TCS-TR-A-06-19

TCS Technical Report

A Polynomial Space and Polynomial Delay Algorithm for Enumerating Maximal Two-Dimensional Patterns with Wildcards

by

HIROKI ARIMURA AND TAKEAKI UNO

Division of Computer Science

Report Series A

July 18, 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

A Polynomial Space and Polynomial Delay Algorithm for Enumerating Maximal Two-Dimensional Patterns with Wildcards

Hiroki Arimura[1] and Takeaki Uno[2]

[1] Hokkaido University, Kita 14-jo, Nishi 9-chome, Sapporo 060-0814, JAPAN
arim@ist.hokudai.ac.jp
[2] National Institute of Informatics, Tokyo 101–8430, JAPAN
uno@nii.jp

Abstract. In this paper, we consider the problem of finding maximal patterns in an two-dimensional text for the class of two-dimensional picure patterns with wild cards (don't cares). A maximal pattern for two-dimensional text is a generalization of a 1-dimensional sequential motif in (Parida et al., SODA'00; Pisanti et al., MFCS'03; Pelfrene etal., CPM'03), and it is such a representative pattern that is not properly contained in any larger pattern with the same location lists under invariance with respect to parallel motion. Since a text may contain exponentially many maximal patterns in general, it is natural to consider an output-sensitive algorithms for enumerating all patterns in the class. Then, we present a polynomial space and polynomial delay algorithm for enumerating all maximal patterns in a text array in exact polynomial time per discovered pattern in the total size of the input text. This result is achieved by backtracking through depth-first search of a tree-shaped search route over all maximal patterns, similar to (Arimura et al., ISAAC'05). We also discuss the lowerbound results on the number of maximal patterns and the #P-hardness of conting, and efficient input polynomial time computation of constant patterns with matrix suffix trees.

1 Introduction

By rapid increase of electronic data in various forms, two dimensional pattern matching have received much attention with potential applications in multimedia, digital libraries, computer vision, geographic information systems and semi-structured data processing. In this paper, we consider a new class of data processing problem for two-dimensional matrix data, called the *two-dimensional pattern discovery* problem, where an input is a $n \times n$ text matrix $T[1..n, 1..n]$ over an alphabet Σ, the problem is to find all patterns within a pattern class and satisfying specified constraints without repetition.

Maximal pattern discovery problem. We study the *maximal pattern discovery problem for the class of square patterns with wild cards* defined as follows. For an alphabet $\Sigma = \{a, b, \ldots\}$ of *constant letters* and the *wild card* (or *don't care*) $\circ \notin \Sigma$, a *two-dimensinal square pattern* with wild cards is an $m \times m$ matrix $P[1..m, 1..m]$ over $\Sigma \cup \{\circ\}$, where $\circ$ matches itself and any constant letters. Then, a pattern $P[1..m, 1..m]$ has the location list $\mathcal{L}(P) = \{d_1, \ldots, d_m\} \subseteq [1..n] \times [1..n]$ $(m \geq 0)$ which is the set of all locations (or occurrences) of P in T. A pattern P with wild cards is *maximal* in a given text T iff (i) P appears in T at some specified frequency and (ii) there exists no pattern Q appearing in T that properly contains P and having the same location list under the invariance with respect to parallel motion. We denote by $\mathcal{P}$ and $\mathcal{M}$ the class of patterns and the class of all maximal two-dimensional square patterns over $\Sigma \cup \{\circ\}$.

The maximal pattern discovery problem can be regarded as a two-dimensional string version of *maximal motif discovery* problems in bioinformatics, extensively studied for the last years [15, 18, 17], and as a variation of *closed itemset mining* and *closed semi-structured data mining* in database and artificial intelligence areas [11, 20, 21].

Since there are exponentially many patterns in an $n \times n$ text (Theorem 2), it is natural to consider a pattern discovery problem as an enumeration problem. Our particular goal here is to devise a polynomial space and polynomial delay algorithm, which is one of the efficient kinds of output-polynomial time enumeration algorithms, for solving the maximal pattern discovery problem for two-dimensional patterns with wild cards.

However, preliminary analyses of the problem shows that (Theorem 3) the set of maximal patterns can be exponentially succinct than the set of all frequent patterns, and furthermore that (Theorem 4) the counting version of maximal pattern problem is #P-hard problem indicating that enumeration is a not trivial problem. These results indicates that a simple generate-and-test approach based on frequent pattern discovery does not work for our maximal pattern problem.

Main results of this paper. We present an efficient algorithm MaxMatrix that, given an $n{\times}n$ input text $T[1..n, 1..n]$, enumerates all maximal two-dimensional square patterns $P \in \mathcal{M}$ with wild cards without duplicates in $O(|\Sigma|N\ell)$ exact time (or the *delay*) per maximal pattern using $O(M\ell)$ space, where $M = m^2$ is the size of $m \times m$ pattern P and $\ell = |\mathcal{L}(P)| = O(N)$ is the number of occurrences of P in T (Theorem 7).

Then, we modify this algorithm to remove the dependency of $|\Sigma|$ resulting another algorithm with $O(N\ell)$ exact time per pattern. Since the class of square patterns with wildcards exactly corresponds to the class of *two-dimensional patterns of arbitrary shape*, the above results applies to the latter class of patterns. As a special case, we observe that the maximal pattern enumeration problem for constant patterns is solvable in input polynomial time using dictionary structures (Theorem 1).

The key of our algorithm is a tree-shaped search route built over the space of all maximal patterns. To realize this idea, we develop a technique, called the prefix-preserving closure extension, by combining the previously proposed notions of *tail extension* and the *closure operations*.

