Polynomial Time Pattern Matching Algorithm for Ordered Graph Patterns

Takahiro Hino[1], Yusuke Suzuki[1], Tomoyuki Uchida[1], and Yuko Itokawa[2]

Department of Intelligent Systems, Hiroshima City University, Japan
{hino@ml.info., y-suzuki@, uchida@}hiroshima-cu.ac.jp
and Faculty of Psychological Science, Hiroshima International University, Japan
y-itoka@he.hirokoku-u.ac.jp

Abstract. Ordered graphs, each of whose vertices has a unique order on neighbouring vertices, can represent graph structured data such as Web pages, TEX sources, CAD and MAP. In this paper, in order to design computational machine learning for such data, we propose an ordered graph pattern with structural variables and each of whose vertices has an order on neighbouring vertices and variables. We have also defined an ordered graph language for the ordered graph pattern. We present a polynomial time pattern matching algorithm for solving a membership problem which is to determine whether or not the ordered graph language for a given ordered graph pattern contains a given ordered graph.

1 Introduction

Tree structure data such as Web pages and TEX sources are modelled by trees, called ordered trees, each of which contains internal nodes with ordered children. Graph structure data such as CAD and Map are modelled by planar graphs, called planar maps, that have vertices with ordered neighbours. Jiang and Bunke [1] introduced the class of ordered graphs, in which the vertices incident to a vertex have a unique order.

Suzuki et al. [3] introduced a rooted ordered term tree, which is a rooted ordered tree that has structural variables. Extending the concept of ordered graphs, Kawamoto et al. [2] proposed a planar map pattern to represent common structural features of rooted planar maps in which a single edge in the outer face is directed in the clockwise direction. In this paper, we propose a new ordered graph pattern that has structural variables and each of whose vertices has an order on neighbouring vertices and structural variables. Planar map patterns [2] and rooted ordered term trees [3] are regarded as restricted ordered graph patterns. An ordered graph pattern g and ordered graphs F_1, F_2, G_1 and G_2 are shown in Fig. 1 as examples.

For an ordered graph pattern g, we define an ordered graph pattern language, denoted by $L(g)$, as the set of all ordered graphs generated by replacing all variables of g with arbitrarily ordered graphs. The purpose of our research is to show polynomial time learnability of the class of ordered graph pattern languages. As a preliminary step in this process, in this paper we present a polynomial time

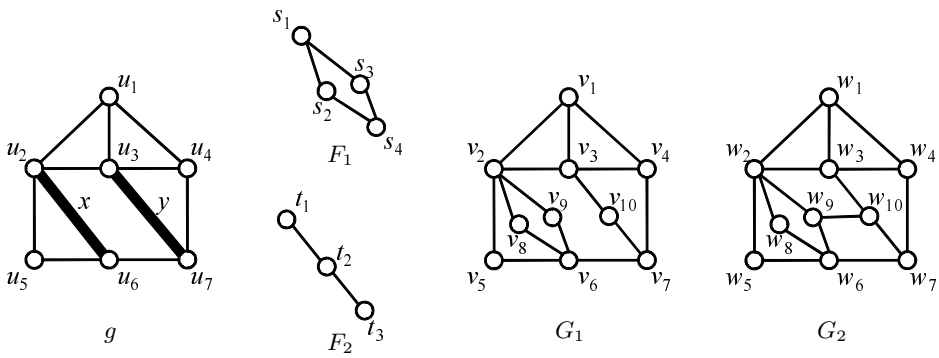

Fig. 1. Ordered graph pattern g and ordered graphs F_1, F_2, G_1 and G_2. Thick lines of g denote variables and an ordering list of each vertex is arranged in the clockwise direction.

pattern matching algorithm for solving a membership problem which is to determine whether or not the ordered graph language $L(g)$ for a given ordered graph pattern g contains a given ordered graph G.

