Proc. 2009 SIAM International Conference on Data Mining (SDM'09), SIAM, April 2009. (to appear)

Polynomial-Delay and Polynomial-Space Algorithms for Mining Closed Sequences, Graphs, and Pictures in Accessible Set Systems

Hiroki Arimura
Hokkaido University, Grad. School of IST
N14 W9, Sapporo 060-0814, JAPAN
arim@ist.hokudai.ac.jp

Takeaki Uno
National Institute of Informatics
Tokyo 101–8430, JAPAN
uno@nii.jp

Abstract

In this paper, we study efficient closed pattern mining in a general framework of *set systems*, which are families of subsets ordered by set-inclusion with a certain structure, proposed by Boley, Horváth, Poigné, Wrobel (PKDD'07 and MLG'07). By modeling semi-structured data such as sequences, graphs, and pictures in a set system, we systematically study efficient mining of closed patterns. For a class of accessible set systems with a tree-like structure, we present an efficient depth-first search algorithm that finds all closed sets in accessible set systems without duplicates in polynomial-delay and polynomial-space w.r.t. the total input size using efficient oracles for the membership test and the closure computation for the pattern class. From the above results, we show that the closed pattern mining problems are efficiently solvable both in time and space for the following classes: *convex hulls, picture patterns* in 2-D planes, *maximal bi-cliques, closed relational graphs, closed patterns for rigid motifs with wildcards*.

1 Introduction

1.1 Closed pattern mining. *Frequent itemset mining* (FIM) has been extensively studied for the last decade, and still expanded in the various directions such as constrained mining, semi-structured mining, and closed/maximal set mining. The problem of *closed set mining* (CIM) is one of the most extensively studied topics for years that attracts many researchers from both practical and theoretical views [3, 4, 5, 6, 7, 11, 12, 17, 18, 22, 25, 26, 28, 27, 31]. A *closed itemset* (a *closed sets*, for short) is a representative among an equivalence class of itemsets that have the same set of occurrences in common in a given transaction database $\mathcal{D}$. The collection $\mathcal{C}$ of all closed sets in $\mathcal{D}$ contains the same information as the original collection $\mathcal{F}$ of frequent itemsets, while the size of $\mathcal{C}$ is possibly much smaller than $\mathcal{F}$. Hence, from the engineering point of view, CIM is considered useful for increasing the comprehensibility and the efficiency of FIM [26, 31]. As an example of the latter aspects, LCM algorithm [25, 26] makes full use of closed pattern mining to achieve high-throughput computation of frequent/closed/maximal itemsets in large data sets.

Besides this, from the theoretical point of view, considerable efforts have been paid for establishing theoretical foundations of closed pattern mining. Boros, Gurvich, Khachiyan, Makino [14] are first to show that the closed set mining problem is solvable in incremental polynomial time in the input database size. Uno and Arimura [25, 26] found that previous depth-first frequent set miners can be naturally generalized to CIM, and that their algorithm LCM achieves polynomial-delay and polynomial-space complexity in theory [25] and also quite efficient in practice [26]. On the other hand, some authors show that enumeration of closed patterns is non-trivial task than that of frequent patterns by proving the $\#P$-hardness of the enumeration problem [7, 30]. Furthermore, extensions of CIM for various subclasses of semi-structured data such as sequences, trees, and graphs are ongoing [3, 4, 5, 6, 7, 12, 17, 18, 22, 28, 27].

1.2 Closed pattern mining in accessible set systems. Interestingly, most of known results for efficient closed pattern mining use a *closure operator* as its key component, which is a mapping $\mathbb{C}$ that returns the smallest closed pattern $\mathbb{C}(X)$ containing a given possibly non-closed pattern X. In their PKDD2007 paper, Boley, Horváth, Poigné, Wrobel [9] introduced a theoretical framework based on *set systems* for analyzing closed pattern mining, borrowed from optimization theory, and they showed a nice result that for any finite set system $(\mathcal{F}, E)$ equipped with a closure operator $\mathbb{C} : \mathcal{F} \to \mathcal{F}$, where E is a finite set and $\mathcal{F} \subseteq 2^E$ is a family of its subsets, if the system is *strongly accessible* then the family of all closed sets $\mathbb{C} = \mathbb{C}(\mathcal{F})$ is

polynomial-delay enumerable assuming that total ordering $\leq$, membership oracle $\mathbb{M}_{\mathcal{F}}$, and closure operator $\mathbb{C}$ for $\mathcal{F}$ are all polynomial-time computable (Theorem 4 in [9]). In the proof of the theorem, they built a DAG $G_{\mathbb{C}}^{\mathrm{BHPW}} = (V, E)$ with the vertex set is $V = \mathcal{C} = \mathbb{C}(\mathcal{F})$, and the directed edge set is $E = \{ (C, D) \in V^2 \mid (\exists e \in E \setminus C)\ D = \mathbb{C}(C \cup \{e\})$ and $C \cup \{e\} \in \mathcal{F} \}$. Then, Boley, Horváth, Poigné, Wrobel [9] propose an enumeration method for $\mathcal{C}$ by breadth-first or depth-first search over the graph $G_{\mathbb{C}}^{BHPW}$. In their search over a general directed graph (not a tree), a crucial point for the correctness is need for explicitly testing duplication of enumerated solutions. To solve this, they used prefix trees as a table for storing discovered closed sets so far.

1.3 Potential problems with the previous approaches. Generally speaking, although enumeration algorithms with table-checking does not always yield a polynomial-delay algorithm, they showed that at most $O(n^2)$ successive closure computation with operator $\mathbb{C}$ are sufficient to know the existence of the next closed set and to find it if exists in the enumeration. Therefore, they succeeded to show the *polynomial-delay* enumerability of closed sets $\mathcal{C} = \mathbb{C}(\mathcal{F})$ over a strongly accessible set systems, and thus from the view of the time complexity, the algorithm in [9] seems satisfactory. On the other hand, from the view of the space complexity, it seems that there still is a room for improvements. The algorithm in [9] requires the memory space proportional to the number of all solutions, which may be exponential in the total input size.

1.4 Main result: a poly-delay and poly-space algorithm. In this paper, we study a systematic way to design high-throughput and light-weight mining algorithms for a wider range of closed pattern problems. Our strategy is as follows. Given a concrete class of patterns in a semi-structured mining problem, encode this class in an *accessible set systems*. We then show that two major representation frameworks, called *closure-based closed set systems* and *intersection-closed closed set systems* are equivalent each other. Thus, we can use either of them to design an efficient algorithm depending on the character of the patterns.

Now, our goal is to develop a polynomial-delay and polynomial-space algorithm for the closed set mining problem. A basic idea is to build a tree-like search route over all closed sets in an accessible set system, and perform a depth-first search over them. We first analyze the existence theorem for *inductive generators* for closed sets in Boley *et al.* [9], and gives a computational version of the theorem by careful analysis of the interaction between the closure operator and the lexicographic structure of subsets of $\mathcal{F}$. Especially, we strengthen the theorem by relaxing the constraint of Boley *et al.* [9] by replacing the requirement for *strong accessibility* with accessibility. Then, we give a construction of a virtual search structure, called a *family tree*, which is a spanning tree for $\mathcal{C}$ formed by a set of *reverse edges* from children to its parents, for the search space consisting of all closed sets in $\mathcal{C} = \mathbb{C}(\mathcal{F})$. After showing some technical lemmas for inverting the reverse edges, we present a polynomial-delay and polynomial-space algorithm CloGenDFS that enumerates all closed sets in $\mathcal{C} = \mathbb{C}(\mathcal{F})$ without duplicates from E using polynomial-time computable total ordering $\leq$, membership oracle $\mathbb{M}_{\mathcal{F}}$, and closure operator $\mathbb{C}$ for $\mathcal{F}$.

1.5 Applications and Contributions. Finally, we demonstrate the usefulness of our result by applying it to a variety of closed pattern mining problems in semi-structured data. We show that the closed pattern mining problems are solvable for the following classes of objects:

(i) Convex hulls in 2-D points [19].
(ii) Maximal bi-cliques (bi-clusters) [23].
(iii) Closed subgraphs in relational graphs [29].
(iv) Closed rigid motifs with wildcards [7, 17, 18].
(v) Closed rigid sequences from itemset streams [17].
(vi) Closed pictures in the 2-D plane under translation and rotation [6].

In the above results, (ii), (iii) and (iv) are reformulation of previously known results, and (i), (v), (vi) are new results for open problems.

Contributions of the main result are summarized as follows. For the closed set mining problem in an accessible set system, this paper

- shows the *polynomial-delay and polynomial-space* complexity for the closed set mining problem in an accessible set system at the first time [1].
- presents a *systematic construction* of a *depth-first algorithms* for closed pattern mining.
- shows that it is possible to relax the constraint for set systems by replacing the constraint of strong accessibility with *accessibility*.
- demonstrates the power of the above construction by building polynomial-delay and polynomial-space closed mining algorithms for old and new classes.

[1] In the preparation of this paper, the present author knew from [15] that after the publication of their paper [9], the authors of [9] had independently obtained a polynomial-delay and polynomial-space algorithm for this problem

1.6 Organization. This paper is organized as follows. In Section 2, we review the basic definitions and notations In Section 3, we examine the output polynomial-time algorithm presented by Boley *et al.* in [9]. In Section 4, we present our polynomial-delay and polynomial-space algorithm for closed set mining problem for an accessible set system. In Section 5 and Section 6, we discuss possible applications to the closed semi-structured data mining. In Section 6, we conclude.

2 Preliminaries

In this section, we define basic notions and notations necessary for the rest of this paper. For definitions that is now defined here, please consult text books, e.g., [24].

2.1 Basic definitions. We denote by $\mathbb{Z}$ and $\mathbb{N} = \{0, 1, 2, \ldots\}$ the sets of all integers and natural numbers, resp. For a set E, $|E|$ denotes the cardinality of E, and 2^E denotes the class of all subsets of E. For sets A, B, we define $B^A = \{ f \mid f : A \to B \}$ the class of all function f from A to B. For a partial order $(A, \leq)$, the *greatest lower bound* (g.l.b.) of $S \subseteq A$ is defined by the $z = \sqcap S \in A$ such that (i) $z \leq x$ for all $x \in S$ and (ii) no $z' > z$ satisfies $z' \leq x$ for all $x \in S$. We write $x_1 \sqcup x_2$ for $\sqcup\{x_1, x_2\}$.

