Mach Learn (2009) 76: 137–173
DOI 10.1007/s10994-009-5115-9

Learning block-preserving graph patterns and its application to data mining

Hitoshi Yamasaki · Yosuke Sasaki · Takayoshi Shoudai · Tomoyuki Uchida · Yusuke Suzuki

Received: 15 November 2008 / Revised: 3 April 2009 / Accepted: 17 April 2009 /
Published online: 12 June 2009

Abstract Recently, due to the rapid growth of electronic data having graph structures such as HTML and XML texts and chemical compounds, many researchers have been interested in data mining and machine learning techniques for finding useful patterns from graph-structured data (graph data). Since graph data contain a huge number of substructures and it tends to be computationally expensive to decide whether or not such data have given structural features, graph mining problems face computational difficulties. Let $\mathcal{C}$ be a graph class which satisfies a connected hereditary property and contains infinitely many different biconnected graphs, and for which a special kind of the graph isomorphism problem can be computed in polynomial time. In this paper, we consider learning and mining problems for $\mathcal{C}$. Firstly, we define a new graph pattern, which is called a block preserving graph pattern (bp-graph pattern) for $\mathcal{C}$. Secondly, we present a polynomial time algorithm for deciding whether or not a given bp-graph pattern matches a given graph in $\mathcal{C}$. Thirdly, by giving refinement operators over bp-graph patterns, we present a polynomial time algorithm for finding a minimally generalized bp-graph pattern for $\mathcal{C}$. 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. Many pharmacologic chemical compounds are known to be represented by outerplanar graphs.

Editors: Filip Zelezny and Nada Lavrac.

H. Yamasaki · Y. Sasaki · T. Shoudai (✉)
Department of Informatics, Kyushu University, Fukuoka 819-0395, Japan
e-mail: shoudai@i.kyushu-u.ac.jp

H. Yamasaki
e-mail: h-yama@i.kyushu-u.ac.jp

Y. Sasaki
e-mail: yosuke.sasaki@i.kyushu-u.ac.jp

T. Uchida · Y. Suzuki
Department of Intelligent Systems, Hiroshima City University, Hiroshima 731-3194, Japan

T. Uchida
e-mail: uchida@hiroshima-cu.ac.jp

Y. Suzuki
e-mail: y-suzuki@hiroshima-cu.ac.jp

The class of connected outerplanar graphs $\mathcal{O}$ satisfies the above conditions for $\mathcal{C}$. Next, we propose two incremental polynomial time algorithms for enumerating all frequent bp-graph patterns with respect to a given finite set of graphs in $\mathcal{O}$. Finally, by reporting experimental results obtained by applying the two graph mining algorithms to a subset of the NCI dataset, we evaluate the performance of the two graph mining algorithms.

Keywords Pattern discovery · Graph mining · Graph-structured pattern · Inductive inference · Outerplanar graph

1 Introduction

Recently, due to the rapid growth of available data, there are growing expectations and desires for discovering interesting and useful patterns which are hidden in datasets. Particularly, many researchers are interested in knowledge discovery from data having structures such as sequences, trees, or graphs (Agrawal and Srikant 1994; Cook and Holder 2007; Han and Kamber 2001; Han et al. 2004). Graph-structured data (including tree structured data) widely appears in various practical fields. For example, web documents such as HTML and XML texts can be expressed by ordered trees, chemical compounds can be expressed by graphs whose vertices and edges correspond to atoms and bonds between atoms respectively, and glycans can be expressed by trees whose nodes and edges correspond to single sugars and covalent bonds, respectively. Moreover, graphs are suitable for representation of relationships between entities, such as link structures of the World Wide Web, gene regulatory networks, and so on. For such graph data, data mining and machine learning techniques for finding their characteristic structures will be useful for many practical applications.

Many graph mining techniques of extracting frequently occurring subgraphs in graph datasets have been proposed (see Cook and Holder 2007). In addition, many researchers have developed frequent substructure mining methods for specific graph classes, e.g. FreqT (Asai et al. 2002, 2003), Find_Freq_CPP (Uchida et al. 2004) and TreeMiner (Zaki 2002) for trees (typically rooted), SUBDUE (Cook and Holder 1994), AGM (Inokuchi et al. 2000), FSG (Kuramochi and Karypis 2001), gSpan (Yan and Han 2002), and GBI (Yoshida and Motoda 1995) for more general graphs than trees. Especially, Horváth et al. (2006) proposed a frequent subgraph mining algorithm for outerplanar graphs, which are planar graphs embedded in the plane in such a way that all of vertices lie on the outer boundary. For graph mining algorithms, graph and subgraph isomorphisms play a key role throughout the computations. It is known that the subgraph isomorphism problem for general graphs is NP-complete, and the graph isomorphism problem for general graphs appears to be hard. Hence, compared with traditional data mining like frequent itemset mining, graph mining problems face computational difficulties, because graph-structured data have a huge number of substructures. However, for some special classes of graphs such as trees, outerplanar graphs (Lingas 1989), planar graphs all of whose blocks are triconnected (Hopcroft and Wong 1974), the graph isomorphism problem can be decided efficiently. In this paper, we focus attention on such classes of graphs, denoted by $\mathcal{C}$, for which a special kind of the graph isomorphism problem can be solved in polynomial time. The purpose of this paper is to show an efficient graph mining algorithm, based on computational and algorithmic learning theory, for solving a problem of extracting structural features from graph data which can be expressed by graphs in $\mathcal{C}$.

Graph mining strategies of extracting frequent *maximal* substructures contribute speed-up and efficiency improvement of systems analyzing graph data. Consider the set $D =$

$\{G_1, G_2, G_3, G_4\}$ of graphs given in Fig. 1. Any graph in D has both graphs F_1 and F_2 also shown in Fig. 1 as subgraphs. Since F_2 has F_1 as a subgraph and is a maximal subgraph common to all graphs in D, F_2 is considered a more meaningful structural feature of D than F_1. As other effective mining strategies, extracting frequent graph-structured patterns, which can express not only frequent substructures but also associated structural relations between substructures common to graph data, have been proposed, e.g., algorithms for term graph patterns having tree structures (Shoudai et al. 2001) and for gap tree patterns under into-matching semantics (Arimura et al. 2001).

As one of such graph-structured patterns, we proposed a graph pattern having edge labels and internal structured variables, called a *term graph pattern*, in (Uchida et al. 1995). A variable in a term graph pattern consists of some number of vertices and can be substituted by an arbitrary term graph pattern. We say that a term graph pattern g *matches* a graph G if G is obtained from g by replacing all variables with certain graphs. In Fig. 1, as an example of term graph patterns, we give a term graph pattern f having only one variable labeled with x and a term graph pattern g having two graphs F_1 and F_2 as subgraphs and having variables (v_1, v_2) and (v_3) labeled with x and y, respectively. The term graph pattern g matches the graph G in Fig. 1 because G is obtained from g by replacing variables (v_1, v_2) and (v_3) with graphs G_1 and G_2, respectively, Consider the set $D = \{G_1, G_2, G_3, G_4\}$ of graphs in Fig. 1 again. The term graph pattern f in Fig. 1 matches all of graphs in D. On the other hand, the term graph pattern g matches two graphs G_3 and G_4, but matches neither G_1 nor G_2. Since f matches any graph, regardless of D, we can see that the structural feature expressed by f is meaningless. Moreover, since exactly one half of D has the structural feature expressed by g, we can see that g is meaningful. In this paper, we define a new graph-structured pattern, called a *block preserving graph pattern* (*bp-graph pattern* for short) as a special type of a connected term graph pattern. Especially, a bp-graph pattern forming an outerplanar graph structure is called a *block preserving outerplanar graph pattern* (*bpo-graph pattern* for short). In Fig. 1, as an example of bpo-graph patterns, we give two bpo-graph patterns f and g.

For considering learnabilities of bp-graph patterns, we use the framework of inductive inference. Inductive inference is a process of hypothesizing a general rule from examples. Angluin (1980b) and Shinohara (1982) showed that if a class $\mathcal{L}$ has finite thickness, and the matching problem and the minimal language (MINL, for short) problem for $\mathcal{L}$ are computable in polynomial time then $\mathcal{L}$ is polynomial time inductively inferable from positive data. Based on this framework, in this paper, we consider the polynomial time learnabilities of a graph class $\mathcal{C}$. Let $\mathcal{O}$ be the set of connected outerplanar graphs. In (Sasaki et al. 2008), we presented a polynomial time algorithm for deciding whether or not a given bp-graph pattern matches a given graph in $\mathcal{O}$. Let $\mathcal{C}$ be a graph class which satisfies a connected hereditary property and contains infinitely many different biconnected graphs, and for which a special kind of the graph isomorphism problem can be computed in polynomial time. In this paper, we consider a general framework of the matching problem for bp-graph patterns and present a polynomial time algorithm for deciding whether or not a given bp-graph pattern matches a given graph in $\mathcal{C}$. This follows up on our previous work where we presented polynomial time matching algorithms for term graph patterns having tree structures (term tree patterns) (Miyahara et al. 2000; Suzuki et al. 2003), having two-terminal series parallel (TTSP) graph structures (Takami et al. 2009), and having interval graph structures (Yamasaki and Shoudai 2007).

A bp-graph pattern p is said to be *minimally generalized* explaining a given finite set S of graphs if $S \subseteq L(p)$ and there is no bp-graph pattern q such that $S \subseteq L(q) \subsetneq L(p)$, where

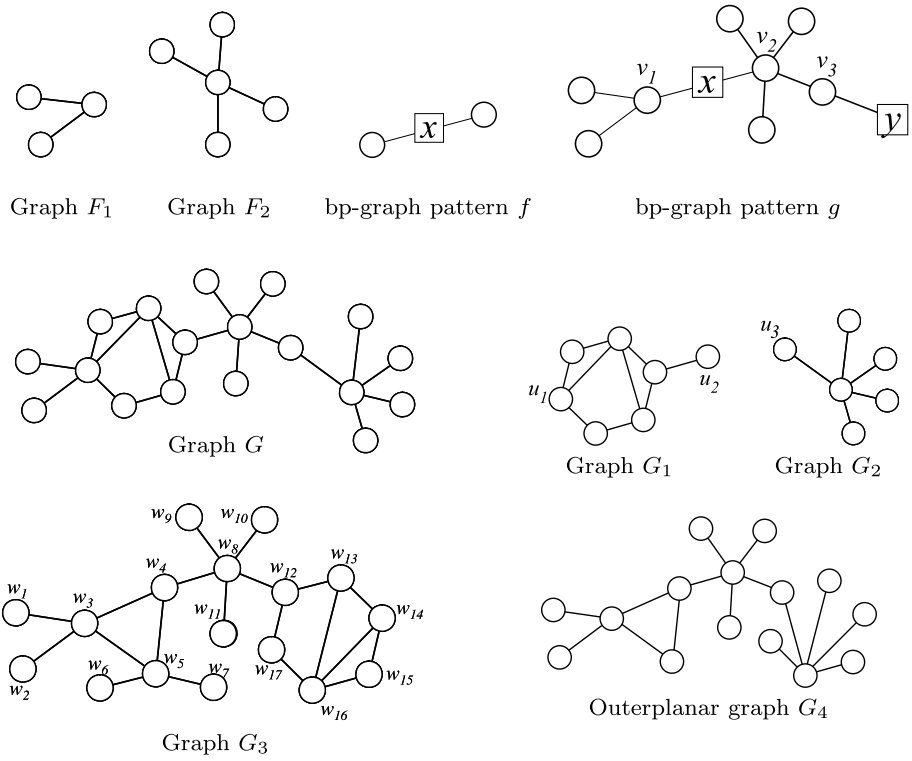


Fig. 1 Graphs G, G_1, G_2, G_3, G_4 and bp-graph patterns g, f. A variable is drawn by a *box with lines* to its elements. The *label inside a box* represents the variable label of the variable

for a bp-graph pattern g, $L(g)$ denotes the set of all graphs obtained from g by replacing all variables in g with arbitrary connected graphs. Hence, it is natural that a minimally generalized bp-graph pattern is more suitable for explaining a given set of graphs. For example, since there exist bp-graph patterns like g in Fig. 1 such that $\{G,\ G_3,\ G_4\} \subseteq L(g) \subsetneq L(f)$, the bp-graph pattern f in Fig. 1 is not minimally generalized with respect to $\{G,\ G_3,\ G_4\}$. An algorithm is said to be an MINL algorithm for $\mathcal{C}$ if it finds a minimally generalized bp-graph pattern explaining a given finite set of graphs in $\mathcal{C}$. In Yamasaki et al. (2008), we presented a polynomial time MINL algorithm for $\mathcal{O}$. In this paper, we give a polynomial time MINL algorithm for $\mathcal{C}$. Hence, we show that $\mathcal{C}$ is polynomial time inductively inferable from positive data. As other related works, we proposed polynomial time MINL algorithms for term graph patterns having ordered tree structures (Suzuki et al. 2006), TTSP graph structures (Takami et al. 2009), and interval graph structures (Yamasaki and Shoudai 2007).

Many pharmacologic chemical compounds are known to be represented by outerplanar graphs. For example, 94.3% of all elements in the NCI dataset (NCI 2000), which is one of popular chemical databases, are expressed by outerplanar graphs. Let S be a finite subset of $\mathcal{O}$. Next, we give two graph mining algorithms that generate all frequent bp-graph patterns for S. The first algorithm is based on an Apriori-like technique which is used in a subgraph mining algorithm for outerplanar graphs presented by Horváth et al. (2006). The second algorithm is a version of the first one substantially improved by using refinement operators

which are used in our MINL algorithm for bp-graph patterns. Both algorithms enumerate all frequent bp-graph patterns for S in incremental polynomial time. In order to show the performance of our graph mining algorithms, we report experimental results of applying our graph mining algorithms to a data mining problem for enumerating all frequent bp-graph patterns for a subset of the NCI dataset (NCI 2000) which are expressed by $\mathcal{O}$.

This paper is organized as follows. In Sect. 2, we define a block preserving graph pattern which is a special type of a term graph pattern. In Sect. 3, we propose a polynomial time algorithm for the matching problem for $\mathcal{C}$. In Sect. 4, we propose a polynomial time MINL algorithm for $\mathcal{C}$. In Sect. 5, we propose two graph mining algorithms for generating all frequent bp-graph patterns in a set of graphs in $\mathcal{O}$. Then, we evaluate the performance of our graph mining algorithms. In Sect. 6, we conclude this work and show directions for future work.

2 Block preserving graph pattern

A graph pattern is defined as a graph-structured pattern with internal variables, which represents characteristic common structures in graph data. In this section, we introduce a new graph pattern which is called a *block preserving graph pattern*. For a connected undirected graph G, the sets of all vertices and edges of G are denoted by $V(G)$ and $E(G)$, respectively. For a subset U of $V(G)$, the *induced subgraph* of G by U, denoted by $G[U]$, is a subgraph $G[U] = (U, \{\{u, v\} \in E(G) \mid u \in U \text{ and } v \in U\})$ of G. For a vertex v in $V(G)$, v is called a *cutpoint* of G if $G[V(G) - \{v\}]$ is disconnected. G is said to be *biconnected* if G has no cutpoint. For a subset U of $V(G)$, the induced subgraph $G[U]$ is said to be a *biconnected component (or bicomponent)* if it is biconnected and there is no proper superset U' of U such that $G[U']$ is biconnected. A biconnected component is called a *block* if it contains at least 3 vertices, otherwise a *bridge*. For example, for the graph G_3 in Fig. 1, the vertices w_4 and w_{12} are cutpoints, the induced subgraphs $G_3[\{w_3, w_4, w_5\}]$ and $G_3[\{w_{12}, w_{13}, w_{14}, w_{15}, w_{16}, w_{17}\}]$ are blocks and the induced subgraphs $G_3[\{w_4, w_8\}]$ and $G_3[\{w_8, w_{12}\}]$ are bridges.

A list is denoted by a collection of elements enclosed in parentheses, e.g. (u_1, u_2, u_3). The k-th element in a list σ is denoted by $\sigma[k]$. For a set and a list S, we denote by $|S|$ the number of elements in S. Let Λ and Δ be two alphabets each of whose elements is called a *vertex label* and an *edge label*, respectively. Let X be an infinite alphabet where $X \cap (\Lambda \cup \Delta) = \emptyset$. A symbol in X is called a *variable label*.

Definition 1 (Block preserving graph patterns) Let G be a connected undirected graph. Let E_b be a subset of the set of bridges of G and V' a subset of $V(G)$. A (Λ, Δ)-*labeled block preserving graph pattern* p *on an underlying graph* G is defined as a triple (V_p, E_p, H_p) where $V_p = V(G)$, $E_p = E(G) - E_b$, and $H_p = \{(u) \mid u \in V'\} \cup \{(u, v) \mid \{u, v\} \in E_b\}$. The elements in V_p, E_p, and H_p are called *vertices*, *edges*, and *variables*, respectively. In particular, a variable in H_p is called a *bridge variable* if it contains two vertices, otherwise a *terminal variable*. All vertices and edges are labeled with symbols in Λ and Δ, respectively, and all variables are labeled with mutually distinct symbols in X. The vertices in a variable are called *ports*. A *bicomponent* is defined as one of a block, a bridge, or a bridge variable.

