TCS-TR-A-06-20

TCS Technical Report

Effcient Algorithms for Mining Maximal Flexible Patterns in Texts and Sequences

by

Hiroki Arimura and Takeaki Uno

Division of Computer Science

Report Series A

July 19, 2006

Hokkaido University
Graduate School of
Information Science and Technology

Email: Arimura@ist.hokudai.ac.jp

Phone: +81-011-706-7678
Fax: +81-011-706-7890

Effcient Algorithms for Mining Maximal Flexible Patterns in Texts and Sequences

Hiroki Arimura and Takeaki Uno

Graduate School of Information Science and Technology, Hokkaido University
Kita 14 Nishi 9, Sapporo 060-0814, Japan

National Institute of Informatics
2-1-2, Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan

Abstract

In this paper, we study the maximal pattern discovery problem in a given sequence for the class $\mathcal{ERP}$ of *flexible patterns* with applications to text mining, where a flexible pattern is a sequence of constant and wildcards for possibly empty strings such as AB*B*ABC, and also known as *erasing regular patterns*. We first discuss the framework of optimal pattern discovery for predictive mining and text classification, and then show its connection to maximal pattern discovery. Then, we introduce a new notion of maximality of patterns based on the position occurrences of patterns, called *position-maximality*. We present an efficient algorithm PosMaxFlexMotif that, given an input string of length n over an alphabet Σ, enumerates all maximal patterns of $\mathcal{ERP}$ without duplicates in $O(kmn^2)$ time per maximal pattern using $O(mn)$ space, where $m = |P|$ is the size of the pattern P to be enumerated, and $k = O(m)$ is the number of variables in P. This implies as corollary that the position-maximal enumeration problem for flexible patterns is output-polynomial time solvable. Then, we apply the above result to maximal pattern discovery in terms of the maximality based on document occurrence as a sound pruning technique.

Keywords: Data mining, Enumeration algorithms, Sequence databases, Maximal pattern, Motif Discovery

1. Introduction

By the rapid growth of the amount of human-readable electronic data on networks and storages, there are potential demands for the efficient computational methods to extract useful information and knowledge from massive amount of electronic data scattered over the network. Some prominent examples of such knowledge discovery tasks are: Automatic classification of natural language texts and web pages, characteristic and descriptive pattern discovery, prediction of trends from market data, detection of malicious activities from audit data, and clustering of documents. Pattern discovery is one of the fundamental technologies to find a class of patterns appearing in a

data set satisfying given constraints, and plays an important role in many knowledge discovery tasks.

In this paper, we consider the *maximal pattern discovery problem* [9, 10, 13, 14] in a set of sequences. A pattern is *maximal* if there is no properly *more specific* pattern w.r.t. some generalization ordering over the class of patterns that has the same occurrence, or equivalently, has the same frequency, in a given set of input sequences. Since the set of all maximal patterns are typically much smaller than the set of all patterns appearing in a data set while the formar contains the complete information of the latter, maximal pattern discovery has merits in efficiency and comprehensiveness. On the other hands, the computational complexity of maximal pattern discovery is higher than that of frequent pattern disocovery. For example, there are few classes of patterns for which the maximal pattern discovery problem have output-polynomial time complexity.

The class of patterns we consider is the class $\mathcal{ERP}$ of erasing regular patterns of Shinohara [12], which is also called flexible patterns in bioinformatics area [9]. A *flexible pattern* over an alphabet Σ is a sequence of constant letters in Σ and wildcards $*$ for possibly empty strings such as AB*B*ABC. From the view of polynomial time enumeration, the class $\mathcal{ERP}$ is a super class of the class of subsequence patterns [7] for which polynomial output maximal pattern enumeration algorithm is known [14]. However, there is no output-polynomial algorithm for the maximal pattern problem for $\mathcal{ERP}$. A potential problem for the class $\mathcal{ERP}$ of flexible patterns is, unlike classes of itemsets and rigid patterns [3, 9, 10], there are no unique maximal pattern in each equivalence class of patterns. To overcome this problem, we introduce a weaker notion of maximality for $\mathcal{ERP}$ than the original one, called *position-maximality*, by generalizing the notion of maximality implicitly used in [14] for maximal subsequence patterns. Then, we study the output-sensitive computational complexity of the maximal pattern enumeration problem in terms of the position-maximality.

As a main result of this paper, we present an efficient enumeration algorithm PosMaxFlexMotif that discovers all position-maximal patterns appearing in a given string without duplicates in $O(|\Sigma|kmn^2)$ time per maximal pattern using $O(mn)$ space, where $|\Sigma|$ is the size of an alphabet, $m = |P|$ is the size of the pattern P to be enumerated, $k = O(m)$ is its number of variables, and n is the total length of an input string. As a corollary, this enumeration algorithm achieves one of the most strict notion of output-polynomial time enumerabilities, known as *polynomial-space and polynomial-delay* enumerability. where the delay is the maximum time required to compute each pattern. This result generalizes the output-polynomial complexity of [14] for subsequence patterns to the class $\mathcal{ERP}$. The polynomial-space and delay property indicates that it can be used as a light-weight and high-throughput algorithm for pattern discovery.

We presented a heuristics algorithm DocMaxFlexMotif for enumerating document-maximal patterns in a collection of strings for the class $\mathcal{ERP}$ with non-trivial pruning strategies derived from the previous algorithm PosMaxFlexMotif. Finally, we

discuss the application of the maximal pattern discovery to the optimal pattern discovery problem in machine learning and knowledge discovery with application to text mining.

2. Preliminaries

Let A be an alphabet of symbols. We denote by A^* the set of all finite strings over A and define $\Sigma^+ = \Sigma^* - \{\varepsilon\}$. Let $s = a_1 \cdots a_n \in A^*$ be a string over A of length n. We denote $|s| = n$ the length of s and by ε the empty string. For any indices $1 \leq i \leq j \leq n$, we denote by $s[i] = a_i$ the i-th letter of s, and by $s[i..j]$ the substring $s[i..j] = a_i \cdots a_j$ that starts with i and ends with j. For a set $S \subseteq A^*$, we denote by $|S|$ the cardinality of S and by $||S|| = \sum_{s \in S} |s|$ be the total size of the strings in S.