2.2 Set systems. Let $E = \{a_0, a_1, \ldots\}$ be a countable set of elements, called a *domain* (or, a *ground set*). We assume a total order $\leq_E$ associated with E. Typically, we can build such orders from natural total orders over numbers and letters.

Definition 1. A *set system* over E is a pair $(\mathcal{F}, E)$ of a family $\mathcal{F} = \{X_0, X_1, \ldots\} \subseteq 2^{\mathcal{F}}$ of subsets of E, where each subset $X \in \mathcal{F}$ is called a *feasible set*. We associate to $\mathcal{F}$ the set-inclusion $\subseteq$ as a natural ordering over $\mathcal{F}$.

In the context of data mining, we often call $X \in \mathcal{F}$ a *pattern* instead of a feasible set. In this paper, E may be infinite, but all feasible sets of $\mathcal{F}$ are finite. If it is clear from context, we may omit the domain E and write $\mathcal{F}$ for $(\mathcal{F}, E)$.

For a set system $\mathcal{F} \subseteq 2^E$, the *total size* of $\mathcal{F}$ is $||\mathcal{F}|| = \sum_{A \in \mathcal{F}} |A|$. A set $X \in \mathcal{F}$ is *maximal* (*minimal*) within $\mathcal{F}$ if there is no $Y \in \mathcal{F}$ such that X is a proper subset (a proper superset) of Y. For any $\mathcal{G} \subseteq \mathcal{F}$, $\min \mathcal{G}$ ($\max \mathcal{G}$, resp.) denotes the unique *minimum* (*maximum*) element in $\mathcal{G}$ w.r.t. $\subseteq$ if it exists.

Definition 2. (accessible set system) A set system $(\mathcal{F}, E)$ is *accessible* if for every non-empty $X \in \mathcal{F}$, there exists some $a \in X$ such that $X \setminus \{a\} \in \mathcal{F}$.

Definition 3. (strongly accessible set system) A set system $(\mathcal{F}, E)$ is *strongly accessible* if for every $X, Y \in \mathcal{F}$ such that $X \subset Y$ ($X \neq Y$), there exists some $a \in Y \setminus X$ such that $X \cup \{a\} \in \mathcal{F}$.

By definition, if an accessible set system $\mathcal{F}$ consists of finite sets only, then $\mathcal{F}$ always contains $\emptyset$. Any strongly accessible, finite set system is also an accessible set system, the converse does not hold in general. Let $\mathcal{C} \subseteq \mathcal{F}$ be any subfamily of feasible sets, and $X \in \mathcal{F}$ be any feasible set. Then, we define the upper set $\mathcal{C}^{\mathrm{up}}(X)$ and the lower set $\mathcal{C}_{\mathrm{low}}(X)$ of X by $\mathcal{C}^{\mathrm{up}}(X) = \{ C \in \mathcal{C} \mid X \subseteq C \}$ and $\mathcal{C}_{\mathrm{low}}(X) = \{ C \in \mathcal{C} \mid C \subseteq X \}$, resp.

2.3 Closed sets and closure operators. A *closure operator* for a set system $(\mathcal{F}, E)$ is any mapping $\mathbb{C} : \mathcal{F} \to \mathcal{F}$ such that for any subsets $X, Y \in \mathcal{F}$,

(C1) Extensibility: $X \subseteq \mathbb{C}(X)$ holds.
(C2) Monotonicity: $X \subseteq Y$ implies $\mathbb{C}(X) \subseteq \mathbb{C}(Y)$ holds.
(C3) Idempotence: $\mathbb{C}(\mathbb{C}(X)) = \mathbb{C}(X)$ holds.

Definition 4. ($\mathbb{C}$-closed set) A feasible set $X \in \mathcal{F}$ is $\mathbb{C}$-*closed* (or, *closed*) if $\mathbb{C}(X) = X$. We denote by $\mathcal{C} = \mathbb{C}(\mathcal{F}) = \{ \mathbb{C}(X) \mid X \in \mathcal{F} \}$ the family of all $\mathbb{C}$-closed sets in $\mathcal{F}$.

The *smallest closed set* in $\mathcal{C}$ is denoted by $\bot = \mathbb{C}(\emptyset)$. Even if $\mathcal{F}$ is accessible, $\mathcal{C} = \mathbb{C}(\mathcal{F})$ may not be an accessible set system. Now, we give a useful characterization of closure systems below. Let E be a countable set of elements, and $\mathcal{F}$ be a set system over E. Then, a set system $\mathcal{C} \subseteq \mathcal{F}$ is said to be *closed under intersection* if for every $C_1, C_2 \in \mathcal{F}$, $C_1, C_2 \in \mathcal{C}$ implies $C_1 \cap C_2$. The following property holds since $\cap C^{up}(X)$ is the unique mininmal element in $C^{up}(X)$.

Property 2.1. *If $\mathcal{C}$ is closed under intersection, $\cap \mathcal{C}^{up}(X)$ is the unique minimum element of $\mathcal{C}^{up}(X)$ w.r.t. the set-inclusion $\subseteq$.*

A set system $\mathcal{C}$ has *finite descending chain property* (*DCC* or, is *Noetherian*) if there is no properly decreasing infinite chain $C_0 \supset C_1 \supset \cdots \supset C_i \supset \cdots$ ($i \in \mathbb{N}$) in $\mathcal{C}$. Clearly, any family $\mathcal{C}$ consisting of finite subsets of E always satisfies DCC.

Definition 5. Let $\mathcal{C}$ be a subfamily of $\mathcal{F}$. Then, a triple $(\mathcal{F}, \mathcal{C}, \mathbb{C})$ is a *closure-based closed set system* if it satisfies the following (A1)–(A2):

(A1) $\mathbb{C}$ is a closure mapping for $\mathcal{F}$.
(A2) $\mathcal{C} = \{ \mathbb{C}(X) \mid X \in \mathcal{F} \}$.

If $\mathcal{C}$ is closed under intersection $\cap$, then we define the *merge* operator on $\mathcal{C}$ by $\textsc{Merge}_{\subseteq} X = \cap \mathcal{C}^{up}(X)$ for any $X \in \mathcal{F}$.

DEFINITION 6. Let $\mathcal{C}$ be a subfamily of $\mathcal{F}$. Then, a triple $(\mathcal{F}, \mathcal{C}, \mathbb{C})$ is a *intersection-based closed set system* if it satisfies the following (B1)–(B2):

(B1) $\mathcal{C}$ is closed under intersection.

(B2) $\mathbb{C}(X) = \text{MERGE}_{\subseteq}(X)$ holds.

THEOREM 2.1. *Let $(\mathcal{F}, E)$ be a Noetherian set system. For any triple $\mathcal{S} = (\mathcal{F}, \mathcal{C}, \mathbb{C})$, $\mathcal{S}$ is a closure-based closed set system if and only if $\mathcal{S}$ is an intersection-based closed set system.*

Proof. (Sketch) Suppose that $\mathcal{S} = (\mathcal{F}, \mathcal{C}, \mathbb{C})$ satisfies (A1) and (A2). For any $Z = \mathbb{C}(X) \cap C(Y)$ with $\mathbb{C}(X), \mathbb{C}(Y) \in \mathcal{C}$ and $\mathbb{C}(X) \neq \mathbb{C}(Y)$, $\mathbb{C}(Z)$ has to be Z from (C2) of the definition of closure mapping. Thus (B1) holds. Similarly, we can see that $\mathcal{C}$ satisfies (B2). Suppose that $\mathcal{S}$ satisfies (B1) and (B2). (C1), (C2) and (C3) hold from (B2). □

From the above theorem, we see that two notions coincide. Thus, we simply call $\mathcal{S} = (\mathcal{F}, \mathcal{C}, \mathbb{C})$ the closed set system in what follows. We can have the same result whether we start either from the closure mapping $\mathbb{C} : \mathcal{F} \to \mathcal{F}$ or from the class of closed sets $\mathcal{C} \subseteq \mathcal{F}$ depending on ease of the treatment.

2.4 Closed set mining problem. To define our data mining problem, we introduce a computational setting for closed set systems. Let $(\mathcal{F}, \mathcal{C}, \mathbb{C})$ be a closed set system. Then, a *membership oracle* for $\mathcal{F}$ is any computable mapping $\mathbb{M}_{\mathcal{F}} : 2^E \to \{0, 1\}$, where for any set $X \subseteq E$, $\mathbb{M}_{\mathcal{F}}(X)$ is 1 if $X \in \mathcal{F}$ and 0 otherwise. In what follows, $\mathcal{F}$ always contains $\emptyset$.

DEFINITION 7. A (computable) *closed set system* is $\mathcal{S} = ((\mathcal{F}, E), \leq_E, \mathbb{M}, \mathbb{C})$, where $(\mathcal{F}, E)$ is a set system over E, $(E, \leq)$ is a total ordering over E, $\mathbb{M} : 2^E \to \{0, 1\}$ is a membership oracle $\mathbb{M}_{\mathcal{F}}$ for $\mathcal{F}$, and $\mathbb{C} : \mathcal{F} \to \mathcal{F}$ is a closure operator for $\mathcal{F}$. $\mathcal{S}$ is said to be *efficient* if $\leq_E$ over E, $\mathbb{M}_{\mathcal{F}}$, and $\mathbb{C}$ for $\mathcal{F}$ are computable in time $h(n)$, $f(m, n)$, and $g(m, n)$, resp., for some polynomials $f(\cdot), g(\cdot, \cdot)$, and $h(\cdot)$ in the size n of a description of family $\mathcal{F}$ and the size m of a set given as an argument.

Now, we state our data mining problem, which is also called the closed set enumeration problem for $\mathcal{S}$.

DEFINITION 8. CLOSED SET MINING PROBLEM FOR CLOSED SET SYSTEM $\mathcal{S}$ (CSM($\mathcal{S}$)):
Given: a description of a (computable) closed set system $\mathcal{S} = ((\mathcal{F}, E), \leq_E, \mathbb{M}, \mathbb{C})$. Task: Generate all sets in $\mathcal{C} = \mathbb{C}(\mathcal{F})$ without duplicates.