Organization of this paper. The rest of this paper is organized as follows. In Section 2, we prepare the basic notions and definitions. Then in Section 3, we show some lower bounds for the number of maximal two-dimensional patterns. In Section 4, we give our polynomial space and polynomial delay enumeration algorithm MaxMatrix for the maximal patterns with wild cards. In Section 5, we discuss future directions.

2 Preliminaries

We prepare basic definitions and notations on two-dimensional pattern matching and discovery according to [3, 12, 13]. Then, we define our problem, the maximal two-dimensional pattern discovery problem by generalizeing the 1-dimensional maximal motif discovery problem [8, 15, 18, 17]

2.1 Basics

We denote by $\mathbb{N} = \{0, 1, 2, \ldots\}$ all natural numbers and by $\mathbb{Z} = \mathbb{N} \cup \{-x \mid x \in \mathbb{N}\}$ be all integers. For any integers $i_1, i_2, j_1 j_2 \in \mathbb{Z}$, we denote the interval $[i_1..i_2] = \{\, x \in \mathbb{Z} \mid i_1 \leq x \leq i_2 \,\}$ and the rectangular region $[i_1..i_2, j_1..j_2] = [i_1..i_2] \times [j_1..j_2]$. We sometimes write $[m]$ and $[m, n]$ to denote $[1..m]$ and $[1..m, 1..n]$, respectively.

Let Δ be an alphabet of symbols. An $n \times n$ *matrix over* Δ, denoted by $A[1..m, 1..n]$, is a mapping $A : [1..m, 1..n] \to \Delta$, where we denote by $dom(A) = [1..m, 1..n]$ the *domain* of A and by $A[i, j] \in \Delta$ the value of A at *point* (or *location*) $(i, j) \in dom(A)$. The subarray of A whose domain is $[i_1..i_2, j_1..j_2] \subseteq dom(A)$ is denoted by $B = A[i_1..i_2, j_1..j_2]$. For $m, n \in \mathbb{N}$, we denote by $\Delta^{m \times n}$ the class of all $m \times n$ matrices over Δ. In what follows, we consider only *square matrices*, which are matrices with $m = n$.

2.2 texts and patterns

Let $\Sigma = \{a, b, \ldots\}$ be an alphabet of *solid characters* (or *constant letters*), and $\circ \notin \Sigma$ be a distinguished letter, called the *wild card* (or *don't care*). A wild card $\circ$ matches any solid character $c \in \Sigma$ and also matches $\circ$ itself. More formally, we define a binary relation $\preceq$ over $\Sigma \cup \{\circ\}$, called the *subsumption* relation, as follows: $\circ \preceq \circ$, $\circ \preceq c$, and $c \preceq c$ for any $c \in \Sigma$.

Definition 1 (input text matrix). An *input text* is a $n \times n$ matrix $T[1..n, 1..n] \in \Sigma^{m \times n}$ of solid letters.

Definition 2 (two-dimensional pattern with wild cards). A *two-dimensional pattern with wild cards* is a $m \times m$ matrix $P[1..m, 1..m] \in (\Sigma \cup \{\circ\}^{m \times m})$, where its border region $border(P) = [1, 1..m] \cup [m, 1..m] \cup [1..m, 1] \cup [1..m, m]$ contains at least one solid letters in Σ.[3]

As a special case, a *two-dimensional constant pattern* is defined as a $m \times m$ matrix $P[1..m, 1..m] \in \Sigma^{m \times m}$ of solid letters only.

[3] This constraint is to avoid useless patterns wiht large borders ocuppied only by wild cards $\circ$.

Definition 3 (occurrence). We say that a square pattern $P[1..m, 1..m] \in (\Sigma \cup \{\circ\})^{m\times m}$ with wild cards *occurs in* another pattern $Q[1..n, 1..n] \in (\Sigma \cup \{\circ\})^{n\times n}$ at position $(x, y) \in dom(Q)$ if $P[i, j] \preceq Q[x + i - 1, y + j - 1]$ holds for every position $(i, j) \in [1..m, 1..m]$ in P. In this case, we write $P \preceq Q$. We define ε occurs in Q at any position (i, j).

The binary relation $\preceq$ is called the *subsumption* for patterns. For constant patterns, $\preceq$ reduces to the submatrix relation. If $P \preceq Q$, then, we say that either P *subsumes* Q, P is *contained by* Q, P is *more general than* Q, or Q is *more specific to* P. If $P \preceq Q$ but $Q \not\preceq P$, then we define $P \prec Q$ and say that either P is *properly contained by* Q and so on. We can see that if $P \preceq Q$ and $Q \preceq P$, then $P = Q$. Furthermore, $\preceq$ is a partial order over $\mathcal{P}$. The converse of Lemma 1 below does not hold in general.

Lemma 1. *For patterns with wild cards P, Q in an text T, if $P \preceq Q$ then $\mathcal{L}(P) \supseteq \mathcal{L}(Q)$.*

Definition 4 (two-dimensional pattern of arbitrary shape). A *two-dimensional pattern of arbitrary shape* (an *arbitrary pattern*) is a mapping $A : dom(A) \to \Sigma$ with a finite domain $dom(A) \subseteq \mathbb{Z}^2$.

In the above definition, the domain $dom(A)$ can be a non-rectangular region of $\mathbb{Z}^2$. We regard $(0, 0)$ as the origin of a pattern A of arbitrary shape. A pattern with arbitrary shape P *occurs* in an input text Q at location (x, y) if $P[i, j] = Q[x + i - 1, y + j - 1]$ for every $(i, j) \in dom(P)$. It is well known that patterns with wildcards can be simulated by two-dimensional pattern of arbitrary shape defined below [3] (detailed is omitted).

Definition 5 (location list). For an input text $T[1..n, 1..n]$ of size $n \geq 0$, the *location list* of pattern $P[1..m, 1..m]$ is the set $\mathcal{L}(x) \subseteq [1..n, 1..n]$ of all the positions $p = (x, y)$ in T at which P occurs.