For strings $u, v, w \in A^*$, we say that u, v, and w, respectively, are a *prefix*, a *substring*, and a *suffix* of a string $s = uvw$. Then, the substring v occurs in s at position $p = |u| + 1$. Equivalently, if $s = s[1] \cdots s[n]$, then v occurs in s at position i iff $v = s[i] \cdots s[i + |v| - 1]$. The *left position* and the *right position* of v corresponding to this occurrence of in t are i and $i + |v| - 1$, respectively. The *reversal* of a string $x = a_1 \cdots a_m$ is defined by $x^{\mathrm{R}} = a_n \cdots a_1$.

2.1. Text and Patterns

In this subsection, we introduce the class $\mathcal{ERP}$ of erasing regular patterns of Shinohara [12], also known as flexible patterns [9]. Let $\Sigma = \{a, b, c, \ldots\}$ be a finite alphabet of *constant symbols*. We assume a special symbol $* \notin \Sigma$ called a *variable* (or *variable-length don't cares*, *VLDC*), which represents arbitrary long possibly-empty finite string in Σ^*. Then, an *erasing regular pattern* (or *pattern*, for short) over Σ is a string $P \in (\Sigma \cup \{*\})^*$ consisting of constant symbols and variables. An erasing regular pattern P is said to be *in canonical form* if it is written as $P = w_0 * w_1 * \cdots * w_m$ for some integer $m \geq 0$ and some non-empty strings $w_0, w_1, \ldots, w_m \in \Sigma^+$. Each constant string w_i is called a *segment* of P. An erasing regular pattern *in canonical form* is also called as a *flexible pattern* or a *VLDC pattern* [9]. We denote by $\mathcal{ERP}$ the class of erasing regular patterns over Σ in canonical form (or equivalently flexible patterns). We note that $\Sigma^* \subseteq \mathcal{ERP}$.

Example 1 Let $\Sigma = \{A, B, C\}$. Then, $P_0 = AB * A * ABC$ is an erasing regular pattern in canonical form, i.e., a flexible pattern. On the other hand, $P_1 = *AC * BB * CB$ and $P_2 = A * *B$ are erasing regular patterns but not in canonical form since P_2 starts with a wildcard and P_3 has consecutive occurrences of wildcards $**$.

Let Σ be a fixed alphabet of constants. An *input string* is a constant string $T = a_1 \cdots a_n$ $(n \geq 0)$ over Σ. Let $P = w_0 * w_1 * \cdots * w_m \in (\Sigma \cup \{*\})^*$ be a pattern with $m \geq 0$ variables in canonical form. A *substitution* for P is any m-tuple $\theta = (u_1, \ldots, u_m) \in ((\Sigma \cup \{*\})^*)^m$ of strings. Then, we define the application of θ to P, denoted by $P\theta$, as the string $P\theta = w_0 u_1 w_1 u_2 \cdots u_m w_m \in (\Sigma \cup \{*\})^*$, where the i-th

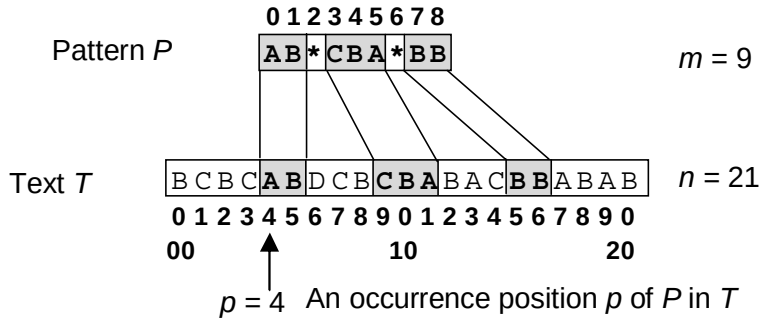

Figure 1: A pattern and its left and right positions in a text.

---

occurrence of variable $*$ is replaced with the i-th string u_i for every $i = 1, \ldots, m$. The string $P\theta$ is said to be an *instance* of P by substition θ.

For strings $P, Q \in (\Sigma \cup \{*\})^*$, P *occurs in* Q *at position* $1 \leq i \leq n$ if there exists some instance I of P that is a substring of Q starting at p, that is, $Q[i\,..\,i+|P\theta|-1] = P\theta$ for some substitution θ for P. Particularly, the positions i and $i + |P\theta| - 1$, respectively, correspond to the left and the right ends of the occurrence of instance $P\theta$, and are called the *left* and the *right* positions of P in T (See Fig. 1). We denote by $LO(P, T)$ and $RO(P, T)$ the set of all left-positions and the set of all right-positions of pattern P in text T, respectively. We refer to $LO(P, T)$ and $RO(P, T)$ as the *left-location list* and the *right-location list* of P in T, respectively. Recall that T^{R} denotes the reversal of T.

Lemma 1 *Let P be a pattern and T be a text. For any position $1 \leq p \leq n$, $p \in RO(P, T)$ if and only if $n - p + 1 \in LO(P^{\mathrm{R}}, T^{\mathrm{R}})$.*

By the symmetry between the left and the right location lists in Lemma 1, it is sufficient to consider only $LO(P, T)$ than $RO(P, T)$. Thus, we only consider the *location list* $O(P, T) = LO(P, T)$ in what follows.