Algorithm CloGenBHPW$(E, \mathbb{M}_{\mathcal{F}}, \mathbb{C})$:
Input: a ground set E, a membership oracle $\mathbb{M}_{\mathcal{F}}$ and a closure operator $\mathbb{C}$ for $\mathcal{F}$;
Output: The family $\mathcal{C} = \mathbb{C}(\mathcal{F})$ of $\mathbb{C}$-closed sets;

```
1: QUEUE := {ℂ(∅)}; //the root element
2: while QUEUE ≠ do
3:    D := Pop(QUEUE);
4:    Print D;    //Do not move it from here!
5:    for each e ∈ E \ C do
6:       if (C ∪ {e} ∉ F) then
7:          Skip e and continue the for-loop;
          //test for strongly accessibility
8:       D := ℂ(C ∪ {e});    //D is always closed.
9:       if (D ∉ TABLE) then
10:         Push(QUEUE, D);
11:         TABLE := TABLE ∪ {D};
12:      end if
13:   end for
14: end while
```

Figure 1: An output-polynomial time algorithm for enumerating $\mathbb{C}(\mathcal{F})$; $QUEUE$ is either a queue or a stack with operations $Push$ and Pop depending on whether the search strategy is BFS or DFS.

In the remainder of this paper, we design algorithms for efficiently solving the closed set mining problem in the sense of enumeration algorithms [3, 8]. Let N be the total input size and M the number of all solutions. An enumeration algorithm $\mathcal{A}$ is of *output-polynomial time*, if $\mathcal{A}$ finds all solutions $S \in \mathcal{S}$ in total polynomial time both in N and M. Also $\mathcal{A}$ is of *polynomial delay*, if the *delay*, which is the maximum computation time between two consecutive outputs, is bounded by a polynomial in N alone. It is obvious that if $\mathcal{A}$ is of polynomial delay, then so is of output-polynomial.

3 Boley *et al.*'s Polynomial-Delay Algorithm

In this section, we review a polynomial-delay algorithm for closed sets in a strongly accessible set system, presented by Boley, Horváth, Poigné, Wrobel [9].

The key of their algorithm is the notion of inductive generator [9]. A *generator* for a closed set $D \in \mathcal{C}$ is any subset $G \subseteq D$ ($G \in \mathcal{F}$) such that $\mathbb{C}(G) = D$. A generator G for $D \in \mathcal{C}$ is called *inductive* if $G = C \cup \{e\}$ for some $C \in \mathcal{D}$ and $e \in D \setminus C$ such that $C \cup \{e\} \in \mathcal{F}$. The pair (C, e) is called an *inductive generator pair* for D. Boley *et al.* [9] showed the existence of an inductive generator for each closed set, which is implicit in a number of CIM algorithms based on bottom-up search.

In Figure 1, we give a description of their algorithm,

Algorithm GetTailElement$(Z, \mathbb{M}_{\mathcal{F}})$:
Input: a set $Z \in \mathcal{F}$ $(Z \subseteq E)$;
Output: the maximum $e \in Z$ with $Z \setminus \{e\} \in \mathcal{F}$;

```
1: S := Z;
2: while (S ≠ ∅) do
3:    e := max S;
4:    if (Z \ {e} ∈ F) then return e;
5:    S := S \ {e};
6: end while
7: //never reach this line
```

Figure 2: An algorithm for finding the tail element

here called CloGenBHPW, which is essentially same to one in [9]. This algorithm traverses the lattice-like structure induced in $\subseteq$ in breadth-first manner by using inductive generators. To avoid duplicates, it stores discovered closed sets so far in a lookup table $TABLE$. They showed that the algorithm CloGenBHPW runs in polynomial-delay, while it is not of polynomial-space in the worst case since the number of solutions in $TABLE$ can be exponential.

THEOREM 3.1. (BOLEY *et al.* [9]) *Let $N = |E|$ be the domain size and $M = |\mathcal{C}|$ be the output size. Then, the algorithm* CloGenBHPW *in Fig. 1 computes all closed sets in $\mathcal{C} = \mathbb{C}(\mathcal{F})$ without duplicates in $O(NM)$ oracle calls and space $O(NM)$,*

4 Poly-Delay and Poly-Space Enumeration of Closed Sets

In this section, we present a polynomial-delay and polynomial-space algorithm CloGenDFS that enumerates all closed sets in $\mathcal{C} = \mathbb{C}(\mathcal{F})$ without duplicates for accessible set system $(\mathcal{F}, E)$.

4.1 Computational version of accessible systems. By definition, a set system $(\mathcal{F}, E)$ is accessible if for every $X \in \mathcal{F} \setminus \{\emptyset\}$, there is some $e \in X$ such that $X \setminus \{e\} \in \mathcal{F}$. Assuming that the closure operator $\mathbb{C}$ is efficiently computable, we can show that the actual *access element* e above can be efficiently computed. The following Lemma 4.1 properly improves the result of Boley *et al.* [9] by replacing strong accessibility with accessibility.

LEMMA 4.1. *Let $\mathcal{S} = ((\mathcal{F}, E), \leq_E, \mathbb{M}, \mathbb{C})$ be an efficient closed set system. If $\mathcal{S}$ is accessible, then for any non-empty set $Z \in \mathcal{F}$,* GetTailElement *in Fig. 2 finds the maximum $e \in Z$ such that $Z \setminus \{e\} \in \mathcal{F}$ in polynomial time in $|Z|$ and $|E|$, where Z uniquely determines e.*

Proof. Since $(\mathcal{F}, E)$ is an accessible, there exists some $e \in Z$ such that $Z \setminus \{e\} \in \mathcal{F}$. GetTailElement finds the largest of such e's, that is, $e_{\max} = \max\{ e \in Z \mid Z \setminus \{e\} \in \mathcal{F} \}$, in linear time in $|Z|$. This shows the lemma. □

Algorithm GetFirstParent$(D, \mathbb{M}_{\mathcal{F}}, \mathbb{C})$:
Input: non-bottom closed set $D \in \mathcal{C} \setminus \{\perp\}$, membership oracle $\mathbb{M}_{\mathcal{F}}$, and closure operator $\mathbb{C}$ for $\mathcal{F}$;

```
1: Z := D;    //the original input
2: while (Z ≠ ∅) do
3:    e := GetTailElement(Z, M_F);
4:    X := Z \ {e};    //X ∈ F from Lemma 4.1
5:    //invariant: C(Z) = D
6:    if (C(X) ≠ Z) then return (C(X), e);
7:    Z := X;
8: end while
9: //never reach this line
```

Figure 3: An algorithm for assigning the unique parent to each closed set D

4.2 Polynomial-delay algorithm. In this subsection, we show that we can assign the unique parent for each closed set. Thm. 4 in Boley *et al.* [9] says that in a strongly accessible set system $(\mathcal{F}, E)$, any closed set $D \in \mathcal{C}$ $(D \neq \perp)$ has an inductive generator $C \cup \{e\}$ such that $C \in \mathcal{C}$ and $\mathbb{C}(C \cup \{e\}) = D$ holds. In Figure 3, we show an algorithm GetFirstParent to assign the unique parent to a given closed set D. From Lemma 4.1, we show the following lemma, which strengthens the above mentioned Thm. 4 in [9].

THEOREM 4.1. (UNIQUE INDUCTIVE GENERATOR IN AN ACCESSIBLE SET SYSTEM) *Let $\mathcal{S} = ((\mathcal{F}, E), \leq_E, \mathbb{M}, \mathbb{C})$ be an efficient closed set system. Assume that $\mathcal{S}$ is accessible. Let $D \in \mathcal{C}$ $(D \neq \perp)$ be a non-bottom closed set. Then, (i)* GetFirstParent *in Fig. 3 computes an inductive generator pair (C, e) for D in $O(m(m(f(m,n) + h(n)) + g(m,n)))$ time in $m = |D|$ and $n = |E|$, and furthermore, (ii) C satisfies that $C \subset D$ and $|C| < |D|$, (iii) the pair (C, e) is uniquely determined by D.*

Proof. By induction on the stage $i \geq 1$, the i-th while-loop, we show that the invariant $\mathbb{C}(Z) = D$ holds at line 5. If $i = 1$ then trivially $Z = D$ holds. Otherwise, $i > 1$, and $\mathbb{C}(Z) = D$ holds at line 5 at stage $i - 1$ by induction hypothesis. If the condition at line 7 holds then the algorithm returns $(\mathbb{C}(X), e)$, and thus, we can show that $\mathbb{C}(X) \subset Z = D$ and $\mathbb{C}(X \cup \{e\}) = \mathbb{C}(Z) = D$ (*1). Otherwise, $\mathbb{C}(X) \subset Z = D$ and $Z := X$ shows the claim at stage i. We can further see that the value

(X, e) is always returned; Otherwise, finally it reaches $Z = \emptyset$ and this equals $\mathbb{C}(Z) = D$; Contradiction. □

Note that the space complexity of the algorithm is $O(m)$ other than the memory spent by the oracles.

DEFINITION 9. The *family tree* for $\mathcal{C} = \mathbb{C}(\mathcal{F})$ is a rooted directed graph $G_{\mathbb{C}} = (\mathcal{C}, \mathcal{E}, \bot)$, where $\mathcal{C}$ is the vertex set consisting of all closed sets, and $\mathcal{E} \subseteq \mathcal{C}^2$ is the set of directed edges, called *reverse edges*, such that for every C, D $(C \subseteq D)$, there is a directed edge from D to C, i.e., $(D, C) \in \mathcal{E}$, iff $(C, e) = \mathsf{GetFirstParent}(D, \mathbb{M}_{\mathcal{F}}, \mathbb{C})$ holds for some $e \in D$, and $\bot \in \mathcal{C}$ is the unique root node.

The crucial point is that we can assign the unique *parent* C with $(C, e) = \mathsf{GetFirstParent}(D) \in \mathcal{C}$ to each *child* $D \in \mathcal{C}$ over the search space $\mathcal{C}$ by using the procedure GetFirstParent. From this, we can build a tree-shaped search route for $\mathcal{C}$ in the sense of [8].

