General Algorithms for Mining Closed Flexible Patterns under Various Equivalence Relations

Tomohiro I, Yuki Enokuma, Hideo Bannai, and Masayuki Takeda

Department of Informatics, Kyushu University, Fukuoka 819-0395, Japan
{tomohiro.i, bannai, takeda}@inf.kyushu-u.ac.jp
yuki.enokuma@i.kyushu-u.ac.jp

Abstract. We address the closed pattern discovery problem in sequential databases for the class of *flexible* patterns. We propose two techniques of coarsening existing equivalence relations on the set of patterns to obtain new equivalence relations. Our new algorithm GenCloFlex is a generalization of MaxFlex proposed by Arimura and Uno (2007) that was designed for a particular equivalence relation. GenCloFlex can cope with existing, as well as new equivalence relations, and we investigate the computational complexities of the algorithm for respective equivalence relations. Then, we present an improved algorithm GenCloFlex+ based on new pruning techniques, which improve the delay time per output for some of the equivalence relations. By computational experiments on synthetic data, we show that most of the redundancies in the mined patterns are removed using the proposed equivalence relations.

1 Introduction

Discovering frequent patterns in sequence databases has great importance in a wide-range of areas, including analysis of customer purchasing histories, Web click streams, DNA/RNA sequences, natural language texts, and so on. Recent decades have seen the series of studies; Agrawal and Srikant [1] was one of the pioneering works on sequential pattern mining, and many studies followed [3, 11, 14].

In practical applications of pattern mining, a typical tradeoff to be considered is: On one hand, we would like to consider for the mining task, a rich set of patterns and a relatively low minimum support threshold so that we may discover interesting, possibly subtle information buried in the data. On the other hand, by choosing such a search space, a mining algorithm may give us a tremendous number of patterns as output, which will definitely be a bottle neck when the results are examined by domain experts. To deal with this problem, an important technique in reducing the number of patterns output without sacrificing their diversity, is to introduce an appropriate equivalence relation $\equiv$ on the pattern set Π, and to output only *closed* patterns, where a pattern P is closed if it is maximal in the equivalence class $[P]_{\equiv}$ to which P belongs under $\equiv$. This problem is referred to as *closed pattern discovery* and has been studied extensively [2, 5–8, 10, 12, 13, 15, 16, 19–21].

In this paper, we consider the closed pattern discovery problem for the class of flexible patterns. The main features of our work are: (1) We focus on the class of *flexible* patterns. (2) We employ occurrence-based equivalence relations. (3) We propose two techniques of coarsening existing equivalence relations. (4) We develop a general algorithm GenCloFlex to handle several equivalence relations, and an improved algorithm GenCloFlex+. The algorithms have polynomial delay time and space guarantees.

(1) Mining flexible patterns: A *flexible* pattern is of the form $\star w_1 \star \cdots \star w_k \star$ where $w_1, \ldots, w_k$ $(k \geq 1)$ are constant strings and $\star$ is a gap symbol that can match any string of any length. Most studies to date target the class of *subsequence* patterns [1, 3, 5–16, 20, 21]. A subsequence pattern is a special case of flexible patterns, having the form $\star a_1 \star \cdots \star a_k \star$ $(k \geq 1)$ where each a_i $(1 \leq i \leq k)$ must be a single character. Thus, flexible patterns are more descriptive and enable us to capture some features that may not be discovered by subsequence patterns. For example, suppose we obtained $\star$l $\star$ o $\star$ v $\star$ e$\star$ as an output of frequent subsequence pattern mining. Since the lengths of gaps between each character is not considered, the pattern does not distinguish its occurrences in texts "love" and "low velocity", and we cannot know from the pattern alone, whether the phrase "love" is actually frequent or not. An output of frequent flexible pattern mining may give us the phrase "$\star$love$\star$" or "$\star$lo $\star$ ve$\star$" in which case this information would be apparent. Thus mining flexible patterns would be appreciated especially for languages which do not have an explicit delimiter between words such as Japanese and Chinese. Also, the information of consecutive characters in a pattern connected without gaps is very important for bio-sequences [18]. To the best of our knowledge, mining of closed flexible patterns with this definition has only been considered in [2]. A different version of closed flexible patterns is proposed in [19], but the definitions are significantly different and incompatible with ours.

(2) Occurrence-based equivalence relations: The definition of *closed* patterns depends on which equivalence relation to use, that is, which patterns we regard as the *same*. An equivalence relation is *finer* if less patterns are considered to be the same: i.e. more attention is paid to differences. An equivalence relation is *coarser* if more patterns are considered to be the same: i.e. less attention is paid to differences.

Most of the existing research on closed pattern mining traditionally use the equivalence relation on Π which is based on the *document occurrence*. Namely, two patterns are equivalent if the sets of strings in sequential database S containing occurrences of the patterns are identical. If a string T in database S contains an occurrence of P, we regard it as just one occurrence even if it contains two or more occurrences of P. For example, consider the occurrences of patterns $P_1 = \star$a $\star$ b $\star$ cd$\star$ and $P_2 = \star$a $\star$ cd$\star$ in string $T =$ .....a...b..cd...a...cd...., where "." denotes any symbol other than a, b, c. We note that P_1 is a super-pattern of P_2 and therefore every occurrence of P_1 implies an occurrence of P_2, independently of strings in sequential database S. Suppose that every other string in S containing P_2 has an occurrence of P_1. Then the document-occurrence-based

equivalence relation regards P_1 and P_2 equivalent. In this case, however, note that the rightmost occurrence of P_2 is not accompanied by an occurrence of P_1. In other words, if we consider the minimal occurrence intervals of the respective patterns, P_1 occurs twice while P_2 occurs only once within T. In some applications, we would like to distinguish between patterns which have different occurrences in such a way. In this paper we pay attention to respective occurrences of patterns, and use *occurrence-based* equivalence relations.

Arimura and Uno [2] defined their equivalence relation $\overset{\mathsf{B}}{\equiv}_S$ on pattern set Π based on the equality on the sets of beginning positions of pattern occurrences. For example, consider the occurrences of $P = \star\mathsf{a}\star\mathsf{b}\star$ in $T_1 = ...\mathsf{a.b}...\mathsf{a}...\mathsf{b..a}....$ The occurrence positions of P are the two occurrence positions of "a", excluding the rightmost one. However, we have four occurrence positions of P in $T_2 = ...\mathsf{a}...\mathsf{a}...\mathsf{a}...\mathsf{b}...$ while we have only one occurrence position of P in $T_3 = ...\mathsf{a}...\mathsf{b}...\mathsf{b}...\mathsf{b}....$ Such a non-symmetric feature is usually not desirable.

Mannila *et al.* [9] defined their equivalence relation $\overset{\mathsf{M}}{\equiv}_S$ on the subsequence pattern set based on the minimal intervals within which patterns occur. In the previous example, the pattern P has only one occurrence in respective strings. In this paper we consider the closed pattern discovery problem mainly under $\overset{\mathsf{M}}{\equiv}_S$ extended to the flexible pattern set and its coarsened variants $\overset{\mathsf{MX}}{\equiv}_S$ and $\overset{\mathsf{MXG}}{\equiv}_S$.

It should be emphasized that occurrence-based equivalence relations such as $\overset{\mathsf{B}}{\equiv}_S$ and $\overset{\mathsf{M}}{\equiv}_S$ are finer than the document-occurrence-based equivalence relation. This means that using such equivalence relations may increase the number of mined closed patterns. Thus it is important to coarsen the equivalence relations.

On the other hand, the goodness of an equivalence relation may vary depending on the nature of the data, the application domain and the goal of pattern mining. For this reason, it is desirable to develop a general, efficient closed pattern mining algorithm for various equivalence relations.

(3) Definition of support: Note that the choice of equivalence relation does not imply a particular definition for the frequency, or *support*, of a pattern. There are several definitions of support of a pattern P in a sequential database S. One definition is the so-called *document frequency*, which is the number of strings in S that contain at least one occurrence of P. A lot of work on closed pattern mining employ this definition. Another definition is the sum of the numbers of minimal intervals in respective strings in S that contain at least one occurrence of P. This definition has been used, for example, in [9] and [22]. Yet another definition can be found in [9], which is the number of windows of a given width in respective strings in S that contain at least one occurrence of P.

Throughout this paper we assume the document frequency. However, our algorithms can be easily modified to cope with the other definitions above when the underlying equivalence relation is in the M family ($\overset{\mathsf{M}}{\equiv}_S$, $\overset{\mathsf{MX}}{\equiv}_S$ and $\overset{\mathsf{MXG}}{\equiv}_S$).

