TCS-TR-A-07-26

TCS Technical Report

Time and Space Efficient Discovery of Maximal Geometric Subgraphs

by

Hiroki Arimura, Takeaki Uno, Shinichi Shimozono

Division of Computer Science

Report Series A

May 7, 2007

Hokkaido University
Graduate School of
Information Science and Technology

Email: arimura@ist.hokudai.ac.jp

Phone: +81-011-706-7678
Fax: +81-011-706-7680

Time and Space Efficient Discovery of Maximal Geometric Subgraphs

May 7, 2007

Abstract

A *geometric graph* is a labeled graph whose vertices are points in the 2D plane with isomorphism invariant under geometric transformations such as translation, rotation, and scaling. While Kuramochi and Karypis (ICDM2002) extensively studied the frequent pattern mining problem for geometric subgraphs, the maximal graph mining has not been considered so far. In this paper, we study the maximal (or closed) graph mining problem for the general class of geometric graphs in the 2D plane by extending the framework of Kuramochi and Karypis. Combining techniques of canonical encoding and a depth-first search tree for the class of maximal patterns, we present *a polynomial delay and polynomial space algorithm*, MaxGeo, *that enumerates all maximal subgraphs* in a given input geometric graph without duplicates. This is the first result establishing the output-sensitive complexity of closed graph mining for geometric graphs. We also show that the frequent graph mining problem is also solvable in polynomial delay and polynomial time.

Keywords: geometric graphs, closed graph mining, depth-first search, rightmost expansion, polynomial delay polynomial space enumeration algorithms.

1 Introduction

Backgrounds. By rapid growth of the amount and the varieties of nonstandard datasets in scientific, spatial, and relational domains, there are increasing demands for efficient methods that extracts useful patterns and rules from weakly structured datasets. *Graph mining* is one of the most promising approaches for knowlege discovery from such weakly structured datasets, and the following topics have been extensively studied for the last years: frequent subgraph mining [6, 11, 16, 26], maximal (closed) subgraph mining [3, 8, 19, 24] and combination with machine learning [20, 27]. See surveys, e.g. [7, 23], for the overviews.

The class of geometric graphs. In this paper, we study a graph mining problem for the class $\mathcal{G}$ of geometric graphs. *Geometric graphs* (*geographs*, for short) [14] are a special kind of vertex- and edge labeled graphs whose vertices have the coordinates in the 2D plane $\mathbb{R}^2$, while vertex and edge labels are used for representing geometric features and their relationships of geometric objects. The matching relation for geographs is defined through invariant under a class of geometric transformations, such as translation, rotation, and scaling in the plane in addition to the usual constraint for graph isomorphism. Geographs are useful in applications concering with geometric configulations, e.g., analysis of chemical compounds, geographic information systems, and knowledge discovery from vision and image data.

Maximal pattern discovery problem. For the class of geometric graphs, Kuramochi and Karypis presented an effcient mining algorithm gFSG for frequent geometric subgraph mining, based on Apriori-like breadth-first search [14]. However, frequent pattern mining has a problem that it can easily produce an extremely large number of solutions, which degrade the performance and the comprehensivity of data mining to a large extent. On the other hands, the *maximal subgraph mining problem*[1] is the problem of finding all *maximal patterns* (closed patterns) appearing in a given input geometric graph D, where a *maximal pattern* is any geometric graph isomorphic to a subgraph of D which does not have any properly larger subgraph having the same set of occurrences in the input database. Since the set $\mathcal{M}$ of all maximal patterns is expected to be much smaller than the set $\mathcal{F}$ of all frequent patterns but still contains the complete information of $\mathcal{M}$, maximal subgraph mining has some advantage as a compact representation to frequent subgraph mining.

Difficulties of maximal pattern mining. However, there are a number of difficulties in maximal subgraph mining for geometric graphs. In general, maximal pattern mining has a large computational complexity [4, 25]. Although a number of efficient maximal pattern algorithms are proposed so far for *sets, sequences*, and *graphs* [3, 8, 19, 21, 24], some algorithms use explicit duplicate detection and maximality test by maintaining a collection of already discovered patterns, but this requires large memory and delay time, and makes it difficult to use efficient search techniques, e.g., depth-first search. For these reasons, output-polynomial time computation for the maximal pattern problem is still a challenge in maximal geometric graphs. Besides this, the invariance under geometric transformation for geometric graphs adds another difficulty to geometric graph mining. Thus, a depth-first algorithm is not known so far even for frequent pattern mininig.

[1] Although the maximal pattern discoery is more often called *closed pattern discovery*, we use the term "maximal" rather than "closed" in this paper for the consistency with works in computational complexity and algorithms area [4, 25].

Main result. The goal of this paper is the development of a time and space efficient algorithm that can work well in theory and practice for maximal geometric graphs. As our main result, we present an efficient depth-first search algorithm MaxGeo that, given an input geometric graph, enumerates all frequent maximal pattern P in $\mathcal{M}$ without duplicates in $O(mnk^2) = O(n^5)$ time per pattern and in $O(m) = O(n^2)$ space, where $k = |P|$ is the size of pattern being enumerated, m is the maximum number of its occurrences, and n is the input size. Thus, this is a polynomial delay and polynomial time algorithm for the maximal pattern discovery problem for geometric graphs. This is the first result establishing the output-sensitive complexity of closed graph mining for geometric graphs.

Other contributions of this paper. To cope with the difficulties mentioned above, we devise some new techniques for geometric graph mining.

(1) We define a polynomial time computable *canonical representation* for all geometric graphs in $\mathcal{G}$, which is invariant under geometric transformations. As bi-product, we give the first polynomial delay and polynomial space algorithm FreqGeo for frequent geometric subgraph mining problem.

(2) We introduce the *intersection* and the *closure operations* for $\mathcal{G}$. Using these tools, we define the *tree-shaped search route* $\mathcal{T}$ *for all maximal patterns* in $\mathcal{G}$. Combining the *closure expansion* [17] and the *rightmost expansion* [6, 15, 26], we propose a new pattern growth technique called the *ppc-expansion* (prefix-preserving closure expansion) for traversing the search tree $\mathcal{T}$ by depth-first search.

(3) As a main result, we present a polynomial delay and polynomial space algorithm MaxGeo based on depth-first search over $\mathcal{T}$ for the maximal geometric subgraph mining problem. This is the first output polynomial time algorithm for the problem.

Related works. There are closely related researches on 1D and 2D point set matching algorithms, e.g. [2], where point sets are simplest kind of geometric graphs. However, since they mainly study exact and approximate matching of point sets, but not matching, the purpose is different from this work.

A number of efficient maximal pattern mining algorithms are presented for subclasses of graph, trees, and sequences, e.g.: general graphs [24], ordered and unordered trees [8], attribute trees [3, 19], and sequences [4, 5, 22]. Some of them have output-sensitive time complexity as follows. The first group deal with mining of "elastic" or "flexible" patterns, where the closure is not defined. CMTreeMiner [8], BIDE [22], and MaxFlex [5] are essentially output-polynomial time algorithms for location-based maxmal patterns though it is implicit. They are originally used as pruning for document-based maximal patterns [5].

The second group deal with mining of "rigid" patterns which have *closure*-like operations. LCM [21] proposes ppc-expansion for maximal sets, and then CloATT [3] and MaxMotif [4] generalize it for trees and sequences. They together with this paper are polynomial delay and polynomial space algorithms.