LEMMA 4.2. (THE FAMILY TREE FOR $\mathcal{C} = \mathbb{C}(\mathcal{F})$) *The family tree $G_{\mathbb{C}} = (\mathcal{C}, \mathcal{E}, \bot)$ for $(\mathcal{F}, E)$ satisfies the following conditions (i) $G_{\mathbb{C}}$ is connected. (ii) $G_{\mathbb{C}}$ is acyclic. (iii) From any non-bottom closed set $C \in \mathcal{C}$ $(C \neq \bot)$, there exists the unique path from C to the root $\bot$. In other words, $G_{\mathbb{C}}$ is a spanning tree over $\mathcal{C}$ rooted at $\mathbb{C}(\bot)$.*

The remaining task is to reverse the directions of edges in $G_{\mathbb{C}}$. It is not sufficient for us to use $D = \mathbb{C}(C \cup \{e\})$ alone since two distinct parents can generate the same child. To avoid this, we use the procedure GetFirstParent to ensure C is the youngest parent for D.

LEMMA 4.3. (ENUMERATION OF CHILDREN) *Let $G_{\mathbb{C}} = (\mathcal{C}, E, \bot)$ be the family tree for $\mathcal{C}$. For any closed set $C, D \in \mathcal{C}$ and any set $D \in \mathcal{F}$, D is a child of C iff the following (i) and (ii) holds for some $e \in E \setminus C$:*

(i) $D = \mathbb{C}(C \cup \{e\})$.

(ii) $(C, e) = \mathsf{GetFirstParent}(D, \mathbb{M}_{\mathcal{F}}, \mathbb{C})$.

In Figure 4, we show our polynomial-delay and polynomial-space algorithm for $\mathbb{C}(\mathcal{F})$. From Lemma 4.3, we have the following lemma.

LEMMA 4.4. *In the family tree $G_{\mathbb{C}} = (V, E, \bot)$ for $(\mathcal{F}, E)$, the algorithm* Expand *in Fig. 4, given any closed set $C \in V = \mathcal{C}$, computes all children of C in $O(f(m,n)m^2n + h(n)m^2n + g(m,n)mn)$ time per C, where $m = |C|$ and $n = |E|$.*

Proof. From the definition of the parent and children, the correctness of the algorithm is obvious. Since one iteration of algorithm Expand involves n closure mapping function evaluations and GetFirstParent, thus we have the announced time complexity. □

Algorithm CloGenDFS$(E, \mathbb{M}_{\mathcal{F}}, \mathbb{C})$:
Input: A ground set E, a membership oracle $\mathbb{M}_{\mathcal{F}}$ and a closure operator $\mathbb{C}$ for $\mathcal{F}$;
Output: Family $\mathcal{C} = \mathbb{C}(\mathcal{F})$ of $\mathbb{C}$-closed sets;
1: Expand$(\mathbb{C}(\emptyset), E, \mathbb{M}_{\mathcal{F}}, \mathbb{C})$;

Algorithm Expand$(C, E, \mathbb{M}_{\mathcal{F}}, \mathbb{C})$:
Input: $E, \mathbb{M}_{\mathcal{F}}, \mathbb{C}$ and a set C;
Output: All sets on the descendant of C in the family tree $G_{\mathbb{C}}$ for $\mathcal{C}$;
1: Print C;
2: **for each** $e \in E$ **do**
3: **if** $C \cup \{e\} \notin \mathcal{F}$ **then** skip e and **continue** the loop;
4: $D = \mathbb{C}(C \cup \{e\})$;
5: **if** (GetFirstParent$(D) = (C, e)$) **then**
6: Expand$(D, E, \mathbb{M}_{\mathcal{F}}, \mathbb{C})$;
7: **end if**
8: **end for**

Figure 4: A polynomial-delay and polynomial-space algorithm CloGenDFS for enumerating all closed sets in an accessible set system and its recursive subprocedure Expand

Combining above discussions, now we have the main result of this paper.

THEOREM 4.2. (POLY-DELAY AND POLY-SPACE ALGORITHM) *Let $(\mathcal{F}, E)$ be a finite accessible set system with polynomial-time computable total ordering $\leq$, membership oracle $\mathbb{M}_{\mathcal{F}}$, and closure operator $\mathbb{C}$ for $\mathcal{F}$. Then, the algorithm* CloGenDFS *in Fig. 4 computes all elements $C \in \mathcal{C} = \mathbb{C}(\mathcal{F})$ without duplicates in $O(f(m,n)m^2n + h(n)m^2n + g(m,n)mn)$ time and $O(m+n)$ space, where $m = |C|$ and $n = |E|$. The space complexity excludes the space for oracle computations.*

If the total order $\leq_E$ and the membership oracle $\mathbb{M}_{\mathcal{F}}$ are constant time computable for $\mathcal{F}$, then $\mathcal{C}$ can be enumerable in $O(m^2n + mn \cdot g(m,n)))$ time per closed set, where $g(m,n)$ is the time for computing the closure $\mathbb{C}(X)$ for each X. We have the following corollary.

COROLLARY 4.1. *For a finite accessible set system $(\mathcal{F}, E)$, the family $\mathcal{C} = \mathbb{C}(\mathcal{F})$ of all closed sets is polynomial-delay and polynomial-space enumerable.*

4.3 Extension to more general structures. Finally, we extend our framework for more general algebraic structure based on generalized notion of closed set systems. Let $(\mathcal{F}, E)$ be an accessible set system and $\sqsubseteq$

is a preorder, i.e., a reflexive and transitive relation, and $\sim$ is an equivalence class over $\mathcal{F}$.

A *generalized closure operator* for $\mathcal{S}$ is any mapping $\mathbb{G} : \mathcal{F} \to \mathcal{F}$ such that for any $X, Y \in \mathcal{F}$, (G1) Extensbility: $X \sqsubseteq \mathbb{G}(X)$ holds; (G2) Monotonicity: $X \sqsubseteq Y$ implies $\mathbb{G}(X) \sqsubseteq \mathbb{G}(Y)$ holds; (G3) Idempotence: $\mathbb{G}(\mathbb{G}(X)) = \mathbb{G}(X)$ holds; (G4) Equivalence: $X \sim Y$ implies $\mathbb{G}(X) = \mathbb{G}(Y)$ holds. A pattern $X \in \mathcal{F}$ is *closed* if $\mathbb{G}(X) = X$. Then, a *generalized closed set system* is a quadruple $\mathcal{S} = (\mathcal{F}, \mathcal{C}, \mathbb{G}, \sim)$, where $\mathcal{F}$ is a class of patterns, $\mathcal{C} = \mathbb{G}(\mathcal{F})$ is the class of all closed sets, $\mathbb{G}$ is a generalized closure operator, and $\sim$ is an equivalence class on $\mathcal{F}$ such that

As seen in Sec. 2.3, closed patterns are defined through g.l.b. $\sqcap$ w.r.t. $\subseteq$. For many pattern classes such as the classes we will study in Sec. 5.5, Sec. 5.6, and Sec. 5.7, the closed patterns are defined based on equivalence under some operations, e.g., *translation* (*displacement*) or *rotation*. Thus, it seems difficult to define a natural notion of $\sqcap$ over whole $\mathcal{F}$, while a natural closure operator $\mathbb{G}$ is rather easy to define.

Now, we will give a special form of generalized closure systems. We start with defining binary relations $\sqsubseteq$ and $\sim$ as follows. Let $\mathcal{D}$ be a class of functions from E to themselves satisfying the followings: (i) any $f \in \mathcal{D}$ is a bijection (one-to-one and onto) from E to E, (ii) $\mathcal{D}$ contains the identity id, and (iii) $\mathcal{D}$ contains the composition $f \circ g$ and the inverse f^{-1} for any $f, g \in \mathcal{D}$. For each set $S \subseteq E$, we extend $f \in \mathcal{D}$ by $f(S) = \{ f(x) \mid x \in S \}$. Any class $\mathcal{D} \in \mathcal{F}^{\mathcal{F}}$ defined in this way is called a *class of displacements*. Then, $\mathcal{D}$ is *rigid* if for any $f, g \in \mathcal{D}$, $f(x_0) = g(x_0)$ for some element $x_0 \in E$ implies $f = g$.

Given a rigid class $\mathcal{D}$ of displacements, we define the preorder $\sqsubseteq$ and an equivalence relation $\sim$ as follows: for any $X, Y \in \mathcal{F}$, $X \sqsubseteq Y$ iff $f(X) \subseteq Y$ for some $\exists f \in \mathcal{D}$, and $X \sim Y$ iff $f(X) = Y$ for some $\exists f \in \mathcal{D}$. Then, a *canonical form function* over $\mathcal{F}$ compatible to $\sim$ is a function that assigns to each set X the unique representative $cano(X) \in \mathcal{F}$ in the equivalence class $\mathcal{E}(X) = \{ Y \in \mathcal{F} \mid X \sim Y \}$. Clearly, $X \sim Y$ implies $cano(X) = cano(Y)$.

Definition 10. (closure-based generalized closed set system) Let $\mathcal{C}$ be a subfamily of $\mathcal{F}$. Then, a triple $(\mathcal{F}, \mathcal{C}, \mathbb{G})$ is a *closure-based generalized closed set system* if it satisfies the following (D1)–(D4):

(D1) There is a closure mapping $\mathbb{C}$ for $\mathcal{F}$ that satisfies conditions (C1)–(C3) in Sec. 2.3.

(D2) There is an equivalence relation $\sim$ over $\mathcal{F}$ induced in a rigid class $\mathcal{D}$ of displacements over E.

(D3) $\mathbb{G}(X) = cano(\mathbb{C}(X))$ for some canonical form function $cano$ over $\mathcal{F}$ compatible to $\sim$.

(D4) $\mathcal{C} = \{ \mathbb{G}(X) \mid X \in \mathcal{F} \}$.

As in Sec. 2.4, we define an efficient version of generalized closed set system $\mathcal{S} = ((\mathcal{F}, E), \leq_E, \mathbb{M}_{\mathcal{F}}, \mathbb{G})$ with efficient oracles for $\leq_E, \mathbb{M}_{\mathcal{F}}$ and $\mathbb{G}$.

By a similar construction for closed set systems, we can show analogue of Theorem 4.1 using a modification of the algorithms CloGenDFS in Sec. 4. Algorithms GetTailElement and CloGenDFS need not to be modified as in Sec. 4. We only modify the algorithm GetFirstParent in Fig. 3 by replacing line 6 of the original code with the next code:

```
6: if (ℂ(X) ≠ Z) then
7:    Let f : E → E be the unique f ∈ 𝒟 such
      that f(ℂ(X)) = cano(ℂ(X)) for a given rigid
      class 𝒟 of displacements;
8:    return (𝔾(X), f(e)) = (f(ℂ(X)), f(e));
9: end if
```

Figure 5: The modified code for line 6 in Algorithm GetFirstParent.