Lemma 2 *The location list $LO(P, T)$ is computable in $O(mn)$ time, where $m = |P|$ and $n = |T|$.*

We define a binary relation $\sqsubseteq$ over $\mathcal{P}$, called the *specifity relation*, as follows [2, 9, 12]. For patterns $P, Q \in ERP$, P is *more specific than* Q, denoted by $P \sqsubseteq Q$, iff P occurs in T at some position $1 \leq p \leq n - 1$. If $P \sqsubseteq Q$ and $Q \not\sqsubseteq P$ hold, then we say that P is properly *more specific than* Q and denote $P \sqsubset Q$. We note that if $P \sqsubseteq Q$ and $Q \sqsubset P$ hold then $P = Q$ holds.

Lemma 3 *Let T be any text. Then, $(\mathcal{ERP}, \sqsubseteq)$ is a partial ordering with the smallest element ε.*

In machine learning area, the learning problem for the class formal languages defined by $\mathcal{ERP}$ has been extensively studied [12]. For a erasing regular pattern $P \in \mathcal{ERP}$, the *language defined by* P is the set $Lang_\Sigma(P) = \{ s \in \Sigma^* : P \sqsubseteq s \} \subseteq \Sigma^*$. It is easy to see that for any regular pattern $P \in (\Sigma \cup \{*\})^*$, there exists some $Q \in \mathcal{ERP}$ in canonical form such that $Lang_\Sigma(P) = Lang_\Sigma(Q)$.

Example 2 For patterns $P_3 = A*BA*B$ and $P_4 = BAB*BA*AB*B$, we can see that $P_3 \sqsubseteq P_4$ as follows. P_3 has an instance $P_4\theta = AB*BA*AB$ that is a substring of P_4 by the substitution $\theta = (B*, *A)$. From this, $P_3 \sqsubseteq P_4$ holds.

2.2. Maximal patterns

Let T be a fixed text of length $n \geq 0$. The *frequency* of a pattern P in T is $|O_T(P)|$. A *minimum support threshold* is a nonnegative integer $0 \leq \sigma \leq n$. A pattern P is *σ-frequent* in T if it has the frequency no less than σ in T, i.e., $|O_T(P)| \geq \sigma$.

Definition 1 A pattern P is *maximal* in T if there is no proper specialization Q of P that has the same location list, i.e., $P \sqsubset Q$ and $O_T(P) = O_T(Q)$.

We see that a pattern $P \in \mathcal{ERP}$ is maximal iff P is a maximal element w.r.t. $\sqsubseteq$ in the equivalence class $[P]_{\equiv_T} = \{ Q \in \mathcal{ERP} : P \equiv_T Q \}$ under the equivalence relation $\equiv_T$ defined by $P \equiv_T Q \Leftrightarrow O_T(P) = O_T(Q)$.

Lemma 4 *The maximal patterns in each equivalence class $[P]_{\equiv_T}$ is not unique in general.*

We denote by $\mathcal{F}_\sigma, \mathcal{M}$, and $\mathcal{M}_\theta = \mathcal{F}_\sigma \cap \mathcal{M}$ the classes of the σ-frequent patterns, the maximal patterns, and the maximal σ-frequent patterns in T. It is easy see that the number of frequent flexible patterns in an input text S is exponential in the total length n of T in the worst case.

Lemma 5 *There is an infinite sequence $(T_i)_{i\geq 0}$ of texts such that the number of maximal flexible patterns in T_i is exponential in $n = |T_i|$, i.e., $|\mathcal{M}_\sigma| = 2^{\Omega(n)}$.*

Now, we state our data mining problem considered in this paper as follows.

Maximal Flexible Pattern Enumeration Problem:
Input: An alphabet Σ, a text T of length n, and a minimum support threshold σ.
Task: To generate all maximal frequent flexible patterns in $\mathcal{M}_\sigma \subseteq \mathcal{ERP}$ in T without duplicates.

It is easy to see that $\mathcal{M}_\theta = \mathcal{F}_\sigma \cap \mathcal{M}$. Therefore, we will first present an efficient enumeration algorithm for $\mathcal{M} = \mathcal{M}_1$, and then extend it for $\mathcal{M}_\sigma$ for every $\sigma \geq 1$.

3. Motivated Applications of Maximal Pattern Discovery

In this section, we discuss potential applications of maximal pattern discovery considered in this paper to predictive mining and classification.

3.1. Predictive Mining and Text Classification

First, we introduce a practical model of machine learning in a noizy environment, known as *robust training* or *agnostic learning* according to, e.g.,[5, 6]. Suppose we are given a finite collection $\mathcal{S} = \{ x_i : i = 1, \ldots, m \} \subseteq \Sigma^*$ of strings, called a *sample*, and a binary labeling function $F : \mathcal{S} \to \{0, 1\}$, called an *objective function*. Each string $s \in \mathcal{S}$ is called a *document* and the value of the funcion $F(s) \in \{0, 1\}$ indicates whether the document is, e.g., interesting or not. Let $\mathcal{P}$ be a class of *classification rules* or *patterns*, where each pattern $P \in \mathcal{P}$ represents a binary function $P : S \to \{0, 1\}$. In our case, for any document $s \in \mathcal{S}$, we define $P(s) = 1$ if P matches s and $P(s) = 0$ otherwise. For a predicate $\pi(x)$ on a string x, $[\pi(x)] \in \{0, 1\}$ is the indicator function that returns 1 or 0 depending on the truth value of $\pi(x)$. Now, we state our pattern discovery problem.

Empirical Error Minimization Problem: ([5, 6])
Input: A sample $\mathcal{S}$ and an objective function $F : \mathcal{S} \to \{0, 1\}$.
Task: To find an optimal pattern $P \in \mathcal{P}$ that minimizes the empirical error

$$\textsc{Err}_{\mathcal{S},F}(P) = \sum_{x \in \mathcal{S}} [P(x) \neq F(x)].$$

among all patterns in $\mathcal{P}$.

In learning theory, it is known that any algorithm that efficiently solves the above empirical error minimization problem can approximate a target concept within a given concept class under arbitrary unknown probability distributions, and thus can work with noisy environments [11].