(4) Polynomial delay time and space: Even in closed frequent pattern mining, the size of the output can be exponentially large, and we cannot hope for an algorithm running in polynomial time with respect to the input size. On the other hand, an enumeration algorithm with *polynomial delay time*, is an

algorithm in which the time *between each consecutive output* is bounded by a polynomial with respect to the size of the input. Such characteristics can be very useful and important for mining algorithms, since it guarantees that the algorithm runs in polynomial time with respect to the size of the output. This means that the time complexity of the algorithm is small when the output size, i.e., the number of closed frequent patterns is small. Even when the output size is large, we can still expect that the next output can be received in a reasonable amount of time. Without this guarantee, we may find out – *after* waiting for a very long time – that there are no more frequent patterns to be discovered.

Space complexity of the mining algorithm is also clearly an important issue. Arimura and Uno [2] addressed the closed pattern discovery problem for the class of flexible patterns and presented the first algorithm MaxFlex with polynomial time delay and polynomial space. Our algorithms also achieve polynomial time delay and polynomial space.

For closed pattern mining under document-based equivalence relations, algorithms such as BIDE [16,17], proposed for subsequence patterns, seem to achieve polynomial space complexity. However, as far as we know, no time delay guarantees have been shown, which may be a consequence of the document-based equivalence relation.

Contributions of this paper: We reiterate the main contributions of this paper: For the frequent closed flexible pattern enumeration problem, we focus on the equivalence relation $\overset{\mathsf{M}}{\equiv}_S$ of [9], and extended it to the class of flexible patterns. We also propose two new equivalence relations by coarsening it. We show GenCloFlex, an algorithm which generalizes the algorithm MaxFlex [2], so that it can cope with existing, as well as new equivalence relations, and investigate its computational complexities for respective equivalence relations. Then we present an improved algorithm GenCloFlex+, based on new pruning techniques which improve the delay time per output for some of the equivalence relations. By computational experiments on synthetic data, we prove that the proposed equivalence relations drastically remove redundancies in the mined patterns.

2 Preliminaries

Let Σ be a non-empty, finite set of symbols. A *string* over Σ is a finite sequence of symbols from Σ. Let Σ^* denote the set of strings over Σ. Strings x, y and z are said to be a *prefix*, *substring* and *suffix* of string $w = xyz$. The *length* of a string w is the number of symbols in w and denoted by $|w|$. The string of length 0 is called the *empty string* and denoted by ε. Let $\Sigma^+ = \Sigma^* - \{\varepsilon\}$. The i-th symbol of a string w is denoted by $w[i]$ for $1 \leq i \leq |w|$. The substring of a string w that begins at position i and ends at position j is denoted by $w[i..j]$ for $1 \leq i \leq j \leq |w|$. That is, $w[i..j] = w[i] \cdots w[j]$. For convenience, let $w[i..j] = \varepsilon$ for $j < i$. The *reversal* w^{rev} of a string $w = w[1..n]$ is defined to be $w[n] \cdots w[1]$. For a finite set S of strings, let $\|S\|$ denote the total length of strings in S and let $S^{rev} = \{w^{rev} \mid w \in S\}$.

An *interval* is an ordered pair $[i, j]$ of integers with $i \leq j$ which represents the set of integers k with $i \leq k \leq j$. Let I be a set of intervals. Let $\mathsf{Beg}(I) = \{i \mid [i, j] \in I\}$ and $\mathsf{End}(I) = \{j \mid [i, j] \in I\}$, and let $\mathsf{Min}(I)$ denote the set of intervals in I which are minimal w.r.t. $\subseteq$. For any set I of intervals and for any integers h, k, let $I \oplus \langle h, k\rangle = \big\{[i + h, j + k] \;\big|\; [i, j] \in I\big\}$. Also, for any set J of integers and for any integer k, let $J \oplus k = \big\{j + k \;\big|\; j \in J\big\}$.

2.1 Equivalence relation

Let A be a set. A *binary relation* on A is a subset of $A \times A$. For binary relations R_1, R_2 on A, let $R_1R_2 = \{\langle a, c\rangle \mid \langle a, b\rangle \in R_1 \wedge \langle b, c\rangle \in R_2\}$. Let $I_A = \{\langle a, a\rangle \mid a \in A\}$. For a binary relation R on A, let $R^0 = I_A$ and $R^n = RR^{n-1}$ $(n > 0)$, and let $R^{-1} = \{\langle b, a\rangle \mid \langle a, b\rangle \in R\}$, $R^+ = \bigcup_{n=1}^{\infty} R^n$ and $R^* = \bigcup_{n=0}^{\infty} R^n$.

A binary relation R on A is said to be *reflexive* if $I_A \subseteq R$; *symmetric* if $R^{-1} \subseteq R$; and *transitive* if $R^+ \subseteq R$. An *equivalence relation* on A is a binary relation on A which is reflexive, symmetric and transitive. For a binary relation R, we often write aRb when $\langle a, b\rangle \in R$.

Let $\equiv$ be an equivalence relation on A. The *equivalence class* of an element x of A under $\equiv$ is $\{y \in A \mid x \equiv y\}$ and denoted by $[x]_{\equiv}$. An equivalence relation $\equiv$ on A is said to be *finer* than another equivalence relation $\equiv'$ on A if $\equiv \subseteq \equiv'$. For any set $\mathcal{R} = \{\equiv_i \mid i \in \Lambda\}$ of equivalence relations on A, let $\wedge\mathcal{R} = \bigcap_{i\in\Lambda} \equiv_i$ and $\vee\mathcal{R} = \big(\bigcup_{i\in\Lambda} \equiv_i\big)^+$. The *equivalence closure* of a binary relation R on A, denoted by $\mathbf{EC}(R)$, is the smallest superset of R that is an equivalence relation on A. For any binary relation R on A, it is known that $\mathbf{EC}(R) = (R \cup R^{-1})^*$.

2.2 Pattern and embedding

Let $\star$ be a special symbol not in Σ, called the *gap*. A *pattern* is of the form $\star w_1 \star \ldots \star w_k \star$ where $k \geq 1$ and $w_1, \ldots, w_k \in \Sigma^+$. Let Π be the set of patterns, and let Π_0 be the set of strings over $\Sigma \cup \{\star\}$ where the $\star$'s do not occur consecutively. We note that $\Pi \subset \Pi_0$. The *size* of a pattern P, denoted by $size(P)$, is the number of symbols in P other than $\star$. The *reversal* of a pattern P, denoted by P^{rev}, is defined in the same way as in the string case. The *degree* of a pattern P is the number of occurrences of $\star$ in P and denoted by $deg(P)$.

For example, let $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$. $P = \star\mathtt{ab} \star \mathtt{a} \star \mathtt{cb}\star$ is a flexible pattern. $size(P) = 5$, $deg(P) = 4$ and $P^{rev} = \star\mathtt{bc} \star \mathtt{a} \star \mathtt{ba}\star$.

A *substitution* of degree d is a d-tuple $\langle \pi_1, \ldots, \pi_d\rangle$ such that $\pi_1, \ldots, \pi_d \in \Pi_0$. A substitution $\langle \pi_1, \ldots, \pi_d\rangle$ is said to be *ground* if $\pi_1, \ldots, \pi_d \in \Sigma^*$. For any pattern $P \in \Pi$ and a substitution $\theta = \langle \pi_1, \ldots, \pi_d\rangle$ with $d = deg(P)$, *the application of θ to P*, denoted by $P\theta$, is the pattern obtained by replacing the i-th occurrence of $\star$ in P with π_i for every $i = 1, \ldots, d$. For any $i = 1, \ldots, d$, let Θ_d^i be the set of substitutions $\theta = \langle \pi_1, \ldots, \pi_d\rangle$ such that (1) $\pi_i \neq \star$ and (2) $\pi_j = \star$ for every j with $1 \leq j \leq d$ and $j \neq i$. Let $\Theta_d = \bigcup_{i=1}^{d} \Theta_d^i$. A substitution $\langle \star, \ldots, \star, \pi, \star, \ldots, \star\rangle \in \Theta_d^i$ is said to be *primitive* if $\pi \in \{\varepsilon\} \cup \{\star a \star \mid a \in \Sigma\}$. A primitive substitution $\langle \star, \ldots, \star, \pi, \star, \ldots, \star\rangle$ is said to be *erasing* if $\pi = \varepsilon$, and

non-erasing if $\pi = \star a \star$ for some $a \in \Sigma$. For any $P, Q \in \Pi$, Q is said to be a *right-extension* of P if there exists a substitution $\theta \in \Theta_d^d$ such that $Q = P\theta$, where $d = deg(P)$.

