大規模幾何データからの高速な極大部分グラフ発見

Efficient Maximal Pattern Discovery from Massive Geometric Graphs

有村博紀 [1], 宇野毅明 [2], 下薗真一 [3]*
Hiroki Arimura[1], Takeaki Uno[1], Shinichi Shimozono[1]

[1] 北海道大学大学院情報科学研究科 Hokkaido University
emailarim@ist.hokudai.ac.jp
[2] 国立情報学研究所 National Institute of Informatics
uno@nii.jp
[3] 九州工業大学情報工学部 Kyushu Institute of Technology
sin@ai.kyutech.ac.jp

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.

1 Introduction

Backgrounds. By rapid growth of both the amount and the varieties of nonstandard datasets in scientific, spatial, and relational domains, there are increasing demands for efficient methods to extract useful patterns and rules from weakly structured datasets. *Graph mining* is one of the most promising approaches for knowledge discovery from such weakly structured datasets. The following topics have been extensively studied for the last years: frequent subgraph mining [7, 12, 17, 27], maximal (closed) subgraph mining [3, 9, 20, 25] and combination with machine learning [21, 28]. See surveys, e.g. [8, 24], for the overviews.

Geometric graphs. In this paper, we study a graph mining problem for the class $\mathcal{G}$ of geometric graphs. *Geometric graphs* (*geographs*, for short) [15] are a special kind of vertex- and edge-labeled graphs whose vertices have the coordinates in the 2D plane $\mathbb{R}^2$, while the labels represent geometric features and their relationships. The matching relation for geographs is defined through the invariance 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 concerning with geometric configurations, 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 efficient mining algorithm gFSG for frequent geometric subgraph mining, based on Apriori-like breadth-first search [15]. However, frequent pattern mining has a problem that it can easily produce an extremely large number of solutions, which degrades the performance and the comprehensivity of data mining to a large extent. On the other hand,

*連絡先：有村博紀，北海道大学大学院情報科学研究科
〒 060-0814 札幌市北区北 14 条西 9 丁目
E-mail: arim@ist.hokudai.ac.jp

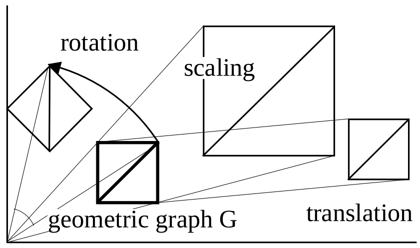

図 1: Three basic types of geometric transformations

the *maximal subgraph mining problem*[1] asks to find only all *maximal patterns* (closed patterns) appearing in a given input geometric graph D, where a *maximal pattern* is a geometric graph which is not included in any properly larger subgraph having the same set of occurrences in D. Since the set $\mathcal{M}$ of all maximal patterns is expected to be much smaller than the set $\mathcal{F}$ of all frequent patterns and still contains the complete information of D, maximal subgraph mining has some advantage as a compact representation to frequent subgraph mining.

Difficulties of maximal pattern mining. However, a number of difficulties in maximal subgraph mining for geometric graphs exist. In general, maximal pattern mining has a large computational complexity [4, 26]. So far, a number of efficient maximal pattern algorithms are proposed for *sets, sequences*, and *graphs* [3, 9, 20, 22, 25]. Some algorithms use explicit duplicate detection and maximality test by maintaining a collection of already discovered patterns. This requires large memory and delay time, and introduces difficulties 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. Moreover, the invariance under geometric transformation for geometric graphs adds another difficulty to geometric graph mining. In fact, no depth-first algorithm has been known so far even for frequent pattern mining.

Main result. The goal of this paper is to develop 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 $O(m(m+n)||D||^2 \log ||D||) = O(n^8 \log n)$ time per pattern and in $O(m) = O(n^2)$ space, with the maximum number m of occurrences of a pattern other than trivial patterns, the number n of vertices in the input graph, and the number $||D||$ of vertices and edges in the input graph. 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 maximal 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 code* for all geometric graphs in $\mathcal{G}$, which is invariant under geometric transformations. As a 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 operation* for $\mathcal{G}$. Using these tools, we define the *tree-shaped search route* $\mathcal{T}$ *for all maximal patterns* in $\mathcal{G}$. We propose a new pattern growth technique arising from reverse search and *closure extension* [18] for traversing the search route $\mathcal{R}$ by depth-first search.

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, 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 [25], ordered and unordered trees [9], attribute trees [3, 20], and sequences [4, 6, 23]. 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 [9], BIDE [23], and MaxFlex [6] are essentially output-polynomial time algorithms for location-based maximal patterns though it is implicit. They are originally used as pruning for document-based maximal patterns [6].