Some of other maximal pattern miners for complex graph classes, e.g.,CloseGraph [24], adopt frequent pattern discovery augmented with,e.g., maximality test and the duplicate detection although it seems difficult to achieve output-polynomial time computability in this approach.

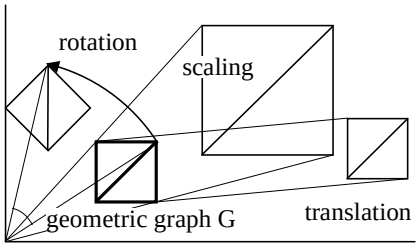

Figure 1: Three basic types of geometric transformarions

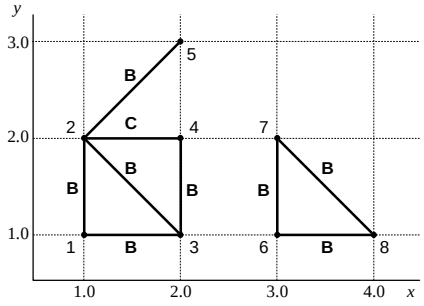

Figure 2: A geometric database D with $V = \{1, \ldots, 9\}$, $\Sigma_{\mathrm{V}} = \emptyset$, and $\Sigma_{\mathrm{E}} = \{\mathsf{B}, \mathsf{C}\}$

Organization of this paper. In Section 2, we introduce maximal pattern mining for geometric graphs. In Section 3, we give the canonical representation and frequent pattern mining. In Section 4, we develop present polynomial delay and polynomial space algorithm MaxGeo for maximal pattern mining, and in Section 5, we conclude.

2 Preliminaries

In this section, we prepare basic definitions and notations for maximal geometric graph mining. We denote by $\mathbb{N}$ and $\mathbb{R}$ the set of all natural numbers and real numbers, resp.

2.1 Geometric transformation and congruence.

We briefly prepare basic of plane geometry [10, 12]. In this paper, we consider geometric objects, such as points, lines, point sets, and polygons, on the *two-dimensional Eucledean space* $\mathbb{E} = \mathbb{R}^2$, also called the *2D plane*. A geometric transformation T is any mapping $T : \mathbb{R}^2 \to \mathbb{R}^2$, which transforms geometric objects into other geometric objects in the 2D plane $\mathbb{R}^2$. In this paper, we consider the class $\mathfrak{T}_{\mathrm{geo}}$ of geometric transformations consisting of three basic types of geometric transformations, *rotation*, and *scaling*, and their combinations. In general, any geometric transformation T can be represented as a *2D affine transformation* $T : \vec{x} \mapsto A\vec{x} + \vec{t}$, where A is a 2×2 nonsingular matrix with $det(A) \neq 0$, and $\vec{t}$ is a 2-vector. Such T is one-to-one. In addition, if $T \in \mathfrak{T}_{\mathrm{geo}}$ then T preserves the angle of two lines and maps an object to a similar object. It is well-known that any affine transformation can be determined by a set of three non-collinear points and their images. For $\mathfrak{T}_{\mathrm{geo}}$, we have the following lemma.

Lemma 1 (determination of unknown transformation) *Given two distinct points in the plane $\vec{x}_1, \vec{x}_2$ and the two corresponding points $\vec{x}'_1, \vec{x}'_2$, there exists a unique geometric transformation T, denoted by $\mathbf{T}(\overline{\vec{x}_1\vec{x}_2}; \overline{\vec{x}'_1\vec{x}'_2})$, such that $T(\vec{x}_i) = \vec{x}'_i$ for every $i = 1, 2$.*

$\mathbf{T}(\overline{\vec{x}_1\vec{x}_2}; \overline{\vec{x}'_1\vec{x}'_2})$ is computable in $O(1)$ time. The above lemma is crucial in the following discussion. For any geometric object O and $T \in \mathfrak{T}_{\mathrm{geo}}$, we denote the image of O via T by $T(O)$. The *inverse image* of O via T is $T^{-1}(O)$.

2.2 Geometric graphs

We introduce the class of geometric graphs according to [14] as follows. Let Σ_{V} and Σ_{E} be mutually disjoint sets of *vertex labels* and *edge labels* associated with total orders $<_\Sigma$. In

what follows, a vertex is always an element of $\mathbb{N}$. A *graph* is a node and edge-labeled graph $G = (V, E, \lambda, \mu)$ with a set of *vertices* V and a set of *edges* $E \subseteq V^2$. Each $x \in V$ has a vertex label $\lambda(x) \in \Sigma_{\mathrm{V}}$, and each $e = xy \in E \subseteq V^2$ represents an unordered edge $\{x, y\}$ with an edge label $\mu(e) \in \Sigma_{\mathrm{E}}$. Two graphs $G_i = (V_i, E_i, \lambda_i, \mu_i)$ $(i = 1, 2)$ are *isomorphic* if they are topologically identical to each other, i.e., $|V_1| = |V_2|$ and there is a bijection $\phi : V_1 \to V_2$ such that for every $xy \in (V_1)^2$, $xy \in E_1$ iff $\phi(x)\phi(y) \in E_2$. The mapping ϕ is called an isomorphism of G_1 and G_2.

A geometric graph is a representation of some geometric object by a set of features and their relationships on a collection of 2D points.

Definition 1 (geometric graph) Formally, a *geometric graph* (or *geograph*, for short) is a structure $G = (V, E, c, \lambda, \mu)$, where (V, E, λ, μ) is an underlying graph and $c : V \to \mathbb{R}^2$ is a one-to-one function called the coordinate function. Each vertex $v \in V$ has the associated coordinate $c(v) \in \mathbb{R}^2$ in the 2D plane as well as its vertex label $\lambda(v)$. We refer to the components V, E, c, λ, μ as $V_G, E_G, c_G, \lambda_G, \mu_G$.

We denote by $\mathcal{G}$ the *class of all geometric graphs* over Σ_{V} and Σ_{E}.

Alternative representation for geograhs. Alternatively, a geometric graph can be simply represented as a collection of labeled object $\underline{G} = \underline{V} \cup \underline{E}$, where $\underline{V} = \{\, \langle v_i, \vec{x}_i, \lambda_i \rangle \mid i = 1, \ldots, n \,\} \subseteq \mathbb{N} \times \mathbb{R}^2 \times \Sigma_{\mathrm{V}}$. and $\underline{E} = \{\, \langle e_i, \mu_i \rangle \mid i = 1, \ldots, m \,\} \subseteq \mathbb{N} \times \mathbb{N} \times \Sigma_{\mathrm{E}}$. Each $\langle v, \vec{x}, \lambda \rangle \in V^\lambda$ is a *labeled vertex* for a vertex v with $c(v) = \vec{x}$ and $\lambda(v) = \lambda$, and each $\langle e, \mu \rangle$ is a *labeled edge* for an edge e with label $\mu(e) = \mu$. A *labeled object* refers to either a labeled vertex or a labeled edge. Let $OL = (\mathbb{N} \times \mathbb{R}^2 \times \Sigma_{\mathrm{V}}) \cup (\mathbb{N} \times \mathbb{N} \times \Sigma_{\mathrm{E}})$ be a domain of labeled object. We assume the lexicographic order $<_O$ L over OL by extending those over $\mathbb{N}, \mathbb{R}^2, \Sigma_{\mathrm{V}}$ and Σ_{E}. Since the correspondence between $\underline{G}$ and G is obvious, we will often use both representations interchangeably. For instance, we may write $G \cup \{\langle v, \vec{x}, \lambda \rangle\}$ or $G - \{\langle e, \mu \rangle\}$. Since c is one-to-one, we may also write $\vec{x} \in G$ instead of $\vec{x} \in c(V_G)$.