The *frequency* of P in T is $|\mathcal{L}(x)|$. A a *quorum* (or a *minimum frequency threshold*) is any positive number $1 \leq \theta \leq n^2$. We say that pattern x is a *θ-frequent* in $T[1..n, 1..n]$ if $|\mathcal{L}(x)| \geq \theta$ holds.

2.3 Maximal Pattern Discovery Problem

Now, we define the class of maximal two-dimensional patterns by generalizing the class of 1-dimensional maximal motifs in [15, 18, 17] as follows. For any $(x_i, y_i) \in \mathbb{Z}^2$, we define $(x_1, y_1)+(x_2, y_2) = (x_1+x_2, y_1+y_2)$. A *displacement* is any two-dimensional vector $d = (x, y) \in \mathbb{Z}^2$ meaning the parallel motion by d. The *shift* of a location list $\mathcal{L} \subseteq \mathbb{Z}^2$ by displacement $d \in \mathbb{Z}^2$ is the set $\mathcal{L} + d = \{\, p + d \,|\, p \in \mathcal{L} \,\}$. We write $\mathcal{L} - x$ to represent the set $\mathcal{L} + y$ with $y = -x$. We regard two matrices P and Q as being *equivalent* if $\mathcal{L}(y) = \mathcal{L}(x) + d$ for some d.

Definition 6 (maximal two-dimensional patterns). A pattern with wild cards $P[1..m, 1..m]$ is *maximal* in text $T[1..n, 1..n]$ if for any pattern with wild cards $Q[1..m', 1..m']$ that properly contains P, i.e., $P \prec Q$, there is no displacement $d \in \mathbb{Z}^2$ such that $\mathcal{L}(y) = \mathcal{L}(x) + d$.

In other words, a pattern P is maximal in T iff there exists no pattern in T that properly contains P and is equivalent to P under the invariance with respect to parallel motion. Without loss of generality, we only consider maximal P with $|\mathcal{L}(P)| \geq 2$. We denote by $\mathcal{F}$ and $\mathcal{M}$ the sets of all frequent patterns and all maximal patterns, respectively. Clearly, $\mathcal{M} \subseteq \mathcal{F} \subseteq \mathcal{P}$ for any T and any θ. Now, we state our problem as follows.

Definition 7 (maximal pattern enumeration problem). Let $\mathcal{C}$ be a class of two-dimensional patterns. The *maximal pattern enumeration problem* for $\mathcal{C}$ is, given an input text $T[1..n, 1..n]$ of size $n \geq 0$, to enumerate all maximal two-dimensional patterns P in T within the class $\mathcal{C}$ without repetition.

Since the number of maximal patterns can be exponential in the input size n as we will see later, the discovery problem naturally becomes an enumeration problem.

2.4 Enumeration algorithms

We introduce terminology for enumeration algorithms according to [14, 19]. Throughout this paper, we adopt the standard RAM model [1] as a model of computation. An *enumeration algorithm* for an enumeration problem Π is an algorithm $\mathcal{A}$ that receives an input and prints all *solutions* without repetition.

Let N, M be the input and the output sizes on an input, and $T_{\mathcal{A}}$ be the total running time of $\mathcal{A}$ for computing all solutions. Among efficient enumeration algorithms, the weakest is an *output-polynomial* algorithm (P-OUTPUT) $\mathcal{A}$, where $T_{\mathcal{A}}$ is bounded by a polynomial $q(N, M)$. $\mathcal{A}$ is of *polynomial enumeration time* (P-ENUM) if the *amortized time* for each solution $x \in \mathcal{S}$ is bounded by a polynomial $p(N)$ in N, i.e., $T_{\mathcal{A}} = O(M \cdot p(N))$. The strongest is a *polynomial delay* (P-DELAY) algorithm (or a *exact polynomial time enumeration* algorithm) $\mathcal{A}$, where the *delay*, which is the maximum computation time between two consecutive outputs, is bounded by a polynomial $p(N)$ in the input size N. $\mathcal{A}$ is of *polynomial space* (P-SPACE) if the maximum size of its working space is bounded by a polynomial $p(N)$.

Our goal is to devise an polynomial space polynomial delay enumeration algorithm for solving the maximal pattern problem for the class of two-dimensional square patterns with wild cards.

3 Lower bound results

We show the following (an upper bound and) lower bound results of the number of maximal two-dimensional square patterns with wild cards.

The maximal pattern enumeration problem seems rather easy for the special case for the class of constant two-dimensinal patterns, where the number of maximal patterns in T is clearly bounded above by $O(n^3)$ ($O(n^2)$ patterns for each size $m = O(n)$). Acturally, the next theorem, which can be easily derived by combining the previous results on suffix trees for matrices [2, 13], says that the maximal pattern enumeration problem is solvable in input polynomial time for the class.

Theorem 1 (constant square patterns are easy to enumerate). *There exists an algorithm for enumerating all of n^3 maximal patterns in $O(n^4) = O(N^2)$ total time for the class of constant two-dimensinal patterns, where $N = n^2$ is the total input size.*

Proof: See Appendix A. □

On the other hand, we have the following result for maximal patterns with wild cards.