An *embedding* of $P \in \Pi_0$ into $Q \in \Pi_0$ is a substitution θ such that $P\theta = Q$.

Definition 1. $P \lhd Q \overset{\text{def}}{\iff}$ *there is an embedding of P into Q that is primitive.*

Let $\lhd^*$ be a partial-order on Π_0 s.t. $P \lhd^* Q \iff$ there is an embedding of P into Q. For any $P \in \Pi$ and $T \in \Sigma^+$, P is said to *occur* in T if $P \lhd^+ T$.

For example, $\star\mathtt{ab} \star \mathtt{a} \star \mathtt{c} \star \lhd \star \mathtt{ab} \star \mathtt{d} \star \mathtt{a} \star \mathtt{c}\star$ due to a non-erasing primitive $\langle \star, \star\mathtt{d}\star, \star, \star\rangle \in \Theta_4$, $\star\mathtt{ab}\star\mathtt{a}\star\mathtt{c}\star\lhd\star\mathtt{ab}\star\mathtt{ac}\star$ due to an erasing primitive $\langle \star, \star, \varepsilon, \star\rangle \in \Theta_4$, and $\star\mathtt{ab} \star \mathtt{a} \star \mathtt{c} \star \lhd^* \star \mathtt{ab} \star \mathtt{d} \star \mathtt{ac}\star$ due to a substitution $\langle \star, \star\mathtt{d}\star, \varepsilon, \star\rangle$.

For an equivalence relation $\equiv$ on Π, a pattern $P \in \Pi$ is said to be *closed* under $\equiv$ if it is maximal in $[P]_\equiv$ w.r.t. $\lhd^*$.

2.3 Existing equivalence relations on Π

Let S be a finite subset of Σ^+. Intuitively, equivalence relations on Π are designed so that P and Q are equivalent if: *Every time P occurs in S, Q also occurs at the same location.* Difference between equivalence relations comes from the difference in definitions of *same location* here. Below, we describe several existing equivalence relations on Π.

An *occurrence interval* of a pattern $P \in \Pi$ in $T \in \Sigma^+$ is an interval $[|w_1| + 1, |T| - |w_d|]$ such that there is a ground embedding $\theta = \langle w_1, \ldots, w_d\rangle$ of P into T. Let $Int_T(P)$ be the set of all occurrence intervals of P in T. We give the definitions of four existing equivalence relations. Let S be a finite subset of Σ^+.

Definition 2 ($\overset{\mathsf{I}}{\equiv}_S, \overset{\mathsf{M}}{\equiv}_S, \overset{\mathsf{B}}{\equiv}_S, \overset{\mathsf{E}}{\equiv}_S$)**.** *For any patterns $P, Q \in \Pi$, let*

$$
\begin{aligned}
P \overset{\mathsf{I}}{\equiv}_S Q &\overset{\text{def}}{\iff} \forall T \in S,\ Int_T(P) = Int_T(Q),\\
P \overset{\mathsf{M}}{\equiv}_S Q &\overset{\text{def}}{\iff} \forall T \in S,\ \mathsf{Min}(Int_T(P)) = \mathsf{Min}(Int_T(Q)),\\
P \overset{\mathsf{B}}{\equiv}_S Q &\overset{\text{def}}{\iff} \forall T \in S,\ \mathsf{Beg}(Int_T(P)) = \mathsf{Beg}(Int_T(Q)),\\
P \overset{\mathsf{E}}{\equiv}_S Q &\overset{\text{def}}{\iff} \forall T \in S,\ \mathsf{End}(Int_T(P)) = \mathsf{End}(Int_T(Q)).
\end{aligned}
$$

The algorithm MaxFlex [2] enumerates all closed flexible patterns in polynomial space and linear-time delay under $\overset{\mathsf{B}}{\equiv}_S$. $\overset{\mathsf{B}}{\equiv}_S$, $\overset{\mathsf{E}}{\equiv}_S$ and $\overset{\mathsf{B}}{\equiv}_S \vee \overset{\mathsf{E}}{\equiv}_S$ are natural extensions of the well-known equivalence relations introduced by Blumer et al. [4] for the class of substring patterns, which are recognized as the basis of index structures for text data, e.g., the suffix trees ($\overset{\mathsf{B}}{\equiv}_S$), the DAWGs ($\overset{\mathsf{E}}{\equiv}_S$), and the compact DAWGs ($\overset{\mathsf{B}}{\equiv}_S \vee \overset{\mathsf{E}}{\equiv}_S$). $\overset{\mathsf{M}}{\equiv}_S$ is an extension of the equivalence relation introduced by Mannila et al. [9] for the class of subsequence patterns.

Equivalence relation function: We note that the equivalence relations above vary depending on S. An *equivalence relation (ER) function* is a function that maps finite subsets S of Σ^+ to equivalence relations $\equiv_S$ on Π. The *reversal* of an ER function Φ is defined by: $\langle P, Q\rangle \in \Phi^{rev}(S) \iff \langle P^{rev}, Q^{rev}\rangle \in \Phi(S^{rev})$.

We say that an ER function Φ is *symmetric* if $\Phi^{rev}(S) = \Phi(S)$ for every S. The ER functions for $\overset{\mathsf{I}}{\equiv}_S$, $\overset{\mathsf{M}}{\equiv}_S$, $\overset{\mathsf{B}}{\equiv}_S \vee \overset{\mathsf{E}}{\equiv}_S$ and $\overset{\mathsf{B}}{\equiv}_S \wedge \overset{\mathsf{E}}{\equiv}_S$ are symmetric, while the reversal of the ER function for $\overset{\mathsf{B}}{\equiv}_S$ is the ER function for $\overset{\mathsf{E}}{\equiv}_S$, and vice versa.

Monotonicity of equivalence relations on Π: An equivalence relation $\equiv$ on Π is said to be *monotone* if $\lhd^+ \cap \equiv = \ (\lhd \cap \equiv)^+$. If $\equiv$ is monotone, then $P_1 \lhd^+ P_2$ and $P_1 \equiv P_2$ implies that $\forall Q \in \Pi$, $(P_1 \lhd^* Q \lhd^* P_2 \implies P_1 \equiv Q \equiv P_2)$. Monotonicity of equivalence relations is a very helpful property for closedness check of a pattern, i.e., for any monotone equivalence relation $\equiv$, a pattern P is closed iff there is no primitive substitution θ such that $P\theta \equiv P$. We will discuss general and efficient algorithms based on monotonicity in Section 4. We remark that $\overset{\mathsf{I}}{\equiv}_S$, $\overset{\mathsf{M}}{\equiv}_S$, $\overset{\mathsf{B}}{\equiv}_S$ and $\overset{\mathsf{E}}{\equiv}_S$ are monotone.

3 Coarsening Existing Equivalence Relations

Since we prefer symmetric equivalence relations, we focus on $\overset{\mathsf{M}}{\equiv}_S$ that is symmetric. It is, however, still too fine and we want a coarser one. In this section, we define two new equivalence relations $\overset{\mathsf{MX}}{\equiv}_S$ and $\overset{\mathsf{MXG}}{\equiv}_S$ by coarsening $\overset{\mathsf{M}}{\equiv}_S$. With one of the techniques, we also coarsen the other equivalence relations $\overset{\mathsf{I}}{\equiv}_S$, $\overset{\mathsf{B}}{\equiv}_S$ and $\overset{\mathsf{E}}{\equiv}_S$. and introduce $\overset{\mathsf{IX}}{\equiv}_S$, $\overset{\mathsf{BX}}{\equiv}_S$ and $\overset{\mathsf{EX}}{\equiv}_S$.

For any pattern $P = \star w_1 \star \cdots \star w_k \star \in \Pi$, let $\overline{P} = w_1 \star \cdots \star w_k$, and thus $P = \star \overline{P} \star$. One technique of coarsening $\overset{\mathsf{M}}{\equiv}_S$ is to extend as long as possible the constant strings at pattern ends to the *outward* without decreasing occurrences.

Here we remark that the next proposition does hold.