3.2. Optimal Pattern Discovery

The above framework can be extended for more general classes of score functions [4]. An *impurity function* is any real-valued function $\psi : [0, 1] \to$ Real such that (i) it takes the maximum value $\psi(1/2)$, (ii) the minimum value $\psi(0) = \psi(1) = 0$, and (iii) $\psi(x)$ is convex, i.e., $\psi(\frac{1}{2}(x + y)) \geq \frac{1}{2}(\psi(x) + \psi(y))$ for every $x, y \in [0, 1]$.

- The information entropy: $\psi_1(x) = -x \log x - (1 - x) \log(1 - x)$.
- The Gini index: $\psi_2(x) = 2x(1 - x)$.

Let N be the number of all examples and M be the number of positive examples in $\mathcal{S}$ with respect to a given an objective function F. Given objective function F, and pattern P, let S_1 and S_0 be the set of the matched examples and the unmatched

examples, where $S_\alpha = \{ s \in S : P(s) = \alpha \}$ for every $\alpha \in \{0,1\}$. The *contingency table* for P is a 4-tuple (M_1, M_0, N_1, N_0), where M_α and N_α are the number of all positive examples and the number of all examples in S_α for every $\alpha \in \{0,1\}$. Then, our goal is to find such rules that decreases the impurities or the ambiguities, $\psi(\frac{M_1}{N_1})$ and $\psi(\frac{M_0}{N_0})$, for S_1 and S_0. Now, we describe our data mining problem, the *optimal pattern discovery problem*, which is parameterized by an impurity function ψ, as follows (See, e.g., Devroy *et al.* [4]).

ψ-Optimal Pattern Discovery Problem:
Input: A sample $\mathcal{S}$ and an objective function $F : \mathcal{S} \to \{0,1\}$.
Task: To find an optimal pattern $P \in \mathcal{P}$ that minimizes the cost

$$\mathrm{G}^{\psi}_{\mathcal{S},F}(P) = N_1 \cdot \psi(\frac{M_1}{N_1}) + N_0 \cdot \psi(\frac{M_0}{N_0}),$$

among all patterns in $\mathcal{P}$, where (M_1, M_0, N_1, N_0) is the contingency table defined by $\mathcal{S}$, F, and the pattern P.

3.3. A Heuristic Algorithm for Optimal Pattern Discovery

In spite of the practical importance of optimal pattern problem, unfortunately, the above optimization problems are known to be computationally hard even for most classes of simple patterns, e.g., half-spaces in $\mathbf{R}^d$ or Boolean conjunctions in $\mathbf{B}^d$ [5]. In particular for the class $\mathcal{ERP}$ of flexible patterns, Miyano, Shinohara, and Shinohara [8] showed that the consistency problem for $\mathcal{ERP}$ is NP-complete. Furthermore, Shimozono, Arimura, and Arikawa [11] showed that the empirical error minimization problem for $\mathcal{ERP}$ is even hard to approximation within arbitrary small ratio.[1] Therefore, a class of straightfoward generate-and-test algorithms, which enumerates frequent patterns with an adequate threshold, are used to solve the optimized pattern discovery problem in practice. However, a potential problem with this approach is that there are a huge number of frequent patterns.

In Fig. 2, we show a heuristic algorithm FindOptimal for discovering top-K optimal patterns in terms of the score function $\mathrm{G}^{\psi}_{\mathcal{S},F}(P)$ based on a generate-and-test strategy using maximal patterns in $\mathcal{M}$ instead of frequent patterns in $\mathcal{F}$. If we set $K = 1$ then the algorithm solves the above ψ-optimal pattern problem. Then, the following theorem gives a justification of such an approach.

Theorem 6 *Let $\mathcal{S} \subseteq \Sigma^*$ and $F : \mathcal{S} \to \{0,1\}$. Let $\mathcal{P} = \mathcal{ERP}$ and $\mathcal{M} \subseteq \mathcal{P}$ be the class of maximal patterns in $\mathcal{S}$. Even if we restrict the class of patterns to $\mathcal{M}$ in $\mathcal{S}$, then this does not lose the optimility of the answers for the empirical error minimization problem and the ψ-optimal pattern discovery problem. That is, $\min\{ \mathrm{Err}_{\mathcal{S},F}(P) : P \in \mathcal{M} \} = \min\{ \mathrm{Err}_{\mathcal{S},F}(P) : P \in \mathcal{P} \}$ and $\min\{ \mathrm{G}^{\psi}_{\mathcal{S},F}(P) : P \in \mathcal{M} \} = \min\{ \mathrm{G}^{\psi}_{\mathcal{S},F}(P) : P \in \mathcal{P} \}$ hold.*

[1] Although the original approximation hardness result in [11] has been shown for the class with proximity constraints, we can obtain the same result for $\mathcal{ERP}$ without proximity constraints from its proof.

Proof: For any pattern P, the cost $G_{\mathcal{S},F}(P)$ is uniquely determined by the contingency table (M_1, M_0, N_1, N_0) corresponding to a sample $\mathcal{S}$, an objective function F, and pattern P. For any patterns P, Q, if $LO(P, S) = LO(Q, S)$ then P and Q realizes the same classification function $P : \mathcal{S} \to \{0, 1\}$, and thus gives the same contingency table. Hence, the result follows. □

Algorithm FINDOPTMAX($\mathcal{S}, F$)
input: A sample $\mathcal{S}$, an objective function $F : \mathcal{S} \to \{0, 1\}$, and an integer $K \geq 1$.
output: The top-K optimal patterns $P_1, \ldots, P_K$ minimizing the cost $G_{\mathcal{S},F}(P)$.
1 Let $\mathcal{Q} := \emptyset$ be an empty priority queue with the cost as the key.
2 **foreach** maximal pattern $P \in \mathcal{M}$ in $\mathcal{S}$ **do begin:**
3 Let $LO(P, \mathcal{S})$ be the location list of P;
4 Compute the contingency table $\tau = (M_1, M_0, N_1, N_0)$ from $LO(P, \mathcal{S})$;
5 $\mathcal{Q} := \mathcal{Q} \cup \{(P, G^{\psi}_{\mathcal{S},F}(P))\}$;
6 **end**
7 **output** the top K patterns P in $\mathcal{Q}$ in terms of $G_{\mathcal{S},F}(P)$;

Figure 2: An algorithm for the optimal pattern discovery via maximal pattern enumeration.