Theorem 2 (exponential lowerbound of maximal patterns). *There is an infinite series of input texts $T_0, T_1, T_2, \ldots,$ such that for every $k = 0, 1, 2, \ldots,$ the number $|\mathcal{M}|$ of maximal two-dimensinal patterns with wild cards in T_i is bounded below by $2^{\Omega(n)}$, where the size of T_i is $n \times n$.*

Proof: See Appendix B. □

As a technical lemma, we construct below a transformation from an well-studied problem of closed itemsets enumeration problem to our maximal two-dimensinal pattern enumeration problem.[4]

Lemma 2 (transformation from closed itemsets to maximal patterns). *For every transaction database $R = \{t_1, \ldots, t_m\}$ consisting of m transactions over n items of total size $N = mn$, there exists an $\tilde{n} \times \tilde{n}$ input matrix T with $\tilde{n}^2 = O(N + m^2)$ such that the number Q of closed itemsets in r is given by $Q = M + m - 1$, where M is the number of maximal patterns with wild cards in T.*

Proof: See Appendix C. □

Theorem 3 (succinctness of maximal patterns). *There is an infinite series of input strings $T_0, T_1, T_2, \ldots,$ such that for every $i \geq 0$ with quorum $\theta = \frac{1}{2}n$, the number $F = |\mathcal{F}|$ of all frequent patterns in T_i is exponential (more precisely $2^{\Omega(n)}$) in the input size n, while the number $M = |\mathcal{M}|$ of maximal patterns in T_i is polynomial in n, where $n = |T_i|$.*

Proof: The theorem is immediate from Lemma 2 above and the succinctness result for closed itemsets in [20], where the frequency constraint is not essential. □

[4] The maximal pattern discover problem in this paper correponds to the closed itemset enumeration for transaction databases [20] and to the maximal motif disocvery in sequences [16], but not to the maximal frequent itemset problem.

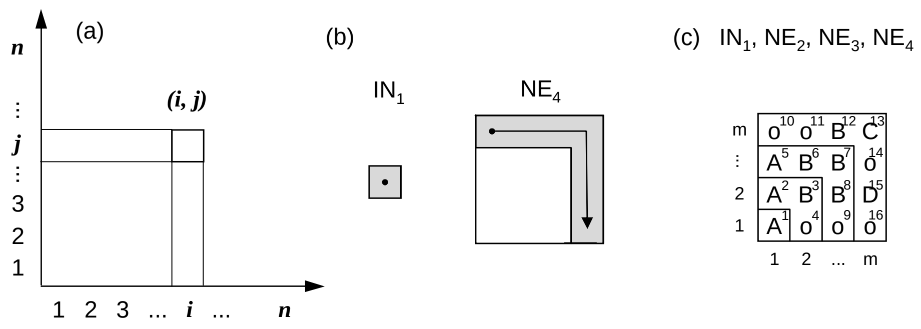


Fig. 1. (a) The coordinate system, (b) two types of shapes, and (c) the partition genereated by a series of shapes $IN_1 + NE_2 + NE_3 + NE_4$.

Theorem 4 (hardness of counting maximal patterns). *The maximal pattern enumeration problem for patterns with wild cards is #P-hard.*

Proof: By reduction from the maximal bipartite clique problem, Yan [21] showed that the closed pattern enumeration problem for itemsets is #P-hard. Hence, the result immediately follows from Lemma 2. □

From Theorem 3, we know that a popular generate-and-test algorithm based on frequent pattern discovery does not work in output-polynomial time (See also for Subsection 4.1). From Theorem 4, we also see that the maximal pattern enumeration problem is not an easy enumeration problem. This justifies our approach to considering output-sensitive enumeration algorithms.

4 Polynomial Space Polynomial Delay Algorithm for Maximal Two-Dimensional Pattern Discovery

4.1 Tail extension

The first step towards an efficient algorithm for maximal pattern enumeration is how to enumerate all two-dimensional maximal patterns without repetition. An obvious way is to store all the patterns discovered so far on main memory and then to make explicit duplicaition check against the stored patterns. However, this appoach requires $|\mathcal{M}| = 2^{\Omega(n)}$ memory, propotional to the output size, in the worst case.

We introduce a systematic way of partitioning matrices according to Amir and Farach [2] and Giancharlo and Grossi [13]. Let $A[1..m, 1..m] \in \Delta^{m\times m}$ be an $m \times m$ matrix. We assume the coordinate system of Fig. 1 (a) for both texts and patterns, which is later extended for $\mathbb{Z}^2$ as seen in Fig. 2. Let us denote by $A + B$ the union of two matrices such that $dom(A) \cap dom(B) = \emptyset$ as defined in the standard way. In what follows, we assume that the locations (i, j) of the plane $\mathbb{Z}^2$ are indexed as in Fig. 2.

We partition $A[1..m, 1..m]$ into a series of shapes $A = IN_1 + NE_2 + NE_3 + \ldots + NE_m$, where IN_1 is the 1×1 square matrix $A[1, 1]$, and NE_i is the L-shaped subarray

$NE_i(A) = A[1..i, i] + A[i, 1..i-1]$ for every $2 \le i \le m$ (Fig. 1 (b)). This series of shapes gives the unique partition of A (Fig. 1 (c)).

Furthermore, we order the positions in the matrix A in the order of $A = IN_1 + NE_2 + NE_3 + \ldots + NE_m$, from inside to outside. Then, inside each L-shape $NE_i(A)$, we order all of its $2i-1$ positions first *from left to right* then *top to bottom* as $[1,i] < [2,i] < \cdots < [i-1,i] < [i,i] < [i,i-1] < \cdots < [i,1]$. This gives a total order, denoted by $<_{\mathrm{ord}}$, over m^2 positions in the square $[1..m, 1..m]$, where the least position is the left bottom corner $[1,1,]$ and the greatest is the right top corner $[m,m]$ as in Fig. 1 (c).