Below, when the alphabets Λ and Δ, and an underlying graph G are clear from the context, a (Λ, Δ)-labeled block preserving graph pattern on G is called *a block preserving*

graph pattern (a bp-graph pattern) simply. A bp-graph pattern is said to be *ground* if it contains no variable. We regard all connected undirected graphs as ground bp-graph patterns. For a bp-graph pattern p, we denote by $V(p)$, $E(p)$ and $H(p)$ the sets of all vertices, edges and variables of p, respectively, and denote by $\lambda_p(v)$ the label of $v \in V(p)$, by $\delta_p(e)$ the label of $e \in E(p)$ and by $x(h)$ the label of $h \in H(p)$. For a vertex $v \in V(p)$, the *degree* of v, denoted by $d_p(v)$, is the total sum of edges adjacent to v and bridge variables including v, i.e., $d_p(v) = |\{u \in V(p) \mid \{u, v\} \in E(p)\}| + |\{u \in V(p) \mid (u, v) \in H(p) \text{ or } (v, u) \in H(p)\}|$. For a set of bp-graph patterns D, we describe the total sum of the numbers of vertices of graph patterns in D as the *vertex size* of D.

Let p and q be bp-graph patterns. We say that p is *isomorphic* to q, denoted by $p \cong q$, 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 $\{u, v\} \in E(p)$, $\delta_p(\{u, v\}) = \delta_q(\{\psi(u), \psi(v)\})$, (4) $(u, v) \in H(p)$ if and only if $(\psi(u), \psi(v)) \in H(q)$, and (5) $(u) \in H(p)$ if and only if $(\psi(u)) \in H(q)$. A bp-graph pattern p' is said to be a *bp-subgraph pattern* of p if $V(p') \subseteq V(p)$, $E(p') \subseteq E(p)$, and $H(p') \subseteq H(p)$. For a bp-graph pattern p and a subset U of $V(p)$, the *induced bp-subgraph pattern* of p by U, denoted by $p[U]$, is a bp-subgraph pattern $p[U] = (U, \{\{u, v\} \mid u \in U \text{ and } v \in U\}, \{(u) \in H(p) \mid u \in U\} \cup \{(u, v) \in H(p) \mid u \in U \text{ and } v \in U\})$.

A *planar graph* is a graph which can be drawn in the plane in such a way that its edges cross only at their endpoints. An *outerplanar graph* is a planar graph which can be drawn in the plane in such a way that all vertices have a border with the outer face. The set of all bp-graph patterns over a vertex label set Λ and an edge label set Δ is denoted by $\mathcal{P}_{\Lambda,\Delta}$ and the set of all ground bp-graph patterns in $\mathcal{P}_{\Lambda,\Delta}$ is denoted by $\mathcal{G}_{\Lambda,\Delta}$. Let $\mathcal{C}$ be a class of graphs (e.g., trees, outerplanar graphs, planar graphs, etc). $\mathcal{P}_{\Lambda,\Delta}[\mathcal{C}]$ denotes the set of all bp-graph patterns p in $\mathcal{P}_{\Lambda,\Delta}$ such that all blocks of the underlying graph of p belong to $\mathcal{C}$ and $\mathcal{G}_{\Lambda,\Delta}[\mathcal{C}]$ denotes the set of all ground bp-graph patterns in $\mathcal{P}_{\Lambda,\Delta}[\mathcal{C}]$.

Definition 2 (Binding) Let p and q be bp-graph patterns in $\mathcal{P}_{\Lambda,\Delta}[\mathcal{C}]$ and x a variable label in X. Let σ be a nonempty list of at most two distinct vertices in q. The form $x := [q, \sigma]$ is called a *binding* for x. Let h be a variable in p which has the variable label x. We can apply a binding $x := [q, \sigma]$ to a variable h if the binding $x := [q, \sigma]$ satisfies that $|\sigma| = |h|$ and $\lambda_q(\sigma[i]) = \lambda_p(h[i])$ for all i $(1 \leq i \leq |\sigma|)$. A new bp-graph pattern $p\{x := [q, \sigma]\}$ is obtained by applying the binding $x := [q, \sigma]$ to the variable h in the following way. For the variable h, we attach q to p by removing the variable h from $H(p)$ and identifying $\sigma[i]$ with $h[i]$ for each i $(1 \leq i \leq |\sigma|)$.

We say that a graph class $\mathcal{C}$ has a *connected hereditary property* if every graph in $\mathcal{C}$ is connected and its connected induced subgraphs are also contained in $\mathcal{C}$. The classes of trees, connected outerplanar graphs, and connected planar graphs have a connected hereditary property.

Proposition 1 *If $\mathcal{C}$ has a connected hereditary property, then $p\{x := [q, \sigma]\}$ is a bp-graph pattern in $\mathcal{P}_{\Lambda,\Delta}[\mathcal{C}]$ for any $p, q \in \mathcal{P}_{\Lambda,\Delta}[\mathcal{C}]$.*

Definition 3 (Substitution) Let $p, q_1, \ldots, q_m$ be bp-graph patterns in $\mathcal{P}_{\Lambda,\Delta}[\mathcal{C}]$. A *substitution* is a finite collection of bindings $\{x_1 := [q_1, \sigma_1], \ldots, x_m := [q_m, \sigma_m]\}$, where $x_1, \ldots, x_m$ are mutually distinct variable labels in X such that for each $i = 1, 2, \ldots, m$, g_i does not have $x_1, x_2, \ldots, x_m$ as variable labels. For a bp-graph pattern p and a substitution θ, $p\theta$ denotes the graph pattern obtained from p and θ by applying all the bindings in θ to p simultaneously.

As an example of bp-graph patterns, in Fig. 1, we give a bp-graph pattern g in $\mathcal{P}_{\Lambda,\Delta}$ having variables (v_1, v_2) and (v_3) labeled with x and y, respectively, so that the bp-graph pattern $g\{x := [G_1, (u_1, u_2)], y := [G_2, (u_3)]\}$ is isomorphic to the graph G in Fig. 1 where G_1 and G_2 are graphs in Fig. 1.

For a bp-graph pattern $p \in \mathcal{P}_{\Lambda,\Delta}[\mathcal{C}]$ and a graph $G \in \mathcal{G}_{\Lambda,\Delta}[\mathcal{C}]$, we say that p *matches* G if there is a substitution θ such that $p\theta \cong G$.

Proposition 2 *Let p be a bp-graph pattern in $\mathcal{P}_{\Lambda,\Delta}[\mathcal{C}]$ and G a graph in $\mathcal{G}_{\Lambda,\Delta}[\mathcal{C}]$. If p matches G, i.e., there is a substitution θ such that $p\theta \cong G$ holds, then any cutpoint in p is mapped to a cutpoint in G by an isomorphism which realizes $p\theta \cong G$.*

For a bp-graph pattern $p \in \mathcal{P}_{\Lambda,\Delta}[\mathcal{C}]$, the *bp-graph pattern language of p with respect to $\mathcal{C}$* is defined as $L_{\mathcal{C}}(p) = \{G \in \mathcal{G}_{\Lambda,\Delta}[\mathcal{C}] \mid p \text{ matches } G\}$. The class of bp-graph pattern languages with respect to $\mathcal{C}$ is defined as $\mathcal{L}_{\Lambda,\Delta}(\mathcal{C}) = \{L_{\mathcal{C}}(p) \mid p \in \mathcal{P}_{\Lambda,\Delta}[\mathcal{C}]\}$.

In Definition 1, we restrict the number of ports of variables in bp-graph patterns to at most 2. It is because the matching problem, as is stated later, even for a tree-structured pattern which has variables with more than 3 ports is NP-complete (Miyahara et al. 2000). The time complexity of the matching problem for a bp-graph pattern all of whose variables have at most 3 ports is not known. In this paper, we focus on outerplanar graphs particularly. We denote by $\mathcal{O}$ the class of all outerplanar graphs. We call a bp-graph pattern in $\mathcal{P}_{\Lambda,\Delta}[\mathcal{O}]$ *a block preserving outerplanar graph pattern (a bpo-graph pattern)*. For example, a bp-graph pattern g in Fig. 1 is a bpo-graph pattern in $\mathcal{P}_{\Lambda,\Delta}[\mathcal{O}]$.

3 A pattern matching algorithm for BP-graph patterns

3.1 General framework

In this section, we give a general framework of a pattern matching algorithm for solving the following problem for any graph class $\mathcal{C}$.

Matching Problem for $\mathcal{P}_{\Lambda,\Delta}[\mathcal{C}]$
Input: A bp-graph pattern $p \in \mathcal{P}_{\Lambda,\Delta}[\mathcal{C}]$ and a graph $G \in \mathcal{G}_{\Lambda,\Delta}[\mathcal{C}]$.
Problem: Decide whether or not p matches G.

For a bp-graph pattern p in $\mathcal{P}_{\Lambda,\Delta}[\mathcal{C}]$ and a vertex r in $V(p)$, let p^r be a rooted bp-graph pattern obtained by specifying r as the root. For bp-graph patterns p and q in $\mathcal{P}_{\Lambda,\Delta}[\mathcal{C}]$, and vertices $r \in V(p)$ and $r' \in V(q)$, we say that p^r is isomorphic to $q^{r'}$ if there exists an isomorphism $\psi : V(p) \to V(q)$ such that $\psi(r) = r'$. Let G be a connected graph in $\mathcal{G}_{\Lambda,\Delta}[\mathcal{C}]$ and s a vertex in $V(G)$. For a rooted bp-graph pattern p^r and a rooted connected graph G^s, we say that p^r matches G^s if there exists a substitution θ such that $p^r\theta$ $(= (p\theta)^r)$ is isomorphic to G^s. It is easy to see the following lemma.

Lemma 1 *Let p be a bp-graph pattern in $\mathcal{P}_{\Lambda,\Delta}[\mathcal{C}]$, G a graph in $\mathcal{G}_{\Lambda,\Delta}[\mathcal{C}]$ and r a vertex of p. Then, p matches G if and only if there is a vertex s in $V(G)$ such that p^r matches G^s.*

For a rooted graph pattern p^r, we define the parent-child relationship of p^r in a way similar to a rooted tree. For a vertex u of p^r, a vertex v of p^r is called a *child* of u if (1) both u and v belong to the same bicomponent, and (2) v becomes unreachable from the root r if

u is removed. A vertex v is called the *parent* of u if u is a child of v. A vertex v is called a *descendant* of u if it is either u itself or a vertex which becomes unreachable from r if u is removed. If u has no child, u is called a *leaf* of p^r. For a rooted bp-graph pattern p^r and a cutpoint u of p^r, we use the following notations.

$Leaf(p^r)$:	The set of all leaves of p^r.
$\mathcal{B}^r(u)$:	The set of all bicomponents of p^r which contain u and children of u. For a leaf v, $\mathcal{B}^r(v) = \emptyset$.
$p^r[u]$:	The bp-subgraph pattern rooted at u which is induced by all descendants of u. For a leaf u in p^r, $p^r[u]$ is the graph having only one vertex u and possibly the terminal variable (u), i.e., $p^r[u] \cong p^r[\{u\}]$. If u belongs to a variable, without loss of generality, we assume that $p^r[u]$ has a terminal variable (u).
$p^r[u, b]$:	The bp-subgraph pattern rooted at u which is induced by all vertices in $b \in \mathcal{B}^r(u)$ and all descendants of vertices in $V(b) - \{u\}$.

Consider the rooted graph $G_3^{w_8}$ having the vertex w_8 as the root, where G_3 is a graph given in Fig. 1. Then, w_4, w_9, w_{10}, w_{11} and w_{12} are the children of the root w_8. Since both w_3 and w_5 are children of w_4 and cutpoints of $G_3^{w_8}$, w_3 and w_5 are child-cutpoints of w_4 and w_4 is the parent-cutpoint of both w_3 and w_5. We can see that $Leaf(G_3^{w_8}) = V(G_3) - \{w_3, w_4, w_5, w_8, w_{12}\}$, $\mathcal{B}^{w_8}(w_4) = \{G_3[\{w_3, w_4, w_5\}]\}$, and both $G_3^{w_8}[w_4]$ and $G_3^{w_8}[w_4, G_3[\{w_3, w_4, w_5\}]]$ are isomorphic to the induced subgraph $G_3[\{w_1, w_2, w_3, w_4, w_5, w_6, w_7\}]$.

Below we also use the above notations for a rooted connected graph G^s in the same meanings.

Definition 4 Let p^r be a rooted bp-graph pattern, G^s a rooted connected graph, and v a vertex in $V(G^s)$. The *correspondence-set (C-set for short)* of v is the set of all vertices u in $V(p^r)$ such that $p^r[u]$ matches $G^s[v]$.

Moreover we use the following notations for a vertex $v \in V(G^s)$ and $B \in \mathcal{B}^s(v)$.

$CS(v)$:	The correspondence-set of v.
$CS(v, B)$:	The set of all pairs (u, b) such that $u \in V(p^r)$, $b \in \mathcal{B}^r(u)$ and $p^r[u, b]$ matches $G^s[v, B]$.
$IS(v)$:	The set of all vertices $u \in V(p^r)$ such that there is a descendant w of v which satisfies $u \in CS(w)$. We call this set the *inheritance-set (I-set for short)* of v.

Given a bp-graph pattern $p \in \mathcal{P}_{\Lambda,\Delta}[\mathcal{C}]$ and a graph $G \in \mathcal{G}_{\Lambda,\Delta}[\mathcal{C}]$, we can decide whether or not p matches G in the following way. First of all, we specify a vertex r of p as the root of p. Second, for each vertex s of G, compute all $CS(v)$ for all vertices v of G^s with respect to p^r. Finally, if there is a vertex $s \in V(G^s)$ whose C-set contains the root r of p^r, we conclude that $p^r[r]$ matches $G^s[s]$, i.e., p matches G. Hence, we give a pattern matching algorithm MATCH-$\mathcal{P}_{\Lambda,\Delta}[\mathcal{C}]$ in Fig. 2 by using a dynamic programming manner. The following lemmas play important roles to show the correctness of our pattern matching algorithm MATCH-$\mathcal{P}_{\Lambda,\Delta}[\mathcal{C}]$.

Lemma 2 *Let p^r be a rooted bp-graph pattern and G^s a rooted connected graph. For any leaf v in $V(G^s)$, $u \in CS(v)$ if and only if u is a leaf and $\lambda_p(u) = \lambda_G(v)$.*

Algorithm: MATCH-$\mathcal{P}_{\Lambda,\Delta}[\mathcal{C}]$;
Input: a bp-graph pattern $p \in \mathcal{P}_{\Lambda,\Delta}[\mathcal{C}]$ and a connected graph $G \in \mathcal{G}_{\Lambda,\Delta}[\mathcal{C}]$;
begin
Let r be a vertex of p;
foreach $s \in V(G)$ **do begin** // *Lemma 1.*
  foreach $v \in Leaf(G^s)$ **do begin** $CS(v) := \emptyset$; $IS(v) := \emptyset$ **end**;
  foreach $(u, v) \in Leaf(p^r) \times Leaf(G^s)$ **do**
    if $\lambda_p(u) = \lambda_G(v)$ **then** add u to $CS(v)$ and $IS(v)$; // *Lemma 2 and Lemma 4 (1).*
  end;
  $W := Leaf(G^s)$;
  // *W is the set of all vertices v in $V(G^s)$ s.t. $CS(v)$ and $IS(v)$ have been computed.*
  foreach $v \in V(G^s) - W$ s.t. all children of v are in W **do begin**
    $CS(v) := \emptyset$;
    foreach $u \in V(p^r)$ s.t. $\lambda_p(u) = \lambda_G(v)$ **do begin**
      if u is a leaf and a port of a terminal variable **then**
        add u to $CS(v)$; // *Lemma 3 (1).*
      if $0 < |\mathcal{B}^r(u)| \leq |\mathcal{B}^s(v)|$ **do begin**
        foreach $(b, B) \in \mathcal{B}^r(u) \times \mathcal{B}^s(v)$ **do begin**
          if MATCHBICOMPONENTS($p^r[u, b], G^s[v, B]$)=TRUE **then**
            add (u, b) to $CS(v, B)$; // *Lemma 5.*
        end;
        Construct a bipartite graph $G_{u,v} = (\mathcal{B}^r(u), \mathcal{B}^s(v), E)$ where
          $E = \{(b, B) \in \mathcal{B}^r(u) \times \mathcal{B}^s(v) \mid (u, b) \in CS(v, B)\}$;
        $m :=$ the size of a maximum bipartite graph matching in $G_{u,v}$;
        if u is a port of a variable and $m = |\mathcal{B}^r(u)|$ **then**
          add u to $CS(v)$ // *Lemma 3 (2).*
        else if $|\mathcal{B}^s(v)| = m$ **then** add u to $CS(v)$ // *Lemma 3 (3)*
      end
    end;
    $IS(v) := CS(v)$;
    foreach child c of v **do** $IS(v) := IS(v) \cup IS(c)$; // *Lemma 4 (2).*
    add v to W
  end;
  if $r \in CS(s)$ **then return** TRUE
end;
return FALSE
end.

Procedure MATCHBICOMPONENTS;
Input: a bp-subgraph pattern $p^r[u, b]$ and a rooted connected graph $G^s[v, B]$;
Output: TRUE or FALSE;
begin
if b is a bridge $\{u, u'\}$ **then begin**
  if B is a bridge $\{v, v'\}$ and $\delta_p(\{u, u'\}) = \delta_G(\{v, v'\})$ and $u' \in CS(v')$ **then**
    return TRUE // *Lemma 5 (1).*
end
else if b is a bridge variable (u, u') **then begin**
  foreach child v' of v in B **do begin**
    if $u' \in IS(v')$ **then return** TRUE // *Lemma 5 (2).*
  end
end
else begin // B is a block.
  if there is an isomorphism $\psi : V(b) \to V(B)$ such that
      $v = \psi(u)$ and $w \in CS(\psi(w))$ for all $w \in V(b) - \{u\}$ **then**
    return TRUE // *Lemma 5 (3).*
end;
return FALSE
end.