From Theorem 6, we know that the algorithm FINDOPTMAX correctly finds top-K optimal patterns in $\mathcal{S}$ minimizing the cost $G^{\psi}_{\mathcal{S},F}(P)$. The efficiency of this heuristics algorithm heavily depends on that of enumeration of maximal patterns in $\mathcal{M}$ at Line 2. Therefore, our goal in the remainder of this paper is to develop a memory and time efficient enumeration algorithm for maximal patterns in $\mathcal{M}$ for the class $\mathcal{ERP}$. Since the number $|\mathcal{M}|$ of maximal patterns is exponential in the total input size, it is natural to consider output-sensitive algorithm for $\mathcal{M}$. In what follows, we refer to the maximum computation time of $\mathcal{A}$ between consecutive outputs as the *delay* of an enumeration algorithm $\mathcal{A}$.

4. Tree-shaped Search Route for Position-Maximal Flexible Patterns

In this section, we will present our algorithm for finding position-maximal patterns in a given input string for the class $\mathcal{ERP}$ of flexible patterns. First, we introduce a tree-shaped search route $\mathcal{T}$ for the space of all maximal patterns in $\mathcal{M}$. Then, in the next section, we give a memory efficient algorithm for enumerating all maximal patterns based on the depth-first search over $\mathcal{T}$. Out strategy is explained as follows: First we define a binary relation between maximal patterns, called the *parent function*, which indicates a reverse edge from a child to its parent. Next, we reverse the direction of the edges to obtain a spanning tree for $\mathcal{M}$.

We start with technical lemmas. Let $P, Q \in \mathcal{ERP}$ be patterns over Σ. Recall that Q is a specialization of P, denoted by $P \sqsubseteq Q$, iff $Q = \alpha(P\theta)\beta$ for some $\alpha, \beta \in (\Sigma \cup \{*\})^*$ and for some substitution θ for P. We distinguish two cases whether $\alpha = \varepsilon$.

Definition 2 A pattern Q is said to be a *prefix specialization* of another pattern P if P occurs in the initial part of Q, i.e., $Q = (P\theta)\beta$ for some string $\beta \in (\Sigma \cup \{*\})^*$ and for some substitution θ for P. If there is no such β and θ, Q is said to be a *non-prefix specialization* of P.

The following two lemmas are essential for flexible patterns.

Lemma 7 *Let $T \in \Sigma^*$ be an input string and $P, Q \in \mathcal{ERP}$ be flexible patterns. Suppose that $P \sqsubseteq Q$. Then, if Q is a prefix specialization of P then $O(P,T) \supseteq O(Q,T)$.*

Proof: Let T be an input string of length n. Let $p \in O(P,T) = LO(P,T)$ be any left-position of P in T. Then, it follows from the definition that If $p \in LO(P,T)$ then some substring H of T starting at position p is an instance of Q. On the other hand, since Q is a prefix specialization of P, some prefix H' of Q is an instance of P. Therefore, some prefix H'' of H, and thus a substring of T, is an instance of P. Since H' is a substring starting at p in T, the lemma is proved. □

Lemma 8 *Let $T \in \Sigma^*$ be an input string and $P, Q \in \mathcal{ERP}$ be flexible patterns. Suppose that $P \sqsubseteq Q$. Then, if Q is a non-prefix specialization of P then $O(P,T) \not\subseteq O(Q,T)$.*

Proof: Let $p_{\max} = \max LO(P,T)$ be the largest left-position of P in T. Since $LO(P,T) \neq \emptyset$ and T has finite length, there always exists such a largest left-position $p_{\max}$ in T. Now we assume to contradict that $O(P,T) \subseteq O(Q,T)$. Then, $p_{\max}$ is also a postition of Q in T. Since Q is a non-prefix specialization of P, P occurs in Q at some position $\delta > 1$. Thus, we know that $q = p_{\max} + \delta - 1$ is a position of P in T. If $\delta > 1$ then q is strictly larger than $p_{\max}$. This contradicts the assumption that $p_{\max}$ is the largest position of P in T, and thus we conclude that $O(P,T) \not\subseteq O(Q,T)$. Hence, the result is proved. □

Corollary 9 *Let $T \in \Sigma^*$ be an input string and $P, Q \in \mathcal{ERP}$ be flexible patterns. Suppose that $P \sqsubseteq Q$. Then, if Q is a non-prefix specialization of P then $O(P,T) \neq O(Q,T)$.*

We define the parent-child relationship between two flexible patterns in $\mathcal{ERP}$ as follows.

Definition 3 Let $Q = w_0 * w_1 * \cdots * w_m$ be a flexible pattern over Σ, where $m \geq 0$ and $w_1, \ldots, w_m \in \Sigma^+$. Then, we define the *parent* of Q, denoted by $\mathcal{P}(Q)$, is the pattern satisfying the the followings (i) or (ii):

(i) If $|w_0| \geq 2$, that is, $w_0 = au_0$ for some letter $a \in \Sigma$ and a non-empty string $u_0 \in \Sigma^+$, then then $\mathcal{P}(Q) = u_0 * w_1 * \cdots * w_m$.

(ii) If $|w_0| = 1$, that is, $w_0 = a$ for some letter $a \in \Sigma$ then $\mathcal{P}(Q) = w_1 * \cdots * w_m$.

In summary, the parent $\mathcal{P}(Q)$ is the flexible pattern obtained from Q by removing the first letter, and then remove the first variable $*$ at the starting position if it exists. The removal of the initial $*$ ensures the canonicity of the resulting pattern.