2.3 Geometric isomorphism and matching

Now, we extend the notions of isomorphisms and matchings for geographs as in [14]. Let $G_1, G_2 \in \mathcal{G}$ be any geographs. Then, G_1 and G_2 are *geometrically isomorphic*, denoted by $G_1 \equiv G_2$, if there are an isomorphism ϕ of G_1 and G_2 and some transformation $T \in \mathfrak{T}_{\mathrm{geo}}$ such that $T(c(x)) = c(\phi(x))$ for every vertex x of G_1. The pair $\langle \phi, T \rangle$ is a *geometric isomorphism* of G_1 and G_2.

Let $G = (V, E, c, \lambda, \mu)$ be a geograph. A geograph H is a *geometric subgraph* of G, denoted by $H \subseteq G$, if G_H is a substructure of G, that is, (i) $V_H \subseteq V$ and $E_H \subseteq E$ hold, and (ii) mappings λ_H, μ_H, and c_H are the restrictions of λ, μ, and c, respectively, on V_H. Now, we define the matching of geographs in terms of geometric subgraph isomorphism.

Definition 2 (geometric matching) A geograph P *geometrically matches* a geograph G (or, P *matches* G) if there exists some geometric subgraph H of G that is geographically isomorphic to P with a geometric isomorphism $\langle \phi, T \rangle$. Then, we call the geometric transformation T a *geometric matching function* from P to G or an occurrence of P in G.

We denote by $\mathcal{M}(P,G) \subseteq \mathfrak{T}_{\rm geo}$ *the set of all geometric matching functions* from P to G. We omit ϕ from $\langle \phi, T \rangle$ above because if P matches G then, there is at most one vertex $v = \phi(u) \in V_G$ of G such that $c(v) = T(c(u))$ for each $u \in V_P$ of P due to the one-to-one condition of c. Clearly, P matches G iff $\mathcal{M}(P,G) \neq \emptyset$. If P matches G then we write $P \sqsubseteq G$ and say P *occurs in* G or P *appears in* G. If $P \sqsubseteq Q$ and $Q \not\sqsubseteq P$ then we define $P \sqsubset Q$. We can observe that if both of $P \sqsubseteq Q$ and $Q \sqsubseteq P$ hold then $P \equiv Q$, that is, P and Q are geometrically isomorphic. If we take the set $\overline{\mathcal{G}}$ of the equivalence classes of geographs modulo geometric isomorphisms, then $\sqsubseteq$ is a partial order over $\overline{\mathcal{G}}$.

2.4 Patterns, occurences, and frequencies

Let $k \geq 0$ be a nonnegative integer. A *k-pattern* (or *k-geograph*) is any geograph $P \in \mathcal{G}$ with k vertices. From the invariance under $\mathfrak{T}_{\rm geo}$, we assume without any loss of generality that any k-pattern P always has the vertex set $V_P = \{1, \ldots, k\}$ and if $k \geq 2$ then P has the fixed coordinates $c(1) = (0,0)$ and $c(2) = (0,1) \in \mathbb{R}^2$ for its first two vertices in the local Cartesian coordinate. An input *geometric database* or of size $n \geq 0$ is a single geograph $D = (V, E, c, \lambda, \mu) \in \mathcal{G}$ with $|V| = n$. D is also called an *input geograph*. If we receive a collection of small geographs as input, then we merge them into a single graph after appropriately renaming vertices and their coordinates. Fig. 2 shows an example of an input geometric database D with $V = \{1, \ldots, 9\}$ over $\Sigma_{\rm V} = \emptyset$, and $\Sigma_{\rm E} = \{\mathsf{B}, \mathsf{C}\}$.

Let $P \in \mathcal{G}$ be any k-pattern. Then, the *location list* of pattern P in D is defined by the set $L(P)$ of all geometric transformations that matches P to the input geograph D, i.e., $L(P) = \mathcal{M}(P, D)$. The *frequency* of P is $|L(P)| \in \mathbb{N}$. Given an integer $0 \leq \sigma \leq n$, called a *minimum support* (or *minsup*), P is σ-frequent in D if $freq(P) \geq \sigma$.

Unlike ordinary graphs, the number of distinct matching functions in $L(P)$ is bounded by polynomial in the input size.

Lemma 2 *For any geograph P, $|L(P)|$ is bounded from above by n^2 under $\mathfrak{T}_{\rm geo}$.*

Proof: From Lemma 1, the images $\vec{x}'_1\vec{x}'_2$ of just two points $\vec{x}_1\vec{x}_2$ in the plane are sufficient to determine $\mathbf{T}(\vec{x}_1\vec{x}_2;\ \vec{x}'_1\vec{x}'_2)$ in $\mathfrak{T}_{\rm geo}$. Thus, the result follows. □

Lemma 3 (monotonicity) *Let P, Q be any geographs. (i) If $P \equiv Q$ then $L(P) = L(Q)$. (ii) If $P \sqsubseteq Q$ then $L(P) \supseteq L(Q)$. (iii) If $P \sqsubseteq Q$ then $|L(P)| \geq |L(Q)|$.*

2.5 Maximal pattern discovery

From the monotonicity of the location list and the frequency in Lemma 3, it is natural to consider maximal subgraphs in terms of $\sqsubseteq$ preserving their location lists as follows.

Definition 3 (maximal geometric patterns) A geometric pattern $P \in \mathcal{G}$ is said to be *maximal* in an input geograph T if there is no other geometric pattern $Q \in \mathcal{G}$ such that (i) $P \subset Q$ and (ii) $L(P) = L(Q)$ hold.

In other words, P is maximal in D if there is no strictly larger pattern than P that has the same location list as P. Equivalently, P is maximal iff any addition of a labeled node or a labeled edge to P makes $L(P)$ strictly smaller than before. We denote by $\mathcal{F}^\sigma \subseteq \mathcal{G}$ be the set of all σ-frequent geometric patterns in D, and by $\mathcal{M} \subseteq \mathcal{G}$ be the set of all maximal geometric patterns in D under $\mathcal{T}$. The set of all σ-frequent maximal patterns is $\mathcal{M}^\sigma = \mathcal{M} \cap \mathcal{F}^\sigma$.

Now, we state our data mining problem as follows.

Definition 4 (maximal pattern enumeration problem) The *maximal geometric pattern enumeration problem* is, given an input geograph $D \in \mathcal{G}$ of size n and a minimum support $1 \leq \sigma \leq n$, to enumerate all frequent maximal geometric patterns $P \in \mathcal{M}^\sigma$ appearing in D within $\mathcal{G}$ without repetition.

Our goal is to devise a light-weight and high-throughput mining algorithm for enumerating all maximal patterns appearing in a given input geograph. This is paraphrased in terms of output-sensitive enumeration algorithms in Section 2.6 as a polynomial delay and polynomial space algorithm for solving this problem. This goal has been open question for $\mathcal{M}$ and even for $\mathcal{F}^\sigma$ so far.

We can define different notion of a location list $D(P)$, called the document list, defined as the set of input graphs in which a pattern appears, and the maximality based on $D(P)$ in a similar way. Acturally, the location-based maximality is a necessary condition for the document-based maximality. However, we do not go further in this direction.