2 Preliminaries

Let Λ and $\mathcal{X}$ be alphabets with $\Lambda \cap \mathcal{X} = \emptyset$. A graph pattern g is a triple (V, E, H), where V is a set of vertices, E is a multiset of edges in $V \times \Lambda \times V$ and H is a set of hyperedges in $V \times \mathcal{X} \times V^*$. Elements in Λ and $\mathcal{X}$ are called *edge labels* and *variable labels*, respectively. A hyperedge in H is called a *variable*. For a variable $h = (u_0, x, u_1, \ldots, u_k) \in H$, each vertex u_i $(0 \leq i \leq k)$ is called a *port* of h. We assume that each variable label x in $\mathcal{X}$ has the *rank*, denoted by $rank(x)$, that is an integer greater than one, and that every variable consisting of k vertices has a variable label whose rank is k. The vertex set, edge set and hyperedge set of g are denoted by $V(g)$, $E(g)$ and $H(g)$, respectively.

An **ordered graph pattern** is a five-tuple $(V, E, H, F, \mathcal{L})$ defined as follows.

(1) (V, E, H) is a graph pattern.
(2) F is a multiset of elements in the set $\{\{u, v\} \mid u \in V, v \in (V \cup H)\}$, such that $F \supseteq \{\{u, v\} \mid (u, a, v) \in E\}$ and the graph pattern $(V, \{\{u, v\} \mid u, v \in V, \{u, v\} \in F\}, \emptyset)$ is connected. An element in F is called an *ordering-edge*.
(3) $\mathcal{L} = \{\ell_F(v) \mid v \in V\}$, called an *ordering-set*, where $\ell_F(v)$, called an *ordering-list* of u on F, is a cyclic list of all elements containing v in F.

An **ordered graph** is an ordered graph pattern without variables. For an ordered graph pattern g, the sets of all ordering-edges and all ordering-lists of g are denoted by $F(g)$ and $\mathcal{L}(g)$, respectively. An ordered graph pattern g is called a **restricted ordered graph pattern** if, for any vertex $v \in V(g)$, the number of variables having v as a port is at most one. The sets of all ordered graph

patterns, all restricted ordered graph patterns and all ordered graphs are denoted by $\mathcal{OGP}$, $\mathcal{ROGP}$ and $\mathcal{OG}$, respectively. We remark that $\mathcal{OG} \subset \mathcal{ROGP} \subset \mathcal{OGP}$ holds. In Fig. 1, we show restricted ordered graph pattern g and ordered graphs F_1, F_2, G_1 and G_2 as examples. The ordered graph pattern g has two variables: (u_2, x, u_6) and (u_3, y, u_7). For the vertex u_2 of g, the ordering-list of u_2 is $\ell_{F(g)}(u_2) = (\{u_2, u_1\}, \{u_2, u_3\}, \{u_2, (u_2, x, u_6)\}, \{u_2, u_5\})$.

For an ordering-list $\ell = (e_1, e_2, \ldots, e_k)$, the integer k is denoted by $|\ell|$. For an ordering-list $\mathcal{L}(g)$, a *size* of $\mathcal{L}(g)$ is defined as $\|\mathcal{L}(g)\| = \sum_{v \in V(g)} |\ell_{F(g)}(v)|$. For an ordered graph pattern g and a subset U of $F(g)$, an *ordering-edge induced subgraph pattern* of g w.r.t. U, denoted by $g\langle U \rangle$, is the graph pattern $f = (V(f), E(f), H(f), U, \mathcal{L}(f))$ such that $V(f) = \{u, v \mid \{u, v\} \in U\}$, $E(f) = \{(u, a, v) \mid \{u, v\} \in U, (u, a, v) \in E(g)\}$, $H(f) = \{h \mid \{u, h\} \in U, h \in H(g)\}$, and $\mathcal{L}(f) = \{\ell_{F(f)}(u) \mid u \in V(f), \ell_{F(f)}(u)$ is the ordering-list obtained by removing all ordering-edges not in the U from $\ell_{F(g)}(u)\}$. We remark that each component in $g\langle U \rangle$ is an ordered graph pattern.