Lemma 10 *For any non-empty flexible pattern $Q \in \mathcal{ERP}$ in cannonical form, its parent $\mathcal{P}(Q)$ is always defined, unique, and a flexible pattern in canonical form, i.e., a member of $\mathcal{ERP}$.*

Let $n \geq 0$ be a positive integer. For a nonnegative integer $0 \leq k \leq n$ and a set $X \subseteq \{1, \ldots, n\}$, we define $X + k = \{\, x + k : x \in X \,\}$. For sets $X, Y \subseteq \{1, \ldots, n\}$, we define $X \bowtie_{\leq} Y = \{\, p \in X : p \leq q \text{ for some } q \in Y \,\}$. By definition, both of $X + k$ and $X \bowtie_{\leq} Y$ are subsets of X. Using these operators, we can describe the location lists of a composite pattern of the form wP or $w * P$, where $w \in \Sigma^+$ and $P \in \mathcal{ERP}$ as follows.

Lemma 11 *Let $T \in \Sigma^*$ be any input string, $w \in \Sigma^+$ be any non-empty constant string, and $P \in \mathcal{ERP}$ be any flexible pattern. Then, the following (a) and (b) hold:*

(a) $LO(wP) = LO(w, T) \cap (LO(P, T) - |w|)$.

(b) $LO(w * P) = LO(w, T) \bowtie_{\leq} (LO(P, T) - |w|)$.

Let $T \in \Sigma^*$ be any input string of length $n \geq 0$. A maximal pattern P is a *root pattern* in T if $O(P, T) = \{1, \ldots, |T|\}$. Now, we show the main result of this section.

Theorem 12 (reverse search property of $\mathcal{M}$) *Let $Q \in \mathcal{M}$ be a maximal pattern in T that is not a root pattern. Then, $\mathcal{P}(Q)$ is also a maximal pattern in T. That is, if $Q \in \mathcal{M}$ then $\mathcal{P}(Q) \in \mathcal{M}$ holds. Furthermore, $|\mathcal{P}(Q)| < |Q|$ holds.*

Proof: Let Q be a maximal pattern that is not a root pattern, and let $P = \mathcal{P}(Q)$ be the parent of Q. In what follows, for any pattern R, we write $LO(R)$ for $LO(R, T)$ by omitting T for simplicity. Suppose to contradict that P is not maximal in T. Then, there exists some proper specialization P' of P, i.e., $P \sqsubset P'$, such that $LO(P) = LO(P')$.

If $P \sqsubset P'$ then P occurs in P' at some position, say, $1 \leq p \leq |P'|$. There are two cases below.

(i) The case where $p \neq 1$: Then, P' is a non-prefix specialization of P. It immediately follows from Lemma 9 that $LO(P) \neq LO(P')$. This is a contradiction.

(ii) The case where $p = 1$: Then, P' is a prefix specialization of P. By the definition of the parent, there are the following cases for P and Q.

(ii.a) The case where $Q = aP$ for some letter $a \in \Sigma$: Let $Q' = aP' \in \mathcal{ERP}$. Since P' is a proper prefix specialization of P, Q' is also a proper specialization of Q, i.e.,

$Q \sqsubset Q'$. In this case, we have $LO(Q') = LO(aP') = LO(a) \cap (LO(P') - |a|)$ by Property (a) of Lemma 11. Since $LO(P') = LO(P)$ by the assumption, it is obvious that $LO(a) \cap (LO(P') - |a|) = LO(a) \cap (LO(P) - |a|)$. Again by applying Property (a) of Lemma 11 to the right hand side, we have $LO(a) \cap (LO(P) - |a|) = LO(aP)$. Thus, we have $Q \sqsubset Q'$ and $LO(Q') = LO(Q)$, which says that Q is not maximal in T. However, this contradicts the assumption.

(ii.b) The case where $Q = a * P$ for some letter $a \in \Sigma$: Let $Q' = a * P' \in \mathcal{ERP}$. Since P' is a proper specialization of P (in this case, P' is not necessarily prefix-specialization), we have $Q \sqsubset Q'$. Furthermore, since $LO(P') = LO(P)$ by assumption, we can also show that $LO(Q') = LO(Q)$ by applying Property (b) of Lemma 11 as in the proof for case (ii.a). Therefore, we see that Q is not maximal in T, and thus, the contradiction is derived.

By combining cases (i), (ii.a), and (ii.b) above, we conclude by contradiction that P is maximal in T. Furthermore, it is clear from the construction that P is strictly shorter than Q in length. Hence, the result is proved. □

Definition 4 A *search graph* for $\mathcal{M}$ w.r.t. $\mathcal{P}$ is a directed graph $\mathcal{T} = (\mathcal{M}, \mathcal{P}, \mathcal{I})$ with roots, where $\mathcal{M}$ is the set of nodes, i.e., the set of all maximal flexible patterns in T, $\mathcal{P}$ is the set of *reverse edge* such that $(P, Q) \in \mathcal{P}$ iff $P = \mathcal{P}(Q)$ holds, and $\mathcal{I} \subseteq \mathcal{M}$ is the set of root patterns in T.

Since each non-root node has the unique parent in $\mathcal{T}$ from Theorem 12, the search graph $\mathcal{T}$ is acturally a directed tree with reverse edges. Therefore, we have the following corollary.

Corollary 13 *Let T be any input string. Then, $\mathcal{T} = (\mathcal{M}, \mathcal{E}, \mathcal{I})$ is a spanning forest for $\mathcal{M}$ with the root set $\mathcal{I}$.*

5. An Algorithm for Position-Maximal Flexible Pattern Enumeration

In Fig. 3, we show a polynomial-space and polynomial-delay enumeration algorithm PosMaxFlexMotif for maximal flexible patterns. Given an input string T of length n, this algorithm enumerates all position-maximal patterns in T without duplicates in polynomial time per maximal pattern using polynomial space in the input size n using depth-first search over the search tree $\mathcal{T}$ over $\mathcal{M}$ based on Corollary 13,

Recall that the search tree $\mathcal{T}$ for $\mathcal{M}$ has reverse edges only, that is, each edge of $\mathcal{T}$ is directed from a child to its parent. Therefore, the first step is to compute all children, given a parent pattern $P \in \mathcal{M}$. This can be done as follows.