The second group deal with mining of "rigid" patterns which have *closure*-like operations. LCM [22] proposes ppc-extension for maximal sets, and then CloATT [3] and MaxMotif [4] generalize it for trees

[1] Although the maximal pattern discovery 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, 26].

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 [25], 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.

Organization of this paper. In Section 2, we introduce maximal pattern mining for geometric graphs. In Section 3, we give the canonical code and frequent pattern mining. In Section 4, we present polynomial delay and polynomial space algorithm MaxGeo for maximal pattern mining, and in Section 5, we conclude. For the details and the proof of this work, see the technical report [5].

2 Preliminaries

In this section, we prepare basic definitions and notations for maximal geometric graph mining. See Appendix for a detailed explanation. 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 [11, 13]. In this paper, we consider geometric objects, such as points, lines, point sets, and polygons, on the *two-dimensional Euclidean 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 $\mathcal{T}_{\rm geo}$ of geometric transformations consisting of three basic types of geometric transformations: *rotation*, *scaling*, and their combinations. In general, any geometric transformation $T \in \mathcal{T}_{\rm geo}$ 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 and onto. In addition, if $T \in \mathcal{T}_{\rm geo}$ then T preserves the angle between two lines. It is well-known that any affine transformation can be determined by a set of three non-collinear points and their images. For $\mathcal{T}_{\rm geo}$, we have the following lemma.

Lemma 1 *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 $i = 1, 2$.*

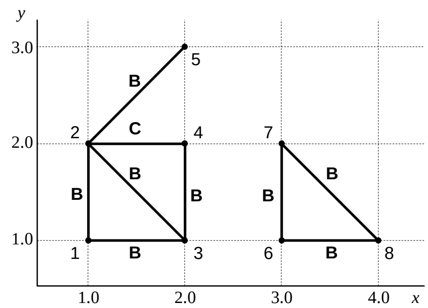

図 2: A geometric database D with $V = \{1, \ldots, 9\}$, $\Sigma_{\rm V} = \emptyset$, and $\Sigma_{\rm E} = \{\mathsf{B}, \mathsf{C}\}$

$\mathbf{T}(\overline{\vec{x}_1\vec{x}_2}; \overline{\vec{x}'_1\vec{x}'_2})$ is computable in $O(1)$ time. The above lemma is crucial in the following discussion. For any geometric object O and $T \in \mathcal{T}_{\rm geo}$, 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 [15] as follows. Let $\Sigma_{\rm V}$ and $\Sigma_{\rm E}$ be mutually disjoint sets of *vertex labels* and *edge labels* associated with total orders $<_\Sigma$ on $\Sigma_{\rm V} \cup \Sigma_{\rm E}$. In what follows, a vertex is always an element of $\mathbb{N}$. A *graph* is a vertex and edge-labeled graph $G = (V, E, \lambda, \mu)$ with a set V of *vertices* and a set $E \subseteq V^2$ of *edges*. Each $x \in V$ has a vertex label $\lambda(x) \in \Sigma_{\rm V}$, and each $e = xy \in E \subseteq V^2$ represents an unordered edge $\{x, y\}$ with an edge label $\mu(e) \in \Sigma_{\rm E}$. 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., there is a bijection $\phi : V_1 \to V_2$ such that (i) $\lambda_1(x) = \lambda_2(\phi(x))$, (ii) for every $xy \in (V_1)^2$, $xy \in E_1$ iff $\phi(x)\phi(y) \in E_2$, and (iii) for any $xy \in E_1$, $\mu_1(xy) = \mu_2(\phi(x)\phi(y))$. 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 labeled graph and $c : V \to \mathbb{R}^2$ is a one-to-one mapping 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 of G as $V_G, E_G, c_G, \lambda_G, \mu_G$.

We here assume that no two vertices or edges have the same coordinates. We denote by $\mathcal{G}$ the *class of all geometric graphs* over $\Sigma_{\rm V}$ and $\Sigma_{\rm E}$.