An ordered graph pattern g is said to be *ordering isomorphic* to an ordered graph pattern f, denoted by $g \cong f$, if there exists a bijection $\varphi : V(g) \to V(f)$ satisfying the following three conditions. (1) $(u, a, v) \in E(g)$ if and only if $(\pi(u), a, \pi(v)) \in E(f)$. (2) $(u_0, x, u_1, \ldots, u_k) \in H(g)$ if and only if $(\pi(u_0), x, \pi(u_1), \ldots, \pi(u_k)) \in H(f)$. (3) $\ell_{F(g)}(v) = (v_1, \ldots, v_k) \in \mathcal{L}(g)$ if and only if $\ell_{F(f)}(\varphi(v)) = (\varphi(v_1), \ldots, \varphi(v_k)) \in \mathcal{L}(f)$, where for a variable $(w_1, \ldots, w_t) \in H(g)$, $\varphi((w_1, \ldots, w_t))$ is defined as $(\varphi(w_1), \ldots, \varphi(w_t))$. Such a bijection φ is called an *ordering isomorphism* between g and f.

We can prove the following lemma by modifying an algorithm in [1].

Lemma 1. *For two ordered graph patterns g and f, a problem of determining whether or not g is ordering isomorphic to f is solvable in $O(\|\mathcal{L}(g)\|^2)$ time.*

Let f and g be ordered graph patterns and x a variable label whose rank is r. Let $\sigma = \langle (u_1, p_1^1), (u_2, p_2^1), \ldots, (u_r, p_r^1) \rangle$ be a list of r pairs in $V(g) \times F(g)$ such that $u_1, u_2, \ldots, u_r$ are distinct and for each i $(1 \leq i \leq r)$, $p_i^1 \in \ell_{F(g)}(u_i)$ holds. Then, the form $x := [g, \sigma]$ is called a *binding* for x. A new ordered graph pattern f' is obtained by applying the binding $x := [g, \sigma]$ to f in the following way. If f has no variable labeled with x, then $f' \cong f$. Otherwise, for the variable $h = (v_1, \ldots, v_r)$ labeled with x, we attach a copy g' of g to f by removing the variable h from $H(f)$ and, for each i $(1 \leq i \leq i)$, by identifying the vertex v_i of f with the vertex u'_i of g', where u'_i of g' corresponds to u_i of g. Then, for each i $(1 \leq i \leq r)$, we update the ordering-list of v_i of f by replacing $h \in \ell_{F(f)}(v_i)$ with the list from p_i^1 to $p_i^{k_i}$ in this order, where $k_i = |\ell_{F(g)}(u'_i)|$. A *substitution* θ is a finite collection of bindings $\{x_1 := [g_1, \sigma_1], \cdots, x_n := [g_n, \sigma_n]\}$ such that for any i, j $(1 \leq i < j \leq n)$, the variable labels x_i and x_j are distinct and for each i $(1 \leq i \leq n)$, the ordered graph pattern g_i has no variable labeled with any variable label in $\{x_1, \ldots, x_n\}$. The ordered graph pattern $f\theta$ is obtained by applying all the bindings $x_i := [g_i, \sigma_i]$ to f. For example, the ordered graph G_1 in Fig. 1 is ordering isomorphic to the ordered graph $g\theta$ obtained by applying the substitution $\theta = \{x := [F_1, \langle (s_1, \{s_1, s_3\}), (s_4, \{s_4, s_2\}) \rangle], y := [F_2, \langle (t_1, \{t_1, t_2\}), (t_3, \{t_3, t_2\}) \rangle]\}$ to g in Fig. 1.

For an ordered graph pattern $g \in \mathcal{OGP}$, the **ordered graph pattern language** of g, denoted by $L(g)$, is defined as the set $\{G \in \mathcal{OG} \mid G \cong g\theta$ for some substitution $\theta\}$. For example, for the ordered graph pattern g in Fig. 1, we can see that $L(g)$ contains G_1but does not contain G_2,since the edge (w_9, a, w_{10}) exists in G_2.

3 Polynomial Time Pattern Matching Algorithm for an Ordered Graph Pattern

For an ordered graph pattern g in $\mathcal{ROGP}$ and an ordered graph G in $\mathcal{OG}$, g is said to *match* G if $L(g)$ contains G. The class $\mathcal{ROGPL}$ of restricted ordered graph pattern languages is defined as $\{L(g) \mid g \in \mathcal{ROGP}\}$. A *membership problem* for the class $\mathcal{ROGPL}$ is a problem of determining whether or not a given restricted ordered graph pattern $g \in \mathcal{ROGP}$ matches a given ordered graph $G \in \mathcal{OG}$.