In the above, we see that if $\mathcal{D}$ is rigid then f is uniquely determined. Assume that $\mathcal{S}$ is accessible and $\mathcal{D}$ is rigid, Then, we can show that for any closed set $D \in \mathcal{C}$ $(D \neq \bot)$, if $(C, f(e)) = \mathsf{GetFirstParent}(D, \mathbb{M}, \mathbb{C})$ with modification in Fig. 5 then $C \in \mathcal{C}$ and $\mathbb{G}(C \cup \{f(e)\}) = D$. From this fact, we can assign the unique parent to each closed pattern in a generalized accessible system as well. Combining above discussions with the proofs for Theorem 4.2, related lemmas, and properties (C1)–(C3) of $\mathbb{C}$, we have the next theorem on CloGenDFS in Fig. 4.

Theorem 4.3. *For an efficient generalized closed set system $\mathcal{S} = ((\mathcal{F}, E), \leq_E, \mathbb{M}, \mathbb{G})$ with a Noetherian domain E, the family $\mathcal{C} = \mathbb{G}(\mathcal{F})$ of all closed sets in $\mathcal{F}$ is polynomial-delay and polynomial-space enumerable.*

5 Reformulating Closed Pattern Classes

Let $\mathcal{C} \subseteq \mathcal{F}$ be a family of (closed) sets over a set system $(\mathcal{F}, S)$. We say that a closure system $((\mathcal{F}, S), \leq_S, \mathbb{M}, \mathbb{C})$ *specifies* $\mathcal{C}$ if $\mathcal{C} = \mathbb{C}(\mathcal{F})$ holds.

5.1 Convex Hulls over a Set of 2D-Points. We define a set of convex hulls. Let $S = \{ p_i = (x_i, y_i) \mid i = 1, \ldots, n \} \subseteq \mathbb{R}^2$ be a set of points on the 2-D plane. Let $\mathcal{F} = 2^S$ be the family of all subsets of S. We define a mapping $\mathbb{C} : 2^S \to 2^S$ by: for any set $X \subseteq S$, $\mathbb{C}(X) = CH(X) \cap S \subseteq \mathbb{R}^2$, where $CH(X)$ is the *convex hull* of X, i.e., the smallest convex polygon containing all elements of X. We simply call $CH(X)$ a *convex hull over* S. Then, the problem is, given S, to enumerate

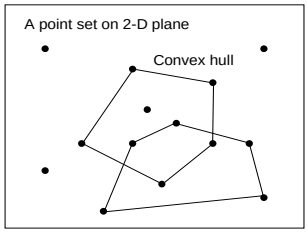

Figure 6: A point set on the 2-D plane and two convex hulls. The problem is to enumerate all possible convex hulls without duplicates.

all members of the set $\mathcal{C} \subseteq 2^S$ of all convex hull over S. $(2^S, S)$ is an accessible set system.

We can show that $\mathbb{C}$ satisfies the condition for closure mapping, and $\mathcal{C} = \{ \mathbb{C}(X) \mid X \in 2^S \}$ holds. Then, it is easy to see that $((2^S, S), \leq_S, \mathbb{M}, \mathbb{C})$ is an accessible set system with polynomial time oracles for $\leq_S$, $\mathbb{M}$, and $\mathbb{C}$.

THEOREM 5.1. (CONVEX HULLS OF 2D-POINTS) *The class $\mathcal{C}_{\rm ch} = \mathbb{C}(2^S)$ of all convex hulls over a given set $S \subseteq \mathbb{R}^2$ of 2D-points is specified by the efficient closure system $((2^E, E), \leq_E, \mathbb{M}, \mathbb{C})$ with $E = S$ as follows:*

- *$\leq_E$ is the lexicographic ordering over $\mathbb{R}^2$.*
- *$\mathbb{M}$ is the membership oracle for 2^E.*
- *For every point set X, $\mathbb{C}(X)$ is given by the convex hull $CH(X)$ of X.*

5.2 Closed itemsets. Let $E = \{1, \ldots, n\}$ be a set of items and a database is a collection $\mathcal{D} = \{t_1, \ldots, t_m\} \subseteq 2^E$. For an *itemset* $X \subseteq E$, its *location list* (*occurrence list*) in $\mathcal{D}$ is the set $\mathcal{D}(X) = \{t \in \mathcal{D} \mid X \subseteq t\}$. An itemset is defined to be closed in $\mathcal{D}$ if there is no superset $Y \supset X$ such that $L(X) = L(Y)$. We denote by $\mathcal{C}_{\rm it}$ the set of all closed itemsets in $\mathcal{D}$. Then, for any itemset $X \subseteq E$, we can define the closure for X by simply taking the intersection $\mathbb{C}_{\rm it}(X) = \bigcap L(X)$ of all transactions that X occurs. This captures $\mathcal{C}_{\rm it}$.

THEOREM 5.2. (CLOSED ITEMSETS) *The class $\mathcal{C}_{\rm it} = \mathbb{C}_{\rm it}(\mathcal{F}_{\rm it})$ of all closed itemsets in a database $\mathcal{D}$ is specified by the efficient closure system $((\mathcal{F}_{\rm it}, E), \leq_E, \mathbb{M}_{\rm it}, \mathbb{C}_{\rm it})$:*

- *$\leq_E$ is the total order over E.*
- *$\mathcal{F}_{\rm it} \subseteq 2^E$ is the set of all itemsets.*
- *$\mathbb{M}_{\rm it}$ is constant mapping that always returns 1.*
- *For every itemset $X \subseteq E$, $\mathbb{C}_{\rm it}(X) = \bigcap L(X)$.*

5.3 Maximal Bipartite Cliques. It is well-known that the class $\mathcal{C}_{\rm it}$ of closed itemsets in a given transaction database $\mathcal{D}$ coincides the maximal bi-cliques over a special bipartite graph [30]. However, below we explicitly give an efficient closure system $\mathcal{S}$ for maximal bi-cliques since $\mathcal{S}$ it is often useful for modification with constraints.

Let$G = (U, V, \mathcal{E})$ be a *bipartite graph* with mutually disjoint vertex sets $U = \{1, \ldots, m\}$ and $V = \{1, \ldots, n\}$, and an edge set $\mathcal{E} \subseteq U \times V$. A *bipartite clique* (*bi-clique*) in G is a pair (X, Y) of subsets X and Y of U and V, resp., such that $\forall x \in X, \forall y \in Y, (x, y) \in \mathcal{E}$. We encode a bi-clique (X, Y) by the edge set $H = \{ (x, y) \in \mathcal{E} \mid x \in X, y \in Y \} = X \times Y$ of its induced graph, and write $H = (X, Y)$. Then, the set-inclusion $\subseteq$ over $U \times V$ gives the ordering over bi-cliques.

Let $E = \mathcal{E}$ be the base set. A bi-clique is *maximal* if it is not properly contained by any other bi-clique, i.e., $X \subseteq X'$, $Y \subseteq Y'$, and $(X, Y) \neq (X', Y')$ hold. We define $\mathcal{C}_{\rm bc}$ to be the set of *all maximal bi-cliques* in G, and $\mathcal{F}_{\rm bi} = \{ (X, Y) \mid X \subseteq C, Y \subseteq D, (C, D) \in \mathcal{C}_{\rm bc} \}$ be the set of all subgraphs which are a subgraph of some bi-clique in $\mathcal{C}_{\rm bc}$. Then, we define the closure $\mathbb{C}(H) = MB(H) = (P, Q)$ by $Q = \{ y \in V \mid \forall x \in X,\ (x, y) \in \mathcal{E} \}$, and $P = \{ x \in U \mid \forall y \in Q,\ (x, y) \in \mathcal{E} \}$. Then, it is shown that $MB(H)$ is the unique maximal bi-clique containing H as subgraph.

THEOREM 5.3. (MAXIMAL BIPARTITE CLIQUES) *The class $\mathcal{C}_{\rm bi} = \mathbb{C}(\mathcal{F}_{\rm bi})$ of all maximal bipartite cliques over a given bipartite graph $G = (U, V, \mathcal{E})$ is specified by the efficient closure system $((\mathcal{F}_{\rm bi}, \mathcal{E}), \leq_E, \mathbb{M}, \mathbb{C})$ with $\mathcal{E} = U \times V$ as follows:*

- *$\leq_E$ is the lexicographic ordering over E.*
- *$\mathcal{F}_{\rm bi}$ is the set of all subgraphs of G which are a subgraph of a bi-clique.*
- *$\mathbb{M}$ is the membership oracle for $\mathcal{F}_{\rm bi}$*
- *For every $H \in \mathcal{F}_{\rm bi}$, $\mathbb{C}(H) = MB(H)$.*

5.4 (Unconnected) Closed Relational Graphs. We define the (non-connected) closed relational graph mining problem as follows [29]. Let us fix a set of vertices $V = \{1, \ldots, n\}$ $(n \geq 0)$. Let $E = V^2$ be the base set. A *relational graph* (a *graph*) over V is any subset $R \subseteq E = V^2$ of directed edges. We denote by $\mathcal{F}_{\rm rg}$ the class of all relational graphs on V. Then, the semantics of a graph $P \in \mathcal{F}_{\rm rg}$ in a *database* $\mathcal{D} = \{G_1, \ldots, G_m\} \subseteq \mathcal{F}_{\rm rg}$ is given by the *location list* $L(P) = \{G_i \in \mathcal{D} \mid P \subseteq G_i\}$. In Fig. 7, we show examples of relational graphs.