2.6 Model of computation

We make the following standard assumptions in computational geometry [18]: For every point $p = (x, y) \in \mathbb{E}$, we assume that its coordinates x and y have infinite precision. Our model of computation is the the *random access machine* (RAM) model with $O(1)$ unit time arithmetic operations over real numbers as well as the standard functions of analysis $((\cdot)^{\frac{1}{2}}, \sin, \cos,$ etc) [1, 18].

An enumeration algorithm $\mathcal{A}$ is an *output-polynomial time* algorithm if $\mathcal{A}$ finds all solutions $S \in \mathcal{S}$ without duplicates on a given input I in total polynomial time both in the input size $n = ||I||$ and the number of output $m = |\mathcal{S}|$. $\mathcal{A}$ is a *polynomial delay and polynomial space* algorithm if the *delay*, which is the maximum computation time between two consecutive outputs, and the maximum space of $\mathcal{A}$ are both bounded by polynomials in $n = |D|$ only. If $\mathcal{A}$ is of polynomial delay then $\mathcal{A}$ is also of output-polynomial time.

3 Algorithm for Frequent Pattern Discovery

3.1 Canonical encoding for geographs

In this subsection, we define a canonical representation of each geograph in $\mathcal{G}$, which is an invariant under geometric isomorphism in $\mathcal{T}_{\text{geo}}$. Let P be any k-pattern with $V_P = \{1, \ldots, k\}$. Recall that the first two vertices of P have the fixed coordinates $c(1) = (0, 0), c(2) = (0, 1) \in \mathbb{R}^2$ in their local 2D plane.

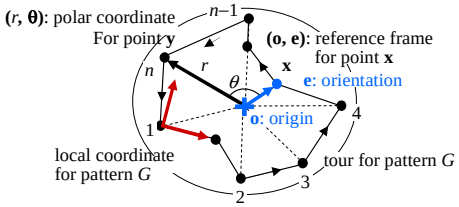

Figure 3: The local frame and reference frame for a pattern

Defining a tour. Let $\vec{o} = \sum_{\vec{x} \in c(P)}$ be the centroid (the *center*) of P. For each *start point* $\vec{x} \in c(P)$, we consider the polar coordinate system for P that has the center $\vec{o}$ and the zero orientation vector $\vec{i} = \vec{x} - \vec{o}$ (See Fig. 3). Then, the *tour* for P w.r.t. $\vec{x}$ is the increasing list $\mathtt{Tour}(P, \vec{x}) = (\vec{x}_1 (= \vec{x}), \ldots, \vec{x}_k) \subseteq \mathbb{R}^2$ of all k points in P in the increasing *angle-distance order* around $\vec{o}$ in the polar coordinate (Fig. 3). That is, $\mathtt{Tour}(P, \vec{x})$ is obtained by sorting the points $\vec{y} \in c(P)$ first in $angle(\vec{y} - \vec{o}, \vec{x} - \vec{o})$ and then in $dist(\vec{y}, \vec{o})$ in $O(k \log k)$ time. Clearly, there are exactly k distinct $\mathtt{Tour}(P, \vec{x})$ depending on the choice of the starting point $\vec{x}$. For $\pi = \mathtt{Tour}(P, \vec{x})$ and each $\vec{y} \in \pi$, let $\mathtt{ord}^\pi(\vec{y})$ is the order of $\vec{y}$ appearing in π, that is, $\mathtt{ord}^\pi(\vec{y}) = i$ if $\vec{y} = \vec{x}_i$.

Alternative representation for geographs. Next, we define a sequential representation for the labeled object representation of geograph P in Sec. 2.2 as follows. For $\pi = \mathtt{Tour}(P, \vec{x})$, we define the set $\mathtt{Lab}^\pi(P) \subseteq OL$ of labeled objects for P as follows:

(i) First, let $\mathtt{Lab}^\pi(P) = \emptyset$.

(ii) For every $x \in V_P$ of P, add $\langle \mathtt{ord}^\pi(x), c(x), \lambda(x) \rangle \in \mathbb{N} \times \mathbb{R}^2 \times \Sigma_{\mathrm{V}}$ into $\mathtt{Lab}$.

(iii) For every $xy \in V_P$, add $\langle \mathtt{ord}^\pi(x')\mathtt{ord}^\pi(y'), c(x')c(y'), \mu(xy) \rangle \in \mathbb{N} \times \mathbb{N} \times \Sigma_{\mathrm{E}}$ into $\mathtt{Lab}$. If xy is directed, then $\mathtt{ord}^\pi(xy) = \mathtt{ord}^\pi(x)\,\mathtt{ord}^\pi(y) \in \mathbb{N}^2$. If xy is undirected, i.e., it represents $\{xy, yx\}$, then $x'y'$ is the lexcographically larger one of xy and $yx \in \mathbb{N}^2$.

Then, the *sequential encoding* for P w.r.t. a tour π, denoted by $\mathtt{e}^\pi(P)$, is the increasing list $\mathtt{e}^\pi(P) = \langle o_1, l_1 \rangle \# \langle o_2, l_2 \rangle \# \cdots \# \langle o_\ell, l_\ell \rangle$ of labeled objects $\langle o_i, l_i \rangle$ separated with a special delimitor $\#$ for some $\ell \geq 0$. This list is obtained in $O(k^2)$ time by sorting the elements of $\mathtt{Lab}^\pi(P)$ in the increasing order w.r.t. the total order $<_{OL}$ over labeled objects.

3.1.1 Canonical Encoding. Now, we define the *canonical encoding for* P, denoted $\mathtt{cano\_e}(P)$, by the lexcographically first sequential encoding of P one among all encodings of P for possible choices of starting point $\vec{x} \in c(P)$:

$$\mathtt{cano\_e}(P) = \min_{<_{\mathrm{lex}, OL}} \{\, \mathtt{e}^\pi(P) \mid \pi = \mathtt{Tour}(P, \vec{x}) \text{ for some } \vec{x} \in c(P) \,\}.$$

Theorem 4 (characterization of canonical encoding) *For any $P, Q \in \mathcal{G}$ of size $k \geq 0$, $\mathtt{cano\_e}(P) = \mathtt{cano\_e}(Q)$ iff $P \equiv Q$ under $\mathcal{T}_{\mathrm{geo}}$.*

We can compute in $O(k^2 \log k)$ time.

The main purpose of $\mathtt{cano\_e}(P)$ is for assigning

By using $\mathtt{cano\_e}(P)$, we define the total order $<_{\mathtt{cano\_e}(P)}$ over the components of $\underline{P}$ as follows: For any labeled objects $\xi, \xi' \in \underline{P}$, we define $\xi <_{\mathtt{cano\_e}(P)} \xi'$ iff ξ appears at an earlier position than ξ' in $\mathtt{cano\_e}(P)$. The *tail* of $\underline{P}$ is defined by $\mathtt{tail}(P) = \max_{<_{\mathtt{cano\_e}(P)}} \underline{P}$, i.e., the last element of the canonical encoding $\mathtt{cano\_e}(P)$ as a sequence.