Fig. 2 An algorithm MATCH-$\mathcal{P}_{\Lambda,\Delta}[\mathcal{C}]$ for deciding whether or not a given bp-graph pattern matches a given connected graph

Proof Since v is a leaf of G^s, $G^s[v]$ contains only one vertex v. On the other hand, if u is a leaf of p^r, $p^r[u]$ is a rooted bp-graph pattern which consists of only one vertex u and possibly contains a terminal variable (u). Therefore, if u is a leaf and $\lambda_p(u) = \lambda_G(v)$, $p^r[u]$ matches $G^s[v]$. Conversely, since $G^s[v]$ consists of only one vertex v, if $p^r[u]$ matches $G^s[v]$, $p^r[u]$ has no vertex other than u. Then u is a leaf such that $\lambda_p(u) = \lambda_G(v)$. □

Lemma 3 *Let p^r be a rooted bp-graph pattern and G^s a rooted connected graph. Let v be a vertex which is not a leaf of G^s. Then, $u \in CS(v)$ if and only if $\lambda_p(u) = \lambda_G(v)$ and either of the following three conditions holds.*

(1) *u is a leaf and a port of a terminal variable.*
(2) *$0 < |\mathcal{B}^r(u)| < |\mathcal{B}^s(v)|$, u is a port of a variable, and there is an injection $\psi : \mathcal{B}^r(u) \to \mathcal{B}^s(v)$ such that $(u, b) \in CS(v, \psi(b))$ for all $b \in \mathcal{B}^r(u)$.*
(3) *$0 < |\mathcal{B}^r(u)| = |\mathcal{B}^s(v)|$ and there is a bijection $\psi : \mathcal{B}^r(u) \to \mathcal{B}^s(v)$ such that $(u, b) \in CS(v, \psi(b))$ for all $b \in \mathcal{B}^r(u)$.*

Proof Let $\mathcal{B}^r(u) = \{b_1, \ldots, b_\ell\}$ and $\mathcal{B}^s(v) = \{B_1, \ldots, B_k\}$. Suppose that the if-statement holds. It is immediate for the case $|\mathcal{B}^r(u)| = 0$. Let θ_i $(1 \le i \le \ell)$ be a substitution such that $p^r[u, b_i]\theta_i \cong G^s[v, \psi(b_i)]$. Let θ be the union of all θ_i $(1 \le i \le \ell)$. If ψ is a bijection and $H(p)$ does not have a terminal variable (u), $p^r[u]\theta \cong G^s[v]$ holds. If ψ is a bijection and there is a terminal variable $h = (u) \in H(p)$, by adding a trivial binding for $x(h)$ to θ, $p^r[u]\theta \cong G^s[v]$ holds. If ψ is not a bijection, $H(p)$ has a terminal variable (u). Let $\tilde{G}$ be the subgraph of $G^s[v]$ induced by v and all vertices of bicomponents in $\mathcal{B}^s(v) - \{\psi(b_1), \ldots, \psi(b_\ell)\}$. In this case, by adding a binding $x(h) := [\tilde{G}, (u)]$ to θ, $p^r[u]\theta \cong G^s[v]$ holds.

Conversely, we suppose that $p^r[u]$ matches $G^s[v]$. If u is a leaf of p^r, since v is not a leaf, there must be a terminal variable (u). Suppose that u is not a leaf of p^r. There exist a substitution θ and an isomorphism φ from $p^r[u]\theta$ to $G^s[v]$. For any i $(1 \le i \le \ell)$, there is an index j $(1 \le j \le k)$ such that the isomorphism φ maps the vertices in b_i into descendants of vertices in B_j. Let ψ be an injection from $\{b_1, \ldots, b_\ell\}$ to $\{B_1, \ldots, B_k\}$ such that $\psi(b_i) = B_j$ if and only if for any $u \in V(b_i)$, $\varphi(u)$ is a descendant of a vertex in $V(B_j)$. If $|\mathcal{B}^r(u)| < |\mathcal{B}^s(v)|$, ψ is an injection which satisfies the statement (1). Otherwise ψ satisfies the statement (2). □

Lemma 4 *Let p^r be a rooted bp-graph pattern and G^s a rooted connected graph. For any $v \in V(G^s)$, $u \in IS(v)$ if and only if either of the following conditions holds.*

(1) *v is a leaf of G^s and $u \in CS(v)$.*
(2) *v is not a leaf of G^s and $u \in CS(v) \cup (\bigcup_{\text{all children } c \text{ of } v} IS(c))$.*

Proof It is immediate from the definitions of C-sets and I-sets. □

Lemma 5 *Let p^r be a rooted bp-graph pattern and G^s a rooted connected graph. Let v be a vertex which is not a leaf of G^s and $B \in \mathcal{B}^s(v)$. Then, $(u, b) \in CS(v, B)$ if and only if $\lambda_p(u) = \lambda_G(v)$ and either of the following three conditions holds.*

(1) *b and B are bridges $\{u, u'\}$ and $\{v, v'\}$, respectively, and $\delta_p(\{u, u'\}) = \delta_G(\{v, v'\})$ and $u' \in CS(v')$.*
(2) *b is a bridge variable (u, u') and there is a child v' of v such that $u' \in IS(v')$.*
(3) *b and B are blocks and there is an isomorphism $\psi : V(b) \to V(B)$ such that $v = \psi(u)$ and $w \in CS(\psi(w))$ for all $w \in V(b) - \{u\}$.*

Proof This is shown in a straightforward way from the definition of bp-graph patterns and Proposition 2. □

The running time MATCH-$\mathcal{P}_{\Lambda,\Delta}[\mathcal{C}]$ depends on the computation between lines 10 and 13 of procedure MATCHBICOMPONENTS. We formally define the following problem.

Block Isomorphism with Vertex Correspondence Sets for $\mathcal{C}$
Input: Two blocks b and B in a specified class $\mathcal{C}$, and correspondence sets $CS(v) \subseteq V(b)$ for all $v \in V(B)$.
Problem: Decide whether or not there is a graph isomorphism $\psi : V(b) \to V(B)$ such that $w \in CS(\psi(w))$ for all $w \in V(b)$.

Since the size of any correspondence set is at most $|V(b)|$, the computational time of the above problem depends only on $|V(b)|$ and $|V(B)|$. We denote by $\mathcal{MC}(n, N)$ the worst computational time of solving **Block Isomorphism with Vertex Correspondence Sets for** $\mathcal{C}$ for blocks b and B whose numbers of vertices are n and N, respectively. Then we have the following theorem.

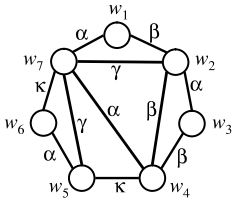
Theorem 1 *Let $\mathcal{C}$ be a class of graphs which satisfies a connected hereditary property. For a given bp-graph pattern $p \in \mathcal{P}_{\Lambda,\Delta}[\mathcal{C}]$ and a given graph $G \in \mathcal{G}_{\Lambda,\Delta}[\mathcal{C}]$, algorithm* MATCH-$\mathcal{P}_{\Lambda,\Delta}[\mathcal{C}]$ *correctly solves Matching Problem for $\mathcal{P}_{\Lambda,\Delta}[\mathcal{C}]$ in $O(nN^2\mathcal{MC}(n, N))$ time, where $n = |V(p)|$ and $N = |V(G)|$.*

Proof The correctness follows from Lemmas 1–5 and the fact that $\mathcal{C}$ has a connected hereditary property. Let m and M be the total numbers of bicomponents in p and G, respectively. At first we consider the running time to decide whether or not $p^r[u]$ matches $G^s[v]$ for $u \in V(p^r)$ and $v \in V(G^s)$. Let $\ell = |\mathcal{B}^r(u)|$ and $k = |\mathcal{B}^s(v)|$. The foreach loop at lines 14–17 of MATCH-$\mathcal{P}_{\Lambda,\Delta}[\mathcal{C}]$ needs $O(\ell k\,\mathcal{MC}(n, N))$ time. At line 19 of MATCH-$\mathcal{P}_{\Lambda,\Delta}[\mathcal{C}]$, we need $O(\ell k\sqrt{\ell + k})$ time to find a maximum bipartite graph matching for the bipartite graph $(\mathcal{B}^r(u), \mathcal{B}^s(v), E)$ by Dinic's algorithm (Dinic 1970; Hopcroft and Karp 1973). Then the running time needed for deciding whether or not $p^r[u]$ matches $G^s[v]$ is $O(\ell k\mathcal{MC}(n, N))$ time, since $\mathcal{MC}(n, N) = \Omega(N)$. Therefore the time for computing $CS(v)$ is $O(mk\mathcal{MC}(n, N))$ time. Considering the two foreach loops at lines 2 and 8, the total running time is $O(NmM\mathcal{MC}(n, N))$ time. Since $m \leq n$ and $M \leq N$, we have this theorem. □

3.2 A pattern matching algorithm for bpo-graph patterns

For an outerplanar graph G, an outerplanar embedding of G is a planar embedding in which all vertices have a border with the outer face. Any block B of G has a unique cycle consisting of all vertices of B, which form the boundary of an outerplanar embedding of B. We call the unique cycle of B the *outer cycle* of B. A *diagonal* is an edge which is contained in B but not on the outer cycle of B.

For a block B in a bpo-graph pattern p, we can index all vertices of B in the clockwise or counterclockwise order of the outer cycle in an outerplanar embedding of B. That is, by specifying a vertex in $V(B)$, called a *start vertex*, and a rotation direction of the outer cycle of B, we can easily construct a numbering function ρ_B from $V(B)$ to the set $\{1, 2, \ldots, |V(B)|\}$. Hereafter, for a vertex v of a block B, we also use the number $\rho_B(v)$ instead of v as a vertex identifier. For example, for a block B given in Fig. 3, $\rho_B(w_i) = i$ for each i $(1 \leq i \leq 7)$ if all vertices are numbered in the clockwise order of the Hamiltonian



B

$\mu_p(B, w_1) = \beta\alpha\beta\kappa\alpha\kappa\alpha \mid$
$1/2(4, \beta)(7, \gamma)/3/4(7, \alpha)/5(7, \gamma)/6/7$

$\overline{\mu_p}(B, w_1) = \alpha\kappa\alpha\kappa\beta\alpha\beta \mid$
$1/2(4, \gamma)(5, \alpha)(7, \gamma)/3/4/5(7, \beta)/6/7$

A block label of B

Fig. 3 A block B and its block label

cycle in the plane embedding of B from the vertex w_1. In the case of the counterclockwise order, $\rho_B(w_1) = 1$ and $\rho_B(w_i) = 9 - i$ for each i $(2 \le i \le 7)$.

Let B be a block of a bpo-graph pattern p, v a start vertex of B and $\ell = |V(B)|$. We suppose that all vertices in B are numbered in the clockwise order of the outer cycle in an outerplanar embedding of B from v. For any i $(1 \le i \le \ell)$, we suppose that the vertex i is adjacent to k_i diagonals $\{i, j_1\}, \{i, j_2\}, \ldots, \{i, j_{k_i}\}$ in B, where $0 \le k_i \le \ell - 3$ and $i \le j_1 < j_2 < \cdots < j_{k_i} \le \ell$. We assume that symbols "/", "(", ")", and "|" are not included in $\Lambda \cup \Delta$. The cycle label of B is defined as $label_c(B, v) = \delta_p(\{1, 2\})\delta_p(\{2, 3\}) \ldots \delta_p(\{\ell, 1\})$, and the diagonal label of B is defined as $label_d(B, v) = \phi_p(1)/\phi_p(2)/\ldots/\phi_p(\ell)$, where for each i $(1 \le i \le \ell)$, $\phi_p(i) = i(j_1, \delta_p(\{i, j_1\}))(j_2, \delta_p(\{i, j_2\})) \ldots (j_{k_i}, \delta_p(\{i, j_{k_i}\}))$. Then, we define a *block label* of B, denoted by $\mu_p(B, v)$, as a label $label_c(B, v) \mid label_d(B, v)$. We denote by $\overline{\mu_p}(B, v)$ the other block label of B in the case of the counterclockwise order. For example, let p be a bpo-graph pattern having a block B given in Fig. 3 as a subgraph. For a start vertex w_1 in B, we give the block labels $\mu_p(B, w_1)$ and $\overline{\mu_p}(B, w_1)$ of B in Fig. 3.

Lemma 6 *Let p^r be a rooted bpo-graph pattern and G^s a rooted connected outerplanar graph. For any vertex $v \in V(G^s)$, let B be a block in $\mathcal{B}^s(v)$. Let $c'_1, \ldots, c'_\ell$ be the children of v which lie on the outer cycle of B in this order. Then, $(u, b) \in CS(v, B)$ if and only if $\lambda_p(u) = \lambda_G(v)$ and u has just ℓ children $c_1, \ldots, c_\ell$ which appear on the outer cycle of b in this order and satisfy either of the following conditions.*

(1) $\mu_p(b, u) = \mu_G(B, v)$ *and* $c_i \in CS(c'_i)$ *for all* i $(1 \le i \le \ell)$.
(2) $\mu_p(b, u) = \overline{\mu_G}(B, v)$ *and* $c_i \in CS(c'_{\ell-i+1})$ *for all* i $(1 \le i \le \ell)$.

Proof Since G is a connected outerplanar graph and B is a block of G, this is shown in a straightforward way from the definition of C-sets. □

Corollary 1 *For a bpo-graph pattern $p \in \mathcal{P}_{\Lambda,\Delta}[\mathcal{O}]$ and a connected outerplanar graph $G \in \mathcal{G}_{\Lambda,\Delta}[\mathcal{O}]$, the problem of deciding whether or not p matches G is correctly solved in $O(nN^2\sqrt{D})$ time, where $n = |V(p)|$, $N = |V(G)|$ and D is the maximum degree of vertices in G.*

Proof From Theorem 1, algorithm MATCH-$\mathcal{P}_{\Lambda,\Delta}[\mathcal{O}]$ correctly solves the problem in $O(nN^2\mathcal{MC}(n, N))$ time. For any block B of an outerplanar graph, since $|E(B)| \le 2|V(B)| - 3$, the length of a block label of B is $O(|V(B)|)$. Moreover, since the outer cycle of a block B is found in linear time (Lingas 1989), we compute a block label of B in $O(|V(B)|)$ time. Therefore we can decide whether or not a block b of p is isomorphic to

B in $O(|V(B)|)$ time. Then we have $\mathcal{MC}(n, N) = O(N)$. Hence, MATCH-$\mathcal{P}_{\Lambda,\Delta}[\mathcal{O}]$ solves the problem in $O(nN^3)$ time.

Below we analyze the running time more accurately. Let $u \in V(p)$ and $v \in V(G)$. Let $\mathcal{B}^r(u) = \{b_1, \ldots, b_\ell\}$ and $\mathcal{B}^s(v) = \{B_1, \ldots, B_k\}$. We decide whether or not $p^r[u, b_i]$ matches $G^s[v, B_j]$ in $O(|V(B_j)|)$ time by using conditions in Lemma 6. At lines 14–17 of algorithm MATCH-$\mathcal{P}_{\Lambda,\Delta}[\mathcal{O}]$, we need $O(\sum_{i=1}^{\ell}\sum_{j=1}^{k}|V(B_j)|) = O(\ell c_v)$ time where c_v is the number of children of v. The running time for deciding whether or not $p^r[u]$ matches $G^s[v]$ is $O(\ell c_v + \ell k\sqrt{\ell + k})$ time. Let m and M be the total numbers of bicomponents in p and G, respectively. Then $CS(v)$ is computed in $O(m(c_v + k\sqrt{k}))$ time. Considering the two foreach-loops at lines 2 and 8, the total time is $O(Nm(N + M\sqrt{D}))$ time. Since $m \leq n$ and $M \leq N$, we have this theorem. □

4 An algorithm for finding a minimally generalized BP-graph pattern

4.1 Minimal language problem for bp-graph patterns

Let $\mathcal{C}$ be a class of graphs. For a bp-graph pattern $p \in \mathcal{P}_{\Lambda,\Delta}[\mathcal{C}]$ and a finite set of connected graphs $S \subset \mathcal{G}_{\Lambda,\Delta}[\mathcal{C}]$, p is said to be a *minimally generalized bp-graph pattern explaining* S if $S \subseteq L_{\mathcal{C}}(p)$ and there is not a bp-graph pattern $q \in \mathcal{P}_{\Lambda,\Delta}[\mathcal{C}]$ such that $S \subseteq L_{\mathcal{C}}(q) \subsetneq L_{\mathcal{C}}(p)$. In this section, we give an algorithm for solving the following problem.

Minimal Language (MINL) Problem for $\mathcal{L}_{\Lambda,\Delta}(\mathcal{C})$
Input: A finite set of graphs $S \subset \mathcal{G}_{\Lambda,\Delta}[\mathcal{C}]$.
Output: A minimally generalized bp-graph pattern $p \in \mathcal{P}_{\Lambda,\Delta}[\mathcal{C}]$ explaining S.

For the class of bp-graph pattern languages $\mathcal{L}_{\Lambda,\Delta}(\mathcal{C})$, from the theoretical result by Angluin (1980a) and Shinohara (1982), if $\mathcal{L}_{\Lambda,\Delta}(\mathcal{C})$ has finite thickness, i.e., for any nonempty finite set $S \subset \mathcal{G}_{\Lambda,\Delta}[\mathcal{C}]$, the cardinality of the set $\{L \in \mathcal{L}_{\Lambda,\Delta}(\mathcal{C}) \mid S \subseteq L\}$ is finite, and the membership problem and the minimal language problem for $\mathcal{L}_{\Lambda,\Delta}(\mathcal{C})$ are solvable in polynomial time then $\mathcal{L}_{\Lambda,\Delta}(\mathcal{C})$ is polynomial time inductively inferable from positive data.

It is easy to see that the following lemma holds.