The *tail* of $m \times m$ matrix $A \in (\Sigma \cup \{\circ\})^{m \times m}$ with wild cards is the largest position $tail(A)$ in the order $<_{\mathrm{ord}}$ that is labeled by a constant letter, i.e., $tail(A) = \max_{<_{\mathrm{ord}}}\{\ [i,j] \in [1..m, 1..m] \mid A[i,j] \in \Sigma\ \}$. For example, the tail of the array in Fig. 1 is the cell with order number 15 at position $[4,2]$ with letter `D`. We denote by $P[\,(i,j) \leftarrow c\,]$ the *matrix obtained from P by substituting letter $c \in \Sigma \cup \{\circ\}$ at location $(i,j) \in dom(P)$.*

Definition 8 (tail extension of a matrix). A $m \times m$ matrix Q is a *tail extension* of a $m' \times m'$ matrix P $(m' \le m)$ if $Q = P[(i,j) \leftarrow c]$ for some solid character $c \in \Sigma$ and some position $(i,j) \in \mathbb{N}^2$ such that $(i,j) >_{\mathrm{ord}} tail(A)$, where $m' = \max\{i,j\}$.

We consider the directed graph $\mathcal{T}_{\mathrm{tail}} = (\mathcal{P}, \mathcal{E})$, called an *enumeration graph*, where the node set $\mathcal{P}$ is the set of all patterns occuring in T at least once and $\mathcal{E}$ is the edge set such that $(P,Q) \in \mathcal{E}$ iff Q is a tail extension of P. Then, we can see that $\mathcal{T}_{\mathrm{tail}}$ is a spanning tree for all members of $\mathcal{P}_m$ since $P = Q[\,tail(Q) \leftarrow \circ\,]$ obtains P as the unique *parent* pattern of Q and $|P|_\Sigma < |Q|_\Sigma$. Thus, we have the next lemma.

Lemma 3. *There is an algorithm that enumerates all matrix patterns with wild cards, up to maximum size $m \ge 0$, without repetition in $O(1)$ time per patterns based on the depth-first search over $\mathcal{T}_{\mathrm{tail}}$.*

Proof: By traversing this tree starting from the empty matrix $\perp$ as the unique root using backtracking, the claimed time complexity is obtained. □

From the above lemma, the following approach for maximal pattern enumeration is possible: it is first to enuemerate all patterns with frequency at least two, and then test if each candidate pattern is maximal and if it is not duplicated. However, From the succinctness result in Theorem 3, this method takes exponential time in the number of maximal patterns in the worst case. Thus, the tail extension alone does not help to give even output polynomial time algorithms.

4.2 Merge and Closure

The second problem to solve is how to efficently check maximality of patterns without storing all discovered patterns so far. First, we extend the merge and the closure operations, which are originally introduced in [5, 9, 17, 18], then we give a characterization of maximal two-dimensional patterns with wildcards. We start with technical definitions.

Definition 9. We define a pseudo two-dimensional matrix to be a mapping $P : \mathbb{Z}^2 \to \Sigma \cup \{\circ\}$ with infinite domain $dom(P) = \mathbb{Z}^2$, where $P[i,j] \in \Sigma$ is defined only for finitely many points $(i,j) \in dom(P)$ and other points are filled with wild cards $\circ$.

We define the *shift of* P by a displacement $d = (x, y) \in \mathbb{Z}^2$, denoted by $(S + d)$, as the pseudo matrix such that $(S + d)[i, j] = S[i - x, j - y]$ for every $(i, j) \in \mathbb{Z}^2$. By this transformation, the origin $o = (0, 0)$ of P moves to the point $d = (x, y)$.

For each two-dimensional pattern $P[1..m, 1..m]$ with wild cards, its *infinite version* is the two-dimensional pseudo matrix $\lfloor P \rfloor : \mathbb{Z}^2 \to \Sigma \cup \{\circ\}$, where the value $\lfloor S \rfloor[i, j] = S[i, j]$ if $(i, j) \in dom(S)$ and $\lfloor S \rfloor[i, j] = \circ$ otherwise. Conversely, for each two-dimensional pseudo matrix $S : \mathbb{Z}^2 \to \Sigma \cup \{\circ\}$, we define the *trimmed version* of S by the $m \times m$ matrix $\lceil S \rceil = S[x..x + m - 1, y..y + m - 1] - (x, y)$ with wildcards, where $x_{\min}$ and $y_{\min}$ ($x_{\min}$ and $y_{\min}$, resp.) are the smallest (largest, resp.) x-coordinates and y-coordinates for the locations of solid letters in S.

Next, we define the merge operator $\oplus$ for two-dimensional pseudo matrices as follows. For letters $a, b \in \Sigma$, we define $a \oplus a = a$ and $a \oplus \circ = \circ \oplus a = a \oplus b = \circ$ if $a \neq b$. This operator $\oplus$ is associative and commutative. The *merge* of two-dimensional pseudo matrices $X, Y : \mathbb{Z}^2 \to \Delta$ is the two-dim pseudo matrix $X \oplus Y : \mathbb{Z}^2 \to \Delta$ defined by $X \oplus Y[i, j] = X[i, j] \oplus Y[i, j]$ for every point $(i, j) \in \mathbb{Z}^2$.

Definition 10. Let T be an input text. The *merge* of a location list $\mathcal{L} = \{d_1, \ldots, d_{|\mathcal{L}|}\} \subseteq dom(T)$ is the pseudo string $\bigoplus \mathcal{L} : \mathbb{Z}^2 \to \Delta$ defined by

$$\bigoplus \mathcal{L} = (\lfloor T \rfloor - d_1) \oplus \cdots \oplus (\lfloor T \rfloor - d_{|\mathcal{L}|}).$$

Then, its trimmed version is a two-dimensional pattern given by $P = \lceil \bigoplus \mathcal{L} \rceil \in \mathcal{P}$.