Proposition 1. *For any $P \in \Pi$ and any substitution $\theta \in (\Theta_d^1 \cup \Theta_d^d)$ with $d = deg(P)$, $|\mathsf{Min}(Int_T(P))| \geq |\mathsf{Min}(Int_T(P\theta))|$ for every $T \in S$.*

Definition 3. $P \lhd_S^{\mathsf{MX}} Q \overset{\text{def}}{\iff}$ *there exists $a \in \Sigma$ such that*

- $Q = \star \overline{P} a \star$ *and* $\mathsf{Min}(Int_T(Q)) = \mathsf{Min}(Int_T(P)) \oplus \langle 0, 1 \rangle$ *for every $T \in S$; or*
- $Q = \star a \overline{P} \star$ *and* $\mathsf{Min}(Int_T(Q)) = \mathsf{Min}(Int_T(P)) \oplus \langle -1, 0 \rangle$ *for every $T \in S$.*

Definition 4 ($\overset{\mathsf{MX}}{\equiv}_S$). *Let* $\overset{\mathsf{MX}}{\equiv}_S = \mathbf{EC}(\overset{\mathsf{M}}{\equiv}_S \cup \lhd_S^{\mathsf{MX}})$.

Proposition 2. *For any $P \in \Pi$, there uniquely exists a pair of strings $u, v \in \Sigma^*$ such that $Q = \star u \overline{P} v \star$ is closed under $\overset{\mathsf{MX}}{\equiv}_S$ and $P \overset{\mathsf{MX}}{\equiv}_S Q$.*

Take an example S which consists of a single string $T_1 =$ `acbmdcacbndcaca`, and a pattern $P = \star$`b`$\star$`d`$\star$. There are two minimal occurrences of P in T_1, and we see they are preceded by "`ac`" and followed by "`cac`" without gap. Thus for any combinations of a suffix u of "`ac`" and a prefix v of "`cac`", $u\overline{P}v$ ($\star$`cb`$\star$`dcac`$\star$ for example) is equivalent to P under $\overset{\mathsf{MX}}{\equiv}_S$.

Another technique of coarsening $\overset{\mathsf{M}}{\equiv}_S$ is to add pattern fragments including gaps to the pattern ends without decreasing occurrences.

Definition 5. $P \lhd_S^{\mathsf{MXG}} Q \stackrel{\text{def}}{\iff} Q \in \{\star a \star \overline{P}\star, \star\overline{P} \star a\star\}$ *with some* $a \in \Sigma$ *and* $|\mathsf{Min}(Int_T(P))| = |\mathsf{Min}(Int_T(Q))|$ *for every* $T \in S$.

Definition 6 ($\stackrel{\mathsf{MXG}}{\equiv}_S$). *Let* $\stackrel{\mathsf{MXG}}{\equiv}_S = \mathbf{EC}(\stackrel{\mathsf{M}}{\equiv}_S \cup \lhd_S^{\mathsf{MXG}})$.

Take an example S which consists of a single string $T_1 = \mathtt{acbmdcagcbnddcaca}$, and a pattern $P = \star\mathtt{b}\star\mathtt{d}\star$. The number of minimal occurrences of P in T_1 is 2. There are several patterns that are equivalent to P under $\stackrel{\mathsf{MXG}}{\equiv}_S$, but not under $\stackrel{\mathsf{M}}{\equiv}_S$ and $\stackrel{\mathsf{MX}}{\equiv}_S$, such as $\star\overline{P}\star\mathtt{c}\star$, $\star\mathtt{a}\star\mathtt{c}\overline{P}\star\mathtt{ca}\star\mathtt{c}\star\mathtt{a}\star$ and $\star\mathtt{a}\star\mathtt{c}\overline{P}\star\mathtt{d}\star$. Note that $Q = \star\overline{P}\star\mathtt{d}\star\mathtt{c}\star$ is not equivalent to P under $\stackrel{\mathsf{MXG}}{\equiv}_S$ because the number of minimal occurrences of Q is 1.

We can prove that $\stackrel{\mathsf{MX}}{\equiv}_S$ and $\stackrel{\mathsf{MXG}}{\equiv}_S$ are monotone from Proposition 1. Also, the ER functions for $\stackrel{\mathsf{MX}}{\equiv}_S$ and $\stackrel{\mathsf{MXG}}{\equiv}_S$ are symmetric. It follows from Definitions 4 and 6 that the next inclusion relation holds.

Theorem 1. $\stackrel{\mathsf{M}}{\equiv}_S \subseteq \stackrel{\mathsf{MX}}{\equiv}_S \subseteq \stackrel{\mathsf{MXG}}{\equiv}_S$.

The technique used in defining $\stackrel{\mathsf{MX}}{\equiv}_S$ extends $\stackrel{\mathsf{I}}{\equiv}_S$, $\stackrel{\mathsf{B}}{\equiv}_S$ and $\stackrel{\mathsf{E}}{\equiv}_S$ as below.

Definition 7. $P \lhd_S^{\mathsf{IX}} Q \stackrel{\text{def}}{\iff}$ *there exists* $a \in \Sigma$ *such that*

- $Q = \star\overline{P}a\star$ *and* $Int_T(Q) = Int_T(P) \oplus \langle 0, 1\rangle$ *for every* $T \in S$*; or*
- $Q = \star a\overline{P}\star$ *and* $Int_T(Q) = Int_T(P) \oplus \langle -1, 0\rangle$ *for every* $T \in S$.

Definition 8. $P \lhd_S^{\mathsf{BX}} Q \stackrel{\text{def}}{\iff}$ *there exists* $a \in \Sigma$ *such that* $Q = \star a\overline{P}\star$ *and* $\mathsf{Beg}(Int_T(Q)) = \mathsf{Beg}(Int_T(P)) \oplus (-1)$ *for every* $T \in S$.

Definition 9. $P \lhd_S^{\mathsf{EX}} Q \stackrel{\text{def}}{\iff}$ *there exists* $a \in \Sigma$ *such that* $Q = \star\overline{P}a\star$ *and* $\mathsf{End}(Int_T(Q)) = \mathsf{End}(Int_T(P)) \oplus 1$ *for every* $T \in S$.

Definition 10 ($\stackrel{\mathsf{IX}}{\equiv}_S$, $\stackrel{\mathsf{BX}}{\equiv}_S$, $\stackrel{\mathsf{EX}}{\equiv}_S$). *Let* $\stackrel{\mathsf{IX}}{\equiv}_S = \mathbf{EC}(\stackrel{\mathsf{I}}{\equiv}_S \cup \lhd_S^{\mathsf{IX}})$, $\stackrel{\mathsf{BX}}{\equiv}_S = \mathbf{EC}(\stackrel{\mathsf{B}}{\equiv}_S \cup \lhd_S^{\mathsf{BX}})$ *and* $\stackrel{\mathsf{EX}}{\equiv}_S = \mathbf{EC}(\stackrel{\mathsf{E}}{\equiv}_S \cup \lhd_S^{\mathsf{EX}})$.

We note that $\stackrel{\mathsf{IX}}{\equiv}_S$, $\stackrel{\mathsf{BX}}{\equiv}_S$, $\stackrel{\mathsf{EX}}{\equiv}_S$ are not monotone.

4 Algorithms for Enumerating Frequent Closed Patterns

Let $Freq_S(P)$ denote the number of strings in S in which P occurs.

*Problem 1 (*FreqCloPatEnum *w.r.t.* $\equiv$*).* Given a finite subset S of Σ^+ and a non-negative integer σ, enumerate all the patterns P closed under $\equiv$ without duplicates such that $Freq_S(P) \geq \sigma$.

Theorem 2 (MaxFlex [2]). *For a finite alphabet* Σ*, there exists an algorithm that solves* FreqCloPatEnum *w.r.t.* $\stackrel{\mathsf{B}}{\equiv}_S$ *in* $O(|\Sigma|\|S\|)$ *time delay and* $O(\|S\|d)$ *space, where* d *is the maximum number of gaps in the output patterns.*

In this section, we consider methods for efficiently solving the problem for the M family ($\stackrel{\mathsf{M}}{\equiv}_S$, $\stackrel{\mathsf{MX}}{\equiv}_S$, $\stackrel{\mathsf{MXG}}{\equiv}_S$), the I family ($\stackrel{\mathsf{I}}{\equiv}_S$, $\stackrel{\mathsf{IX}}{\equiv}_S$), and the E family ($\stackrel{\mathsf{E}}{\equiv}_S$, $\stackrel{\mathsf{EX}}{\equiv}_S$). In the sequel, we exclude the descriptions for the B family ($\stackrel{\mathsf{B}}{\equiv}_S$, $\stackrel{\mathsf{BX}}{\equiv}_S$), since they are simply the reversal of the E family.

4.1 Outline of GenCloFlex

A pattern $P \in \Pi$ is said to be *i-th-gap-closed* under $\equiv$ if $P \not\equiv P\theta$ for every $\theta \in \Theta_d^i$, where $1 \leq i \leq d$ and $d = deg(P)$. For convenience, we say that P is *leftmost-gap-closed* for $i = 1$ and *rightmost-gap-closed* for $i = deg(P)$. A pattern $P \in \Pi$ is said to be *inner-gap-closed* if it is i-th-gap-closed under $\equiv$ for every i with $1 < i < deg(P)$.