Lemma 7 *Let $\mathcal{C}$ be a class of graphs. The class $\mathcal{L}_{\Lambda,\Delta}(\mathcal{C})$ has finite thickness.*

Proof Let S be a nonempty finite subset of $\mathcal{G}_{\Lambda,\Delta}[\mathcal{C}]$ and G a connected graph in S. If g is a bp-graph pattern in $\mathcal{P}_{\Lambda,\Delta}[\mathcal{C}]$ such that $L_{\mathcal{C}}(g)$ includes G (i.e., g matches G), then $|V(g)| \leq |V(G)|$ and $|E(g)| + |H(g)| \leq |E(G)| + |V(G)|$. Moreover, the number of all edge labels and the number of all vertex labels appearing in G are finite. Therefore the set $\{g \in \mathcal{P}_{\Lambda,\Delta}[\mathcal{C}] \mid G \in L_{\mathcal{C}}(g)\}$ is finite, consequently $\mathcal{L}_{\Lambda,\Delta}(\mathcal{C})$ has finite thickness. □

4.2 A general algorithm for finding a minimally generalized bp-graph pattern

First of all, we give the four procedures in Fig. 4, which are used in algorithm MINL[$\mathcal{C}$] (Fig. 8) for finding a minimally generalized bp-graph pattern. These procedures are called *refinement operators*. As for the correctness of algorithm MINL[$\mathcal{C}$], we have the following lemma.

Procedure Λ'-LabeledVertexAddition$(p, h, \mathtt{a})$; (see Figure 5)
Input: a bp-graph pattern $p \in \mathcal{P}_{\Lambda,\Delta}[\mathcal{C}]$, a variable $h \in H(p)$, and a vertex label $\mathtt{a} \in \Lambda'$;
begin
1: **if** h is a bridge variable $[u, w]$ **then begin**
2: $q :=$ bp-graph pattern $(\{u', v, w'\}, \emptyset, \{(u', v), (v, w'), (v)\})$
where $\lambda_q(u') = \lambda_p(u)$, $\lambda_q(w') = \lambda_p(w)$ and $\lambda_q(v) = \mathtt{a}$;
3 **return** $p\{x(h) := [q, (u', w')]\}$
4: **end**
5: **else if** h is a terminal variable (u) **then begin**
6: $q :=$ bp-graph pattern $(\{u', v\}, \emptyset, \{(u', v), (u'), (v)\})$
where $\lambda_q(u') = \lambda_p(u)$ and $\lambda_q(v) = \mathtt{a}$;
7: **return** $p\{x(h) := [q, (u')]\}$
8: **end**
end.

Procedure Υ-BlockReplacement(p, h, B, σ); (see Figure 6)
Input: a bp-graph pattern $p \in \mathcal{P}_{\Lambda,\Delta}[\mathcal{C}]$, a variable $h \in H(p)$,
a block $B \in \Upsilon$ and a list σ of distinct vertices in B where $|\sigma| \leq 2$;
begin
1: $b :=$ bp-graph pattern $(V(B), E(B), \{(v) \mid v \in V(B)\})$;
2: **if** h is a bridge variable (u, w) and $\sigma = (u', w')$ **then**
3: **if** $\lambda_q(u') = \lambda_p(u)$ and $\lambda_q(w') = \lambda_p(w)$ **then return** $p\{x(h) := [b, \sigma]\}$;
4: **if** h is a terminal variable (u) and $\sigma = (u')$ **then**
5: **if** $\lambda_q(u') = \lambda_p(u)$ **then return** $p\{x(h) := [b, \sigma]\}$
end.

Procedure Δ'-LabeledEdgeReplacement(p, h, α); (see Figure 7)
Input: a bp-graph pattern $p \in \mathcal{P}_{\Lambda,\Delta}[\mathcal{C}]$, a bridge variable $h = (u, w) \in H(p)$,
and an edge label $\alpha \in \Delta'$;
begin
1: $q :=$ bp-graph pattern $(\{u', w'\}, \{\{u', w'\}\}, \emptyset)$
where $\lambda_q(u') = \lambda_p(u)$, $\lambda_q(w') = \lambda_p(w)$ and $\delta_q((u', w')) = \alpha$;
2: **return** $p\{x(h) := [q, (u', w')]\}$
end.

Procedure TerminalVariableDeletion(p, h); (see Figure 7)
Input: a bp-graph pattern $p \in \mathcal{P}_{\Lambda,\Delta}[\mathcal{C}]$ and a terminal variable $h = (u) \in H(p)$;
begin
1: $q :=$ bp-graph pattern $(\{u'\}, \emptyset, \emptyset)$ where $\lambda_q(u') = \lambda_p(u)$;
2: **return** $p\{x(h) := [q, (u')]\}$
end.

Fig. 4 Refinement operators: These operators are used in algorithm MINL[$\mathcal{C}$] (Fig. 8). Λ' and Δ' are finite subsets of Λ and Δ, respectively. Υ is the finite set of blocks

Lemma 8 *Let $\mathcal{C}$ be a class of graphs which satisfies a connected hereditary property. If $\mathcal{C}$ contains infinitely many different biconnected graphs or Δ has infinitely many edge labels, for a given nonempty set of graphs $S \subset \mathcal{G}_{\Lambda,\Delta}[\mathcal{C}]$, algorithm* MINL[$\mathcal{C}$] *finds a minimally generalized bp-graph pattern explaining S.*

Proof Let p be a bp-graph pattern computed by algorithm MINL[$\mathcal{C}$] for an input S. Let Λ_S, Δ_S, and Υ_S be the sets of vertex labels, edge labels, and blocks, respectively, which appear in all graphs in S. Let p_2 and p_3 be temporary bp-graph patterns of p immediately after Steps 2 and 3 in algorithm MINL[$\mathcal{C}$] respectively. Since no new vertex is added at Steps 3

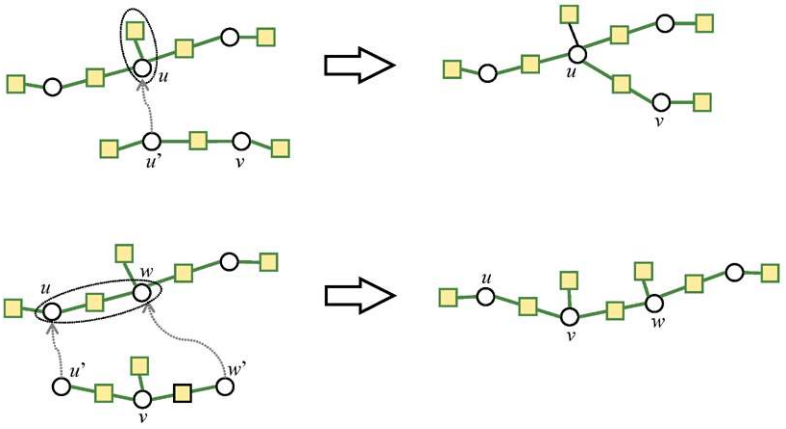


Fig. 5 LABELEDVERTEXADDITION: The upper figure shows the case of a terminal variable and the lower shows the case of a bridge variable

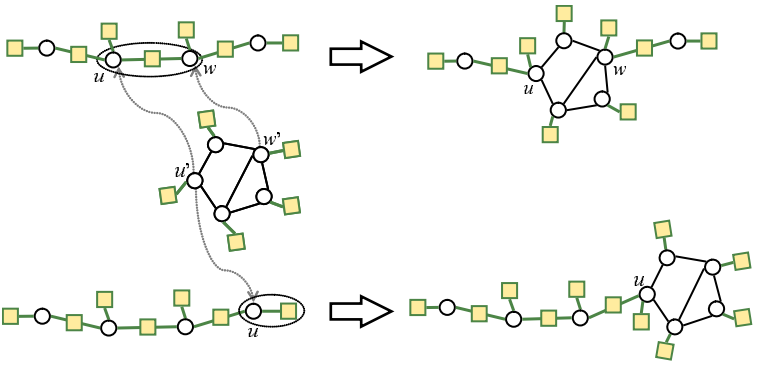


Fig. 6 BLOCKREPLACEMENT: The upper figure shows the case of a bridge variable and the lower shows the case of a terminal variable

and 4 in the algorithm, $|V(p)| = |V(p_2)| = |V(p_3)|$ holds. We show that if there exists a bp-graph pattern q such that $S \subseteq L_{\mathcal{C}}(q) \subseteq L_{\mathcal{C}}(p)$, $q \cong p$ holds.

Since $L_{\mathcal{C}}(q) \subseteq L_{\mathcal{C}}(p)$, $|V(q)| \geq |V(p_2)|$ holds. Let b be a biconnected graphs in $\mathcal{G}_{\Lambda,\Delta}[\mathcal{C}]$, which consists of either a single edge with an edge label in $\Delta - \Delta_S$ or a block which does not belong to Υ_S. Let θ be a substitution for p which replaces all bridge variables with copies of the biconnected graph b and all terminal variables with graphs consisting of a single vertex. Since $q\theta \in L_{\mathcal{C}}(p)$ and $L_{\mathcal{C}}(p) \subseteq L_{\mathcal{C}}(p_2)$, there is a substitution θ_2 for p_2 such that $q\theta \cong p_2\theta_2$. We note that each occurrence of b in $p\theta$ has at most 2 cutpoints. Let θ_2^x be a substitution which is obtained from θ_2 by replacing all occurrences of b and all bridges appearing in θ_2 with bridge variables and removing all bindings for terminal variables in θ_2. Since $S \subseteq L_{\mathcal{C}}(q) \subseteq L_{\mathcal{C}}(p_2\theta_2^x) \subseteq L_{\mathcal{C}}(p_2)$ and refinement operators Λ_S-LABELEDVERTEXADDITION and Υ_S-BLOCKREPLACEMENT cannot apply to p_2 any more, all bp-graph patterns in bindings in θ_2^x are bp-graph patterns consisting of two vertices and one bridge variable connecting the two vertices. Therefore $|V(q)| = |V(p_2)|$, and then we have $|V(q)| = |V(p)|$.

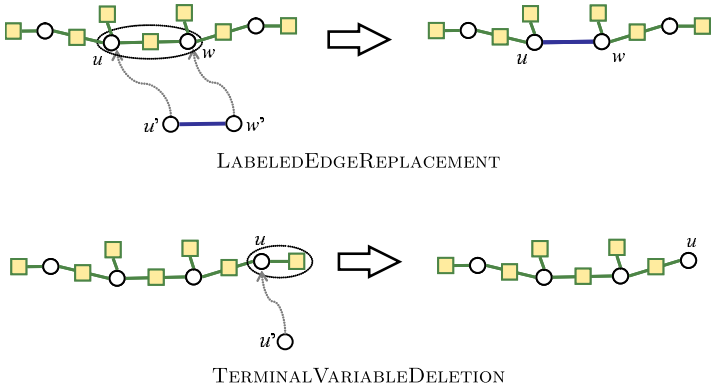


Fig. 7 Refinement operators Δ'-LabeledEdgeReplacement and TerminalVariableDeletion

Since $L_{\mathcal{C}}(q) \subseteq L_{\mathcal{C}}(p) \subseteq L_{\mathcal{C}}(p_3)$, there is a substitution θ_3 for p_3 such that $q\theta \cong p_3\theta_3$. Let θ_3^x be a substitution which is obtained from θ_3 by replacing all occurrences of b in θ_3 with bridge variables and removing all bindings for terminal variables in θ_3. We have $p_3 \cong p_3\theta_3^x$, and then there is a bijection $\zeta : V(q) \to V(p_3)$ satisfying the following conditions.

1. For all $v \in V(q)$, $\lambda_q(v) = \lambda_{p_3}(\zeta(v))$.
2. If $(v, v') \in H(q)$ $(v \neq v')$ then $(\zeta(v), \zeta(v')) \in H(p_3)$ holds.
3. If $\{v, v'\} \in E(q)$ then either $(\zeta(v), \zeta(v')) \in H(p_3)$ or $\{\zeta(v), \zeta(v'))\} \in E(p_3)$ with $\delta_{p_3}(\{\zeta(v), \zeta(v')\}) = \delta_q(\{v, v'\})$.

If $\{v, v'\} \in E(q)$ and $(\zeta(v), \zeta(v')) \in H(p_3)$, a bp-graph pattern obtained from p_3 by substituting an edge of label $\delta_q(\{v, v'\})$ for $(\zeta(v), \zeta(v'))$ also explains S. Therefore since Δ_S-LabeledEdgeReplacement can not apply to p_3 any more, if $\{v, v'\} \in E(q)$ then $\{\zeta(v), \zeta(v')\} \in E(p_3)$ and $\delta_{p_3}(\{\zeta(v), \zeta(v')\}) = \delta_q(\{v, v'\})$ hold.

Let b' $(b' \ncong b)$ be a biconnected graph in $\mathcal{G}_{\Lambda,\Delta}[\mathcal{C}]$, which consists of either a single edge with an edge label in $\Delta - \Delta_S$ or a block not in Υ_S. Let θ' be a substitution for q such that it substitutes copies of b for all bridge variables and copies of b' for all terminal variables. Since $q\theta' \in L_{\mathcal{C}}(p)$, there is a substitution θ_4 for p such that $q\theta' \cong p\theta_4$. Since all terminal variables which are adjacent to bridge variables are removed by TerminalVariableDeletion, all bindings in θ_4 which contain an occurrence of b' are applied to terminal variables. Let θ_4^x be a substitution obtained from θ_4 so that all occurrences of b in bindings in θ_4 are replaced with bridge variables, and all bindings to terminal variables are removed. Then $p \cong p\theta_4^x$. Moreover since at Step 4, terminal variables are removed from p as much as possible while p explains S, we have $q \cong p$. □

In Fig. 9, we give an example process of algorithm MINL[$\mathcal{C}$] of finding a minimally generalized bp-graph pattern explaining an example set of graphs in Fig. 10. The running time of MINL[$\mathcal{C}$] depends only on $\mathcal{MC}(n, N)$ defined in Sect. 3

Theorem 2 *Let $\mathcal{C}$ be a class of graphs which satisfies a connected hereditary property. If $\mathcal{C}$ contains infinitely many different biconnected graphs or Δ has infinitely many edge labels, for a given nonempty set of graphs $S \subset \mathcal{G}_{\Lambda,\Delta}[\mathcal{C}]$, the minimal language problem for $\mathcal{C}$ is correctly computed in $O(N_{\min}^5\mathcal{N}^2 \cdot \mathcal{MC}(N_{\min}, N_{\min}))$ time, where $\mathcal{N}$ is the total sum of numbers of vertices of all graphs in S and $N_{\min} = \min\{|V(G)| \mid G \in S\}$.*

Algorithm MINL[$\mathcal{C}$];
Input: a nonempty set of graphs $S \subset \mathcal{G}_{\Lambda,\Delta}[\mathcal{C}]$;
Output: a minimally generalized bp-graph pattern $p \in \mathcal{P}_{\Lambda,\Delta}[\mathcal{C}]$ explaining S;
begin
// Step 0.
1: Compute the sets Λ_S, Δ_S, and Υ_S of all vertex labels, all edge labels, and all blocks, respectively, which appear in all graphs in S;
// Step 1.
2: $p := (\{u\}, \emptyset, \{(u)\})$ where $\lambda_p(u) \in \Lambda_S$; $H := \{(u)\}$;
// Step 2.
3: **while** $H \neq \emptyset$ **do begin**
4: $h :=$ a variable in H; $H := H - \{h\}$;
5: **foreach** $\mathtt{a} \in \Lambda_S$ **do begin**
6: $p' :=\Lambda_S$-LabeledVertexAddition$(p, h, \mathtt{a})$;
7: **if** $S \subseteq L_{\mathcal{C}}(p')$ **then begin** $H := H \cup (H(p') - H(p))$; $p := p'$; **break end**
8: **end**;
9: **if** p is not updated **then begin**
10: **foreach** $B \in \Upsilon_S$ **do begin**
11: **foreach** list σ of distinct vertices in B s.t. $|\sigma| \leq 2$ **do begin**
12: $p' :=\Upsilon_S$-BlockReplacement(p, h, B, σ);
13: **if** $S \subseteq L_{\mathcal{C}}(p')$ **then begin** $H := H \cup (H(p') - H(p))$; $p := p'$; **break end**
14: **end**;
15: **if** p is updated **then break**
16: **end**
17: **end**
18: **end**;
// Step 3.
19: **foreach** bridge variable $h \in H(p)$ **do begin**
20: **foreach** $\alpha \in \Delta_S$ **do begin**
21: $p' :=\Delta_S$-LabeledEdgeReplacement(p, h, α);
22: **if** $S \subseteq L_{\mathcal{C}}(p')$ **then begin** $p := p'$; **break end**
23: **end**
24: **end**;
// Step 4.
25: **foreach** terminal variable $h \in H(p)$ **do begin**
26: $p' :=$TerminalVariableDeletion(p, h);
27: **if** $S \subseteq L_{\mathcal{C}}(p')$ **then** $p := p'$
28: **end**
end.