A graph X is *maximal* in $\mathcal{D}$ if there is no distinct pattern Y within $\mathcal{X}_{\rm rg}$ that is a proper superset of X

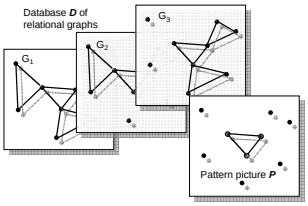

Figure 7: A database of relational graphs $\mathcal{D} = \{G_1, G_2, G_3\}$ (upper left) and a relational graph pattern P (right lower) over a domain V consisting of nine vertices.

having the same set of occurrences, i.e., $L(X) = L(Y)$. $\mathcal{C}_{\rm rg}$ denotes the *set of all maximal relational graphs* in $\mathcal{D}$. Defining the closure of a graph $X \in \mathcal{F}_{\rm rg}$ by the *intersection graph* $\mathbb{C}_{\rm rg}(X) = \bigcap L(X)$, which is the unique maximal graph in the equivalence class of graphs having the same location list. Thus, $\mathcal{C}_{\rm rg} = \mathbb{C}(\mathcal{F}_{\rm rg})$ [29]. Note that even if X and all graphs in $\mathcal{D}$ are connected, $INT(X)$ is not necessarily connected [29].

THEOREM 5.4. (CLOSED RELATIONAL GRAPHS) *The class $\mathcal{C}_{\rm rg} = \mathbb{C}_{\rm rg}(\mathcal{F}_{\rm rg})$ of all maximal relational graph in an input database $\mathcal{D}$ over vertex set $V = \{1, \ldots, n\}$ is specified by the efficient closure system $((\mathcal{F}_{\rm rg}, E), \leq_E, \mathbb{M}_{\rm rg}, \mathbb{C}_{\rm rg})$ with $E = V^2$ as follows:*

- *$\leq_E$ is the lexicographic ordering over V^2.*
- *$\mathcal{F}_{\rm rg} \subseteq 2^E$ is the set of all relational graph over V.*
- *$\mathbb{M}_{\rm rg}$ is the membership oracle for $\mathcal{F}_{\rm rg}$.*
- *For every $X \in \mathcal{F}_{\rm rg}$, $\mathbb{C}_{\rm rg}(X) = \bigcap L(X)$.*

5.5 Closed Rigid Motifs with Wildcards in a Sequence. We study the closed pattern problem for a class of sequence patterns called *rigid motifs with wildcards*, which was introduced by Parida *et al.* [17] in 2000 and studied in [18]. The polynomial delay and space complexity for the class has been open for years and solved positively by [7] in 2005.

Let Σ be an alphabet and $\circ \notin \Sigma$ be a wildcard. Let us fix a finite string, called an *input sequence*, $S = S[1] \cdots S[n] \in \Sigma^*$ of length $n \geq 0$. A *rigid motif* or *motif* (RM) is a string $P = X[1] \cdots X[m] \in (\Sigma \cup \{\circ\})^*$ of length $m \leq n$ such that $X_1 \neq \circ$ and $X_m \neq \circ$. We denote by $\mathcal{P}$ the set of all rigid motifs over $\Sigma \cup \{\circ\}$.

Let $O = \{1, \ldots, n\}$ be the domain of *positions*. For motifs $P[1..m]$ and $Q[1..n]$, P *occurs* in Q at position $p \in O$ if either $P[i] = Q[p + i - 1]$ or $P[i] = \circ$ for every $i = 1, \ldots, m$. The *location list* for P in S is the set $L(P) = \{\ p \in O \mid P \text{ occurs at } p \text{ in } S\ \}$. If P occurs in Q then we write $P \sqsubseteq Q$, and say that Q is *more specific than* P. Then, P is a *maximal motif* in S if there exists no properly specific Q than P that has the same location list under displacement, i.e., there is no Q with $P \sqsubset Q$ and $L(P) = L(Q) + d$ for some $d \in \mathbb{Z}$, where $L(Q) + d = f_d(L(Q))$. We denote by $\mathcal{C}_{\rm rm}$ the *set of all maximal rigid motifs* in S.

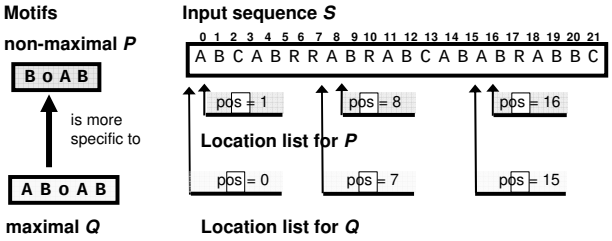

Figure 8: Examples of an input string S (right), a non-maximal motif P (left upper) and a maximal motif Q (left lower) with the location lists $L(P) = \{1, 8, 16\}$ and $L(Q) = \{0, 7, 15\} = L(P) - d$ with displacement $d = 1$.

Now, we give a generalized set system $\mathcal{S}_{\rm rm} = (\mathcal{F}_{\rm rm}, \sqsubseteq_{\rm rm}, \mathcal{C}_{\rm rm}, \mathbb{C}_{\rm rm})$ as follows. Let $E_{\rm rm} = \mathbb{Z} \times \Sigma$ be the domain. We encode a motif $P = P_1 \cdots P_m \in (\Sigma \cup \{\circ\})^*$ by a subset $[P] = \{\,(i, P_i) \mid i = 1, \ldots, m, P_i \in \Sigma\,\} \subseteq E_{\rm rm}$, which is exactly a relational representation of a motif P as a partial function $P : \mathbb{Z} \to \Sigma$. Clearly, $(i, P_i) \in [P]$ iff $P_i \in \Sigma$. Let $L([P]) = L(P)$.

EXAMPLE 1. Let $P = \texttt{B}\circ\texttt{AB}$ and $Q = \texttt{AB}\circ\texttt{AB}$ be a rigid motif with wildcard in Fig. 8. Then, P and Q are encoded by feasible sets $[P] = \{(1, \texttt{B}), (3, \texttt{A}), (4, \texttt{B})\}$ and $[Q] = \{(1, \texttt{A}), (2, \texttt{B}), (4, \texttt{A}), (5, \texttt{B})\} \subseteq \mathbb{Z} \times \Sigma$, resp. Note that $[P]$ has the *hole* at index 2. On the other hand, $\{(1, \texttt{A}), (1, \texttt{B}), (2, \texttt{B})\}$ is not a proper encoding since two distinct pairs $(1, \texttt{A}), (1, \texttt{B})$ have the same index.

We define the embedding $\sqsubseteq$ and the equivalence $\sim$ over $\mathcal{F}$ as follows. Let $\mathcal{D}_1 = \{\ f_d \mid d \in \mathbb{Z}\ \} \subseteq \mathbb{Z}^{\mathbb{Z}}$ be the class of *1-dim rigid displacements* on $\mathbb{Z}$, where $f_d(x) = x + d$. For each $X \in \mathcal{F}$, we define $f_d(X) = \{\ (f_d(i), a) \mid (i, a) \in X\ \}$. Now, the class of all encodings of rigid motifs is $\mathcal{F}_{\rm rm} = \{f_d([P]) \mid P \in \mathcal{P}, d \in \mathbb{Z}\} \subseteq 2^{E_{\rm rm}}$, which is closed under $\mathcal{D}_1$. For any $X \subseteq E_{\rm rm}$, $cano(X) = f_d(X)$ is the unique canonical form of X such that whose indices starts with 1, i.e., $\min f_d(X) = 1$ for some $d \in \mathbb{Z}$. For any motifs $X, Y \in \mathcal{F}_{\rm rm}$, $X \sqsubseteq_{\rm rm} Y$ iff $f_d(X) \subseteq Y$ and $X \sim Y$ iff $f_d(X) = Y$ for some $d \in \mathbb{Z}$. Then, we have the following characterization: for every $1 \leq d \leq n$, a rigid motif $P[1..m]$ occurs in $S[1..n]$ at position d iff $f_d([P]) \subseteq [S]$ iff $[P] \subseteq f_{-d}([S])$. As intended, we see that $(\mathcal{P}, \sqsubseteq)$ and $(\mathcal{F}_{\rm rm}, \sqsubseteq_{\rm rm})$ are ismorphic via $[\cdot]$.

Now, we give a closure mapping for $\mathcal{C}_{rm}$ as follows. Let $P \in \mathcal{P}$ be a rigid motif. Based on [7], we define the *merge* of motif P in S by:

$$\text{MERGE}([P]) = cano\left(\bigcap\{\ f_{-d}([S]) \mid d \in L(P)\ \}\right),$$

where $\cap$ simply denotes the intersection under the set-inclusion. We define the bottom by $\bot_{\rm rm} = \textsc{Merge}(\emptyset)$. Clearly, $\mathcal{F}_{\rm rm}$ forms an accessible set system.

LEMMA 5.1. *For any motif $P \in \mathcal{P}$ with $L(P) \neq \emptyset$, $[Q] = \textsc{Merge}([P])$ is the unique most specific motif within $\mathcal{C}_{\rm rm}$ such that $L(P) = L(Q) + d$ for some $d \in \mathbb{Z}$.*

THEOREM 5.5. (MAXIMAL RIGID MOTIFS) *The class $\mathcal{C}_{\rm rm} = \mathbb{C}_{\rm rm}(\mathcal{F}_{\rm rm})$ of all maximal rigid motifs with wild-cards in an input sequence S is specified by the efficient generalized closure system $((\mathcal{F}_{\rm rm}, E), \leq_E, \mathbb{M}_{\rm rm}, \mathbb{G}_{\rm rm})$ with $E = \mathbb{Z}\times\Sigma$ as follows:*

- *$\leq_E$ is the lexicographic ordering over E.*
- *$\mathcal{F}_{\rm rm} \subseteq 2^E$ is the set of encodings for motifs in $\mathcal{P}$.*
- *$\mathbb{M}_{\rm rm}$ is the membership oracle for $\mathcal{F}_{\rm rm}$, i.e., $\mathbb{M}_{\rm rm}(X) = 1$ iff $X \in \mathcal{F}_{\rm rm}$ for any $X \subseteq E$.*
- *For any $X \in \mathcal{F}_{\rm rm}$, $\mathbb{G}_{\rm rm}(X) = \textsc{Merge}(X)$.*

Proof. We show (D1)–(D4) hold. Since $\mathbb{C}_{\rm rm}([P]) = \textsc{Merge}([P])$ for any $P \in \mathcal{C}$, we see that (C1) and (C3) hold. We have (C2) since $X \sqsubseteq Y$ implies $L(X) \supseteq L(Y)$ and $\textsc{Merge}(X) \subseteq \textsc{Merge}(Y)$ iff $L(X) \supseteq L(Y)$ for any $X, Y \in \mathcal{F}$ as in [7]. (D2)–(D4) are easily shown. □

5.6 Closed Rigid Itemset Sequences. Parida *et al.* [17] also considered a closed pattern problem for the class of rigid motifs over a letter alphabet, where we call them rigid itemset sequences. They are the rigid version of *sequential patterns* by Srikant and Agrawal [21].