Elimination Ordering:

1: Given an labeled object representation of geograph $\underline{P}$;
2: $i = 1$; $j = 1$; $P_1 = \underline{P}$;
3: **while** $P_i \neq \emptyset$ **do**
4: $\quad \langle o, l \rangle = \texttt{tail}(P_i)$ based on the canonical encoding $\texttt{cano\_e}(P_i)$;
5: $\quad$ **if** $L(P_i - \{\langle o, l \rangle\}) \neq L(P_i)$ **then** $\chi_i = \langle o, l \rangle$ and $i = i + 1$;
6: $\quad P_{i+1} = P_i - \{\langle o, l \rangle\}$; $\xi_j = \langle o, l \rangle$ and $j = j + 1$;
7: **end while**
8: **return** $\texttt{elimseq}(P) = (\xi_k, \ldots, \xi_1)$ and $\texttt{critseq}(P) = (\chi_j, \ldots, \chi_1)$;

Figure 4: The procedure for computing the perfect elimination and the perfect critical elimination sequences $\texttt{elimseq}(P)$ and $\texttt{critseq}(P)$ for a geometric graph P

3.2 Perfect elimination sequences

Before studying the enumeration or generation of each pattern, we consider the reverse process of the enumeration, the decomposition of a given geograph. Let $P \in \mathcal{G}$ be any k-geograph. We define the sequences $\texttt{elimseq}(P) = (\xi_k, \ldots, \xi_1) \in OL^*$ and $\texttt{critseq}(P) = (\chi_j, \ldots, \chi_1) = (\xi_{\sigma(j)}, \ldots, \xi_{\sigma(1)}) \in OL^*$ by the procedure *Elimination Ordering* in Fig. 4. They are called the *perfect elimination sequence* and the *perfect critical elimination sequence*, resp. Note that the elimination sequence $(\xi_k, \ldots, \xi_1)$ for P is not the mere reverse of $\texttt{cano\_e}(P)$ since the i-th element ξ_i is selected based on the canonical form of the current geograph P_i not on that of the initial graph $P = P_k$.

Lemma 5 *For any $P, Q \in \mathcal{G}$, $P \equiv Q$ under $\mathcal{T}_{\text{geo}}$ iff $\texttt{elimseq}(P) = \texttt{elimseq}(Q)$.*

In the next Sec. 3.3, we study the application of $\texttt{elimseq}(P)$ to frequent geograph mining . In Sec. 4.3, we consider $\texttt{critseq}(P)$ for maximal pattern enumeration.

3.3 Algorithm for Frequent Pattern Discovery

In Fig. 5, we show an algorithm FreqGeo for frequent geometric subgraph discovery. Starting from the emptygraph $\emptyset$, FreqGeo searches $\mathcal{F}^\sigma$ from smaller to larger by growing P with adding new labeled object ξ_i one by one in the reverse order of the elimination sequence for P. Since $\{\xi_i\}_i = \underline{P}$ holds, we know that at least FreqGeo is complete for $\mathcal{F}^\sigma$. The key of the algorithm is the test for the elimination ordering at Line 4 of Fig. 5. From Lemma 5, we know that distinct elimination sequence results distinct patterns.

There are infinitely many candidates for possible labeled object at Line 2 of Fig. 5. From the next lemma, we can avoid such a blind search by focusing only on *missing labeled objects for P*, which is either labeled vertex or edge ξ such that $L(P) \supseteq L(P \cup \{\xi\}) \neq \emptyset$ holds. From Lemma 1 and Lemma 3, we have the next lemma.

Lemma 6 (missing labeled objects) *Let P be a pattern with nonempty $L(P)$ in D. Any missing object $\xi = \langle o, l \rangle$ for P is the inverse images of some labeled vertex or labeled edge π via T for some matching $T \in L(P)$, that is, $\xi = T^{-1}(\pi)$ for some $\pi \in \underline{D}$.*

FreqGeo(σ : minsup, D : input database)

1: **call** Expand_FG($\emptyset, \sigma, D$);

Expand_FG(P, σ, D)

1: **if** $|L(P)| < \sigma$ **then return else output** P as a frequent subgraph;
2: **for** each possible labeled object $\langle o, l \rangle$ **do**
3: $\quad Q = P \cup \{\langle o, l \rangle\}$;
4: $\quad$ **if** $\langle o, l \rangle \neq \texttt{tail}(Q)$ **then return**;
5: $\quad$ **call** Expand_FG(Q, σ, D);
6: **end for**

Figure 5: A polynomial delay and polynomial space algorithm for the frequent geometric subgraph enumeration problem

From Lemma 6 above, we know that there are at most $O(|L(P)| \cdot ||D||) = O(n^4)$ missing objects. Thus, Line 2 can be done in polynomial time. Combining the above discussion, we have the following theorem.

Theorem 7 (frequent geograph enumeration) *The algorithm* FreqGeo *in Fig. 5 enumerates all σ-frequent geometric subgraphs in a given input database $D \in \mathcal{G}$ in polynomial delay and polynomial space in the total input size $n = ||D||$.*

4 Algorithm for Maximal Pattern Discovery

In this section, we present an efficient algorithm MaxGeo for the maximal pattern enumeration problem for the class of geographs that runs in polynomial delay and polynomial space in the input size.

4.1 Outline of the algorithm

In Fig. 6, we show our algorithm MaxGeo for enumerating all σ-frequent maximal geometric patterns in $\mathcal{M}^\sigma$ using backtracking. The key of the algorithm is a tree-like search route $\mathcal{T} = \mathcal{T}(\mathcal{M}^\sigma)$ for $\mathcal{M}^\sigma$ implicitly defined over $\mathcal{M}^\sigma$. Then, Starting at the root of the search tree $\mathcal{T}$, the algorithm MaxGeo searches $\mathcal{T}$ jumping from a smaller maximal pattern to a larger one in the depth-first manner. Each jump is performed by expanding each maximal pattern in polynomial time using a procedure called the *ppc-expansion* defined in the following subsections. To properly handle the geometric isomorphism among the isomorphic patterns, we introduce the canonical encoding for geometric patterns.

4.2 Intersection and closure operations for geographs

Let G_1 and G_2 be two geographs with $V_{G_1} \cap V_{G_2} \neq \emptyset$. The *maximally common geometric subgraph* (MCGS) of G_1 and G_2 is a maximal geograph $G \in \mathcal{G}$ w.r.t. $\subseteq$ such that $G \subseteq G_i$ for every $i = 1, 2$. The next lemma states that MCGS is unique for geographs, while they are not unique for ordinary graphs.

Lemma 8 (intersection geometric subgraph $G_1 \cap G_2$) *There exists the unique maximally common geometric subgraph of two geographs G_1 and G_2. It is given by the* intersection geograph $G_1 \cap G_2 = (V_{12}, E_{12}, \lambda_{12}, \mu_{12}, c_{12}) \in \mathcal{G}$ *defined below:*

Algorithm MaxGeo: (D : input geograph, σ : *minsup*)
task: finding all σ-frequent maximal patterns of $\mathcal{M}^\sigma$ in D.