Fig. 8 Algorithm MINL[$\mathcal{C}$]: This algorithm finds a minimally generalized bp-graph pattern in $\mathcal{P}_{\Lambda,\Delta}[\mathcal{C}]$ for a given finite set S of $\mathcal{G}_{\Lambda,\Delta}[\mathcal{C}]$

Proof The correctness of algorithm MINL[$\mathcal{C}$] follows from Lemma 8. We analyze the running time of algorithm MINL[$\mathcal{C}$] (Fig. 8) for $\mathcal{C}$. The sets of labels Λ_S and Δ_S can be computed in $O(N_{\min}\mathcal{N})$ time. By using a linear time algorithm for finding all blocks in a graph (Aho et al. 1974), Υ_S is computed in at most $O(\mathcal{N} \cdot \mathcal{MC}(N_{\min}, N_{\min}))$ time. Let $\Upsilon_S = \{B_1, \ldots, B_\ell\}$ ($\ell \geq 0$). It is easy to see that $\sum_{i=1}^{\ell} |V(B_i)| = O(N_{\min})$. Since the number of vertices of the final output bp-graph pattern is not more than $N_{\min}$, during Step 2, Λ_S-LabeledVertexAddition and Υ_S-BlockReplacement are applied to p at most $O(N_{\min}^2|\Lambda_S|) = O(N_{\min}^3)$ times and $O(N_{\min}^2 \sum_{i=1}^{\ell} |V(B_i)|^2) = O(N_{\min}^4)$ times, respectively. During Steps 3 and 4, Δ_S-LabeledEdgeReplacement and TerminalVariableDeletion are applied to p at most $O(N_{\min}|\Delta_S|) = O(N_{\min}^2)$ times and $O(N_{\min})$ times, respectively. For each refinement operation, we have to decide whether or not a tem-

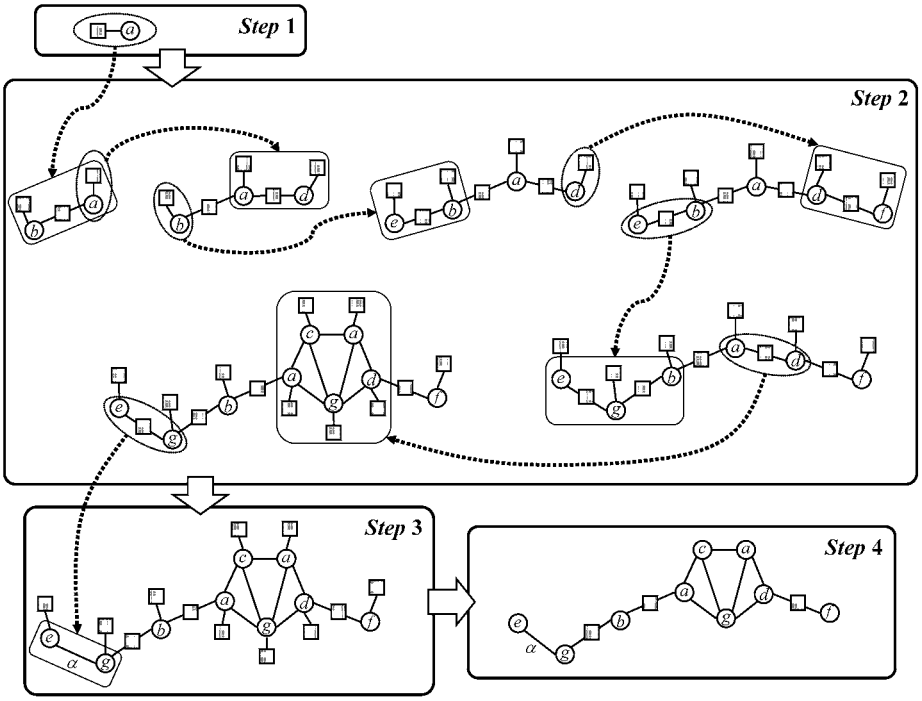


Fig. 9 An example process of finding a minimally generated bp-graph pattern explaining an example set of graphs in Fig. 10

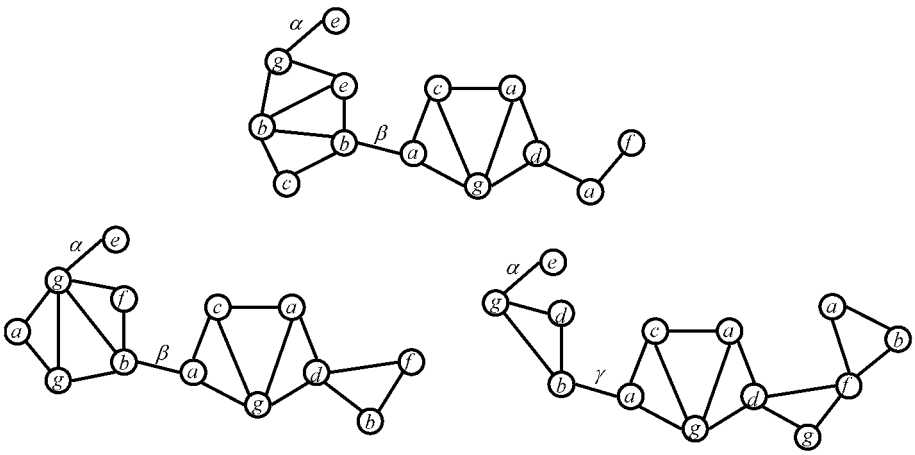


Fig. 10 An example set of outerplanar graphs: We assume that the edges with no label have a unique edge label except for α, β and γ

porary bpo-graph pattern explains S, and from Theorem 1, it needs at most $O(N_{\min}\mathcal{N}^2 \cdot \mathcal{MC}(N_{\min}, N_{\min}))$ time. Then, the total computational time of algorithm MINL[$\mathcal{C}$] is $O(N_{\min}^5\mathcal{N}^2 \cdot \mathcal{MC}(N_{\min}, N_{\min}))$. □

4.3 Minimal language problem for bpo-graph patterns

In this section, we show that algorithm MINL[$\mathcal{O}$] finds a minimally generalized bpo-graph pattern with respect to a given set of connected outerplanar graphs $S \subset \mathcal{G}_{\Lambda,\Delta}[\mathcal{O}]$ in polynomial time.

Corollary 2 *The minimal language problem for $\mathcal{L}_{\Lambda,\Delta}(\mathcal{O})$ is correctly computed in polynomial time.*

Proof Since the class $\mathcal{O}$ contains finitely many different blocks, e.g., circles of arbitrary length, algorithm MINL[$\mathcal{O}$] correctly finds a minimally generalized bpo-graph pattern explaining a given set of connected outerplanar graphs $S \subset \mathcal{G}_{\Lambda,\Delta}[\mathcal{O}]$.

For each refinement operation, we have to decide whether or not a temporary bpo-graph pattern explains S. From Corollary 1, it needs at most $O(N_{\min}\mathcal{N}^2\sqrt{d_{\max}})$ time where $d_{\max}$ is the maximum degree of outerplanar graphs in S. Therefore, in a similar argument to Theorem 2, the total computational time of algorithm MINL[$\mathcal{O}$] is $O(N_{\min}^5\mathcal{N}^2\sqrt{d_{\max}})$. □

Finally we have the following theorem and corollary.

Theorem 3 *The class $\mathcal{L}_{\Lambda,\Delta}(\mathcal{C})$ is polynomial time inductively inferable from positive data if the class $\mathcal{C}$ satisfies the following conditions.*

1. *$\mathcal{C}$ satisfies a connected hereditary property.*
2. *$\mathcal{C}$ contains infinitely many different biconnected graphs or Δ has infinitely many edge labels.*
3. *$\mathcal{MC}(n, N)$ is polynomial with respect to n and N.*

Proof This theorem follows from Lemma 7, Theorems 1 and 2. □

Corollary 3 *The class $\mathcal{L}_{\Lambda,\Delta}(\mathcal{O})$ is polynomial time inductively inferable from positive data.*

5 Pattern enumeration algorithms for frequent BPO-graph pattern problem

Let S be a finite subset of $\mathcal{G}_{\Lambda,\Delta}[\mathcal{O}]$ and p a bpo-graph pattern in $\mathcal{P}_{\Lambda,\Delta}[\mathcal{O}]$. Then, we denote by $O_S(p)$ the set of outerplanar graphs in S which are matched by p, called the *occurrence set* of p with respect to S. The *frequency* of p with respect to S, denoted by $supp_S(p)$, is defined as $supp_S(p) = |O_S(p)|/|S|$. Let t be a real number where $0 < t \leq 1$. A bpo-graph pattern $p \in \mathcal{P}_{\Lambda,\Delta}[\mathcal{O}]$ is *t-frequent* with respect to S if $supp_S(p) \geq t$. We call this real number t a *frequency threshold*. In this section, we give an algorithm for computing the following problem.

Frequent Block Preserving Outerplanar Graph Pattern Problem (FBPOGP Problem)
Input: A finite set of connected outerplanar graphs $S \subset \mathcal{G}_{\Lambda,\Delta}[\mathcal{O}]$ and a frequency threshold t where $0 < t \leq 1$.
Output: The set of all t-frequent bpo-graph patterns in $\mathcal{P}_{\Lambda,\Delta}[\mathcal{O}]$ with respect to S.

In this section, all vertices of the generated bpo-graph patterns are assumed to have terminal variables connecting to each of them. This is because we avoid to generate a huge

number of patterns by the mining process. Then any generated bpo-graph pattern has terminal variables (u) for all vertices u of the bpo-graph pattern and they are not removed from it during the process of a pattern generation.

Let F^t be the set of all t-frequent bpo-graph patterns with respect to S. Because the number of F^t may be exponential in the vertex size of S, we cannot solve the problem for enumerating F^t in polynomial time. We say that an algorithm solves the problem for enumerating F^t in *incremental polynomial time* if the algorithm enumerates the first k patterns in F^t in polynomial time with respect to the vertex sizes of S and the set of these k patterns for any k $(1 \leq k \leq |F^t|)$. In this section, we show that FBPOGP problem is computed in incremental polynomial time.

5.1 An apriori-like algorithm

We give an Apriori-like algorithm for computing FBPOGP problem. The algorithm is similar to the subgraph mining algorithm for outerplanar graphs described in (Horváth et al. 2006). For an integer $k \geq 0$, a *k-bpo-graph pattern* is defined to be a bpo-graph pattern such that the number of all bicomponents of it is equal to k. Let S be a set of outerplanar graphs in $\mathcal{G}_{\Lambda,\Delta}[\mathcal{O}]$ and t a real number where $0 < t \leq 1$. Let F_k^t be the set of all t-frequent k-bpo-graph patterns with respect to S. We give the pattern enumeration algorithm GENPATTERNS-1.0 in Fig. 11. The algorithm generates F_k^t from F_{k-1}^t for any $k \geq 2$ in a breadth-first manner.

First, we describe the detail of line 1 of GENPATTERNS-1.0 in which F_0^t and F_1^t are computed, and then give explanations of line 4 (procedure GENCANDIDATES).

Construct F_0^t and F_1^t w.r.t. S: F_0^t is the set of t-frequent 0-bpo-graph patterns consisting of a single vertex and a terminal variable connecting to the vertex. F_1^t is the set of t-frequent 1-bpo-graph patterns p of the following three types: (a) $p = (V(B), E(B), \{(u) \mid u \in V(B)\})$ where B is a block, and for two vertices u and v, (b) $p = (\{u, v\}, \{(u, v)\}, \{(u), (v)\})$ and (c) $p = (\{u, v\}, \emptyset, \{(u), (v), (u, v)\})$. F_0^t and F_1^t are computable in polynomial time with respect to the vertex size of S. In Fig. 12, we give examples of 0-bpo-graph patterns and 1-bpo-graph patterns.

Procedure GENCANDIDATES**:** In the procedure we generate the set of candidate frequent k-bpo-graph patterns. Each candidate is obtained from two $(k-1)$-bpo-graph patterns which have an isomorphic $(k-2)$-bpo-graph pattern.

For a k-bpo-graph pattern p $(k \geq 2)$, we say that a subgraph pattern p' of p is said to be *terminal* if p' is a 1-bpo-graph pattern and contains exactly one cutpoint of p. The cutpoint of p appearing in p' is called the *connected point* of p' to p. We denote by $p \ominus p'$ the $(k-1)$-bpo-subgraph pattern obtained from p by removing all vertices of p' except for the connected point of p'. We denote by $TB(p)$ the set of all terminal subgraph patterns of a bpo-graph pattern p. For two $(k-1)$-bpo-graph patterns p_1 and p_2, at line 4 we check whether or not $p_1 \ominus p_1'$ is isomorphic to $p_2 \ominus p_2'$ for each pair $p_1' \in TB(p_1)$ and $p_2' \in TB(p_2)$. If it is the case then we pick up all vertices of p_2 which correspond to the connected point u_1 of p_1' to p_1 (line 6). In order to obtain the set of vertices at line 6, for each vertex $u \in V(p_2 \ominus p_2')$ we decide whether or not two rooted bpo-graph patterns $(p_2 \ominus p_2')^u$ and $(p_2 \ominus p_2')^{\psi(u_1)}$ are isomorphic where ψ is an isomorphism from $p_1 \ominus p_1'$ to $p_2 \ominus p_2'$. If $(p_2 \ominus p_2')^u$ is isomorphic to $(p_2 \ominus p_2')^{\psi(u_1)}$, we see that u corresponds to the connected point u_1. For each vertex u which are obtained at line 6, we attach p_1' to p_2 by identifying u_1 with u and generate a new k-bpo-graph pattern (line 8). At line 9, for every candidate pattern p we check that p satisfies a necessary condition of the frequency pattern, that is, it decides whether or not

Algorithm: GenPatterns-1.0;
Input: a finite set of connected outerplanar graphs $S \subset \mathcal{G}_{\Lambda,\Delta}[\mathcal{O}]$
and a frequency threshold t where $0 < t \leq 1$;
Output: the set of all t-frequent bpo-graph patterns in $\mathcal{P}_{\Lambda,\Delta}[\mathcal{O}]$ w.r.t. S;
begin
1: Construct F_0^t and F_1^t w.r.t. S;
2: $k := 2$;
3: **while** $F_{k-1}^t \neq \emptyset$ **do begin**
4: $C_k^t :=$GenCandidates(F_{k-1}^t);
5: $F_k^t := \emptyset$;
6: **foreach** $p \in C_k^t$ **do**
7: **if** $supp_S(p) \geq t$ **then** add p to F_k^t;
8: $k := k + 1$
9: **end**;
10: **return** $\bigcup_{k \geq 0} F_k^t$
end.

Procedure GenCandidates(F_{k-1}^t);
Input: a set of frequent $(k-1)$-bpo-graph patterns F_{k-1}^t where $k \geq 2$;
Output: a set of candidates for frequent k-bpo-graph patterns C_k^t;
begin
1: $C_k^t := \emptyset$;
2: **foreach** $(p_1, p_2) \in F_{k-1}^t \times F_{k-1}^t$ **do begin**
3: **foreach** $(p_1', p_2') \in TB(p_1) \times TB(p_2)$ **do begin**
4: **if** $p_1 \ominus p_1' \cong p_2 \ominus p_2'$ **then begin**
5: Let u_1 be the connected point of p_1' to p_1
and ψ an isomorphism from $p_1 \ominus p_1'$ to $p_2 \ominus p_2'$.
6: $U := \{u \in V(p_2 \ominus p_2') \mid u$ maps to $\psi(u_1)$ by an automorphism of $p_2 \ominus p_2'\}$;
7: **foreach** $u \in U$ **do begin**
8: $p :=$ bpo-graph pattern obtained from p_1' and p_2 by identifying u_1 with u;
9: **if** $p \notin C_k^t \ \wedge \ p \ominus p' \in F_{k-1}^t$ for all $p' \in TB(p)$ **then** add p to C_k^t
10: **end**
11: **end**
12: **end**
13: **end**;
14: **return** C_k^t
end.

Fig. 11 An algorithm for generating all frequent bpo-graph patterns with respect to a given set of connected outerplanar graphs

$p \ominus p'$ is in F_{k-1}^t for each terminal subgraph pattern p' of p. If p is t-frequent, $p \ominus p'$ must be in F_{k-1}^t for each vertex $u \in V(p)$. This is a useful heuristic procedure to make a candidate set for F_k^t smaller.

In Fig. 13, we give an example process of generating candidate k-bpo-graph patterns from two $(k-1)$-bpo-graph patterns in the above way.