Let $\Sigma = \{1, \ldots, s\}$ $(m \geq 0)$ be an alphabet of items. An *input sequence* of length $n \geq 0$ is a sequence $S = (S_1, \ldots, S_n) \in (2^\Sigma)^*$, of n itemsets. A *itemset sequence pattern* (or *pattern*) of size $m \geq 0$ is a sequence $P = (P_1, \ldots, P_m) \in (2^\Sigma)^*$, of m itemsets such that $P_1 \neq \emptyset$ and $P_m \neq \emptyset$. We denote by $\mathcal{P}_{\rm sp}$ the set of all itemset sequence patterns. For $P, Q \in \mathcal{P}_{\rm sp}$, P *occurs* in Q at position p if $P[i] \subseteq Q[p + i - 1]$ for every $i = 1, \ldots, |P|$. Then, we write $P \sqsubseteq Q$, and say that Q is *more specific than* P. As in Sec. 5.5 for rigid motifs, we similarly define the *location list* $L(P)$, the *equivalence* $\sim$, and the *canonical form* $cano(P)$ for P. Namely, P is *maximal* in S if there is no Q such that $P \sqsubset Q$ and $L(P) = L(Q) + d$ for some $d \in \mathbb{Z}$. We denote by $\mathcal{C}_{\rm rm}$ the *set of all maximal itemset sequence patterns* in S.

Then, we encode a sequential itemset pattern $P = (P_1, \ldots, P_m)$ by the set $[P]_{\rm sp} = \{(i, a) \mid i = 1, \ldots, m, a \in P_i\} \subseteq \mathbb{Z}\times\Sigma$. We denote by $\mathcal{F}_{\rm sp}$ the class of all encodings for $\mathcal{P}$. For the class of 1-dim displacements $\mathcal{D}_1$, we can show that for any $1 \leq d \leq n$, P occurs at d in S iff $[P]_{\rm sp} \subseteq f_{-d}([S]_{\rm sp})$.

EXAMPLE 2. Let $R = \texttt{[A][BC][A][][B]}$ be a sequential itemset pattern over $\Sigma = \{\texttt{A}, \texttt{B}, \texttt{C}\}$, where the indices are starting from 1 and $\texttt{[BC]}$ and $\texttt{[]}$ denote the sets $\{\texttt{B}, \texttt{C}\}$ and $\emptyset$, resp. Then, $[R] = \{(1, \texttt{A}), (2, \texttt{B}), (2, \texttt{C}), (3, \texttt{A}), (5, \texttt{B})\} \subseteq \mathbb{Z}\times\Sigma$. Note that this contains two distinct pairs at index 2, while it contains no pair at index 4.

As in Sec. 5.5, we define $\sqsubseteq$ and $\sim$ over $\mathcal{F}_{\rm sp}$, the location list $L(X)$, and the canonical form $cano(X)$ for encoding X. Similarly, we define the *merge* of $P \in \mathcal{P}$ in S by $\textsc{Merge}_{\rm sp}([P]) = cano\left(\bigcap\{\, f_d([S]) \mid d \in L(P)\,\}\right)$. For any $P \in \mathcal{P}$ such that $L(P) \neq \emptyset$, $Q = \textsc{Merge}_{\rm sp}(P)$ is the unique most specific pattern w.r.t. $\sqsubseteq$ within $\mathcal{C}_{\rm sp}$ such that $L(P) = L(Q) + d$ for some $d \in \mathbb{Z}$. Let $\bot = \textsc{Merge}(\emptyset)$. $\mathcal{F}_{\rm sp}$ forms an accessible set system.

THEOREM 5.6. (CLOSED RIGID ITEMSET SEQUENCES) *The class $\mathcal{C}_{\rm sp} = \mathbb{C}(\mathcal{F}_{\rm sp})$ of all maximal itemset sequence patterns in $S \in (2^\Sigma)^*$ is specified by the efficient generalized closure system $((\mathcal{F}_{\rm sp}, E), \leq_E, \mathbb{M}_{\rm sp}, \mathbb{G}_{\rm sp})$ with $E = \mathbb{Z}\times\Sigma$ as follows:*

- *$\leq_E$ is the lexicographic ordering over E.*
- *$\mathcal{F}_{\rm sp} \subseteq 2^E$ is the set of all encodings for $\mathcal{P}_{\rm sp}$.*
- *$\mathbb{M}_{\rm sp}$ is the membership oracle for $\mathcal{F}_{\rm sp}$.*
- *For any $X \in \mathcal{F}_{\rm sp}$, $\mathbb{G}_{\rm sp}(X) = \textsc{Merge}_{\rm sp}(Q)$.*

5.7 Closed Picture Patterns on 2-D plane. Below, we introduce the class of 2-D picture motifs under translation and rotation. For details of the proofs, please consult [6].

Let Σ be an alphabet of letters and $\circ \notin \Sigma$ be a wildcard. For $n \in \mathbb{N}$, we define $[n] = \{1, \ldots, n\}$. As a domain, we consider subspace of 2-D discrete space $\mathbb{Z}^2 = \mathbb{Z}\times\mathbb{Z}$. A *picture* is any mapping $P : [m]\times[m] \to \Sigma \cup \{\circ\}$ that assigns to each point $(x, y) \in dom(P) = [m]\times[m]$ in a rectangular domain a letter $P(x, y) \in \Sigma$. We denote by $\mathcal{P} = \mathcal{P}_{\rm pt}$ the class of all picture patterns.

We consider two types of (discrete) geometric transformations $f : \mathbb{Z}^2 \to \mathbb{Z}^2$. (i) if f is a *translation* with displacement $(p, q) \in \mathbb{Z}^2$ then $f(x, y) = (x + p, y + q)$. (ii) if f is a *rotation* with angle $\theta \in \{0\pi, \frac{1}{2}\pi, 1\pi, \frac{3}{2}\pi\}$ around $(0, 0)$ then $f(x, y) = (ax - by, bx + ay)$, where $a = \cos\theta$ and $b = \sin\theta$. $\mathcal{G}$ denotes the set of all geometric transformations generated by (i) and (ii) above. For a picture $P \in \mathcal{P}_{\rm pt}$, we extend $f \in \mathcal{G}$ by $f(P)(x, y) = P(f(x), f(y))$. It is known that any $f \in \mathcal{G}$ can be completely specified by the image $f(\mathbf{e})$ of a fixed unit vector $\mathbf{e}$ in $\mathbb{Z}^2$, e.g., $\mathbf{e}_1 = ((0, 0), (1, 0))$. We write $P \sim Q$ if $P = f(Q)$ for some $f \in \mathcal{G}$.

Let us fix an *input picture* $S : [n]\times[n] \to \Sigma \cup \{\circ\}$ of size $n\times n$. Let $P, Q \in \mathcal{P}$ be pictures in $\mathbb{Z}^2$. P *is included in* Q, denoted by $P \subseteq Q$, if if either $P(x, y) = Q(x, y)$ or $P(x, y) = \circ$ holds for every $(x, y) \in dom(P)$. P *occurs* in Q via transformation $f \in \mathcal{G}$, if $f(P) \subseteq Q$, where f is

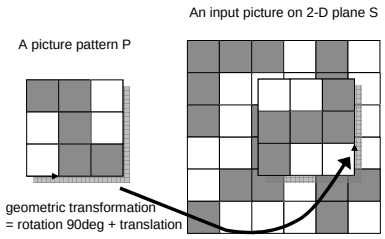

Figure 9: A pict pattern (left) and an input picture (right) with an occurrence of a pict pattern by geometric transformation obtained by a 90 degree rotation and some translation.

called a *occurrence* of P. The *location list* for P in S is the set $L(P) = \{\ f \in \mathcal{G} \mid P \text{ occurs in } S \text{ via } f\ \}$. If P occurs in Q by some f then we write $P \sqsubseteq Q$, and say that Q is *more specific than* P. Then, pattern $P \in \mathcal{P}_{\rm pt}$ is *closed* in S if there exists no properly specific Q than P that has the same location list to P, i.e., no Q such that $P \sqsubset Q$ and $L(P) = g(L(Q))$ for some $g \in \mathcal{G}$, where $g(F) = \{\ f \circ g \mid f \in F\ \}$ for $F \subseteq \mathcal{G}$. Let $\mathcal{C}_{\rm pt}$ be the set of all closed rigid motifs in an input picture S.

Now, we give a generalized set system $\mathcal{S}_{\rm pt} = (\mathcal{F}_{\rm pt}, \sqsubseteq_{\rm pt}, \mathcal{C}_{\rm pt}, \mathbb{C}_{\rm pt})$ as follows. Let $E = \mathbb{Z}^2 \times \Sigma$ be a domain. We encode a picture pattern $P : [m]\times[m] \to \Sigma \cup \{\circ\}$ by the set $[P] = \{\ (x, y, c) \mid (x, y) \in [m]\times[m], c = P(x, y) \in \Sigma\ \} \subseteq E$. For each $f \in \mathcal{G}$, we extend f by $f([P]) = \{\ (f(x, y), c) \mid (x, y, c) \in [P]\ \}$. Then, we define the class of all encodings closed under $\mathcal{G}$ by the set $\mathcal{F}_{\rm pt} = \{\ f([P]) \in 2^E \mid P \in \mathcal{P}, f \in \mathcal{G}\ \}$. By this encoding, we see that $P \preceq Q$ iff $[P] \subseteq [Q]$, and that $P \sqsubseteq Q$ iff $f([P]) \subseteq [Q]$ for some $f \in \mathcal{G}$. In what follows, we identify P and $[P]$ if no conflict occurs.