An equivalence relation $\equiv$ on Π is said to be *rightmost-gap-independent* if any $P \in \Pi$ satisfies the following condition: For every i with $1 \leq i < deg(P)$, if P is not i-th-gap-closed under $\equiv$, then every right-extension P' of P is not i-th-gap-closed under $\equiv$. We remark that the M, I and E families are rightmost-gap-independent.

As a generalization of MaxFlex [2], we describe GenCloFlex, an algorithm for solving FreqCloPatEnum w.r.t. *any* equivalence relation that is rightmost-gap-independent. We define a rooted search-tree **ST** over $\Pi \cup \{\bot\}$ by:

- For any $a \in \Sigma$, the parent of $\star a \star$ is $\bot$.
- For any $a \in \Sigma$ and for any $w_1, \ldots, w_k \in \Sigma^+$, the parent of $\star w_1 \star \cdots \star w_k a \star$ and $\star w_1 \star \cdots \star w_k \star a \star$ is $\star w_1 \star \cdots \star w_k \star$.

Lemma 1. *For any $P, Q \in \Pi$, $P \lhd^* Q$ implies $Freq_S(P) \geq Freq_S(Q)$.*

Lemma 2 (general pruning rule). *Let $\equiv$ be any rightmost-gap-independent equivalence relation on Π. Under $\equiv$, if $P \in \Pi$ is not i-th-gap-closed for some i with $1 \leq i < deg(P)$, then no descendant of P in* **ST** *is closed.*

Algorithm 1 outlines a general algorithm for FreqCloPatEnum under *any* rightmost-gap-independent equivalence relation. The algorithm performs a depth-first-traversal of **ST**, with pruning based on Lemmas 1 and 2. We note that the algorithm does not build **ST** actually.

4.2 Closedness tests

We now consider how to realize the inner-, the leftmost- and the rightmost-closedness tests. An equivalence relation $\equiv$ on Π is said to be *inner-gap-monotone* if for any $P \in \Pi$ and any $\theta \in \Theta_d - (\Theta_d^1 \cup \Theta_d^d)$ with $d = deg(P)$, $P \equiv P\theta$ implies that $\forall Q \in \Pi$, $(P \lhd^* Q \lhd^* P\theta \implies P \equiv Q \equiv P\theta)$. We remark that the M, I and E families are all inner-gap-monotone.

Lemma 3 (inner-gap-closedness test). *Let $\equiv$ be any inner-gap-monotone equivalence relation on Π. Let $P \in \Pi$. Then, for any i with $1 < i < deg(P)$, P is i-th-gap-closed under $\equiv$ if $P \not\equiv P\theta$ for every primitive substitution θ in Θ_d^i.*

For a monotone equivalence relation $\equiv$, the leftmost- (resp. rightmost-) gap-closedness of $P \in \Pi$ can also be tested by checking whether $P \not\equiv P\theta$ for every non-erasing primitive substitution θ in Θ_d^1 (resp. Θ_d^d). For $\overset{\mathsf{IX}}{\equiv}_S$ and $\overset{\mathsf{EX}}{\equiv}_S$ that are not monotone, we have the following lemma:

Algorithm 1: General Algorithm GenCloFlex for FreqCloPatEnum

Input: a finite subset S of Σ^+ and a non-negative integer σ.
Output: non-duplicate list of patterns P with $Freq_S(P) \geq \sigma$ that are closed under a rightmost-gap-independent equivalence relation $\equiv$.

```
1 foreach a ∈ Σ do Expand(⋆a⋆);

procedure Expand(P);
1 let d := deg(P);
2 if Freq_S(P) < σ then return ;
3 if (P is not leftmost-gap-closed) or (P is not inner-gap-closed) then
4     return; // Pruning
  // Now P is leftmost-gap-closed and inner-gap-closed
5 if P is rightmost-gap-closed then // P is closed
6     report P;
7 foreach a ∈ Σ do
8     Expand(⋆P̄ ⋆ a⋆);
9     Expand(⋆P̄a⋆);
```

Table 1. Time complexities of the leftmost-, the rightmost- and the inner-gap-closedness tests for respective equivalence relations.

	leftmost-gap-closedness	rightmost-gap-closedness	inner-gap-closedness
$\overset{\mathsf{I}}{\equiv}_S$	(always true)	(always true)	$O(\|\|S\|\|d)$
$\overset{\mathsf{IX}}{\equiv}_S$	$O(\|\|S\|\|)$	$O(\|\|S\|\|)$	$O(\|\|S\|\|d)$
$\overset{\mathsf{M}}{\equiv}_S$	(always true)	(always true)	$O(\|\|S\|\|d)$
$\overset{\mathsf{MX}}{\equiv}_S$	$O(\|\|S\|\|)$	$O(\|\|S\|\|)$	$O(\|\|S\|\|d)$
$\overset{\mathsf{MXG}}{\equiv}_S$	$O(\|\|S\|\|)$	$O(\|\|S\|\|)$	$O(\|\|S\|\|d)$
$\overset{\mathsf{E}}{\equiv}_S$	$O(\|\|S\|\|)$	(always true)	$O(\|\|S\|\|)$
$\overset{\mathsf{EX}}{\equiv}_S$	$O(\|\|S\|\|)$	$O(\|\|S\|\|)$	$O(\|\|S\|\|)$

Table 2. Delay time per output for respective equivalence relations.

	GenCloFlex	GenCloFlex+
$\overset{\mathsf{I}}{\equiv}_S$	$O(\|\Sigma\|\|\|S\|\|d)$	$O(\|\Sigma\|\|\|S\|\|d)$
$\overset{\mathsf{IX}}{\equiv}_S$	$O(\|\Sigma\|\|\|S\|\|^2 d)$	$O(\|\Sigma\|\|\|S\|\|d)$
$\overset{\mathsf{M}}{\equiv}_S$	$O(\|\Sigma\|\|\|S\|\|d)$	$O(\|\Sigma\|\|\|S\|\|d)$
$\overset{\mathsf{MX}}{\equiv}_S$	$O(\|\Sigma\|\|\|S\|\|^2 d)$	$O(\|\Sigma\|\|\|S\|\|d)$
$\overset{\mathsf{MXG}}{\equiv}_S$	$O(\|\Sigma\|\|\|S\|\|^2 d)$	$O(\|\Sigma\|\|\|S\|\|^2 d)$
$\overset{\mathsf{E}}{\equiv}_S$	$O(\|\Sigma\|\|\|S\|\|)$ [2]	$O(\|\Sigma\|\|\|S\|\|)$
$\overset{\mathsf{EX}}{\equiv}_S$	$O(\|\Sigma\|\|\|S\|\|^2)$	$O(\|\Sigma\|\|\|S\|\|)$

Lemma 4 (leftmost-, rightmost-gap-closedness tests for $\overset{\mathsf{IX}}{\equiv}_S$, $\overset{\mathsf{EX}}{\equiv}_S$, $\overset{\mathsf{MX}}{\equiv}_S$). *Let $\equiv \in \{\overset{\mathsf{IX}}{\equiv}_S, \overset{\mathsf{EX}}{\equiv}_S, \overset{\mathsf{MX}}{\equiv}_S\}$ and $P \in \Pi$. P is leftmost-gap-closed under $\equiv$ if $P \not\equiv \star a\overline{P}\star$ for every $a \in \Sigma$. P is rightmost-gap-closed under $\equiv$ if $P \not\equiv \star\overline{P}a\star$ for every $a \in \Sigma$.*

Lemma 5. *The time complexities of the leftmost-, the rightmost- and the inner-gap-closedness tests for $P \in \Pi$ with $d = deg(P)$ are summarized in Table 1.*

Proof. The leftmost- and the rightmost-gap-closedness tests for $\overset{\mathsf{I}}{\equiv}_S$ and $\overset{\mathsf{M}}{\equiv}_S$ and the rightmost-gap-closedness test for $\overset{\mathsf{E}}{\equiv}_S$ are unnecessary by their definitions. The leftmost-gap-closedness test for $\overset{\mathsf{E}}{\equiv}_S$ takes $O(\|S\|)$ time as shown in [2]. By Lemma 4 the leftmost-gap-closedness test (resp. the rightmost-gap-closedness test) for $\overset{\mathsf{MX}}{\equiv}_S$ can be performed simply by checking whether all minimal occurrences of P are directly preceded by (resp. followed by) a same symbol. This

takes $O(\|S\|)$ time. Similarly, the rightmost-gap-closedness test for $\overset{\mathsf{EX}}{\equiv}_S$ and the leftmost- and the rightmost-gap-closedness test for $\overset{\mathsf{IX}}{\equiv}_S$ take $O(\|S\|)$ time. Now we consider the rightmost-gap-closedness test for $\overset{\mathsf{MXG}}{\equiv}_S$. What we have to do is to check whether there exists a non-erasing primitive substitution θ in Θ_d^d which preserves $|\mathsf{Min}(Int_T(P))|$ in every $T \in S$. Let $e_1, \ldots, e_m$ be the increasing sequence of ending positions of $\mathsf{Min}(Int_T(P))$. Let $I_i = [e_1 + 1, e_{i+1}]$ for $i = 1, \ldots, m-1$ and let $I_m = [e_m + 1, |T|]$. We can build the list of symbols common to the substrings of T implied by $I_1, \ldots, I_m$ in $O(|T|)$ time. The test thus takes $O(\|S\|)$ time. The leftmost-gap-closedness test also takes $O(\|S\|)$ time.