To solve the membership problem for $\mathcal{ROGPL}$, we give the polynomial time algorithm Matching-$\mathcal{ROGP}$ shown in Fig. 2. We present Procedure Coding of Matching-$\mathcal{ROGP}$ in Fig. 3 by modifying polynomial time coding algorithms for ordered graphs [1]. An ordered graph pattern g and an ordered graph G is said to *match on code* if there exists a set $U \subseteq V(G)$ and an ordering-edge $e \in F(G)$ such that for an ordering-edge $d \in F(g)$, Coding$(d, g, V(g))$ is equal to the sequence obtained by replacing '$?^+$' with 'x' in the sequence Coding(e, G, U). For a restricted ordered graph pattern $g \in \mathcal{ROGP}$ and an ordered graph $G \in \mathcal{OG}$, Procedure CodeMatch of Matching-$\mathcal{ROGP}$, which determines whether or not g and G match on code, can be obtained by modifying Procedure Coding. For example, for the ordered graph pattern g in Fig. 1 and the ordering-edge $\{u_1, u_2\}$, Procedure Coding$(\{u_1, u_2\}, g, \emptyset)$ outputs the code

Code(g)="#234#14x5#164#13x2#27#37x#5x6#".

For the ordered graph pattern G_1 in Fig. 1, the ordering-edge $\{v_1, v_2\}$ and the set of vertices $P = \{v_8, v_9, v_{10}\}$ of G_1, Procedure Coding$(\{v_1, v_2\}, G_1, P)$ outputs the code Code(G_1)="#234#14??5#164#13?2#27#37?#5??6#".
If all '$?^+$'s are replaced with 'x' in Code(G_1), then the resultant code of Code(G_1) is equal to Code(g). We can then say that g and G_1 match on code.

Lemma 2. *For a restricted ordered graph pattern g and an ordered graph G, if g matches G, g and G match on code.*

For an ordered graph pattern g and an ordered graph G such that g and G match on code, we say that g and G *match on graph structure* if the following conditions hold. Let W be the set of all ordering-edges used in Coding$(e, g, \emptyset)$ for some ordering-edge $e \in F(g)$. Let $\varphi : V(g) \to V(G)$ be a one-to-one mapping obtained by Coding$(e, g, \emptyset)$.

(1) φ is an ordering isomorphism between the ordered graph $g\langle F_g \rangle$ and the ordered graph $G\langle W \rangle$, where $F_g = F - \{\{u, h\} \mid h \in H(f)\}$.
(2) For each variable $(u_0, x, u_1, \ldots, u_k) \in H(g)$, all vertices $\varphi(u_1), \varphi(u_1), \ldots, \varphi(u_k)$ in $V(G)$ are in the same component of $G\langle F(G) - W \rangle$.

(3) For any two distinct variables $(u_0, x, u_1, \ldots, u_k), (v_0, x, v_1, \ldots, v_k) \in H(g)$ having the same variable label x, the component in $G\langle F(G) - W\rangle$ having $\varphi(u_0), \varphi(u_1), \ldots, \varphi(u_k)$ is ordering isomorphic to the component in $G\langle F(G) - W\rangle$ having $\varphi(v_0), \varphi(v_1), \ldots, \varphi(v_k)$.

For a restricted ordered graph pattern $g \in \mathcal{ROGP}$ and an ordered graph $G \in \mathcal{OG}$ such that g and G match on code, Procedure StructureMatch of Matching-$\mathcal{ROGP}$ determines whether or not g and G match on graph structure.

Lemma 3. *For a restricted ordered graph pattern g and an ordered graph G, if g and G match on code and on graph structure, g matches G.*

From lemmas 1, 2 and 3, the following theorem holds.

Theorem 1. *For a restricted ordered graph pattern g and an ordered graph G, the membership problem for g and G is solvable in polynomial time.*

Proof. (sketch) We can see that the termination of Matching-$\mathcal{ROGP}$ is certainly held. Given a restricted ordered graph pattern $g \in \mathcal{ROGP}$ and an ordered graph $G \in \mathcal{OG}$, Matching-$\mathcal{ROGP}$ correctly determines whether or not $L(g)$ contains G. For a particular ordering-edge e of g, Coding$(e, g, \emptyset)$ generates the code in $O(\|\mathcal{L}(g)\|)$ time. For each of the $O(\|\mathcal{L}(G)\|)$ ordering-edges of G, CodeMatch needs $O(\|\mathcal{L}(g)\|)$ time, and StructureMatch needs $O(|V(g)| \times \|\mathcal{L}(G)\|)$ time. Hence the total time for all executions in the algorithm is $O(|V(g)| \times \|\mathcal{L}(G)\|^2)$.