Alternative representation for geographs. Alternatively, a geometric graph can be simply represented as a collection of labeled objects $\underline{G} = \underline{V} \cup \underline{E}$, where $\underline{V} = \{ \langle \vec{x}_i, \lambda_i \rangle \mid i = 1, \ldots, n \} \subseteq \mathbb{R}^2 \times \Sigma_\mathrm{V}$, and $\underline{E} = \{ \langle e_i, \mu_i \rangle \mid i = 1, \ldots, m \} \subseteq \mathbb{R}^2 \times \mathbb{R}^2 \times \Sigma_\mathrm{E}$. Each $\langle \vec{x}, \lambda \rangle$ is a *labeled vertex* for a vertex v with $c(v) = \vec{x}$ and $\lambda(v) = \lambda$, and each $\langle c(v), c(u), \mu \rangle$ is a *labeled edge* for an edge $e = vu$ with label $\mu(e) = \mu$. A *labeled object* refers to either a labeled vertex or a labeled edge. Let $OL = (\mathbb{R}^2 \times \Sigma_\mathrm{V}) \cup (\mathbb{R}^2 \times \mathbb{R}^2 \times \Sigma_\mathrm{E})$ be a domain of labeled objects. We assume the lexicographic order $<_{OL}$ 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 $G \setminus \{\langle e, \mu \rangle\}$. 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 [15]. 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 a transformation $T \in \mathcal{T}_\mathrm{geo}$ 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 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 $\mathcal{M}(P, G) \subseteq \mathcal{T}_\mathrm{geo}$ *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. 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, occurrences, 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 $\mathcal{T}_\mathrm{geo}$, we assume without any loss of generality that if P is a k-pattern then $V_P = \{1, \ldots, k\}$, 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* of size $n \geq 0$ is a single geograph $D = (V, E, c, \lambda, \mu) \in \mathcal{G}$ with $|V| = n$. We denote $|V| + |E|$, which is total size of D, by $||D||$. D is also called an *input geograph*. Fig. 2 shows an example of an input geometric database D with $V = \{1, \ldots, 9\}$ over $\Sigma_\mathrm{V} = \emptyset$, and $\Sigma_\mathrm{E} = \{\mathsf{B}, \mathsf{C}\}$.

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 $|L(P)| \in \mathbb{N}$. For an integer $0 \leq \sigma \leq n$, called a *minimum support* (or *minsup*), P is σ-frequent in D if its frequency is no less than σ.

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 no greater than n^2 under $\mathcal{T}_\mathrm{geo}$.*

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) $P \sqsubset Q$ 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's. Equivalently, P is maximal iff any addition of a labeled object 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 (data mining 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 every frequent maximal geometric pattern $P \in \mathcal{M}^\sigma$ appearing in D without outputting no isomorphic two.

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 an open question for $\mathcal{M}$ and even for $\mathcal{F}^\sigma$ so far.

We can define a different notion of 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. Actually, 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 [19]: 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, 19].

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 and the output size. $\mathcal{A}$ is *polynomial delay* if the *delay*, which is the maximum computation time between two consecutive outputs is bounded by polynomials in the input size. If $\mathcal{A}$ is polynomial delay then $\mathcal{A}$ is also output-polynomial time. $\mathcal{A}$ is a *polynomial space* algorithm if the maximum space $\mathcal{A}$ uses is bounded by a polynomial in the input size.

Algorithm MaxGeo: (D : geograph, σ : *minsup*)
task: finding all σ-frequent maximal patterns of $\mathcal{M}^\sigma$ in D.
$\bot = \mathrm{Clo}(\emptyset)$; //*The bottom maximal geograph*
Call the recursive procedure Expand_MaxGeo($\bot, \mathbf{0}, \sigma, D$);

Algorithm Expand_MaxGeo(P, π, σ, D)
if P is not σ-frequent **then return**; //*Frequency test*
else output P as a σ-frequent maximal geograph;
for each missing labeled object $\xi = \langle o, \ell \rangle$ **do**
  $Q = \mathrm{Clo}(P \cup \{\xi\})$;
  if ($\xi \leq_{\mathtt{cano\_e}(Q)} \pi$) **then return**;
  if ($(\exists \tau \in Q - P)\ \tau <_{\mathtt{cano\_e}(Q)} \xi$) **then return**;
  call Expand_MaxGeo(Q, ξ, σ, D);
endfor

図 3: A polynomial delay and polynomial space algorithm MaxGeo for the maximal geometric subgraph enumeration problem

3 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.

3.1 Outline of the algorithm

In Fig. 3, 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 $\mathcal{R} = \mathcal{R}(\mathcal{M}^\sigma)$ implicitly defined over $\mathcal{M}^\sigma$. Then, starting at the root of the search route $\mathcal{R}$, the algorithm MaxGeo searches $\mathcal{R}$ by 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, thus the algorithm is polynomial delay.

3.2 Canonical encoding for geographs

In this subsection, to properly handle the geometric isomorphism among the isomorphic patterns, we introduce the canonical code for geometric patterns, which is invariant under geometric transformations in $\mathcal{T}_{\mathrm{geo}}$. 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.