1: $\bot = \mathrm{Clo}(\emptyset)$; *//The bottom maximal geograph*
2: Call the recursive procedure Expand_MaxGeo($\bot, \mathbf{0}, \sigma, D$);

Algorithm Expand_MaxGeo(P, π, σ, D)

1: **if** P is not σ-frequent **then return**; *//Frequency test*
2: **else output** P as a σ-frequent maximal geograph;
3: **for** each missing labeled object $\xi = \langle o, \ell \rangle$ **do** *//Lemma 6*
4: $Q = \mathrm{Clo}(P \cup \{ \xi \})$; *//Cond.(i) of Def.7*
5: **if** ($\xi \leq_{\mathtt{cano\_e}(Q)} \pi$) **then return**; *//Cond.(ii) of Def.7*
6: **if** ($(\exists \tau \in Q - P)\ \tau <_{\mathtt{cano\_e}(Q)} \xi$) **then return**; *//Cond.(iii) of Def.7*
7: **call** Expand_MaxGeo(Q, ξ, σ, D); *//Recursive call for children*
8: **endfor**

Figure 6: A polynomial delay and polynomial space algorithm MaxGeo for the maximal geometric subgraph enumeration problem

- $V_{12} = \{1, \ldots, k\}$ *for some* $k \geq 0$.
- *For every* $\vec{x} \in \mathbb{R}^2$, $\vec{x} \in V_{12}$ *iff* $\vec{x} \in c(V_1) \cap c(V_2)$ *and* $\lambda_1(\vec{x}) = \lambda_2(\vec{x})$.
- *For every* $\vec{x}\vec{y} \in \mathbb{R}^2 \times \mathbb{R}^2$, $\vec{x} \in E_{12}$ *iff* $c(E_1) \cap c(E_2)$ *and* $\mu_1(\vec{x}\vec{y}) = \mu_2(\vec{x}\vec{y})$.

where $P_i = (V_i, E_i, \lambda_i, \mu_i, c_i)$ *for* $i = 1, 2$. *Furthermore,* $G_1 \cap G_2$ *is computable in* $O(||G_1|| + ||G_2||)$ *time, where* $||G_i|| = |V_i| + |E_i|$.

Proof: By the construction, V_{12} and E_{12} are the largest sets satisfying the above conditions. It follows from this that that $G_1 \cap G_2$ is the unique MCGS for both G_1 and G_2. □

The intersection operation $\cap$ is reflexive, commutative, and associative over $\mathcal{G}$. For a set $\mathbf{G} = \{G_1, \ldots, G_m\}$ of geographs, we define $\cap\mathbf{G} = G_1 \cap G_2 \cap \cdots \cap G_m$. We can see that the size and the computation time of $\cap\mathbf{G}$ are bounded by $O(||\mathbf{G}||)$. Some literatures [13] give the intersection of labeled graphs or first-order models based on the *cross product* of two structures. However, their iterative applications causes exponentially large intersection unlike $\cap\mathbf{G}$ above. Gariiga *et al.*[9] discuss related issues.

Now, we define the closure operation for $\mathcal{G}$.

Definition 5 (closure operator for geographs) Let $P \in \mathcal{G}$ be geograph of size ≥ 2. Then, the *closure* of P in D is define by the geograph $\mathrm{Clo}(P)$:

$$\mathrm{Clo}(P) = \bigcap \{\, T^{-1}(D) \mid T \in L(P) \,\}.$$

Theorem 9 (correctness of the closure operation) *Let* P *be a geograph of size* ≥ 2 *and* D *be an input database. Then,* $\mathrm{Clo}(P)$ *is the unique, maximal geograph w.r.t.* $\sqsubseteq$ *satifying* $L(\mathrm{Clo}(P)) = L(P)$.

Proof: We give a sketch of the proof. Let $T \in \mathcal{T}_{\mathrm{geo}}$ be any geometric tansformation. Then, we can see that P matches D via T iff P is a direct geometric subgraph (substructure) of the inverse image of D via T, i.e., $P \subseteq T^{-1}(D)$. Thus, taking the intersection of the inverse image $T^{-1}(D)$ for all matching T of P, we obtain the unique maximal subgraph having $L(P)$. □

Lemma 10 *For any geographs $P, Q \in \mathcal{G}$, the following properties hold:*

(i) $P \subseteq \mathrm{Clo}(P)$. *(ii)* $L(\mathrm{Clo}(P)) = L(P)$. *(iii)* $\mathrm{Clo}(P) = \mathrm{Clo}(\mathrm{Clo}(P))$.

(iv) $P \subseteq Q$ *iff* $L(P) \supseteq L(Q)$ *for any maximal* $P, Q \in \mathcal{M}$.

(v) $\mathrm{Clo}(P)$ *is the unique, smallest maximal point set containing* P.

(vi) For the empty graph $\emptyset$, $\bot = \mathrm{Clo}(\emptyset)$ *is the smallest element of* $\mathcal{M}$.

Theorem 11 (characterization of maximal geographs) *Let D be an input geograph and $P \in \mathcal{G}$ be any geograph. Then, P is maximal in D iff* $\mathrm{Clo}(P) = P$.

4.3 Defining the tree-shaped search route

In this subsection, we define a tree-like search route $\mathcal{T} = (\mathcal{M}^\sigma, \mathcal{P}, \bot)$ for the depth-first search of all maximal geographs based on a so-called parent function. By inspecting the procedure for elimination sequences in Fig. 4 carefully, we obtain the following definitions.

Let $Q \in \mathcal{M}$ be a maximal pattern such that $Q \neq \bot$. Suppose we have the canonical encoding $\mathtt{cano\_e}(Q)$ and the associated total ordering $<_{\mathtt{cano\_e}(Q)}$ over the elements of $\underline{Q}$. For any element $\xi \in \underline{Q}$, define the ξ-prefix of Q as the pattern $Q[\xi]$ with $\underline{Q[\xi]} = \{\, \pi \in \underline{Q} \mid \pi \leq_{\mathtt{cano\_e}(Q)} \xi \,\}$. Then, the *core index* of Q is

$$\mathtt{core\_i}(Q) = \max_{\leq \mathtt{cano\_e}(Q)} \{\; \xi \in \underline{Q} \mid L(Q[\xi]) = L(Q) \;\}.$$

We can show that if $Q \neq \bot$ then $\mathtt{core\_i}(Q)$ is always defined. We assume that $\mathtt{core\_i}(\bot) = \mathbf{0}$ for a special labeled object $\mathbf{0}$ such that $\mathbf{0} < \xi$ for any $\xi \in OL$.

$Q[\mathtt{core\_i}(Q)] \subseteq Q$ is a shortest prefix of Q satisfying $L(Q[\xi]) = L(Q)$. Moreover, if we remove $\mathtt{core\_i}(Q)$ from the prefix $Q[\mathtt{core\_i}(Q)]$, then we have the properly shorter prefix, and then the location list changed. Now, we define the parent function $\mathcal{P}$ that gives the predecessor of Q in terms of its perfect critical elimination sequence.

Definition 6 (parent function $\mathcal{P}$) The *parent* of any maximal pattern $Q \in \mathcal{M}$ ($Q \neq \bot$) is defined by $\mathcal{P}(Q) = \mathrm{Clo}(Q[\xi] - \{\xi\})$, where $\xi = \mathtt{core\_i}(Q)$ is the core index of Q.

Lemma 12 *$\mathcal{P}(Q)$ is (i) always defined, (ii) unique, (iii) is a maximal pattern in $\mathcal{M}$. Moreover, $\mathcal{P}$ satisfies that (iv)* $\underline{\mathcal{P}(Q)} \subset \underline{Q}$, *(v)* $|\,\underline{\mathcal{P}(Q)}\,| < |\,\underline{Q}\,|$, *and (vi)* $L(\mathcal{P}(Q)) \supset L(Q)$.

The parent $\mathcal{P}(Q)$ corresponds to the maximum critical object in $\underline{Q}$ in $\mathtt{critseq}(Q)$. Now, we define the *search graph* for $\mathcal{M}^\sigma$ as a rooted directed graph $\mathcal{T}(\mathcal{M}^\sigma) = (\mathcal{M}^\sigma, \mathcal{P}, \bot)$, where $\mathcal{M}^\sigma$ is the vertex set, $\mathcal{P}$ is the set of reverse edges, and $\bot$ is the root. Then, the unique path from each maximal pattern Q to $\bot$ spells out $\mathtt{critseq}(Q)$.