Lemma 4. *Let $\mathcal{L}, \mathcal{L}'$ be any location lists and $d \in \mathbb{Z}^2$ be any displacement.*

1. *If $\mathcal{L} \supseteq \mathcal{L}' + d$ then $\bigoplus \mathcal{L} \preceq \bigoplus \mathcal{L}'$.*
2. *$\bigoplus \mathcal{L} = \bigoplus (\mathcal{L} + d)$ for any integer d.*

Proof: 1: For sets $A, A' \subseteq \Delta$ of letters, if $A \supseteq A'$ then $\oplus A \preceq \oplus A'$. Let $p = (i, j)$ be any position and put $A = \{\, T[p + d] \mid d \in \mathcal{L} \,\}$ and $A' = \{\, T[p + d] \mid d \in \mathcal{L}' \,\}$. Since $A \subseteq A'$, we can see that $\bigoplus \mathcal{L}[p] = \oplus A \preceq \oplus A' = \bigoplus \mathcal{L}'[p]$. This proves the property. 2: Immediately from the above.

Lemma 5. *Let $\mathcal{L} \subseteq [1..n, 1..n]$ be any location list in T. Then, $P = \lceil \bigoplus \mathcal{L} \rceil$ is a maximal pattern.*

The following definition is a generalization of the closure operator for itemsets [20] and sequence motifs [17, 18].

Definition 11 (closure operation). Given a pattern x and an input string s, the maximal pattern $Clo(x) = \lceil \bigoplus \mathcal{L}(x) \rceil$ is called the *closure of x on s.*

Lemma 6. *The closure $Clo(P)$ of a pattern P is unique and computable in $O(\ell n^2) = O(\ell N)$ time from P and $\mathcal{L}(P)$, where $N = n^2$ and $\ell = |\mathcal{L}(P)| \leq n$.*

Lemma 7 (properties of closure). *Let P, Q be any patterns occurring in T and P, Q be any location lists.*

1. $P \preceq Clo(P)$.
2. $Clo(P) = Clo(Clo(P))$.
3. $\mathcal{L}(P) = \mathcal{L}(Clo(P))$.
4. *If $P \preceq Q$ then $Clo(P) \preceq Clo(Q)$.*
5. *$Clo(P)$ is the unique maximal member in $\{ Q \mid P$ and Q are equivalent on $T \}$, the equivalence class of patterns with the same location lists.*

The following theorem is a two-dimensional version of a therem in [17].

Theorem 5 (characterization of maximal patterns). *Let θ a quorum and $P[1..m, 1..m]$ be a pattern in a text $T[1..n, 1..n]$. Then, the following (i)–(iii) are equivalent:*

(i) *P is a maximal pattern.*
(ii) $P = \lceil \bigoplus \mathcal{L} \rceil$ *for some* $\mathcal{L} \subseteq [1..n, 1..n]$.
(iii) $P = Clo(P)$.

Let $\overline{\perp}$ be the (smallest) maximal pattern equivalent to the empty $\perp$ pattern, called the *root maximal pattern*. Recall that in Lemma 1, the subsumption $\preceq$ and the inclusion $\supseteq$ between location lists do not coincide for two-dimensional patterns with wild cards. The following lemma says that they coincide for maximal patterns.

Lemma 8. *For maximal patterns $P, Q \in \mathcal{M}$ on an input text T, $P \preceq Q$ iff $\mathcal{L}(P) \supseteq \mathcal{L}(Q)$.*

A possible use of the closure operation is to define the *closure extension Q* of a square matrix $P[1..m, 1..m]$ by $Q = Clo(P[(i, j) \leftarrow c])$. Then, repeating the closure extension iteratively patterns starting from the root maximal pattern $\overline{\perp}$, this method acturally generates all maximal patterns. Since the enumeration graph for the patterns based on the closure extension is a DAG but not a tree unlike Lemma 3, we need explicit duplicate check storing all discovered patterns using the memory proportional to the output size M. Thus, this approach, may result an output polynomial time algorithm, but cannot be a polynomial delay polynomial space algorithm. Thus, the closure extension alone still does not help to solve our problem.

4.3 Prefix Preserving Extension for Two-dimensional Patterns

Now, we give a new extension method that grows a given parent maximal pattern to generate more specific children maximal patterns by combining the tail extension and the closure operator above. In this subsection, we adopt the framework of the *reverse search* of Avis and Fukuda [10] in order to build an enuemration algorithm based on backtracking over a tree-shaped search route.

An enumeration tree for $\mathcal{M}$. First, we construct an enumeration tree $\mathcal{T}_{\mathrm{ppc}} = (\mathcal{M}, \mathcal{E}_{\mathrm{ppc}})$ for all maximal two-dimensional patterns of $\mathcal{M}$ appearing in an input text T. To construct the tree $\mathcal{T}_{\mathrm{ppc}}$, we associate with each non-bottom pattern $Q \in \mathcal{M}\backslash\{\bot\}$ the unique parent pattern $P = Pa(Q) \in \mathcal{M}$ of smaller size as follows.

Let $Q[1..m, 1..m]$ be an $m \times m$ pattern with wild cards. Recall that the indexes of Q is totally ordered by $<_{\mathrm{ord}}$ according to the shape sequence $IN_1, SE_2, SE_3, \ldots, SE_m$ as in Fig. 1 (c). Given any position (i, j), the *prefix with tail* (x, y) is the subset $prefix(i, j) = \{\ (i, j) \in [1..m, 1..m] \mid (0, 0) \leq_{\mathrm{ord}} (i, j) \leq_{\mathrm{ord}} (x, y)\ \}$, where $m = \max\{x, y\}$. Then, we denote by $Q \cap prefix(x, y)$ the submatrix of Q with the domain $dom(Q) \cap prefix(x, y)$.

Definition 12 (core position). The *core position* of Q, denoted by $core\_p(Q)$, is the smallest position that the corresponding prefix preserves the location list of Q, that is,

$$core\_p(Q) = \min\{\ (x, y) \in dom(Q) \mid \mathcal{L}(Q \cap prefix(x, y)) = \mathcal{L}(Q)\ \}.$$

Then, we call the *core prefix* of Q the pattern $R = prefix(x, y) \cap Q$.

