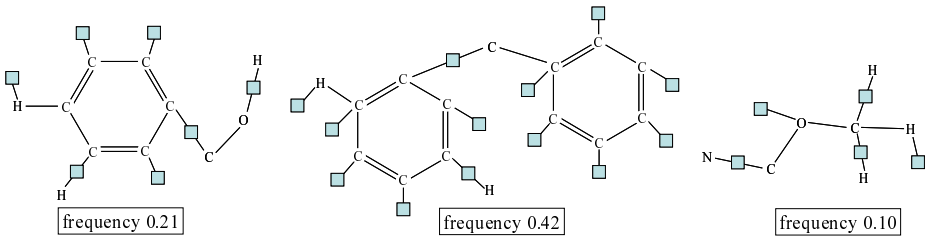

Fig. 6. Examples of 6-patterns generated by our system on the chemical test data

erplanar graphs. Finally, by reporting experimental results on a subset of the NCI dataset, we have evaluated the performance of our algorithm. In experiments, a huge number of frequent bpo-graph patterns were found. Many found frequent bpo-graph patterns may be useless or unimportant from the chemical viewpoints. Hence, we are considering a problem of extracting all frequent minimally generalized bpo-graph patterns. Moreover, we are studying the polynomial time learnability of the class of bpo-graph patterns.

In [3], we introduced unordered term trees, which are unordered tree patterns with internal structured variables. The variables in unordered term trees are defined in a similar way to bpo-graph patterns. We showed in [3] that a matching problem of deciding whether or not a given unordered tree is matched by a given unordered term tree with variables of more than 3 ports is NP-complete. From this result, we easily to show that it is hard to solve in polynomial time a matching problem for bpo-graph patterns extended to have variables consisting of more than 3 ports. Hence, we are considering a polynomial time matching algorithm for bpo-graph pattern extended to have variables consisting of at most 3 ports. Furthermore, we are planing to mining frequent graph patterns on other classes of graphs like planar graphs.

References

1. T. Horváth, J. Roman, and S. Wrobel. Frequent subgraph mining in outerplanar graphs. *Proc. 12th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pages 197–206, 2006.
2. A. Lingas. Subgraph isomorphism for biconnected outerplanar graphs in cubic time. *Theoretical Computer Science*, 63:295–302, 1989.
3. T. Miyahara, T. Shoudai, T. Uchida, K. Takahashi, and H. Ueda. Polynomial time matching algorithms for tree-like structured patterns in knowledge discovery. *Proc. 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-2000), Springer-Verlag, LNAI 1805*, pages 5–16, 2000.
4. National Cancer Institute - Chemical Dataset, `http://cactus.nci.nih.gov/`
5. R. Shamir and D. Tsur. Faster subtree isomorphism. *Journal of Algorithms*, 33(2):267–280, 1999.

6. Y. Suzuki, T. Shoudai, T. Uchida, and T. Miyahara. Ordered term tree languages which are polynomial time inductively inferable from positive data. *Theoretical Computer Science*, 350:63–90, 2006.
7. R. Takami, Y. Suzuki, T. Uchida, T. Shoudai, and Y. Nakamura. Polynomial time inductive inference of TTSP graph languages from positive data. *Proc. 15th International Conference on Inductive Logic Programming (ILP-2005), Springer-Verlag, LNAI 3625*, pages 366–383, 2005.
8. T. Uchida, T. Shoudai, and S. Miyano. Parallel algorithm for refutation tree problem on formal graph systems. *IEICE Transactions on Information and Systems*, E78-D(2):99–112, 1995.
9. H. Yamasaki and T. Shoudai. A polynomial time algorithm for finding linear interval graph patterns. *Proc. 4th Annual Conference on Theory and Applications of Models of Computation (TAMC-2007), Springer-Verlag, LNCS 4484*, pages 67–78, 2007.