Lemma 14 *For any maximal flexible patterns $P, Q \in \mathcal{M}$, $P = \mathcal{P}(Q)$ if and only if there exists some constant letter $a \in \Sigma$ such that either (i) $Q = aP$ or (ii) $Q = a*P$ holds.*

Algorithm PosMaxFlexMotif(Σ, $\mathcal{S}$, σ)
input: An alphabet Σ, an input string $\mathcal{S}$,
minimum frequency threshold $0 \leq \sigma \leq |\mathcal{S}|$;
output: All maximal patterns in $\mathcal{M}$;
1 Let $\perp$ be the maximal pattern in T which equivalent to ε (the root pattern).
2 EnumMaximal(ε, σ);

Procedure EnumMaximal($P, LO(P), \sigma$)
input: A maximal pattern P and its left-location list $LO(P)$.
output: All maximal patterns that are descendants of P.
1 Compute $LO(P)$;
2 **if** $|LO(P)| = 0$ **then return**;
3 **if** P is not maximal in $\mathcal{S}$ **then return**;
4 Output P;
5 **foreach** $a \in \Sigma$ **do begin:**
6 EnumMaximal($aP, LO(aP), \sigma$);
7 EnumMaximal($a * P, LO(a * P), \sigma$);
7 **end**

Figure 3: An algorithm PosMaxFlexMotif for enumerating all maximal flexible patterns in an input sequence.

Furthermore, since $\mathcal{P}(Q)$ is defined also for non-maximal flexible patterns Q, we know that any flexible pattern can be obtained from finite applications of the operations in above Lemma 14 to the empty pattern ε. Then, Theorem 12 gives a sound pruning strategy that once an enumerated pattern P gets non-maximal then we can immediately prune all the descendants of P.

Secondly, we discuss how to efficiently test the maximality of a given pattern P. The refinement operator for $\mathcal{ERP}$ was introduced by Shinohara [12]. The following version is due to [2].

Definition 5 A basic refinement of a pattern P is any pattern Q obtained from P by applying one of the following operations (r1) and (r2):

(r1) Q is obtained by replacing some segment $w \in \Sigma^+$ in P with either aw, wa, $a*w$, $w*a$ for some $a \in \Sigma$.

(r2) Q is obtained by replacing a pair of consecutive segments $v * w \in \Sigma^+\{*\}\Sigma^+$ in P with vw.

For a pattern P, we define $\rho(P) \subseteq \mathcal{ERP}$ to be the set of all basic refinements of P.

Lemma 15 *A flexible pattern P is maximal in T if and only if there is no basic refinement $Q \in \rho(P)$ such that $LO(P,T) = LO(Q,T)$.*

Corollary 16 *The maximality of a flexible pattern P in an input string T is decidable in $O(|\Sigma|m^2n)$ time, where $m = |P|$ and $n = |T|$.*

On the shape of $\mathcal{T}$, we have the following lemma.

Lemma 17 *Let $P \in \mathcal{M}$ be any maximal pattern in $\mathcal{T}$ and $m = |P|$. Then,*

(i) The depth of P in $\mathcal{T}$ (the length of the unique path from the root to P) is at most m.

(ii) The branching of P in $\mathcal{T}$ (the number of the children for P) is at most $O(|\Sigma|m)$.

By combining the above lemmas, we have the main result of this paper. This says that our algorithm PosMaxFlexMotif is a memory and time efficient algorithm for discovering maximal flexible patterns.

Theorem 18 *Let Σ be an alphabet and $T \in \Sigma^*$ be an input string of length $n \geq 0$. Then, the algorithm* PosMaxFlexMotif *in Fig. 3 enumerates all position-maximal patterns P in T without duplicates in $O(|\Sigma|kmn^2)$ delay per maximal pattern using $O(mn)$ space, where $m = |P|$ and $k = O(m)$ are the size and the number of variables of the pattern P to be enumerated.*

Corollary 19 *The maximal pattern enumeration problem for the class $\mathcal{ERP}$ of flexible patterns (or erasing regular patterns) w.r.t. position-maximality is solvable in polynomial-space and polynomial-delay in the total input size.*

6. A Practical Algorithm for Discovering document-maximal flexible patterns in a set of input strings

In this section, we consider the maximality based on the document occurrences and present a heuristic algorithm for solving the document-maximal pattern enumeration problem for $\mathcal{ERP}$ using the algorithm in the previous section as a pruning technique.

Let Σ be a fixed alphabet of constants. An *input string set* is a collection of constant strings over Σ

$$\mathcal{S} = \{s_1, \ldots, s_d\} \subseteq \Sigma^*,$$

where each member $s_i \in \Sigma^*$ is called a *document* of S $(i = 1, \ldots, d)$. For a flexible pattern $P \in \mathcal{ERP}$, the *document-location list* of P is defined by the set $DO(P, \mathcal{S}) = \{\, 1 \leq i \leq d : P \sqsubseteq s_i \,\}$ of the indices of the documents in which P occurs. The *document frequency* of P in $\mathcal{S}$ is defined by $|DO(P, \mathcal{S})|$.

For $0 \leq \sigma \leq |\mathcal{S}|$, a pattern P is *document σ-frequent* if $|DO(P, \mathcal{S})| \geq \sigma$. A pattern P is *document-maximal* in $\mathcal{S}$ if there is no proper specialization Q of P that has the same document-location list in $\mathcal{S}$, i.e., $P \sqsubset Q$ and $DO(P, \mathcal{S}) = DO(Q, \mathcal{S})$. The following lemma justifies the pruning strategy at Line 2 of Algorithm DocMaxFlexMotif in Fig. 4.