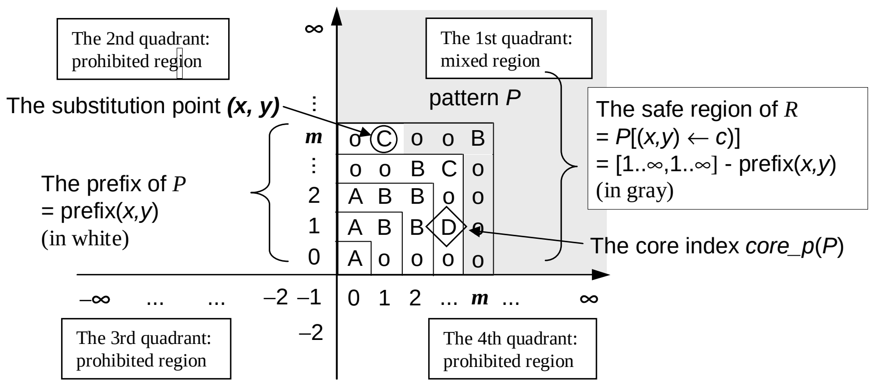
Definition 13 (parent in ppc-extension). Let Q be an $m \times m$ pattern with wild cards and $(x, y) = core\_p(Q)$ is the core position of Q. Then, the parent of Q is the pattern $Pa(Q)$ given by

$$Pa(Q) = Clo(CP[tail(CP) \leftarrow \circ]),$$

where $CP = prefix(x, y) \cap Q$ is the core prefix of Q and $CP[tail(CP) \leftarrow \circ]$ is the pattern obtained from CP by removing its tail by substitution a wild card for the tail position.

On the other words, $Pa(Q)$ is the two-dimensional pattern obtained from the core prefix of Q by removing the solid letter at the tail position. Now, we have the parent-child relation for $\mathcal{T}_{\mathrm{ppc}} = (\mathcal{M}, \mathcal{E}_{\mathrm{ppc}})$. $\mathcal{M}$ is the set of all maximal patterns appearing in an input text T, and $\mathcal{E}$ is the edge set such that $(P, Q) \in \mathcal{E}$ iff P is a parent of Q in ppc-extension (Definition 13).

Lemma 9. *The* enumeration graph $\mathcal{T}_{\mathrm{ppc}}$ *for $\mathcal{M}$ is a spanning tree for $\mathcal{M}$.*



The 2nd quadrant: prohibited region
The 1st quadrant: mixed region
pattern *P*
The substitution point *(x, y)*
The safe region of *R* = *P*[(*x*,*y*) ← *c*)] = [1..∞,1..∞] - prefix(*x*,*y*) (in gray)
The prefix of *P* = prefix(*x*,*y*) (in white)
The core index *core_p*(*P*)
The 3rd quadrant: prohibited region
The 4th quadrant: prohibited region

Fig. 2. The safe region and the prohibited region in the prefix-preserving closure extension (ppc-extension).

Proof: Since the core prefix and the closure are uniquely determined from their inputs, the parent $Pa(Q)$ is well-defined and unique. Thus, each node Q other than the bottom $\perp$ has the unique parent. On the other hand, since the core prefix is the shortest prefix preserving the location list $\mathcal{L}(Q)$, $Pa(Q)$ must have a properly larger location list, i.e., $\mathcal{L}(Pa(Q)) \supset \mathcal{L}(Q)$. From Lemma 8, we have that $Pa(Q) \prec Q$, and that $|Pa(Q)|_\Sigma < |Q|_\Sigma$. Therefore, we know that $\mathcal{T}_{\mathrm{ppc}}$ has no infinitely descending sequences w.r.t. $\succ$. Combining the above arguments, we proved the lemma. □

A depth-first search algorithm for $\mathcal{M}$. Now, we have an enumeration tree $\mathcal{T}_{\mathrm{ppc}}$ for maximal two-dimensional patterns in $\mathcal{M}$. However, it is a bit complicated to perform a depth-first search on $\mathcal{T}_{\mathrm{ppc}}$ because the edges of the tree are directed from a child to the parent in the reverse direction. Thus, the remaining task is to give an efficient procedure to, given a parent maximal pattern P, enumerate all of its children Q uniquely. This is achieved by the following PPC-extension, which is obtained by combining the tail extension and the closure operation.

Let P be any maximal pattern. If we substitute any solid letter $c \in \Sigma$ for any location $(x, y) \in dom(x, y)$ of a wild card, i.e., $P[x, y] = \circ$, then from the maximality of P, we see that the resulting pattern $R = P[\,(x, y) \leftarrow c\,]$ has a properly smaller location list $\mathcal{L}(R) \subset \mathcal{L}(P)$ than the original one. Then, we compute the merge $Q = \oplus\mathcal{L}(R)$ of $\mathcal{L}(R)$. By Theorem 5, Q is ensured to be a maximal pattern in T, and from Lemma 8, Q is properly more general than P, i.e., $P \prec Q$. However, the wrong choice of the substituted position (x, y) causes the duplicated generation of the same pattern through exponentially many distinct paths from the root. Thus, we need careful application of the substitution and the merge.

By definition, $Q = \oplus\mathcal{L}(R)$ above is a pseudo pattern $Q : \mathbb{Z}^2 \to (\Sigma \cup \{\circ\})$ whose domain $dom(Q)$ may contain some positions with negative coordinates. Let $[1..\infty, 1..\infty] = \mathbb{N}^2$ be the first quadrant of the plane $\mathbb{Z}^2$ as an open region. Let $(x, y) \in dom(P)$ be the index of P to be subsituted. Then, the *safe region* for ppc-extension is defined by the set $[1..\infty, 1..\infty] - \mathit{prefix}(x, y) = \{\ (i, j) \in \mathbb{N}^2 \mid (i, j) >_{\mathrm{ord}} (x, y)\ \}$, where the total order $<_{\mathrm{ord}}$ is naturally extended for $[1..\infty, 1..\infty]$.