Example 3. In Fig. 9, we show a 3×3 picture pattern $P = [[\mathtt{W}, \mathtt{B}, \mathtt{B}], [\mathtt{W}, \mathtt{B}, \mathtt{W}], [\mathtt{B}, \mathtt{B}, \mathtt{W}]]$ over $\Sigma = \{\mathtt{W} = \textit{white}, \mathtt{B} = \textit{black}\}$ in vector-of-rows representation, where rows are ordered from the bottom to the top, and columns are from left to right. Then, P is encoded by the set

$$[P] = \left\{ \begin{array}{lll} (1,1,\mathtt{W}), & (2,1,\mathtt{B}), & (3,1,\mathtt{B}), \\ (1,2,\mathtt{W}), & (2,2,\mathtt{B}), & (3,2,\mathtt{W}), \\ (1,3,\mathtt{B}), & (2,3,\mathtt{B}), & (3,3,\mathtt{W}) \end{array} \right\}.$$

The *sequential code* for $X \in \mathcal{F}_{\rm pt}$, denoted by $code(Q)$, is the sequence of the triples in X obtained by scanning pixels of P, e.g., in the left-to-right and the bottom-to-top order. Using the lexicographic order $<_{\rm lex}$ on $\mathbb{Z}^2 \times \Sigma$, we define the *canonical form* of P by $cano(P) = \min_{<_{\rm lex}}\{\ code(Q) \mid P \sim Q, \min_x dom(Q) = \min_y dom(Q) = 1\}$, where $\min_x S = \min\{x \mid (x, y) \in S\}$ and $\min_y S = \min\{\ y \mid (x, y) \in S\ \}$ for finite $S \subseteq E$.

Clearly, $\mathcal{F}_{\rm pt}$ forms an accessible set system. Then, we define the merge $\textsc{Merge}_{\rm pt} : \mathcal{F} \to \mathcal{F}$ for pictures as follows. For any picture P, the *merge* $\textsc{Merge}_{\rm pt}(P)$ is defined by the intersection of inverse images by $L(P)$

$$\textsc{Merge}_{\rm pt}(P) = cano\left(\bigcap\{\ f^{-1}(S) \mid f \in L(P)\ \}\right).$$

Lemma 5.2. ([6]) *For any picture $P \in \mathcal{F}_{\rm pt}$ with $L(P) \neq \emptyset$, $Q = \textsc{Merge}_{\rm pt}(P)$ is the unique most specific pucture pattern within $\mathcal{C}_{\rm pt}$ such that $L(P) = g(L(Q))$ for some $g \in \mathcal{G}$. Moreover, $\mathcal{C}_{\rm pt} = \textsc{Merge}_{\rm pt}(\mathcal{F}_{\rm pt})$.*

Theorem 5.7. (Closed 2-D picture patterns) *The class $\mathcal{C}_{\rm pt} = \mathbb{C}(\mathcal{F}_{\rm pt})$ of all closed 2-D picture patterns in an input picture $S : [n]^2 \to \Sigma \cup \{\circ\}$ is specified by the efficient generalized closure system $((\mathcal{F}_{\rm pt}, E), \leq_E, \mathbb{M}_{\rm pt}, \mathbb{G}_{\rm pt})$ with $E = \mathbb{Z}^2 \times \Sigma$ as follows:*

- *$\leq_E$ is the lexicographic ordering over E.*
- *$\mathcal{F}_{\rm pt}$ is the set of all canonical encoding for picture patterns in $\mathcal{F}_{\rm pt}$.*
- *$\mathbb{M}_{\rm pt}$ is the membership oracle for $\mathcal{F}_{\rm pt}$*
- *For any $X \in \mathcal{F}_{\rm pt}$, $\mathbb{G}_{\rm pt}(X) = \textsc{Merge}_{\rm pt}(X)$.*

5.8 Summary of the results. Applying our main theorem, we have the following result.

Theorem 5.8. *The closed pattern mining problem for the following classes are polynomial-delay and polynomial space enumerable: relational graphs [29] without connectivity constraint, convex hulls [19] over a set of 2D-points (new result), maximal bi-cliques over a bipartite graph (known in [23]), maximal rigid sequence motifs with wildcards from an input sequence. (known in [7]), maximal rigid itemset sequences [17] (new result) picture patterns on 2-D plane (new, [6])*

6 Conclusion

This paper studied the closed set mining problem in an abstract framework, called accessible set systems, and show an efficient polynomial-delay and polynomial-space algorithm for the problem. This result improves the space complexity of the previous research [9] exponentially in the worst case. We also presented applications of the proposed method to a variety of closed pattern mining problems for semi-structured data.

Garriga, Khardon, and deRaedt [12] studied the polynomial-delay and polynomial-space algorithms in the context of closed pattern mining in First-order logic based on intersection of models and graphs. It is a future research to compare two approaches.

In this paper, we mainly studied a group of pattern classes called *rigid patterns* [3, 4, 6, 7, 12, 17, 18, 22], with closure mappings $\mathbb{C}$. On the other hand, although classes of *flexible patterns* [5, 28, 27] have no notion of

closure mappings $\mathbb{C}$, they still allow efficient depth-first search. Thus, it an interesting future problem to extend the approach of this paper to these classes.

Acknowledgment

The first author is grateful to Tamas Horvath for his valuable discussions on their polynomial-delay algorithm in [9] and closed pattern mining, to Roni Khardon for his indication of the general approach to closed pattern mining [12]. The authors also would like to thank Michele Sebag, Nicloas Spyratos, Shin-ichi Minato, Kouichi Hirata, Akihiro Yamamoto, Makoto Haraguchi, Tatsuya Asai, Tetsuji Kuboyama, Takuya Kida for their discussions, This research was partly supported by MEXT Grant-in-Aid for Scientific Research (A), 20240014, 2008, and the MEXT/JSPS Global COE Program, 2007–2011 at Hokkaido University.

References

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

[2] T. Asai, H. Arimura, T. Uno, S. Nakano, Discovering frequent substructures in large unordered trees *Proc. DS'03*, LNCS 2843, 47–61, 2003.

[3] H. Arimura, Efficient algorithms for mining frequent and closed patterns from semi-structured data (invited talk), *Proc. PAKDD'08, LNAI 5012*, 2–13, 2008.

[4] H. Arimura, T. Uno and S. Shimozono, Time and space efficient discovery of maximal geometric graphs, *Proc. DS'07*, LNAI 4755, 42–55, 2007.

[5] H. Arimura and T. Uno, Mining maximal flexible patterns in a sequence, *Proc. LLLL'07*, LNAI, 2008.

[6] H. Arimura and T. Uno, A Polynomial Space and Polynomial Delay Algorithm for Enumerating Maximal Two-Dimensional Patterns with Wildcards, Technical Report TCS-TR-A-06-19, DCS, Hokkaido U., 2006.

[7] H. Arimura and T. Uno, An efficient polynomial space and polynomial delay algorithm for enumeration of maximal motifs in a sequence, *J. Comb. Optim.*, 13, 243–262, 2006.

[8] D. Avis, K. Fukuda, Reverse search for enumeration, Discrete Applied Mathematics, 65(1–3), 21–46, 1996.

[9] M. Boley, T. Horváth, A. Poigné, and S. Wrobel, Efficient closed pattern mining in strongly accessible set systems (Extended Abstract), *Proc. PKDD 2007*, LNCS 4702, 382–389, 2007.

[10] M. Boley, T. Horváth, A. Poigné, and S. Wrobel, Efficient closed pattern mining in strongly accessible set systems, *Proc. MLG 2007*, 2007.

[11] M. de Brecht, M. Kobayashi, H. Tokunaga, A. Yamamoto, Inferability of Closed Set Systems from Positive Data, JSAI 2006, LNCS 4384, 265–275, 2007.

[12] G. C. Garriga, R. Khardon, L. De Raedt, On Mining Closed Sets in Multi-Relational Data, *Proc. IJCAI 2007*, 804–809, 2007.

[13] A. Inokuchi, T. Washio, H. Motoda, Complete mining of frequent patterns from graphs: mining graph data, *Machine Learning*, 50(3), 321–354, 2003.

[14] E. Boros, V. Gurvich, L. Khachiyan, K. Makino, On the Complexity of Generating Maximal Frequent and Minimal Infrequent Sets, *Proc. STACS'02*, LNCS 2285, 133–141, 2002.

[15] Tamas Horváth, Private communication, Oct. 2008.

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

[17] L. Parida, I. Rigoutsos, A. Floratos, D. Platt, and Y. Gao, *Proc. SODA'00*, 297–308, SIAM, 2000.

[18] N. Pisanti, *et al.*, A basis of tiling motifs for generating repeated patterns and its complexity for higher quorum, *Proc. MFCS'03*, LNCS 2747, 622–631, 2003.

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

[20] J. Ramon, S. Nijssen, General Graph Refinement with Polynomial Delay, *Proc. MLG'07*, 2007

[21] R. Srikant, R. Agrawal, Mining Sequential Patterns: Generalizations and Performance Improvements, *Proc. EDBT'96*, LNCS 1057, 3–17, 1996.

[22] A. Termier, M.-C. Rousset, M. Sebag, K. Ohara, T. Washio, H. Motoda, DryadeParent, An Efficient and Robust Closed Attribute Tree Mining Algorithm, *IEEE Trans. Knowl. Data Eng.*, 20(3), 300–320, 2008.

[23] S. Tsukiyama, M. Ide, H. Ariyoshi and I. Shirakawa, A new algorithm for generating all the maximum independent sets. *SIAMJ. Comp.*, 6, 505–517, 1977.

[24] J. D. Ullman, A. V. Aho and J. E. Hopcroft, The Design and Analysis of Computer Algorithms, Addison-Wesley, 1974.

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

[26] T. Uno, M. Kiyomi, H. Arimura, LCM ver. 2: Efficient Mining Algorithms for Frequent/Closed/Maximal Itemsets, *Proc. the ICDM Workshop FIMI'04*, 2004.

[27] J. Wang and J. Han, BIDE: efficient mining of frequent closed sequences, Proc. ICDE'04, 2004.

[28] X. Yan and J. Han, R. Afshar, CloSpan: mining closed sequential patterns in large databases, Proc. SDM 2003, SIAM, 2003.

[29] X. Yan, X. J. Zhou, J. Han, Mining closed relational graphs with connectivity constraints, *Proc. KDD'05*, 324–333, 2005.

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

[31] M. J. Zaki, C.-J. Hsiao, Efficient Algorithms for Mining Closed Itemsets and Their Lattice Structure, IEEE Trans. Knowl. Data Eng., 17(4), 462–478, 2005.

[32] M. J. Zaki. Efficiently mining frequent trees in a forest, *Proc. SIGKDD'02*, ACM, 2002.