Lemma 9 *For any $k \geq 2$, procedure* GenCandidates *in Fig. 11 computes a set of k-bpo-graph patterns C_k^t which contains all t-frequent k-bpo-graph patterns in polynomial time with respect to the vertex size of F_{k-1}^t.*

Proof Let $\mathcal{N}$ be the vertex size of F_{k-1}^t and n the maximum number of vertices of bpo-graph patterns in F_{k-1}^t. Then the maximum number of vertices of bpo-graph patterns in C_k^t is at

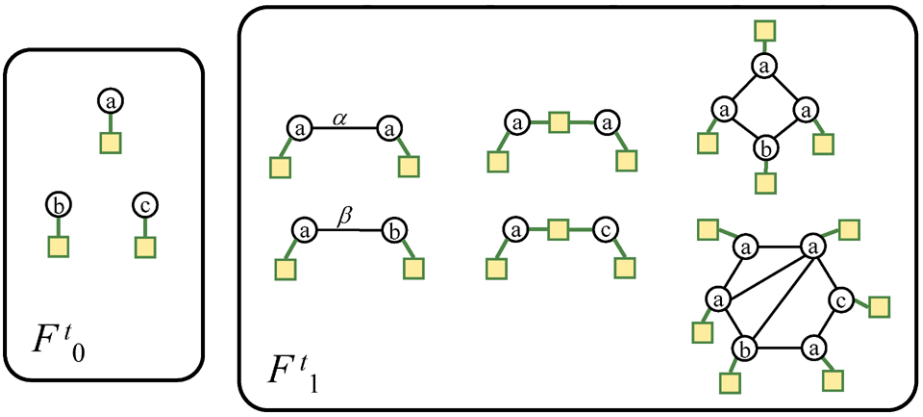


Fig. 12 Examples of 0-bpo-graph patterns and 1-bpo-graph patterns

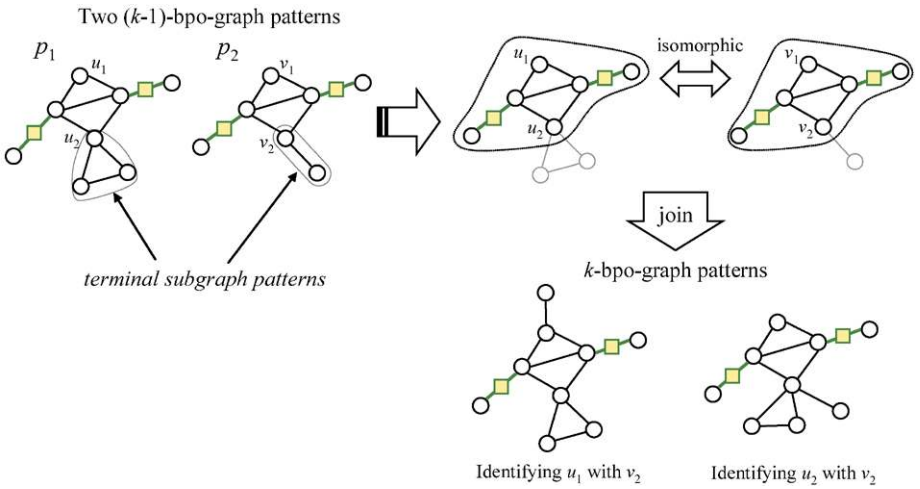


Fig. 13 An example of a generation of k-bpo-graph patterns from two $(k-1)$-bpo-graph patterns p_1 and p_2. In this case, the $(k-2)$-bpo-graph pattern which is subgraph isomorphic to p_1 and p_2 has more than one automorphisms, therefore, we can generate more than one k-bpo-graph patterns from p_1 and p_2. We omit terminal variables in this figure

most $2n$. The double foreach-loop at lines 2 and 3 iterates $O(\mathcal{N}^2)$ times. At line 4 we decide in linear time whether or not the two bpo-graph patterns $p_1 \ominus p'_1$ and $p_2 \ominus p'_2$ are isomorphic each other by using a linear time isomorphism algorithm for planar graphs in (Hopcroft and Wong 1974). The set of vertices U at line 6 is computed in $O(n^2)$ time. In order to quickly decide whether or not a bpo-graph pattern is included in a set of bpo-graph pattern, we adopt a canonical string representation described in (Horváth et al. 2006), which satisfies that two outerplanar graphs have the same canonical string if and only if they are isomorphic. We transform a bpo-graph pattern p into its canonical string in $O(|V(p)|^2 \log |V(p)|)$, and computes the if-statement at line 9 in $O(n \log |C^t_k| + n(n^2 \log n + n \log |F^t_{k-1}|))$ time. Since the number of k-bpo-graph patterns obtained from two $(k-1)$-bpo-graph patterns p_1 and p_2 is at most $|V(p_1)| \cdot |V(p_2)|$, the number of bpo-graph patterns in C^t_k is at most $\mathcal{N}^2$. Then

the total time complexity of procedure GENCANDIDATES is $O(n^3\mathcal{N}^2(n\log n+\log\mathcal{N}))$. It is polynomial time with respect to $\mathcal{N}$. □

Frequency calculation: At line 6 of algorithm GENPATTERNS-1.0, we decide whether or not a candidate k-bpo-graph pattern is t-frequent with respect to S. We use a well-known heuristics to reduce the number of pattern matching operations. Let p be a candidate k-bpo-graph pattern obtained from two $(k-1)$-bpo-graph patterns p_1 and p_2. In order to compute the occurrence set $O_S(p)$, it is enough to compute the occurrence set of p with respect to $O_S(p_1)\cap O_S(p_2)$ because p satisfies that $L_{\mathcal{O}}(p)\subseteq L_{\mathcal{O}}(p_1)$ and $L_{\mathcal{O}}(p)\subseteq L_{\mathcal{O}}(p_2)$.

From Theorem 1 and Lemma 9, we have the following lemma.

Lemma 10 *Let S be a finite set of connected outerplanar graphs in $\mathcal{G}_{\Lambda,\Delta}[\mathcal{O}]$ and t a frequency threshold where $0<t\leq 1$. For any $k\geq 2$, F_k^t is computed from F_{k-1}^t in polynomial time with respect to the vertex sizes of F_{k-1}^t and S by algorithm* GENPATTERNS-1.0.

Finally we have the following theorem.

Theorem 4 *Algorithm* GENPATTERNS-1.0 *correctly solves FBPOGP problem in incremental polynomial time.*

5.2 A refinement-based algorithm

In this section, we give an improved version of algorithm GENPATTERNS-1.0 and describe pattern enumeration algorithm GENPATTERNS-2.0 in Fig. 14. Algorithm GENPATTERNS-2.0 consists of two main parts, a generation process of the most generalized bpo-graph patterns and a refinement process of them. First we describe the detail of the generation of most generalized bpo-graph patterns at lines 5–9 of GENPATTERNS-2.0 and then give explanations of GENREFINEDPATTERNS at line 11 and COUNTFREQUENCY at line 12.

Generation of the most generalized bpo-graph patterns in F_k^t. For a set of bpo-graph patterns F and a bpo-graph pattern $p\in F$, we say that p is the *most generalized bpo-graph pattern* in F if there does not exist a bpo-graph pattern $q\in F$ such that $L_{\mathcal{O}}(p)\subsetneq L_{\mathcal{O}}(q)$. We have the following lemma.

Lemma 11 *A bpo-graph pattern $p\in F_k^t$ is the most generalized bpo-graph pattern in F_k^t if and only if p has no bridge and no block which has at most 2 cutpoints.*

Proof If p has a bridge or a block which has at most 2 cutpoints, a bpo-graph pattern q obtained from p by replacing the bridge or block with a bridge variable is in F_k^t and satisfies $L_{\mathcal{O}}(p)\subsetneq L_{\mathcal{O}}(q)$. Therefore p is not the most generalized bpo-graph pattern in F_k^t.

Conversely we suppose that q is a k-bpo-graph pattern such that $L_{\mathcal{O}}(p)\subsetneq L_{\mathcal{O}}(q)$. Let G be an outerplanar graph which is obtained from p by replacing all bridge variables with cycles of length $|V(p)|+1$, and all terminal variables with single vertices. Since $G\in L_{\mathcal{O}}(q)$, there is a substitution θ such that $G\cong q\theta$. Since both p and q are k-bpo-graph patterns, any binding for a bridge variable in θ is to replace it with either one labeled edge or one block. If θ contains a binding for a bridge variable which is replaced with a graph having at most $|V(p)|$ vertices, from the construction of G, p has a bridge or a block which has at most 2 cutpoints. Otherwise $p\cong q$ holds. It contradicts that $L_{\mathcal{O}}(p)\subsetneq L_{\mathcal{O}}(q)$. □

Algorithm: GENPATTERNS-2.0;
Input: a set of connected outerplanar graphs S and a frequency threshold $t(0 < t \leq 1)$;
Output: the set of all t-frequent bpo-graph patterns w.r.t. S;
begin
1: Construct F_0^t and F_1^t w.r.t. S;
2: Let F_b be the set of all bpo-graph patterns in F_1^t which have no bridge variable.
3: $k := 2$;
4: **while** $F_{k-1}^t \neq \emptyset$ **do begin**
5: $F_M :=$ the set of all bpo-graph patterns in F_{k-1}^t which are subgraph patterns of the most generalized bpo-graph patterns in F_k^t;
6: $F'_M := \emptyset$; $F_k^t := \emptyset$;
7: $C_M :=$GENCANDIDATES-R(F_M);
8: **foreach** $p \in C_M$ **do**
9: **if** $supp_S(p) \geq t$ **then** add p to F'_M and F_k^t;
10: **foreach** $p_M \in F'_M$ **do begin**
11: $C :=$GENREFINEDPATTERNS(p_M, F_{k-1}^t, F_b);
12: $F :=$COUNTFREQUENCY$(C, O_S(p_M), t|S|)$;
13: $F_k^t := F_k^t \cup F$
14: **end**;
15: $k := k + 1$
16: **end**;
17: **return** $\bigcup_{k \geq 0} F_k^t$
end.

Fig. 14 An algorithm for generating all frequent bpo-graph patterns with respect to a given set of connected outerplanar graphs

From Lemma 11, any $(k-1)$-bpo-graph pattern which is a subgraph pattern of the most generalized bpo-graph patterns in F_k^t satisfies that (1) there is no bridge and (2) each block has at least 2 cutpoints. At line 5 of GENPATTERNS-2.0 we extract the set F_M of all bpo-graph patterns satisfying (1) and (2) from F_k^t.

At line 7, we generate candidates of the most generalized bpo-graph patterns in F_{k+1}^t from F_M. We use a similar method to procedure GENCANDIDATES in algorithm GENPATTERNS-1.0 in order to generate candidates, but we note that for an input set F_M, procedure GENCANDIDATES generates extra bpo-graph patterns which have a block having at most 2 cutpoints. So, we use procedure GENCANDIDATES-R which is improved so as to generate only bpo–graph patterns which meet the requirements to be most generalized. The set of all most generalized bpo-graph patterns is computable in polynomial time with respect to the vertex sizes of F_M and S. It is the same time complexity as that of algorithm GENPATTERNS-1.0.

Procedure GENREFINEDPATTERNS**.** We describe procedure GENREFINEDPATTERNS in Fig. 15. The procedure is based on the following lemma.

Lemma 12 *For any t-frequent k-bpo-graph pattern $p \in F_k^t$, there exists the most generalized bpo-graph pattern p_M in F_k^t and a substitution θ such that $p \cong p_M\theta$.*

Proof Let p_M be a bpo-graph pattern which is obtained from p by replacing all bridges and blocks containing at most 2 cutpoints by bridge variables. It is clear that p_M is the most generalized bpo-graph pattern in F_k^t. Let θ be a substitution which replaces all variables made by the above operation with the original bridges and blocks, then $p \cong p_M\theta$ holds. □

Procedure GENREFINEDPATTERNS(p_M, F^t_{k-1}, F_b);
Input: one of the most generalized k-bpo-graph patterns p_M,
a set of frequent $(k-1)$-bpo-graph patterns F^t_{k-1} and
a set of frequent 1-bpo-graph patterns which have no bridge variable F_b;
Output: a set of k-bpo-graph patterns p such that $L_{\mathcal{O}}(p) \subseteq L_{\mathcal{O}}(p_M)$;
begin
1: $C := \emptyset$; $C_1 := \{p_M\}$; $C_2 := \emptyset$;
2: **while** $C_1 \neq \emptyset$ **do begin**
3: **foreach** $p \in C_1$ **do begin**
4: $D :=$REPLACEVARIABLE(p, F^t_{k-1}, F_b); $C_2 := C_2 \cup D$
5: **end**;
6: $C := C \cup C_2$; $C_1 := C_2$; $C_2 := \emptyset$
7: **end**;
8: **output** C
end.

Procedure REPLACEVARIABLE(p, F^t_{k-1}, F_b);
Input: a k-bpo-graph pattern p and sets of bpo-graph patterns F^t_{k-1} and F_b;
Output: a set of k-bpo-graph patterns obtained from p by applying refinement operators;
begin
1: $C := \emptyset$;
2: **foreach** $((u_1, u_2), p_b) \in H(p) \times F_b$ **do begin**
3: **foreach** $(v_1, v_2) \in V(p_b) \times V(p_b)$ **do begin**;
4: **if** $v_1 \neq v_2 \ \wedge \ \lambda_p(u_1) = \lambda_{p_b}(v_1) \ \wedge \ \lambda_p(u_2) = \lambda_{p_b}(v_2)$ **then begin**
5: $q := p\{x((u_1, u_2)) := [p_b, (v_1, v_2)]\}$;
6: **if** $q \notin C \ \wedge \ q \ominus q' \in F^t_{k-1}$ for all $q' \in TB(q)$ **then** add q to C;
7: **end**
8: **end**
9: **end**;
10: **output** C
end.

Fig. 15 Procedure GENREFINEDPATTERNS: We denote by $TB(p)$ the set of all terminal bpo-subgraph patterns of p

GENREFINEDPATTERNS receives one of the most generalized t-frequent k-bpo-graph patterns p_M, F^t_{k-1} and the set F_b of t-frequent 1-bpo-graph patterns which have no bridge variable. And it returns a superset of the set of t-frequent k-bpo-graph patterns p such that $L_{\mathcal{O}}(p) \subseteq L_{\mathcal{O}}(p_M)$. For a bpo-graph pattern p which has ℓ bridge variables ($\ell \geq 1$), procedure REPLACEVARIABLE proceeds to generate bpo-graph patterns q which have $\ell - 1$ bridge variables, in a similar way to refinement operators LABELEDEDGEREPLACEMENT and BLOCKREPLACEMENT in Sect. 4.

Procedure COUNTFREQUENCY**.** We describe this procedure in Fig. 16. The procedure exactly selects all t-frequent k-bpo-graph patterns from the output of procedure GENREFINEDPATTERNS. Let p_M be the most generalized k-bpo-graph pattern in F^t_k and $U = O_S(p_M)$. Let C be the output set of GENREFINEDPATTERNS for an input p_M. Since any bpo-graph pattern $p \in C$ satisfies $L_{\mathcal{O}}(p) \subseteq L_{\mathcal{O}}(p_M)$, the occurrence set of p with respect to S is computed from U. If there is a bpo-graph pattern $q \in C$ such that $L_{\mathcal{O}}(q) \subseteq L_{\mathcal{O}}(p)$ and the occurrence set of q has already been computed, the occurrence set of p is computed only from $U - O_S(q)$. In this way, we reduce the number of pattern matchings in this procedure. First of all, COUNTFREQUENCY computes all occurrence sets of bpo-graph patterns in C which have no variable, and decide whether or not they are frequent (lines 3–6). Next, for all $i \geq 1$, the procedure computes the occurrence sets of all bpo-graph patterns in C

Procedure CountFrequency(C, U, τ);
Input: the output set C of GenRefinedPatterns for an input k-bpo-graph pattern p_M, the occurrence set U of p_M w.r.t. S and a real number $\tau = t|S|$;
Output: the set of bpo-graph patterns $p \in C$ such that $|O_U(q)| \geq \tau$;
begin
1: Let C_i $(0 \leq i \leq k)$ be the set of all bpo-graph patterns in C which have exactly i bridge variables.
2: $F := \emptyset$;
3: **foreach** $p \in C_0$ **do begin**
4: $O_U(p) := \{G \in U \mid p \text{ matches } G\}$;
5: **if** $|O_U(p)| \geq \tau$ **then** add p to F
6: **end**;
7: **for** $i := 1$ **to** k **do begin**
8: **foreach** $p \in C_i$ **do begin**
9: $W := \emptyset$;
10: **foreach** $q \in C_{i-1}$ such that $L_{\mathcal{O}}(q) \subseteq L_{\mathcal{O}}(p)$ **do** $W := W \cup O_U(q)$;
11: $O_U(p) := \{G \in U - W \mid p \text{ matches } G\} \cup W$;
12: **if** $|O_U(p)| \geq \tau$ **then** add p to F
13: **end**
14: **end**;
15: **output** F
end.

Fig. 16 An algorithm for counting frequencies of given bpo-graph patterns

which have i variables by using the occurrence sets of bpo-graph patterns which have $i-1$ variables (lines 7–14). To be precise, in order to compute the occurrence set of a bpo-graph pattern p, first we compute the union W of occurrence sets of bpo-graph patterns which are obtained by applying procedure ReplaceVariable to p (line 10). Then we compute the occurrence set of p with respect to $U - W$ and obtain $O_U(p)$ (line 11).

We have the following lemma.

Lemma 13 *Let S be a finite set of connected outerplanar graphs in $\mathcal{G}_{\Lambda,\Delta}[\mathcal{O}]$, t a frequency threshold where $0 < t \leq 1$, and F_b a set of t-frequent 1-bpo-graph patterns which have no bridge variable. For any $k \geq 2$, F_k^t is computed from F_{k-1}^t in polynomial time with respect to the vertex sizes of F_{k-1}^t, F_b and S by algorithm* GenPatterns-2.0.

Proof Let $\mathcal{N}$ be the vertex size of F_{k-1}^t, n the maximum number of vertices of bpo-graph patterns in F_{k-1}^t, and $\mathcal{N}_b$ the vertex size of F_b. All $(k-1)$-bpo-subgraph patterns of each candidate k-bpo-graph pattern generated by GenRefinedPatterns are t-frequent because of the reduction method at line 6 of ReplaceVariable. Therefore, because each candidate is also obtained from two $(k-1)$-bpo-graph patterns, the number of all k-bpo-graph patterns outputted by GenRefinedPatterns during the foreach-loop at line 10 of GenPatterns-2.0 is less than the number of k-bpo-graph patterns outputted by procedure GenCandidates in algorithm GenPatterns-1.0, that is, at most $\mathcal{N}^2$. In procedure ReplaceVariable, the double loop at lines 2 and 3 iterates $O(k\mathcal{N}_b^2)$ time, and the if-statement at line 4 is computed in polynomial time with respect to n and $\mathcal{N}$. Hence, procedure GenRefinedPatterns works in polynomial time with respect to n, $\mathcal{N}$ and $\mathcal{N}_b$. From Theorem 1, procedure CountFrequency works in polynomial time with respect to $\mathcal{N}$ and the vertex size of S. Consequently, algorithm GenPatterns-2.0 outputs F_k^t in polynomial time with respect to $\mathcal{N}$, $\mathcal{N}_b$ and the vertex size of S. □

From Lemma 13, we have the following theorem.