Theorem 13 (reverse search property) *For every σ, the search graph $\mathcal{T}(\mathcal{M}^\sigma)$ is a spanning tree with the root $\bot$ over all the maximal patterns in $\mathcal{M}^\sigma$.*

4.4 A polynomial space polynomial delay algorithm

The remaining thing is to show how we can efficiently traverse the search tree $\mathcal{T}(\mathcal{M}^\sigma)$ starting from the $\perp$. However, this is not a straightforward task since $\mathcal{T}(\mathcal{M}^\sigma)$ only has the reverse edges. To cope with this difficulty, we introduce the notion of the prefix-preserving closure expansion by combining the rightmost expansion [6, 15, 24, 26] and the closure extension [17]. For discussions on closure expansion, see Sec. B of appendix.

Prefix-preserving closure extension for geographs. For a maximal pattern $Q \neq \perp$ and its predecessor $\mathcal{P}(Q)$, we can show from the definition of $\mathcal{P}$ and Lemma 10 that if $\xi = \texttt{core\_i}(Q)$ then $\pi \in Q[\xi]$ iff $\pi \in Q$ for all $\pi \in \underline{Q}$ such that $\pi <_{\texttt{core\_i}(Q)} \xi$, that is ξ-prefix other than ξ itself is preserved by the application of $\mathcal{P}$.

Definition 7 (ppc-expansion) Let P and Q be any patterns such that $Q \neq \perp$. Suppose that P is maximal in D. Q is a *prefix-preserving closure expansion* (ppc-expansion) of P if Q satisfies the following conditions (i) – (iii):

(i) $Q = \mathrm{Clo}(P \cup \{\xi\})$ for some labeled object $\xi = \langle o, l \rangle \in OL$, called a *key*.

(ii) The key ξ satisfies: $\texttt{core\_i}(P) <_{\texttt{cano\_e}(Q)} \xi$.

(iii) There is no element $\tau \in Q - P$ such that: $\tau <_{\texttt{cano\_e}(Q)} \xi$.

Theorem 14 (correctness of ppc-expansion) *Let P and Q be any maximal patterns such that $Q \neq \perp$. $P = \mathcal{P}(Q)$ holds iff $Q = \mathrm{Clo}(P \cup \{\xi\})$ is a ppc-expansion of P with some labeled object $\xi \in OL$. In this case, $\xi = \texttt{core\_i}(Q)$ always holds. Furthermore, if two ppc-extension have mutually distinct keys then the resulting ppc-extensions are also mutually distinct.*

The correctness and the time and space complexity. Based on Theorem 14, we can show the correctness and the time and space complexity of our algorithm MaxGeo in Fig. 6 based on the ppc-extension. Now, we show the main theorem of this paper.

Theorem 15 (correctness and complexity of MaxGeo) *Given a minimum support parameter $1 \leq \sigma \leq n$ and an input geograph D of length n, the algorithm* MaxGeo *in Fig. 6 enumerates all σ-frequent maximal geograph P of $\mathcal{M}^\sigma$ in $O(mnk^2) = O(n^5)$ delay per maximal geograph and using $O(km)$ space, where k is the maximum size of patterns and $m = O(n^2)$ is the maxmum size of the location lists.*

Overall, the algorithm MaxGeo runs in $O(n^5)$ time in the size of input geograph n. If the maximum size k of maximal geographs is bounded by a (small) constant, the algorithm runs in $O(mn)$ per pattern P and will be fast when $m = |L(P)|$ is not large.

Corollary 16 *The maximal geograph enumeration problem is solvable in polynomial delay and polynomial space.*

5 Conclusion

In this paper, we presented a polynomial delay and polynomial space algorithm that discover all maximal geographs in a given geometric configulations without duplicates. As future works, we are working on implementation and experimental evaluation of the algorithm. Extensions with approximation and constraints, with applications to image processing and geographic information systems, are other future problems.

References

[1] Aho, A. V., Hopcroft, J. E., Ullman, J. D., *Data Structures and Algorithms*, 1983.

[2] T. Akutsu, H. Tamaki, T. Tokuyama, Distribution of distances and triangles in a point set and algorithms for computing the largest common point sets, *Discr. & Comp. Geom.*, 20(3), 307–331, 1998.

[3] H. Arimura, T. Uno, An output-polynomial time algorithm for mining frequent closed attribute trees, In *Proc. ILP'05*, LNAI 3625, 1–19, August 2005.

[4] H. Arimura, T. Uno, A polynomial space and polynomial delay algorithm for enumeration of maximal motifs in a sequence, In *Proc. ISAAC'05*, LNCS, 2005.

[5] H. Arimura, T. Uno, Effcient algorithms for mining maximal flexible patterns in texts and sequences, TCS-TR-A-06-20, DCS, Hokkaido Univeristy, 2006. (submitting)

[6] T. Asai, K. Abe, S. Kawasoe, H. Arimura, H. Sakamoto, and S. Arikawa. Efficient substructure discovery from large semi-structured data. In *Proc. SDM'02*, 2002.

[7] Y. Chi, R. R. Muntz, S. Nijssen, J. N. Kok, Frequent subtree mining – An overview, *Fundam. Inform.*, 66, 1–2, 161–198, 2005.

[8] Y. Chi, Y. Yang, Y. Xia, and R. R. Muntz, CMTreeMiner: mining both closed and maximal frequent subtrees, In *Proc. PAKDD'04*, 2004.

[9] G. C. Garriga, R. Khardon, L. De Raedt, On mining closed sets in multi-relational data, In *Proc. IJCAI 2007*, 804–809, 2007.

[10] C. Guerra, Vision and image processing algorithms, *Algorithms and Theory of Computation Handbook*, Chapter 22, **22**-1–**22**-23, CRC Press, 1999.

[11] A. Inokuchi, T. Washio, H. Motoda, An apriori-based algorithm for mining frequent substructures from graph data, In *Proc. PKDD'00*, 13–23, LNAI 1910, 2000.

[12] A. Jain, *Fundamentals of Digital Image Processing*, Prentice-Hall, 1986.

[13] R. Khardon, Learning function-free horn expressions, *Machine Learning* 37(3), 241–275, 1999.

[14] M. Kuramochi, G. Karypis, Discovering frequent geometric subgraphs, In *Proc. IEEE ICDM'02*, 258–265, 2002.

[15] S. Nakano, Efficient generation of plane trees, *Information Processing Letters*, 84, 167–172, Elsevier,2002.

[16] S. Nijssen, J. N. Kok, Effcient discovery of frequent unordered trees, In *Proc. MGTS'03*, 2003.

[17] N. Pasquier, Y. Bastide, R. Taouil, L. Lakhal, Discovering frequent closed itemsets for association rules, In Proc. ICDT'99, 398–416, 1999.

[18] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*, Springer, 1985.

[19] A. Termier, M.-C. Rousset, M. Sebag, DRYADE: a new approach for discovering closed frequent trees in heterogeneous tree databases, In *Proc. ICMD'04*, 2004.

[20] K. Tsuda, T. Kudo, Clustering graphs by weighted substructure mining, *Proc. ICML 2006*, 953–960, 2006.

[21] T. Uno, T. Asai, Y. Uchida, H. Arimura, An efficient algorithm for enumerating closed patterns in transaction databases, In *Proc. DS'04*, LNAI 3245, 16–30, 2004.