Algorithm DocMaxFlexMotif(Σ, $\mathcal{S}$, σ)
input: An alphabet Σ, an input set $\mathcal{S} = \{s_1, \ldots, s_k\} \subseteq \Sigma^*$, $0 \leq \sigma \leq |\mathcal{S}|$;
output: All document-maximal patterns in $\mathcal{DM}$;
1 Let $\perp$ be the maximal pattern in T which equivalent to ε (the root pattern).
2 Let $S = s_1 \# \cdots \# s_k$ be an input string ($\# \notin \Sigma$).
3 DocEnumMaximal($\perp, LO(\perp, S), \sigma$);

Procedure DocEnumMaximal($P, LO(P), \sigma$)
input: A maximal pattern P and its left-location list $LO(P, S)$.
output: All maximal patterns that are descendants of P.
1 Compute $DO(P, \mathcal{S})$ from $LO(P, S)$;
2 **if** $|DO(P, \mathcal{S})| < \sigma$ **then return**;
3 **if** P is not position-maximal in $\mathcal{S}$ w.r.t. $LO(P, S)$ **then return**;
4 **if** P is document-maximal in $\mathcal{S}$ **then output** P;
5 **foreach** $a \in \Sigma$ **do begin:**
6 DocEnumMaximal($aP, LO(aP, S), \sigma$);
7 DocEnumMaximal($a*P, LO(a*P, S), \sigma$);
7 **end**

Figure 4: An algorithm PosMaxFlexMotif for enumerating all maximal flexible patterns in an input sequence.

Lemma 20 (pruning by monotonicity) *If $P \sqsubseteq Q$ then $DO(P, \mathcal{S}) \supseteq DO(Q, \mathcal{S})$.*

Let $\mathcal{S} = \{s_1, \ldots, s_k\} \subseteq \Sigma^*$ be an input document set and let $\# \notin \Sigma$) be a new delimitor symbol. Then, we define an input string $S = s_1 \# \cdots \# s_k$ obtained from $\mathcal{S}$ by concatenating all documents by the delimitor $\#$. The following lemma ensure the soundness of the pruning strategy at Line 3 of Algorithm DocMaxFlexMotif in Fig. 4.

Lemma 21 (pruning by position-maximality) *Let P be any flexible pattern over Σ. If P is document-maximal in $\mathcal{S}$ then P is also a position-maximal in S.*

Based on the above lemmas, we show the algorithm DocMaxFlexMotif for enumerating all document-maximal frequent flexible patterns in a set of strings in Fig. 4. Unfortunately, this algorithm is not shown to be of output-polynomial time.

Theorem 22 *Let Σ be an alphabet and $T \in \Sigma^*$ be an input string of length $n \geq 0$. Then, the algorithm* DocMaxFlexMotif *in Fig. 4 enumerates all document-maximal patterns P in T without duplicates using $O(mn)$ space, where $m = |P|$ is the size of the pattern P to be enumerated and $k = O(m)$ is the number of variables of P.*

7. Conclusion

In this paper, we consider the maximal pattern discovery problem for the class $\mathcal{ERP}$ of flexible patterns [9], which is also known as erasing regular patterns in machine learning. The motivation of this study is applications to the optimal pattern discovery problem in machine learning and knowledge discovery. As a main result we present a polynomial-space and polynomial-delay algorithm for enumerating all maximal patterns appearing in a given string without duplicates in terms of postion-maximality defined through the equivalence relation between the location lists. As another application, we presented a heuristics algorithm for enumerating document-maximal patterns in a collection of strings for the class $\mathcal{ERP}$ with non-trivial pruning strategies.

References

[1] D. Avis and K. Fukuda, Reverse Search for Enumeration, *Discrete Applied Mathematics,* Vol. 65, 21–46, 1996.

[2] H. Arimura, R. Fujino, T. Shinohara, Protein motif discovery from positive examples by minimal multiple generalization over regular patterns, Proc. GIW'94, 39-48, Dec. 1994.

[3] H. Arimura, T. Uno, A polynomial space and polynomial delay algorithm for enumeration of maximal motifs in a sequence, Proc. ISAAC'05, LNCS 3827, Springer, Dec. 2005.

[4] L. Devroy, L. Gyrfi and G. Lugosi, A Probabilistic Theory of pattern Recognition, Springer Verlag, 1996.

[5] M. J. Kearns, R. E. Shapire, L. M. Sellie, Toward efficient agnostic learning, Machine Learning, 17(2–3), 115–141, 1994.

[6] W. Maass, Efficient agnostic PAC-learning with simple hypothesis, In Proc. COLT94, 67–75, 1994.

[7] H. Mannila, H. Toivonen, A. I. Verkamo, Discovery of frequent episodes in event sequences, Data Min. Knowl. Discov., 1(3), 259–289, 1997.

[8] S. Miyano, A. Shinohara, T. Shinohara, Polynomial-time learning of elementary formal systems, New Generation Comput. 18(3): 217–242, 2000.

[9] L. Parida, I. Rigoutsos, *et al.*, Pattern discovery on character sets and real-valued data: linear-bound on irredandant motifs and efficient polynomial time algorithms, In *Proc. SODA'00*, 2000.

[10] N. Pisanti, M. Crochemore, R. Gross, M.-F. Sagot, A basis of tiling motifs for generating repeated patterns and its complexity of higher quorum, In *Proc. MFCS'03*, 2003.

[11] S. Shimozono, H. Arimura, S. Arikawa, Efficient discovery of optimal word-association patterns in large text databases, New Generation Comput. 18(1), 49–60, 2000.

[12] T. Shinohara, Polynomial time inference of extended regular pattern Languages. *Proc. RIMS Symp. on Software Sci. & Eng.*, 115–127, 1982.

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

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