Theorem 5 *Algorithm* GENPATTERNS-2.0 *correctly solves FBPOGP problem in incremental polynomial time.*

5.3 Maximal frequent bpo-graph pattern problem

A bpo-graph pattern $p \in \mathcal{P}_{\Lambda,\Delta}[\mathcal{O}]$ is said to be a *maximal t-frequent bpo-graph pattern* with respect to a set of connected outerplanar graphs $S \subseteq \mathcal{G}_{\Lambda,\Delta}[\mathcal{O}]$ if p is t-frequent with respect to S and there is no t-frequent bpo-graph pattern q with respect to S satisfying $L_{\mathcal{O}}(q) \subsetneq L_{\mathcal{O}}(p)$. In this section, we give procedures for computing the following problem.

Maximal Frequent BPO-Graph Pattern Problem (MFBPOGP Problem)
Input: A finite set of connected outerplanar graphs $S \subset \mathcal{G}_{\Lambda,\Delta}[\mathcal{O}]$ and a frequency threshold t where $0 < t \leq 1$.
Output: The set of all maximal t-frequent bpo-graph patterns in $\mathcal{P}_{\Lambda,\Delta}[\mathcal{O}]$ with respect to S.

Let M_k^t be the set of all maximal t-frequent k-bpo-graph patterns with respect to S. For a t-frequent bpo-graph pattern p, we decide whether or not p is maximal t-frequent by using LABELEDVERTEXADDITION, BLOCKREPLACEMENT and LABELEDEDGEREPLACEMENT in Sect. 4. For each bridge variable h of p, if at least one bpo-graph pattern q which is obtained from p by applying either of the three refinement operators to h is t-frequent, p is not maximal t-frequent. For more details, we computes M_k^t by applying the following three maximality tests on F_k^t and F_{k+1}^t generated by GENPATTERNS-2.0.

Maximality Test 1: For a t-frequent k-bpo-graph pattern p, let $R(p)$ be the set of k-bpo-graph patterns generated by procedure REPLACEVARIABLE in Fig. 15. If $F_k^t \cap R(p) \neq \emptyset$, we conclude that p is not maximal t-frequent. Let Δ_S^t and Υ_S^t be the sets of all edge labels and blocks appearing in at least $t|S|$ graphs in S, respectively. This maximality test corresponds to Δ_S^t-LABELEDEDGEREPLACEMENT and Υ_S^t-BLOCKREPLACEMENT for each bridge variable of p. This maximality test are executed in procedure COUNTFREQUENCY.

Let Λ_S^t be the set of all vertex labels appearing in at least $t|S|$ graphs in S. The following two maximality tests correspond to Λ_S^t-LABELEDVERTEXADDITION to each variable of p. For these maximality tests, we need only a subset of Λ_S^t.

Maximality Test 2: For a k-bpo-graph pattern $p \in F_k^t$, if there exists a $(k+1)$-bpo-graph pattern $p' \in F_{k+1}^t$ which has p as its bpo-subgraph pattern, p is not maximal t-frequent. Let Λ_1 be the set of all vertex labels appearing in F_1^t. This maximality test corresponds to Λ_1-LABELEDVERTEXADDITION for each terminal variable of p. In order to do this maximality test easily, we make the bidirectional links between p and all $(k+1)$-bpo-graph patterns p' which have p as its bpo-subgraph pattern. These links are added when the if-statement at line 6 of every REPLACEVARIABLE application is executed.

Maximality Test 3: Let Λ_2 be the set of all vertex labels of cutpoints of bpo-graph patterns in F_2^t consisting of exactly two bridge variables. For each bridge variable h of p, if at least one of $(k+1)$-bpo-graph patterns p' obtained from p by applying Λ_2-LABELEDVERTEXADDITION to h is in F_{k+1}^t, p is not maximal t-frequent.

The next theorem is proved in a similar way to Lemma 8.

Theorem 6 *Let p be a t-frequent bpo-graph pattern with respect to a given set of connected outerplanar graphs $S \subseteq \mathcal{G}_{\Lambda,\Delta}[\mathcal{O}]$. Procedures Maximality Tests* 1–3 *correctly decide whether or not p is maximal t-frequent with respect to S.*

5.4 Experimental results

We have implemented our frequent bpo-graph pattern mining algorithms and tested on chemical datasets. In our experiments, we used a dataset containing 250,251 chemical compounds, which is available from National Cancer Institute (NCI) (NCI 2000). The NCI dataset contains 236,180 compounds which can be expressed by connected or disconnected outerplanar graphs. In particular, we used a set of 223,912 compounds which can be expressed by connected outerplanar graphs. In the dataset, there are 81 distinct vertex labels which correspond to types of atoms, three distinct edge labels which correspond to types of bonds, and 10,374 pairwise non-isomorphic blocks. Though, there are only 6 atom types and one block, that is, C, Cl, H, N, O, S, and the benzene ring, whose frequencies are beyond 0.1. We experimented on various subsets of the original dataset of 223,912 compounds, however frequent atom types and blocks of datasets used in our experiments are almost the same as those of the original dataset. The implementation is by javac 1.5.0_07 and on an Intel Core 2 Duo E6300 (1.86 GHz) machine with 2.00 GB RAM, running the Linux operating system. All the running times reported as follows are in seconds.

5.4.1 Apriori-like algorithm versus refinement-based algorithm

First, we carried out experiments to compare performances of algorithm GENPATTERNS-1.0 with performances of algorithm GENPATTERNS-2.0. We used a dataset consisting of 100 outerplanar molecular graphs, which are chosen from NSC number 1 to 100. We set frequency thresholds to be 0.5, 0.4, 0.3, 0.2 and 0.1, and tested on the dataset with respect to the frequencies. We show the results with frequency thresholds 0.5 and 0.4 in Table 1, with frequency thresholds 0.3 and 0.2 in Table 2, and with frequency threshold 0.1 in Table 3. We only show experimental results obtained by experiments which finished in about 250,000 seconds (about 70 hours). For frequencies 0.3, 0.2 and 0.1, algorithm GENPATTERNS-1.0 did not complete within the time frame.

In the tables, C_k^t means the set of all candidate k-bpo-graph patterns outputted by GENCANDIDATES-R and GENREFINEDPATTERNS during the k-th iteration of the while-loop at lines 4–16 in algorithm GENPATTERNS-2.0 (see Fig. 14). In the tables we omit the number of candidates of algorithm GENPATTERNS-1.0 (i.e., the number of candidate k-bpo-graph patterns generated by GENCANDIDATES), since it was almost the same as the number of candidates of algorithm GENPATTERNS-2.0. This is because both algorithms apply the same reduction method at line 9 in GENCANDIDATES and line 6 in REPLACEVARIABLE. The average of $|C_k^t|/|F_k^t|$ is very close to 1. If the reduction method does not applied, the numbers of candidates rise about 3–7 times. Therefore, we succeed to reduce candidate bpo-graph patterns and the running time for computing the frequencies of candidates.

We utilize only the most generalized $(k-1)$-bpo-graph patterns from F_{k-1}^t to generate candidates of t-frequent k-bpo-graph patterns. Therefore GENPATTERNS-2.0 becomes faster than GENPATTERNS-1.0. For example, for frequency threshold 0.3, the total running time of GENPATTERNS-1.0 for the generation of $F_{10}^{0.3}$ from $F_0^{0.3}$ is 122,015 seconds. On the other hand, the total running time of GENPATTERNS-2.0 is 426 seconds, which is reduced to about 1/286 of that of GENPATTERNS-1.0. For frequency 0.1, GENPATTERNS-2.0 could find as many as 48 million frequent bpo-graph patterns. The column specified with M_k^t shows the numbers of bpo-graph patterns in M_k^t, which are found to be extremely smaller than $|F_k^t|$ for any frequency threshold. The numbers of frequent patterns and maximal frequent patterns depend on an input dataset. In this case it would appear that the dataset does not contain scattering data and there are many graphs which have large common structures.

Table 1 Experimental results on a dataset containing 100 outerplanar molecular graphs with frequency thresholds 0.5 and 0.4. T_1 (resp. T_2) is the running time in seconds for the generation of F_k^t from F_{k-1}^t by GENPATTERNS-1.0 (resp. GENPATTERNS-2.0) and T_M is the running time for the generation of M_k^t from F_k^t and F_{k+1}^t by maximality tests

k	$\|S\| = 100, t = 0.5$					
	$\|C_k^t\|$	$\|F_k^t\|$	$\|M_k^t\|$	T_1	T_2	T_M
0	10	4	0	0.09	0.07	0
1	53	10	0	0.7	0.2	0.02
2	32	23	0	1	2	0.02
3	154	96	3	3	3	0.08
4	314	280	8	7	2	0.15
5	753	713	4	16	5	0.15
6	1368	1332	15	35	6	0.23
7	1714	1696	15	68	6	0.23
8	1373	1367	18	96	5	0.18
9	606	606	34	69	3	0.04
10	107	107	13	19	1	0
11	0	0	0	1	0	0
12	0	0	0	0	0	0
13	0	0	0	0	0	0
14	0	0	0	0	0	0
15	0	0	0	0	0	0
Total	6484	6234	110	316	34	1
k	$\|S\| = 100, t = 0.4$					
	$\|C_k^t\|$	$\|F_k^t\|$	$\|M_k^t\|$	T_1	T_2	T_M
0	10	4	0	0.1	0.1	0
1	53	13	1	0.6	0.5	0.01
2	51	31	5	1	1	0.03
3	216	122	2	4	3	0.05
4	463	231	5	9	3	0.25
5	1293	1233	13	27	8	0.25
6	3010	2961	16	82	10	0.55
7	6290	6261	32	277	15	1.3
8	11042	11026	23	1261	32	2.8
9	15326	15326	26	6590	64	3.5
10	15781	15781	30	18013	81	3.0
11	11540	11540	29	38740	60	1.7
12	5596	5596	8	30777	25	0.57
13	1556	1556	6	10942	6	0.08
14	170	170	2	1231	1	0.01
15	0	0	0	29	0	0
Total	72397	72051	198	107984	308	14

Table 2 Experimental results on a dataset containing 100 outerplanar molecular graphs with frequency thresholds 0.3 and 0.2

k	$\|S\| = 100, t = 0.3$					
	$\|C_k^t\|$	$\|F_k^t\|$	$\|M_k^t\|$	T_1	T_2	T_M
0	10	4	0	0.1	0.1	0
1	53	15	0	0.6	0.7	0.02
2	72	39	5	1	1	0.06
3	250	146	2	3	3	0.09
4	622	555	6	8	3	0.19
5	1871	1798	16	34	8	0.46
6	5057	4996	21	127	13	1.1
7	12432	12390	39	564	27	3.4
8	26350	26330	67	2451	61	5.3
9	46706	46677	27	16624	107	12
10	67938	67938	41	102200	202	17
11	80945	80945	50	–	256	19
12	77220	77220	27	–	240	16
13	56164	56164	3	–	160	9.7
14	28744	28744	0	–	70	3.5
15	9112	9112	5	–	21	0.6
16	1360	1360	4	–	3	0.01
17	0	0	0	–	0.05	0
18	0	0	0	–	0.05	0
19	0	0	0	–	0.05	0
Total	414906	414433	313	122015	1176	89
k	$\|S\| = 100, t = 0.2$					
	$\|C_k^t\|$	$\|F_k^t\|$	$\|M_k^t\|$	T_1	T_2	T_M
0	10	4	0	0.1	0.07	0
1	53	17	1	0.6	0.2	0.03
2	89	58	11	1	2	0.02
3	316	185	7	4	3	0.06
4	816	728	10	10	4	0.17
5	2753	2574	17	40	9	0.59
6	8129	7921	46	166	17	1.9
7	22258	22151	68	782	33	6.4
8	55359	55303	113	5839	98	13
9	123632	123547	126	43091	178	31
10	242757	242754	67	–	424	63
11	412249	412249	57	–	977	108
12	585305	585305	78	–	2002	140
13	666128	666128	75	–	2795	139
14	575174	575174	12	–	2562	94
15	352960	352960	1	–	1339	48
16	139752	139752	4	–	505	14
17	29912	29912	8	–	127	1.2
18	2048	2048	1	–	13	0.02
19	0	0	0	–	0	0
Total	3219682	3218770	702	49933	11088	661

Table 3 Experimental results on a dataset containing 100 outerplanar molecular graphs with frequency threshold 0.1

k	$\|S\| = 100, t = 0.1$					
	$\|C_k^t\|$	$\|F_k^t\|$	$\|M_k^t\|$	T_1	T_2	T_M
0	10	5	0	0.07	0.08	0
1	63	24	0	0.7	0.2	0.02
2	150	99	8	1	2	0.04
3	499	296	17	5	3	0.07
4	1202	1041	8	9	4	0.25
5	4412	4128	17	36	9	0.91
6	15369	15010	19	177	23	3.6
7	49739	49379	60	1379	43	15
8	147809	147461	114	14431	133	47
9	401598	401296	248	188536	491	154
10	978266	978153	233	–	1968	307
11	2100702	2100696	171	–	5687	884
12	3909387	3909387	116	–	11624	1464
13	6206092	6206092	182	–	27025	2191
14	8225737	8225737	230	–	49285	2944
15	8889001	8889001	80	–	89743	2724
16	7673045	7673045	20	–	42183	2028
17	5215896	5215896	10	–	16588	1202
18	2751960	2751960	26	–	4354	426
19	1087072	1087072	7	–	1649	164
20	290560	290560	0	–	387	25
21	39776	39776	1	–	67	0.17
22	0	0	0	–	1	0
Total	47988345	47986114	1193	204575	251269	14582

5.4.2 Scalability

We carried out experiments to test the scalability of algorithm GENPATTERNS-2.0. First, we evaluated the scalability with respect to the dataset size. We used four datasets which contain 1,000, 2,000, 5,000, and 10,000 outerplanar molecular graphs, respectively. Each dataset was randomly chosen from the original dataset, so that the average number of vertices of graphs is about 30. We set frequency thresholds to be 0.5, 0.4, 0.3 and 0.2, and tested on each dataset with respect to the frequencies. The results are given in Table 4. As expected, as the dataset size increases the running time increases. For example, for frequency 0.2, on datasets of 1,000, 2,000, 5,000, and 10,000 compounds, the times needed for generation per pattern are 12, 23, 52 and 95 milliseconds, respectively. We can see that the time needed per pattern scales linearly with respect to the dataset size.

Next, we evaluated the scalability with respect to the graph size, i.e., the number of vertices. We created three datasets which contain 1,000 outerplanar molecular graphs respectively, so that each dataset has a different average graph size. The average graph sizes of datasets we created are 21, 29 and 38, respectively. We modulated the frequency thresh-

Table 4 Dataset size and running time. t is the frequency threshold, $k_{\max}$ is the maximum level (i.e., the number of bicomponents) of t-frequent bpo-graph patterns, $F^t = \bigcup_{k=0}^{k_{\max}} F_k^t$, $M^t = \bigcup_{k=0}^{k_{\max}} M_k^t$, and T is the running time in seconds for the generation of all t-frequent bpo-graph patterns by GENPATTERNS-2.0

t	$\lvert S\rvert = 1,000$				$\lvert S\rvert = 2,000$			
	$k_{\max}$	$\lvert F^t\rvert$	$\lvert M^t\rvert$	T	$k_{\max}$	$\lvert F^t\rvert$	$\lvert M^t\rvert$	T
0.5	8	612	64	33	7	505	48	52
0.4	9	2429	216	95	9	1887	187	147
0.3	13	13468	850	326	12	10148	732	429
0.2	16	133202	5454	1618	15	101130	4917	2369
t	$\lvert S\rvert = 5,000$				$\lvert S\rvert = 10,000$			
	$k_{\max}$	$\lvert F^t\rvert$	$\lvert M^t\rvert$	T	$k_{\max}$	$\lvert F^t\rvert$	$\lvert M^t\rvert$	T
0.5	8	543	55	134	8	846	73	403
0.4	9	2090	211	430	10	3629	281	947
0.3	12	11349	824	1293	13	20972	1447	4297
0.2	15	130812	7102	6809	16	275710	13703	26198

Table 5 Average graph size and running time. t is the frequency threshold, $k_{\max}$ is the maximum level of t-frequent bpo-graph patterns, $F^t = \bigcup_{k=0}^{k_{\max}} F_k^t$, $M^t = \bigcup_{k=0}^{k_{\max}} M_k^t$, and T is the running time in seconds for the generation of all t-frequent bpo-graph patterns by GENPATTERNS-2.0

t	$\lvert S\rvert = 1,000$, Avg = 21				$\lvert S\rvert = 1,000$, Avg = 29				$\lvert S\rvert = 1,000$, Avg = 38			
	$k_{\max}$	$\lvert F^t\rvert$	$\lvert M^t\rvert$	T	$k_{\max}$	$\lvert F^t\rvert$	$\lvert M^t\rvert$	T	$k_{\max}$	$\lvert F^t\rvert$	$\lvert M^t\rvert$	T
0.5	6	189	22	11	8	807	83	45	11	8149	502	348
0.4	7	435	61	18	9	2914	208	90	13	39718	2137	1065
0.3	8	1331	163	31	11	11375	695	210	16	257329	10455	3698
0.2	10	6532	516	76	14	67017	2893	658	19	3263868	61891	22446

old from 0.5 to 0.2, and tested on each dataset with respect to the frequency threshold. The results are given in Table 5. The table shows that as we increase the average graph size the number of frequent bpo-graph patterns increases, but the time needed for generation per pattern does not increase. For example, for frequency 0.2, on datasets whose average graph sizes are 21, 29 and 38, the times needed per pattern are 12, 10 and 7 milliseconds, respectively. For each dataset and frequency threshold, more than 90% of the total time is the time for counting frequency. From Theorem 1, if we naively counted frequencies, the time needed per pattern would vary directly with the square of the average graph size. The reason why it was not the case is that as the number of bpo-graph patterns generated from one most generalized bpo-graph pattern increases, the occurrence sets of bpo-graph patterns overlaps more often and the number of pattern matching operations in COUNTFREQUENCY decreases.