4 Conclusion and Future Work

We have formally defined a new ordered graph pattern with structural variables and each of whose vertices has an ordered list. We have also defined an ordered graph pattern language $L(g)$ for an ordered graph pattern g. When an ordered graph G and an ordered graph pattern g in which any two variables have no common ports are given, we have proposed a polynomial time matching algorithm to determine whether or not $L(g)$ contains G.

It is easy to prove that the class $\mathcal{ROGPL}$ has finite thickness. We therefore consider a minimal language (MINL) problem for $\mathcal{ROGPL}$ which is, given a subset S of $\mathcal{OG}$ as input, to find a restricted ordered graph pattern g such that $L(g)$ is minimal in $\{L \mid L \in \mathcal{ROGPL}, L \supseteq S\}$. As future work, if we can present a polynomial time algorithm of solving the MINL problem, we can prove that the class of restricted ordered graph pattern languages is polynomial time inductively inferable from positive data. In 2000, we defined a layout term graph that is a graph pattern consisting of variables and graph structures having geometric information [4]. We also proposed a Layout Formal Graph System (LFGS) as a new logic programming system that uses a layout term graph as a term. LFGS directly deals with graphs having geometric information just like first order terms. By extending the concept of an ordered graph pattern and modifying the results of this paper, we aim to design an effective knowledge discovery system using LFGS as knowledge representation.

Algorithm MATCHING-$\mathcal{ROGP}$(g: ordered graph pattern, G:ordered graph);
begin
  Let u be a vertex of g and $\{u, v\}$ in $\ell_g(u)$;
  C :=CODING($\{u, v\}, g, \emptyset$);
  foreach u' in $V(G)$ **do**
    foreach e in $\ell_G(u')$ **do**
      if CODEMATCH(C, e, G) and STRUCTUREMATCH(C, g, G) **then return** *true*;
  return *false*
end.

Fig. 2. Algorithm MATCHING-$\mathcal{ROGP}$.

Procedure CODING($\{u, v\}$:ordering-edge, g:ordered graph pattern, V:set of vertices);
begin
  Initially assign the number '1' to u and the number '0' to all other vertices of g;
  Assign the number '-1' to all vertices in V;
  Enqueue $\{u, v\}$ in a queue Q; Let C := '#' be a code of g; $i := 2$;
  while Q is not empty **do begin**
    Let $\{s, t\}$ be the ordering-edge dequeued from Q;
    foreach $\{s, w\}$ in $\ell_g(s)$ in order of $\ell_g(s)$ starting from t **do begin**
      if w is a variable **then** $C := C \circ$ 'x';
      else if w is assigned by '0' **then begin**
        Assign i to w; $i := i + 1$; $C := C \circ i$; enqueue $\{w, s\}$ in Q **end**;
        else if w is assigned by '-1' **then** $C := C \circ$ '?';
          else $C := C \circ j$; /* j is the number assigned to w */
    end;
    $C := C \circ$ '#'
  end;
  return C
end;

Fig. 3. Procedure CODING. For a code C and a character 'a', $C \circ a$ means the sequence obtained by concatenating 'a' at the end of C.

References

1. X. Jiang and H. Bunke. On the coding of ordered graphs. *Computing*, 61(1):23–38, 1998.
2. S. Kawamoto, Y. Suzuki, and T. Shoudai. Learning characteristic structured patterns in rooted planar maps. In *IMECS 2010*, pages 465–470, 2010.
3. Y. Suzuki, T. Shoudai, T. Uchida, and T. Miyahara. Ordered term tree languages which are polynomial time inductively inferable from positive data. *Theor. Comput. Sci.*, 350(1):63–90, 2006.
4. T. Uchida, Y. Itokawa, T. Shoudai, T. Miyahara, and Y. Nakamura. A new framework for discovering knowledge from two-dimensional structured data using layout formal graph system. In *ALT 2000*, pages 141–155, 2000.