We now suppose $d > 2$ for the inner-gap-closedness test. For the E family, it suffices to determine whether $P\theta$ occurs within the leftmost occurrence interval of P. This can be done in $O(\|S\|)$ time independently of d by using an auxiliary data structure of size $O(\|S\|d)$ as shown in [2]. For the M family, we have to check it over all minimal occurrence intervals of P, and the same technique cannot be applied. This takes $O(\|S\|d)$ time and space. For the I family, we basically check it over all occurrence intervals of P. For the erasing primitive substitution θ in Θ_d^2 (resp. Θ_d^{d-1}), it suffices to check whether $P\theta$ begins (resp. ends) at every beginning (resp. ending) positions of occurrence intervals of P. For the other erasing primitive substitutions or for the non-erasing primitive substitutions, it suffices to consider only the minimal occurrence intervals. □

4.3 Improved algorithms for respective equivalence relations

We introduce new efficient pruning techniques based on common extensions. Especially, the techniques improve the time complexity for $\overset{\mathsf{MX}}{\equiv}_S$, $\overset{\mathsf{IX}}{\equiv}_S$ and $\overset{\mathsf{EX}}{\equiv}_S$, as shown in Table 2.

Let $\equiv \in \{\overset{\mathsf{M}}{\equiv}_S, \overset{\mathsf{MX}}{\equiv}_S, \overset{\mathsf{MXG}}{\equiv}_S, \overset{\mathsf{I}}{\equiv}_S, \overset{\mathsf{IX}}{\equiv}_S, \overset{\mathsf{E}}{\equiv}_S, \overset{\mathsf{EX}}{\equiv}_S\}$. The *longest common extension* of $P \in \Pi$ under $\equiv$ is the longest string $v \in \Sigma^*$ such that for every $T \in S$,

- $\mathsf{Min}(Int_T(\star\overline{P}v\star)) = \mathsf{Min}(Int_T(P)) \oplus \langle 0, |v| \rangle$ when $\equiv \in \{\overset{\mathsf{M}}{\equiv}_S, \overset{\mathsf{MX}}{\equiv}_S, \overset{\mathsf{MXG}}{\equiv}_S\}$;
- $Int_T(\star\overline{P}v\star) = Int_T(P) \oplus \langle 0, |v| \rangle$ when $\equiv \in \{\overset{\mathsf{I}}{\equiv}_S, \overset{\mathsf{IX}}{\equiv}_S, \overset{\mathsf{E}}{\equiv}_S, \overset{\mathsf{EX}}{\equiv}_S\}$.

When $v \neq \varepsilon$, $c = v[1]$ is said to be the *common extension* of P under $\equiv$.

The following lemmas help us to skip unnecessary closedness tests.

Lemma 6 (skipping leftmost- and inner-gap-closedness tests). *Let* $\equiv \in \{\overset{\mathsf{M}}{\equiv}_S, \overset{\mathsf{MX}}{\equiv}_S, \overset{\mathsf{MXG}}{\equiv}_S, \overset{\mathsf{I}}{\equiv}_S, \overset{\mathsf{IX}}{\equiv}_S, \overset{\mathsf{E}}{\equiv}_S, \overset{\mathsf{EX}}{\equiv}_S\}$ *and let* $c \in \Sigma$ *be the common extension of* $P \in \Pi$ *under* $\equiv$. *If* P *is leftmost- and inner-gap-closed under* $\equiv$, $\star\overline{P}c\star$ *is also leftmost- and inner-gap-closed under* $\equiv$.

Lemma 7 (skipping rightmost-gap-closedness tests). *Let* $\equiv \in \{\overset{\mathsf{MX}}{\equiv}_S, \overset{\mathsf{MXG}}{\equiv}_S, \overset{\mathsf{IX}}{\equiv}_S, \overset{\mathsf{EX}}{\equiv}_S\}$ *and let* $c \in \Sigma$ *be the common extension of* $P \in \Pi$ *under* $\equiv$. *Then* P *is not rightmost-gap-closed under* $\equiv$.

For the M family, we can utilize the following lemma for pruning.

Lemma 8 (pruning for the M family). *Let* $\equiv \in \{\overset{\mathsf{M}}{\equiv}_S, \overset{\mathsf{MX}}{\equiv}_S, \overset{\mathsf{MXG}}{\equiv}_S\}$ *and let* $c \in \Sigma$ *be the common extension of* $P \in \Pi$ *under* $\equiv$*. Then among the descendants of* P *in* **ST***, only descendants of* $\star\overline{P}c\star$ *can be closed under* $\equiv$*.*

Proof. Since $\mathsf{Min}(Int_T(\star\overline{P}c\star)) = \mathsf{Min}(Int_T(P)) \oplus \langle 0, 1\rangle$ for every $T \in S$, $\star\overline{P}\star c\star$ is not closed due to $\star\overline{P}\star c\star \equiv \star\overline{P}c\star$. Since $\langle \star\overline{P} \star a\star, \star\overline{P}c \star a\star\rangle \in \lhd^+ \cap \equiv$, $\star\overline{P} \star a\star$ is not closed for any $a \in \Sigma - \{c\}$. Let $P = \star w_1 \star \cdots \star w_k\star$, where $w_1, \ldots, w_k \in \Sigma^+$. Let $P' = \star w_1 \star \cdots \star w_{k-1} \star b \star w_k a\star$, where b is the first symbol of w_k. Since $\langle \star\overline{P}a\star, P'\rangle \in \lhd^+ \cap \equiv$, $\star\overline{P}a\star$ is not closed for any $a \in \Sigma - \{c\}$. Since $\equiv$ is monotone, the lemma holds. □

For the I and E families, we can utilize the following lemmas for pruning.

Lemma 9 (pruning for the I and E families). *Let* $\equiv \in \{\overset{\mathsf{I}}{\equiv}_S, \overset{\mathsf{IX}}{\equiv}_S, \overset{\mathsf{E}}{\equiv}_S, \overset{\mathsf{EX}}{\equiv}_S\}$ *and let* $c \in \Sigma$ *be the common extension of* $P \in \Pi$ *under* $\equiv$*. Then among the descendants of* P *in* **ST***, only descendants of* $\star\overline{P}c\star$ *and of* $\star\overline{P}\star c\star$ *can be closed under* $\equiv$*.*

Proof. $\star\overline{P}a\star$ does not occur in S for any $a \in \Sigma - \{c\}$. Since $\langle \star\overline{P}\star a\star, \star\overline{P}c\star a\star\rangle \in \lhd^+ \cap \equiv$, $\star\overline{P}\star a\star$ is not inner-gap-closed for any $a \in \Sigma - \{c\}$. Since $\equiv$ is inner-gap-monotone, the lemma holds. □

Lemma 10. *Let* $\equiv \in \{\overset{\mathsf{I}}{\equiv}_S, \overset{\mathsf{IX}}{\equiv}_S, \overset{\mathsf{E}}{\equiv}_S, \overset{\mathsf{EX}}{\equiv}_S\}$ *and let* $c \in \Sigma$ *be the common extension of* $P \in \Pi$ *under* $\equiv$*.* $|\mathsf{End}(Int_T(P))| = |\{i \mid \min\{\mathsf{End}(Int_T(P))\} < i \leq |T|, T[i] = c\}|$ *for every* $T \in S \iff \star\overline{P}\star c\star \equiv \star\overline{P}c\star$*.*