Suppose that the vertex set V_P of P has at least 2 vertices. Let $\vec{o} = (\sum_{v \in V_P} c(v))/|V_P|$ be the centroid (the *center*) of the vertices in P, which is the averages of x-coordinates and y-coordinates of all vertices in P. We choose a point $\vec{x} \in P, \vec{x} \neq \vec{o}$ having the minimum Euclidean distance to $\vec{o}$ called *base point*. Denote by Q the pattern obtained by transforming P in a polar coordinate system such that $\vec{o}$ is mapped to the origin and $\vec{x}$ is mapped to $(1,0)$, where the first element of the coordinates gives the angle. We define the coordinate of the origin by $(0,0)$. Let $O = \underline{V}_Q \cup \{\langle c(v), c(u) - c(v), \mu_{uv}\rangle, \langle c(u), c(v) - c(u), \mu_{uv}\rangle \mid uv \in E_Q\}$. Then, the code $\mathtt{Code}(P, \vec{x})$ of P is defined by the elements of O sorted in the lexicographic order.

Clearly, there are at most k distinct $\mathtt{Code}(P, \vec{x})$ depending on the choice of the base point $\vec{x}$. Then, the *canonical code* $\mathtt{Code}^*(P)$ for pattern P is defined by the lexicographically minimum code among the codes of P. A pattern P is said to be *canonical* if (i) it has no vertex, (ii) it has one vertex at $(0,0)$, or (iii) its vertices are indexed in the order of its canonical code.

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

A code can be computed in $O(k^2 \log k)$ time for any k-pattern P and base point $\vec{x}$, then the code for another base point is obtained by shifting it. Hence, we can compute the canonical code of P in $O(k^2 \log k)$ time. The purpose of the canonical code and the canonical pattern is to define a representative pattern among the geometric isomorphic patterns. Thus, our task is to enumerate all σ-frequent canonical patterns.

3.3 Finding missing objects

There are infinitely many candidates for possible labeled object at Line 3 of Fig. 3. From the next lemma, we can avoid such a blind search by focusing only on *missing 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 5 (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 $\pi \in \underline{D}$.*

From Lemma 5 above, we know that there are at most $O(|L(P)| \cdot ||D||) = O(|V|^2(|V| + |E|))$ missing objects.

Combining the discussions in the previous subsection and this subsection, we have the following theorem as a biproduct of canonical coding.

Theorem 6 (frequent geographs) *The algorithm* FreqGeo *in Fig.* **??** *enumerates all σ-frequent geometric graphs in a given input database $D \in \mathcal{G}$ in polynomial delay and polynomial space in the total input size.*

3.4 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 geograph which is represented by labeled objects common to both G_1 and G_2. MCGS is unique for geographs, while they are not unique for ordinary graphs.

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 computation time for $\cap\mathbf{G}$ are bounded by $O(||\mathbf{G}|| \log ||\mathbf{G}||)$. Some literatures [14] give an intersection of labeled graphs or first-order models in a different way which is based on the *cross product* of two structures. However, their iterative applications causes exponentially large intersection unlike $\cap\mathbf{G}$ above. Gariiga *et al.*[10] 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 $\mathrm{Clo}(P)$:

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

Theorem 7 (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. $\sqsubseteq$ satisfying $L(\mathrm{Clo}(P)) = L(P)$.*

Lemma 8 *For any geographs $P, Q \in \mathcal{G}$, the following properties hold:*

- *(i)* $P \sqsubseteq \mathrm{Clo}(P)$. *(ii)* $L(\mathrm{Clo}(P)) \equiv L(P)$. *(iii)* $\mathrm{Clo}(P) \equiv \mathrm{Clo}(\mathrm{Clo}(P))$.
- *(iv) $P \sqsubseteq Q$ iff $L(P) \supseteq L(Q)$ for any maximal $P, Q \in \mathcal{M}$.*
- *(v)* $\mathrm{Clo}(P)$ *is the unique, smallest maximal geograph containing P.*

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

Theorem 9 (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* $\mathrm{Clo}(P) \equiv P$.

3.5 Defining the tree-shaped search route

In this subsection, we define a tree-like search route $\mathcal{R} = (\mathcal{M}^\sigma, \mathcal{P}, \bot)$ for the depth-first search of all maximal geographs based on a so-called parent function.

Let $Q \in \mathcal{M}$ be a maximal pattern with vertices at least 2 such that $Q \neq \bot$. For any labeled object $\xi \in \underline{Q}$, define the ξ-prefix of Q as the pattern $Q[\xi]$ which is the collection of the labeled objects prior to ξ in $\mathtt{Code}^*(Q)$. Then, the *core index* $\mathtt{core\_i}(Q)$ of Q is the labeled object ξ such that $L(Q[\xi']) \neq L(Q)$ holds for any ξ' prior to ξ in $\mathtt{Code}^*(Q)$. We can show that if $Q \neq \bot$ then $\mathtt{core\_i}(Q)$ is always defined.