[22] J. Wang, J. Han, BIDE: Efficient Mining of Frequent Closed Sequences, In *Proc. IEEE ICDE'04*, 79–90, 2004.

[23] T. Washio, H. Motoda, State of the art of graph-based data mining, *SIGKDD Explor.*, 5, 1, 59–68, 2003.

[24] X. Yan, J. Han, CloseGraph: mining closed frequent graph patterns In *Proc. KDD'03*, 2003.

[25] G. Yang, The complexity of mining maximal frequent itemsets and maximal frequent patterns, In *Proc. KDD'04*, 344–353, 2004.

[26] M. J. Zaki, Efficiently mining frequent trees in a forest, In *Proc. KDD'02*, 71–80, 2002.

[27] M. J. Zaki, C. C. Aggarwal, XRules: an effective structural classifier for XML data, In *Proc. KDD'03*, 316–325, 2003.

Note: The contents of this appendix is not included in the submission draft and can be ignored.

A Appendix: Basics in geometry

In this paper, we consider points and geometric objects on the *two-dimensional Eucledean space* (or the *plane*, for short) $\mathbb{E}$.

Throughout this paper, we fix the Cartesian coordinate system for $\mathbb{E}$. Each point $\vec{p}$ in the plane $\mathbb{E}$ is represented by a pair of real numbers $\vec{p} = (\vec{p}.x, \vec{p}.y) = (x, y) \in \mathbb{R}^2$, where x and y are called the x- and the y-coordinates of p, resp. A point $\vec{x}$ is also regarded as a *vector* in $\mathbb{R}^2$ as well. Let $\vec{e}_x = (1, 0)$ and $\vec{e}_y = (0, 1)$ be a pair of the orthogonal unit vectors for *x-axis* and *y-axis*. Therefore, our Cartesian coordinate system is specified by the axes $\psi = (\vec{e}_x, \vec{e}_y)$. For vectors $\vec{x} = (x, y) \in \mathbb{R}^2$, the *norm* or the *length* of $\vec{x}$ is defined by $||\vec{x}|| = (x^2 + y^2)^{\frac{1}{2}} \geq 0$.

Let $\vec{x}_1, \vec{x}_2 \in \mathbb{R}^2$ be two points in $\mathbb{R}^2$. Then, the *distance* of $\vec{x}_1$ and $\vec{x}_2$ is defined by $||\vec{x}_1 - \vec{x}_2||$. The line segment between $\vec{x}_1$ and $\vec{x}_2$ is denoted by $\overline{\vec{x}_1\vec{x}_2}$. The *angle* of two line segments $\vec{s}_1$ and $\vec{s}_2$ is denoted by $angle(\vec{s}_1, \vec{s}_2)$. We denote the *scalar product* of vector $\vec{x} \in \mathbb{R}^2$ with $c \in \mathbb{R}$ by $c\,\vec{x}$, and the *addition* of two vectors $\vec{x}, \vec{y}$ by $\vec{x} + \vec{y}$. The product of a 2×2 matrix A and a 2-vector $\vec{x}$ is denoted by $A\vec{x}$. Then, the determinant of A is written $det(A)$. If $det(A) \neq 0$, A is called *non-singlar* and always has its *inverse* A^{-1}.

In this paper, we consider three basic types of geometric transformations, *translation* M, *rotation* R, and *scaling* S defined as follows for any vector $\vec{x} \in \mathbb{R}^2$:

$$M : \vec{x} \mapsto \vec{x} + \begin{pmatrix} t_1 \\ t_2 \end{pmatrix}, \qquad R : \vec{x} \mapsto \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \vec{x}, \qquad S : \vec{x} \mapsto \begin{pmatrix} s & 0 \\ 0 & s \end{pmatrix} \vec{x},$$

where (t_1, t_2) is called a displacement, $0 \leq \theta < 2\pi$ is a rotaion angle (in clockwise), and $s > 0$ is a scaling factor of the transformations.

In general, any geometric transformation T can be represented as a *2D affine transformation* $T : \vec{x} \mapsto A\vec{x} + \vec{t}$, where A is a 2×2 nonsingular matrix with $det(A) \neq 0$, and $\vec{t}$ is a 2-vector. Such T is is one-to-one, and any transformation can be determined a set of three non-collinear points and their images. An affine transformation $T \in \mathfrak{T}_{\text{affine}}$ has the following nice properties: it maps parallel lines to parallel lines, is one-to-one and preserved the coordinates detemined by three non-collinear points. Also, any transformation can be determined a set of three non-collinear points and their images.

B Discussion: Critical perfect elimination sequences and application to maximal pattern enumeration

Given a k-pattern Q, its perfect critical elimination sequence $\texttt{critseq}(Q) = (\chi_j, \ldots, \chi_1)$, discussed in Sec. 3.2, is uniquely determined by the procedure *Elimination Ordering* shown in Fig. 4. The next lemma says that the critical elimination sequence has the complete information on the original pattern Q as expected.

Lemma 17 *For any $Q \in \mathcal{G}$,* $\text{Clo}(\texttt{critseq}(Q)) = Q$.

ClosureMain(σ : minsup, D : input database)

1: **global:** $Pool = \emptyset$;
2: **call** ClosureExpansion($\bot = \mathrm{Clo}(\emptyset), \sigma, D$);

ClosureExpansion(P, σ, D)

1: **global:** $Pool$; //*A pool of already discovered patterns*
2: **if** $|L(P)| < \sigma$ **then return**
3: **else if** $P \notin Pool$ **then**
4: **output** P as a frequent maximal subgraph; $Pool = Pool \cup \{P\}$;
5: **for** each possible labeled object $\langle o, l \rangle$ **do**
6: $Q = \mathrm{Clo}(P \cup \{\langle o, l \rangle\})$;
7: **call** ClosureExpansion(Q, σ, D);
8: **end for**

Figure 7: An output-polynomial time algorithm for enumerating all frequent-maximal patterns without duplicates. This algorithm is neither of polynomial delay or polynomial space.

Proof: To see this, suppose we reverse the elimination process of *Elimination Ordering* by the following procedure: STEP 1: $Q = \emptyset$; STEP 2: **for** $i = j$ **downto** 1 **do** $Q = \mathrm{Clo}(Q \cup \{\chi_i\})$; STEP 3: **return** Q; Then, this process finally reaches the original k-pattern Q. Since we can show that this iterative process is equivalent to $\mathrm{Clo}(\texttt{critseq}(Q))$, the proof is completed. □

From the above arguments, we can show that any maximal pattern Q can be generated from a properly smaller maximal pattern, say P, and some labeled object ξ by iterativelly applying the operation $Q = \mathrm{Clo}(P \cup \{\xi\})$ called the *closure extension* [17]. In Fig. 7, we present a recursive algorithm ClosureMain for the frequent maximal geograph enumeration problem.

Proposition 18 ClosureMain *is an output-polynomial time algorithm for the frequent maximal geograph enumeration problem, but neither a polynomial delay or polynomial space algorithm.*

To avoid duplication, ClosureMain has to use explicit maximality test at Line 3 of Fig. 7 by maintaining a possibly exponentially large pool $Pool$ of discovered patterns. Unfortunately, however, ClosureMain may visit an identical maximal pattern possibly exponentially many times in the worst case through distinct search route. Therefore, we conclude that ClosureMain is neither polynomial delay or polynomial space. In the next subsection, we will show how to improve this approach.