Proof. Since c is the common extension of P under $\equiv$, $Int_T(\star\overline{P}c\star) = Int_T(P) \oplus \langle 0, 1\rangle$ for every $T \in S$. Adding to this, the left-hand condition implies that $Int_T(\star\overline{P}\star c\star) = Int_T(\star\overline{P}c\star)$ for every $T \in S$. Hence the $\Longrightarrow$ statement holds. The $\Longleftarrow$ statement follows from the fact that $|\mathsf{End}(Int_T(\star\overline{P}\star c\star))| > |\mathsf{End}(Int_T(\star\overline{P}c\star))|$ for some $T \in S$ if the left-hand condition does not hold. □

For any $\equiv \in \{\overset{\mathsf{M}}{\equiv}_S, \overset{\mathsf{MX}}{\equiv}_S, \overset{\mathsf{MXG}}{\equiv}_S, \overset{\mathsf{I}}{\equiv}_S, \overset{\mathsf{IX}}{\equiv}_S, \overset{\mathsf{E}}{\equiv}_S, \overset{\mathsf{EX}}{\equiv}_S\}$ and $P \in \Pi$, the common extension c of P under $\equiv$ is said to *make a branch* if $\star\overline{P}\star c\star \not\equiv \star\overline{P}c\star$. Clearly from Lemma 8, any common extension does not make a branch for the M family. For the I and E families, it follows from Lemma 10 that a common extension c makes a branch iff $|\mathsf{End}(Int_T(P))| < |\{i \mid \min\{\mathsf{End}(Int_T(P))\} < i \leq |T|, T[i] = c\}|$ for some $T \in S$.

The algorithm based on Lemmas 6, 7, 8, 9 and 10 can be summarized as Algorithm 2. We remark that the longest common extension v can be represented in constant space, by the pair of a pointer to some position in $T \in S$ where v occurs and length $|v|$.

4.4 Time complexities

Theorem 3 (GenCloFlex,GenCloFlex+). *For a finite alphabet* Σ*, Algorithms 1 and 2 solve* FreqCloPatEnum *for respective equivalence relations with time delay shown in Table 2 and* $O(\|S\|d)$ *space, where* d *is the maximum number of gaps in the output patterns.*

Algorithm 2: Improved Algorithm GenCloFlex+ for FreqCloPatEnum

Input: a finite subset S of Σ^+ and a non-negative integer σ.
Output: non-duplicate list of patterns P with $Freq_S(P) \geq \sigma$ that are closed under $\equiv \in \{\overset{\mathsf{M}}{\equiv}_S, \overset{\mathsf{MX}}{\equiv}_S, \overset{\mathsf{MXG}}{\equiv}_S, \overset{\mathsf{I}}{\equiv}_S, \overset{\mathsf{IX}}{\equiv}_S, \overset{\mathsf{E}}{\equiv}_S, \overset{\mathsf{EX}}{\equiv}_S\}$.

1 **foreach** $a \in \Sigma$ **do** `CheckExtension(`$\star a \star$`)`;

procedure `ExpandWithCommonExtension(`P, v`)`;
1 **if** P is rightmost-gap-closed **then** `// When` $|v| > 0$`, use Lemma 7`
2 **report** P;
3 **if** $|v| > 0$ **then**
4 let $c := v[1]$;
5 `ExpandWithCommonExtension(`$\star\overline{P}c\star$, $v[2..|v|]$`)`;
6 **if** c makes a branch **then**
7 `CheckExtension(`$\star\overline{P} \star c\star$`)`;
8 **else**
9 **foreach** $a \in \Sigma$ **do**
10 `CheckExtension(`$\star\overline{P} \star a\star$`)`;
11 `CheckExtension(`$\star\overline{P}a\star$`)`;

procedure `CheckExtension(`P`)`;
1 **if** $Freq_S(P) < \sigma$ **then return** ;
2 **if** (P is not leftmost-gap-closed) **or** (P is not inner-gap-closed) **then**
3 **return**; `// Pruning`
`// Now` P `is leftmost-gap-closed and inner-gap-closed`
4 compute the longest common extension v of P under $\equiv$;
5 `ExpandWithCommonExtension(`P, v`)`;

Proof. Let $CT_{\equiv}$ denote the cost of the closedness test under $\equiv$, i.e., $CT_{\equiv} \in O(\|S\|)$ if $\equiv$ is the E family, $CT_{\equiv} \in O(\|S\|d)$ if $\equiv$ is the M or I family.

For GenCloFlex: For $\overset{\mathsf{M}}{\equiv}_S$, $\overset{\mathsf{I}}{\equiv}_S$ and $\overset{\mathsf{E}}{\equiv}_S$, the rightmost-gap-closedness test is unnecessary and therefore the condition of the **if**-statement at Line 5 is always satisfied and pattern P is always reported. As a result, the delay time per output is obtained by $O(|\Sigma| \times CT_{\equiv})$. On the other hand, for $\overset{\mathsf{MX}}{\equiv}_S$, $\overset{\mathsf{MXG}}{\equiv}_S$, $\overset{\mathsf{IX}}{\equiv}_S$ and $\overset{\mathsf{EX}}{\equiv}_S$, pattern P is reported only when the condition is satisfied. Nevertheless, we can find a frequent closed pattern after going down **ST** at most $\|S\|$ unreported patterns, and hence, the delay for output is $O(|\Sigma|\|S\| \times CT_{\equiv})$.

For GenCloFlex+: At Line 4 of Procedure `CheckExtension`(P), we compute the longest common extension of P under $\equiv$. It is equivalent to compute the longest common prefix of $\bigcup_{T \in S}\{T[j+1..|T|] \mid j \in \mathsf{End}(\mathsf{Min}(Int_T(P)))\}$ (resp. $\bigcup_{T \in S}\{T[j+1..|T|] \mid j \in \mathsf{End}(Int_T(P))\}$) for the M family (resp. for the I and E families), and hence, is done in $O(\|S\|)$ time.

In the case of the I or E family, we also compute the positions where the common extension makes a branch as follows.
For every $T \in S$ in which P occurs, do the following:

1. Compute $F(a) = |\{i \mid \min\{\mathsf{End}(Int_T(P))\} < i \leq |T|, T[i] = a\}|$ for any symbol a used in v.
2. For each $j = 1, \ldots, |v|$ in increasing order, do the following:
   (a) If $|\mathsf{End}(Int_T(P))| < F(v[j])$ then $v[j]$ makes a branch.
   (b) Decrement $F(v[j])$ by 1.

Thus we can compute the branching positions in v in $O(\|S\|)$ time.

Here we estimate the delay time per output for $\equiv \in \{\overset{\mathsf{MX}}{\equiv}_S, \overset{\mathsf{IX}}{\equiv}_S, \overset{\mathsf{EX}}{\equiv}_S\}$. Let us consider the call $\texttt{CheckExtension}(P)$ such that P is not rightmost-gap-closed, i.e., there exists $c \in \Sigma$ with $P \equiv \star\overline{P}c\star$. From the definition of $\overset{\mathsf{MX}}{\equiv}_S$, $\overset{\mathsf{IX}}{\equiv}_S$ and $\overset{\mathsf{EX}}{\equiv}_S$, $\star\overline{P}v\star$ is closed, where v is the longest common extension of P under $\equiv$. Since v can be computed in $O(\|S\|)$ time and the closedness test for P takes $O(CT_{\equiv})$ time, we can output $\star\overline{P}v\star$ in $O(CT_{\equiv})$ time just after $\texttt{CheckExtension}(P)$ is executed. Hence the delay from a previous output to $\star\overline{P}v\star$ is $O(|\Sigma| \times CT_{\equiv})$. □

The closedness checks for $\overset{\mathsf{MXG}}{\equiv}_S$ take a constant factor more time than those for $\overset{\mathsf{M}}{\equiv}_S$ and $\overset{\mathsf{MX}}{\equiv}_S$. However, it should be noted that the total theoretical asymptotic worst case time complexities for $\overset{\mathsf{MXG}}{\equiv}_S$ is equal to or smaller than those for $\overset{\mathsf{M}}{\equiv}_S$ and $\overset{\mathsf{MX}}{\equiv}_S$, due to the smaller search space for $\overset{\mathsf{MXG}}{\equiv}_S$ in **ST**. Hence, $\overset{\mathsf{MXG}}{\equiv}_S$ never falls far behind $\overset{\mathsf{M}}{\equiv}_S$ and $\overset{\mathsf{MX}}{\equiv}_S$, and can be much faster.