$Q[\mathtt{core\_i}(Q)] \subseteq Q$ 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 a properly shorter prefix, and then the location list changes. Now, we define the parent function $\mathcal{P}$ that gives the predecessor of Q.

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

Lemma 10 $\mathcal{P}(Q)$ *is (i) always defined, (ii) unique, and (iii) 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)* $L(\mathcal{P}(Q)) \supset L(Q)$.

Now, we define the *search route* for $\mathcal{M}^\sigma$ as a rooted directed graph $\mathcal{R}(\mathcal{M}^\sigma) = (\mathcal{M}^\sigma, \mathcal{P}, \bot)$, where $\mathcal{M}^\sigma$ is the vertex set, $\mathcal{P}$ is the set of reverse edges, and $\bot$ is the root. For the search route, we have the following theorem.

Theorem 11 (reverse search property) *For every* σ, *the search route* $\mathcal{R}(\mathcal{M}^\sigma)$ *is a spanning tree with the root* $\bot$ *over all the maximal patterns in* $\mathcal{M}^\sigma$.

3.6 A polynomial space polynomial delay algorithm

The remaining thing is to show how we can efficiently traverse the search route $\mathcal{R}(\mathcal{M}^\sigma)$ starting from $\bot$. However, this is not a straightforward task since $\mathcal{R}(\mathcal{M}^\sigma)$ only has the reverse edges. To cope with this difficulty, we introduce the technique so called reverse search [?] and the closure extension [18].

Lemma 12 *For maximal patterns* Q *and* P, P *is the parent of* Q *only if* $Q \equiv clo(P \cup \xi)$ *holds for a missing object* ξ *for* P.

The operation of adding a labeled object and taking its closure is called *closure extension*. Lemma 12 says that any maximal geometric pattern can be obtained by applying to $\bot$ closure extensions repeatedly.

From Lemma 12, we can see that to find all children of a pattern P, we have to examine the closure extension for all missing objects for P. Clearly, a closure extension $Q = Clo(P \cup \xi)$ of P is a child of P if its parent is P. Since the parent of Q can be obtained by computing its canonical code, we can check whether a closure extension is a child or not in $O(k^2 logk)$ time where k is the number of labeled objects in $\underline{Q}$. Since the computation of $clo(Q)$ takes $O(|L(Q)| \times ||D|| \log ||D||)$ time, we obtain the following theorem.

Theorem 13 (correctness and complexity of MaxGeo) *Given an input geograph* D *with vertex set* V *and a minimum support threshold* $\sigma > 0$, *the algorithm* MaxGeo *in Fig. 3 enumerates all* σ*-frequent maximal geographs in* $O((m||D||) \times ((m+n)||D|| \log ||D||)) = O(m(m+n)||D||^2 \log |D|)$ *per maximal geograph with* $O(||D||)$ *space, where* $m = O(n^2)$ *is the maximum size of the location lists.*

If σ is not so small, then the number of missing objects to examine will be small, such as $O(n)$, then the computation time will be short. It is expected in practical computation. Moreover, in practice, usually almost all (maximal) patterns to be output have small frequency close to σ, thus computation time for the closure operation is rather short. According to the computational experiments in [4, 22], practical computation time is very small in such cases.

Corollary 14 *The maximal geograph enumeration problem is solvable in polynomial delay and polynomial space.*

4 Conclusion

In this paper, we presented a polynomial delay and polynomial space algorithm that discovers all maximal geographs in a given geometric configuration without duplicates. As future works, we are working on implementation and experimental evaluation of the algorithm. Dealing with input of many geographs and document occurrence is a straightforward work.

Dealing with polygons is also straightforward, by using sophisticated labels to identify edges of polygons as a group. Extensions with approximation and constraints, with applications to image processing and geographic information systems, are other future problems.

参考文献

[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, S. Shimozono, Time and Space Efficient Discovery of Maximal Geometric Graphs, In *Proc. DS2007*, LNAI, Springer, 2007. (to appear) (Also appeared as TCS Technical Report Series A, Hokkaido University, Division of Computer Science, TCS-TR-A-07-26, 2007.)

[6] 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)

[7] 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.

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

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

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

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

[12] 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.

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

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

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

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

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

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

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

[20] 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.

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

[22] 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.

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

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

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

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

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

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