5.4.3 Study with respect to bridge variables

We divided sets of frequent bpo-graph patterns discovered in experiments on the dataset containing 10,000 compounds with respect to the number of bridge variables. The results

Table 6 The number of frequent bpo-graph patterns with respect to the number of bridge variables, which are discovered in the experiment on the dataset containing 10,000 compounds. $F^t(r)$ (resp. $M^t(r)$) means the set of all t-frequent bpo-graph patterns (resp. maximal t-frequent bpo-graph patterns) with r bridge variables where $r \geq 0$

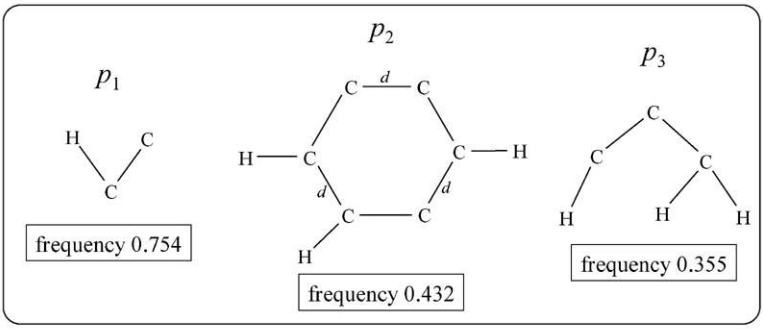
r	$t = 0.5$		$t = 0.4$		$t = 0.3$		$t = 0.2$	
	$\lvert F^t(r)\rvert$	$\lvert M^t(r)\rvert$	$\lvert F^t(r)\rvert$	$\lvert M^t(r)\rvert$	$\lvert F^t(r)\rvert$	$\lvert M^t(r)\rvert$	$\lvert F^t(r)\rvert$	$\lvert M^t(r)\rvert$
0	12	0	27	0	48	1	122	2
1	55	3	145	12	339	15	1104	35
2	126	10	367	39	1115	96	4552	173
3	194	28	622	56	2321	159	12052	562
4	200	16	780	77	3569	314	23873	1251
5	147	12	731	52	4200	318	37266	2189
6	78	3	533	35	3889	286	47046	2684
7	28	1	284	6	2841	158	48743	2690
8	6	0	107	4	1621	78	41697	1969
9	0	0	28	0	730	21	29526	1247
10	0	0	5	0	235	1	17227	649
11	0	0	0	0	56	0	8240	171
12	0	0	0	0	7	0	3130	69
13	0	0	0	0	1	0	921	9
14	0	0	0	0	0	0	185	3
15	0	0	0	0	0	0	25	0
16	0	0	0	0	0	0	1	0

are given in Table 6. For any frequency threshold, the number of frequent bpo-graph patterns with no bridge variable is relatively smaller than the total number of frequent bpo-graph patterns. Our method succeeds in generating many frequent bpo-graph patterns in which some frequent subgraphs are connected to one another with bridge variables.

For frequency 0.2, we counted pairs of bpo-graph patterns with no bridge variable which appear in the same bpo-graph pattern with one bridge variable, and found out 644 distinct pairs. Since there are 1,104 bpo-graph patterns with one bridge variable whose frequencies are over 0.2, it means that there exists $1104/644 \fallingdotseq 1.7$ bpo-graph patterns with one bridge variable with respect to one pair of bpo-graph patterns with no bridge variable. Figure 17 shows some of discovered bpo-graph patterns whose frequencies are over 0.2. The upper in Fig. 17 are bpo-graph patterns with no bridge variable, and the lower are bpo-graph patterns with one bridge variable which are obtained from the upper bpo-graph patterns by connecting with a bridge variable. By using graph pattern with internal variables, our method successfully generates graph patterns representing more characteristic common structures compared to subgraph mining.

6 Conclusion and future work

In this paper, from the viewpoint of computational and algorithmic learning theory, we have considered a graph mining problem of extracting structural features from graph data. Firstly, we have defined a block preserving graph pattern, called a bp-graph pattern, as a new graph



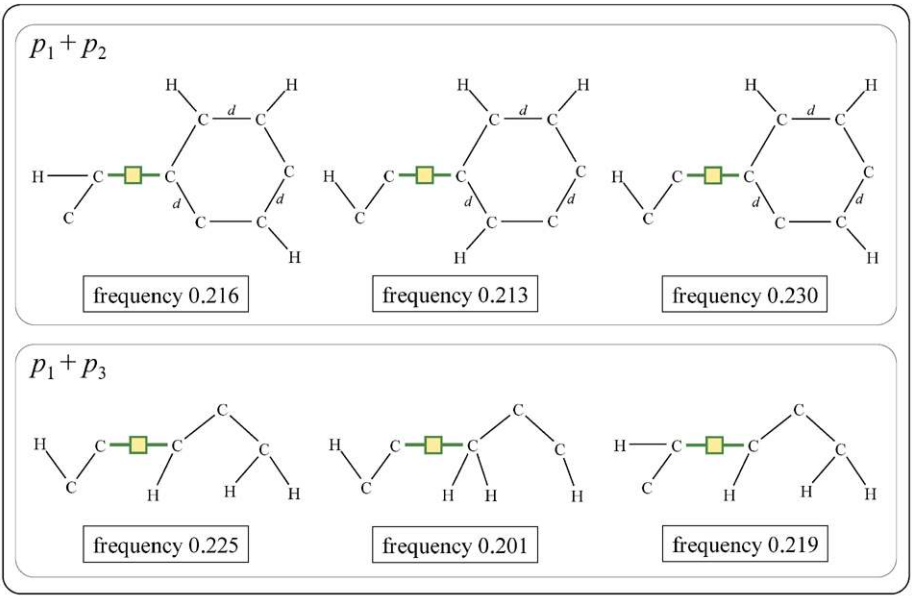


Fig. 17 Examples of discovered bpo-graph patterns in the experiment on the dataset containing 10,000 compounds and frequency 0.2. In the figure, edges with label d mean double bonds between atoms, and edges with no label mean single bonds

pattern having structured variables. Secondly, we have presented a polynomial time algorithm for solving a pattern matching problem of deciding whether or not a given bp-graph pattern matches a given connected graph in a special graph class $\mathcal{C}$. $\mathcal{C}$ is defined as the set of graphs that satisfies a connected hereditary property, contains infinitely many different biconnected graphs, and for which a special kind of the graph isomorphism problem can be computed in polynomial time. Thirdly, by giving refinement operators over bp-graph patterns, we have presented a polynomial time algorithm for finding a minimally generalized bp-graph pattern explaining a given set of connected graphs in $\mathcal{C}$. Therefore, we have shown that the class of graph languages generated by bp-graph patterns for $\mathcal{C}$ is polynomial time inductively inferable from positive data. The set of connected outerplanar graphs $\mathcal{O}$ satis-

fies the above conditions for $\mathcal{C}$. Based on these results, we have proposed two incremental polynomial time algorithms for enumerating all frequent bp-graph patterns with respect to a given finite set of graphs in $\mathcal{O}$. One is an Apriori-like algorithm GENPATTERNS-1.0 and the other is a refinement-based algorithm GENPATTERNS-2.0. Finally, in order to evaluate the performance of the two algorithms, we have reported experimental results obtained by applying the two graph mining algorithms to a subset of the NCI dataset (NCI 2000), almost of whose elements are expressed by graphs in $\mathcal{O}$.

A bpo-graph pattern represents an expressive graph structure among blocks. However, it cannot represent internal structures common to different blocks if the blocks are not isomorphic. Hence, we are also considering extensions of representational power of bp-graph patterns. In (Yamasaki and Shoudai 2008), we introduced a more expressive outerplanar graph pattern than bpo-graph patterns, called an externally extensible outerplanar graph pattern (eeo-graph pattern for short). Unlike bpo-graph patterns, eeo-graph patterns have variables which are contained in blocks. We have proposed a polynomial time matching algorithm for eeo-graph patterns (Yamasaki and Shoudai 2008). An eeo-graph pattern is one of extensions of bp-graph patterns, but eeo-graph patterns are applied only to outerplanar graphs. Based on the concept of term graph patterns (Uchida et al. 1995), we will study more expressive graph patterns.

The pattern matching problem for a tree-structured pattern which has variables with more than 3 ports is NP-complete (Miyahara et al. 2000). However, it is open whether or not there is a polynomial time matching algorithm for the class of bpo-graph patterns all of whose variables have at most 3 ports. In the framework of exact learning model, Okada et al. (2007) showed polynomial time learnabilities of finite unions of some kinds of term graph patterns. We will consider polynomial time learnabilities of finite unions of bp-graph patterns for $\mathcal{C}$ using queries.

As other future works, we will study practical applications of graph mining algorithms proposed in this paper. In the field of frequent itemset mining, several efficient methods for document clustering using frequent itemsets are proposed (Fung et al. 2003; Kryszkiewicz and Skonieczny 2006). In the field of graph classification, several efficient classifiers using kernel methods or boosting methods are proposed (Kashima and Koyanagi 2002; Kuboyama et al. 2006; Kudo et al. 2004). Using the learning algorithms, we will propose efficient graph mining techniques of clustering graph data.

References

Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules. In *Proceedings of the 20th VLDB conference* (pp. 487–499).

Aho, A. V., Hopcroft, J. D., & Ullman, J. D. (1974). *The design and analysis of computer algorithms*. Reading: Addison-Wesley.

Angluin, D. (1980a). Finding patterns common to a set of strings. *Journal of Computer and System Science*, *21*, 46–62.

Angluin, D. (1980b). Inductive inference of formal languages from positive data. *Information and Control*, *45*, 117–135.

Arimura, H., Sakamoto, H., & Arikawa, S. (2001). Efficient learning of semi-structured data from queries. In *LNCS(LNAI): Vol. 2225. Proceedings of the 12th workshop on algorithmic learning theory* (pp. 315–331). Berlin: Springer.

Asai, T., Abe, K., Kawasoe, S., Arimura, H., Sakamoto, H., & Arikawa, S. (2002). Efficient substructure discovery from large semi-structured data. In *Proceedings of the second SIAM international conference on data mining (SDM-2002)* (pp. 158–174).

Asai, T., Arimura, H., Uno, T., & Nakano, S. (2003). Discovering frequent substructures in large unordered trees. In *LNCS(LNAI): Vol. 2843. Discovery science (DS-2003)* (pp. 47–61). Berlin: Springer.

Cook, D. J., & Holder, L. (1994). Substructure discovery using minimum description length and background knowledge. *Journal of Artificial Intelligence Research*, *1*, 231–255.

Cook, D. J., & Holder, L. (2007). *Mining graph data*. New York: Wiley-Interscience.

Dinic, E. A. (1970). Algorithm for solution of a problem of maximum flow in a network with power estimation. *Soviet Mathematics Doklady*, *11*, 1277–1280.

Fung, B. C. M., Wang, K., & Ester, M. (2003). Hierarchical document clustering using frequent itemsets. In *Proceedings of the 3rd SIAM international conference on data mining (SDM-2003)* (pp. 59–70).

Han, J., & Kamber, M. (2001). *Data mining: concepts and techniques*. San Mateo: Morgan Kaufmann.

Han, J., Pei, J., Yin, Y., & Mao, R. (2004). Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Mining and Knowledge Discovery*, *8*(1), 53–87.

Hopcroft, J., & Karp, R. (1973). An $n^{5/2}$ algorithm for maximum matching in bipartite graphs. *SIAM Journal on Computing*, *2*, 225–231.

Hopcroft, J. E., & Wong, J. K. (1974). Linear time algorithm for isomorphism of planar graphs. In *Proceedings of the 6th annual ACM symposium on theory of computing* (pp. 172–184).

Horváth, T., Roman, J., & Wrobel, S. (2006). Frequent subgraph mining in outerplanar graphs. In *Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 197–206).

Inokuchi, A., Washio, T., & Motoda, H. (2000). An apriori-based algorithm for mining frequent substructures from graph data. In *LNCS: Vol. 1910. Proceedings of the 4th European conference on principles of data mining and knowledge discovery (PKDD-2000)* (pp. 12–23). Berlin: Springer.

NCI (2000). The NCI Open Database, Release 2, August 2000 2D file. National Cancer Institute. http://cactus.nci.nih.gov/ncidb2/download.html. Accessed 1 November 2008.

Kashima, H., & Koyanagi, T. (2002). Kernels for semi-structured data. In *Proceedings of the 19th international conference on machine learning (ICML-2002)* (pp. 291–298).

Kryszkiewicz, M., & Skonieczny, L. (2006). Hierarchical document clustering using frequent closed sets. In *Advances in soft computing*. *Proceedings of the international conference on intelligent information systems 2006: new trends in intelligent information processing and web mining* (pp. 489–498). Berlin: Springer.

Kuboyama, T., Hirata, K., Aoki, K. F., Kashima, H., & Yasuda, H. (2006). A gram distribution kernel applied to glycan classification and motif extraction. In *Proceedings of the 17th international conference on genome informatics (GIW-2006)* (pp. 25–34).

Kudo, T., Maeda, E., & Matsumoto, Y. (2004). An application of boosting to graph classification. In *Proceedings of the 18th annual conference on neural information processing systems (NIPS-2004)*.

Kuramochi, M., & Karypis, G. (2001). Frequent subgraph discovery. In *Proceedings of the 2001 IEEE international conference on data mining* (pp. 313–320).

Lingas, A. (1989). Subgraph isomorphism for biconnected outerplanar graphs in cubic time. *Theoretical Computer Science*, *63*, 295–302.

Miyahara, T., Shoudai, T., Uchida, T., Takahashi, K., & Ueda, H. (2000). Polynomial time matching algorithms for tree-like structured patterns in knowledge discovery. In *LNCS(LNAI): Vol. 1805*. *Proceedings of the 4th Pacific-Asia conference on knowledge discovery and data mining (PAKDD-2000)* (pp. 5–16). Berlin: Springer.

Okada, R., Matsumoto, S., Uchida, T., Suzuki, Y., & Shoudai, T. (2007). Exact learning of finite unions of graph patterns from queries. In *LNCS(LNAI): Vol. 4754*. *Proceedings of the 18th international conference on algorithmic learning theory (ALT-2007)* (pp. 298–312). Berlin: Springer.

Sasaki, Y., Yamasaki, H., Shoudai, T., & Uchida, T. (2008). Mining of frequent block preserving outerplanar graph structured patterns. In *LNCS(LNAI): Vol. 4894*. *Proceedings of the 17th international conference on inductive logic programming (ILP 2007)* (pp. 239–253). Berlin: Springer.

Shinohara, T. (1982). Polynomial time inference of extended regular pattern languages. In *LNCS(LNAI): Vol. 147*. *RIMS symposia on software science and engineering* (pp. 115–127). Berlin: Springer.

Shoudai, T., Uchida, T., & Miyahara, T. (2001). Polynomial time algorithms for finding unordered tree patterns with internal variables. In *LNCS: Vol. 2138*. *Proceedings of the 13th international symposium on fundamentals of computation theory (FCT-2001)* (pp. 335–346). Berlin: Springer.

Suzuki, Y., Shoudai, T., Miyahara, T., & Uchida, T. (2003). A polynomial time matching algorithm of structured ordered tree patterns for data mining from semistructured data. In *LNCS(LNAI): Vol. 2583*. *Proceedings of the 12nd international workshop on inductive logic programming (ILP-2002)* (pp. 270–284). Berlin: Springer.

Suzuki, Y., Shoudai, T., Uchida, T., & Miyahara, T. (2006). Ordered term tree languages which are polynomial time inductively inferable from positive data. *Theoretical Computer Science*, *350*, 63–90.

Takami, R., Suzuki, Y., Uchida, T., & Shoudai, T. (2009). Polynomial time inductive inference of TTSP graph languages from positive data. *IEICE Transactions on Information and Systems*, *E92-D*(2), 181–190.

Uchida, T., Mogawa, T., & Nakamura, Y. (2004). Finding frequent structural features among words in tree-structured documents. In *LNCS(LNAI): Vol. 3056*. *Proceedings of the 8th Pacific-Asia conference on knowledge discovery and data mining (PAKDD-2004)* (pp. 351–360). Berlin: Springer.

Uchida, T., Shoudai, T., & Miyano, S. (1995). Parallel algorithm for refutation tree problem on formal graph systems. *IEICE Transactions on Information and Systems*, *E78-D*(2), 99–112.

Yamasaki, H., & Shoudai, T. (2007). A polynomial time algorithm for finding linear interval graph patterns. In *LNCS: Vol. 4484*. *Proceedings of the 4th international conference of theory and applications of models of computation (TAMC-2007)* (pp. 67–78). Berlin: Springer.

Yamasaki, H., & Shoudai, T. (2008). Mining of frequent externally extensible outerplanar graph patterns. In *Proceedings of the 7th international conference on machine learning and applications (ICMLA'08)* (pp. 871–876). Los Alamitos: IEEE Computer Society.

Yamasaki, H., Sasaki, Y., Shoudai, T., Uchida, T., & Suzuki, Y. (2008). Learning block-preserving outerplanar graph patterns and its application to data mining. In *LNCS(LNAI): Vol. 5194*. *Proceedings of the 18th international conference on inductive logic programming (ILP 2008)* (pp. 330–347). Berlin: Springer.

Yan, X., & Han, J. (2002). gSpan: Graph-based substructure pattern mining. In *Proceedings of the third SIAM international conference on data mining (SDM03)* (pp. 721–724).

Yoshida, K., & Motoda, H. (1995). Clip: concept learning from inference patterns. *Artificial Intelligence*, *75*(1), 63–92.

Zaki, M. J. (2002). Inductive inference by stepwise pair expansion. In *Proceedings of the eighth ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 71–80).