5 Computational Experiments

We implemented our algorithms for I, M and E families in the C language. Recall that $\overset{\mathsf{B}}{\equiv}_S$ is just the reversal of $\overset{\mathsf{E}}{\equiv}_S$, and thus, GenCloFlex $\overset{\mathsf{E}}{\equiv}_S$ can essentially be regarded as MaxFlex [2]. Considering the trade-off in implementation, we used naive matching to compute the longest common extensions, and did not implement the pruning technique based on Lemma 10. All the computational experiments were carried out on Apple Xserve with two Quad-Core Intel Xeon at 2.93GHz (8 CPU x 2 HT), with 24GB Memory 1066MHz DDR3.

We carried out experiments on synthetic data. To create data sets for examining flexible pattern mining algorithms, we modified IBM sequence generator [1], which is widely used in the subsequence pattern mining research area [3, 5–8, 13, 20]. The original program generates random sequences of item sets and embeds copies of some item set sequence as a pattern which is randomly corrupted. Although, originally, each item set is sorted and represented as a sorted integer sequence, we use the unsorted sequence representation. Each such sequence in the pattern is considered as a segment of the flexible pattern. In this way, we are able to generate a data set of integer strings in which some flexible patterns are embedded, where each segment is damaged in the same manner as the original program (See [1] for more details).

There are several parameters: [D] number of generated strings in 1000s, [C] average length of strings, [N] alphabet size in 1000s, [P] number of patterns, [L] average number of segments of patterns and [S] average length of segments of

Table 3. Experiments on Synthetic Data Sets (threshold value is fixed to $\sigma = 10$)

	D5C40N1P500L4S2		D5C40N1P500L4S4		D5C40N1P500L4S6	
algorithm/equiv	patterns	seconds	patterns	seconds	patterns	seconds
GenCloFlex $\overset{\mathsf{I}}{\equiv}_S$	271,412	1155	216,513	1119	201,746	972
GenCloFlex+ $\overset{\mathsf{I}}{\equiv}_S$	271,412	1105 (96%)	216,513	967 (86%)	201,746	800 (82%)
GenCloFlex+ $\overset{\mathsf{IX}}{\equiv}_S$	255,910 (94%)	1104 (96%)	182,868 (84%)	970 (87%)	159,898 (79%)	777 (80%)
GenCloFlex $\overset{\mathsf{M}}{\equiv}_S$	294,183	1225	285,954	1434	305,893	1360
GenCloFlex+ $\overset{\mathsf{M}}{\equiv}_S$	294,183	1161 (95%)	285,954	1116 (78%)	305,893	948 (70%)
GenCloFlex+ $\overset{\mathsf{MX}}{\equiv}_S$	253,928 (86%)	1075 (88%)	180,576 (63%)	928 (65%)	162,658 (53%)	781 (57%)
GenCloFlex+ $\overset{\mathsf{MXG}}{\equiv}_S$	247,036 (84%)	1060 (87%)	165,175 (58%)	874 (61%)	143,930 (47%)	731 (54%)
GenCloFlex $\overset{\mathsf{E}}{\equiv}_S$	311,526	1380	328,368	1717	341,138	1545
GenCloFlex+ $\overset{\mathsf{E}}{\equiv}_S$	311,526	1266 (92%)	328,368	1368 (80%)	341,138	1100 (71%)
GenCloFlex+ $\overset{\mathsf{EX}}{\equiv}_S$	270,511 (87%)	1180 (86%)	209,971 (64%)	1146 (67%)	184,289 (54%)	904 (59%)

patterns. We omitted scalability tests since we clearly showed time complexities of our algorithms. Instead we are interested in how the parameter [S] affects efficiency of our algorithms, and therefore experimented on three data sets with parameters D5C40N1P500L4S2, D5C40N1P500L4S4 and D5C40N1P500L4S6.

In Table 3, we compare the number of output closed patterns and computational time, where xx% is the relative ratio compared to GenCloFlex of respective families. The result shows that redundant output patterns which can be removed by our coarsened equivalence relations increase as the average length of frequent segments embedded in a data set becomes longer. Thus our algorithms with coarsened equivalence relations would be effective especially for data which is expected to contain long frequent segments such as bio-sequences.

6 Conclusion

We addressed the closed pattern discovery problem for the class of flexible patterns. We focused on the minimal-occurrence-interval based equivalence relation $\overset{\mathsf{M}}{\equiv}_S$ on the set of patterns introduced by Mannila et al. [9], and proposed two new equivalence relations by coarsening it. We investigated the properties of equivalence relations on the patterns from viewpoints of closed pattern enumeration, and as a generalization of the algorithm proposed by Arimura and Uno [2], we showed a general algorithm for enumerating closed patterns for existing and newly proposed equivalence relations of various kinds. Then we accelerated the algorithm by a set of new pruning techniques. Computational experiments on synthetic data implied that the proposed equivalence relations successfully remove some redundancy in the output patterns compared to the existing equivalence relations. Finally, although the results are not shown due to space limitation, we applied our algorithm on real data: Waka poems – traditional Japanese poetry with over 1300-year history – and confirmed that our algorithms with coarsened equivalence relations output reasonable amounts of mined patterns and increase their readability.

References

1. Agrawal, R., Srikant, R.: Mining sequential patterns. In: ICDE. pp. 3–14 (1995)
2. Arimura, H., Uno, T.: Mining maximal flexible patterns in a sequence. In: JSAI'07. pp. 307–317 (2008)
3. Ayres, J., Flannick, J., Gehrke, J., Yiu, T.: Sequential pattern mining using a bitmap representation. In: KDD. pp. 429–435 (2002)
4. Blumer, A., Blumer, J., Haussler, D., McConnell, R., Ehrenfeucht, A.: Complete inverted files for efficient text retrieval and analysis. J. ACM 34(3), 578–595 (1987)
5. Ding, B., Lo, D., Han, J., Khoo, S.C.: Efficient mining of closed repetitive gapped subsequences from a sequence database. In: ICDE. pp. 1024–1035 (2009)
6. Lo, D., Cheng, H., Lucia: Mining closed discriminative dyadic sequential patterns. In: EDBT. pp. 21–32 (2011)
7. Lo, D., Ding, B., Lucia, Han, J.: Bidirectional mining of non-redundant recurrent rules from a sequence database. In: ICDE. pp. 1043–1054 (2011)
8. Lo, D., Khoo, S.C., Li, J.: Mining and ranking generators of sequential patterns. In: SDM. pp. 553–564 (2008)
9. Mannila, H., Toivonen, H., Verkamo, I.A.: Discovery of frequent episodes in event sequences. Data Mining and Knowledge Discovery 1(3), 259–289 (1997)
10. Parida, L., Rigoutsos, I., Floratos, A., Platt, D.E., Gao, Y.: Pattern discovery on character sets and real-valued data: linear bound on irredundant motifs and an efficient polynomial time algorithm. In: Proc. SODA. pp. 297–308 (2000)
11. Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., Hsu, M.: Prefixspan: Mining sequential patterns by prefix-projected growth. In: ICDE. pp. 215–224 (2001)
12. Pisanti, N., Crochemore, M., Grossi, R., Sagot, M.F.: A basis of tiling motifs for generating repeated patterns and its complexity for higher quorum. In: Proc. MFCS. pp. 622–631 (2003)
13. Raïssi, C., Calders, T., Poncelet, P.: Mining conjunctive sequential patterns. In: ECML/PKDD (1). p. 19 (2008)
14. Srikant, R., Agrawal, R.: Mining sequential patterns: Generalizations and performance improvements. In: EDBT. pp. 3–17 (1996)
15. Tatti, N., Cule, B.: Mining closed strict episodes. In: ICDM. pp. 501–510 (2010)
16. Wang, J., Han, J.: BIDE: Efficient mining of frequent closed sequences. In: ICDE. pp. 79–90 (2004)
17. Wang, J., Han, J., Li, C.: Frequent closed sequence mining without candidate maintenance. IEEE Transactions on Knowledge and Data Engineering 19(8), 1042–1056 (2007)
18. Wang, K., Xu, Y., Yu, J.X.: Scalable sequential pattern mining for biological sequences. In: CIKM. pp. 178–187 (2004)
19. Wu, H.W., Lee, A.J.: Mining closed flexible patterns in time-series databases. Expert Systems with Applications 37(3), 2098 – 2107 (2010)
20. Yan, X., Han, J., Afshar, R.: CloSpan: Mining closed sequential patterns in large databases. In: SDM (2003)
21. Zaki, M.J.: Spade: An efficient algorithm for mining frequent sequences. Machine Learning 42(1/2), 31–60 (2001)
22. Zhou, W., Liu, H., Cheng, H.: Mining closed episodes from event sequences efficiently. In: PAKDD. pp. 301–318 (2010)