Definition 14 (ppc-extension). Let $P \in \mathcal{M}$ be any maximal two-dimensional pattern in T. Then, a two-domensional pattern Q is said to be a *prefix-preserving closure extension* (*ppc-extension*, for short) of P if the following (i)–(iii) are satisfied:

(i) *Closure extension*: Q is obtained by substitution of a solid character and application of the closure operation, i.e, $Q = Clo(P[\,(x,y) \leftarrow c\,]) = \lceil \oplus\mathcal{L}(P[\,(x,y) \leftarrow c\,]) \rceil$ for some $c \in \Sigma$ and some position $(x,y) \in dom(P)$ satisfying $P[x,y] = \circ$.

(ii) *Tail extension*: The position (x,y), called the substitution point, properly follows the core position of P, i.e., $(x,y) >_{\mathrm{ord}} core\_p(P)$.

(iii) *Prefix checking*: The extension adds new solid letters to P only to its safe region, i.e., $dom(\oplus\mathcal{L}(P[\,(x,y) \leftarrow c\,])) \subseteq dom(P) \cup \{\,(i,j) \in \mathbb{N}^2 \mid (i,j) >_{\mathrm{ord}} (x,y)\,\}$.[5]

Lemma 10. *Let T be an $n \times n$ input text. Then, all ppc-extensions of a maximal $m \times m$ pattern P can be computed in $O(|\Sigma|N^2)$, where $N = n^2$.*

Lemma 11. *Let P be any maximal pattern and $Q = \lceil \oplus\mathcal{L}(P[\,(x,y) \leftarrow c\,]) \rceil$ be a ppc-extension of P. Then, (x,y) is the core position of Q.*

Theorem 6 (correctness of ppc-extension). *For any maximal two-dimensional pattern P, Q with wild cards such that $Q \neq \overline{\bot}$. Then, (1) $P = Pa(Q)$ if and only if (2) $Q = \lceil \oplus\mathcal{L}(P[\,(x,y) \leftarrow c\,]) \rceil$ is a ppc-expansion of P for some solid letter $c \in \Sigma$ and for some location $(x,y) \in dom(P)$ of wild card $\circ$. Furthermore, there exists exactly one pair of (x,y) and c satisfying condition (2) for each Q.*

Proof: A sketch of proof. (*only-if part*) If $P = Pa(Q)$ then $P = Clo(CP[tail(CP) \leftarrow \circ])$ for the core prefix CP of P. If $c = CP[tail(CP)]$ then we can prove that $Q = P[tail(CP) \leftarrow c]$ is a ppc-extension of P. (*if part*) If Q is a ppc-extension of P then $Q = \lceil \oplus\mathcal{L}(P[\,(x,y) \leftarrow c\,]) \rceil$ for some solid letter $c \in \Sigma$ and position $(x,y) >_{\mathrm{org}} core\_p(P)$. Then, we can see that $(x,y) = core\_p(Q)$. Furthermore, we can show that $Pa(Q) = Clo(Q \cap prefix(x,y)[(x,y) \leftarrow \circ]) = P$. This completes the proof. □

Based on Theorem 6, we present in Fig. 3 our algorithm MaxMatrix that enumerates all maximal two-dimensional patterns in a given input text by the depth-first search over the enumeration tree $\mathcal{T}_{\mathrm{ppc}}$ for $\mathcal{M}$ applying the ppc-extension to each maximal patterns. From the aruguments so far, we have the main result of this paper. To diminish the factor on the delay proportional to the depth of the enumeration tree, we use the technique of Uno [19].

Theorem 7 (main theorem). *Given an $n \times n$ input text T, the algorithm* MaxMatrix *in Fig. 3 enumerates all maximal two-dimensional square patterns P with wild cards within $\mathcal{M}$ in $O(|\Sigma|N\ell)$ amortized time per maximal pattern with $O(M\ell)$ space and $O(|\Sigma|N\ell)$ delay, where $N = n^2$ is the total input size, $M = m^2$ is the size of $m \times m$ pattern P being enumerated, and $\ell = |\mathcal{L}(P)| = O(N)$ is the number of occurrences of P.*

[5] Note that we did not apply the trimming operation $\lceil\cdot\rceil$ in order to avoid shifting the origin of $\oplus\mathcal{L}(R)$ until this cheking step is done.

```
Algorithm MaxMatrix($\Sigma$: alphabet of constants, $T[1..n,1..n]$: input text matrix)
0  $\overline{\perp} = Clo(\perp)$;                        //the root pattern.
1  call Expand($\overline{\perp}, (0,0), T$);              //$(0,0) <_{\rm ord} (i,j)$ for any $(i,j)$.

Procedure Expand($P$: motif, $(p_x,p_y)$: core position, $T$: input matrix)
1  //assumption: $core\_p(P) = (p_x,p_y)$ is ensured.
2  output $P$;
3  foreach $(x,y) \in [1..n,1..n]$ s.t. $(x,y) >_{\rm ord} (p_x,p_y)$ do
4    foreach  $c \in \Sigma$ do begin
5      $R = \oplus\mathcal{L}(P[\,(x,y) \leftarrow c\,])$;            //ppt-extension.
6      $SafeRegion(x,y) = \{\ (i,j) \in [1..n,1..n] \mid (i,j) >_{\rm ord} (x,y)\ \}$;
7      if $(dom(\oplus\mathcal{L}(R)) \backslash dom(P)) \subseteq SafeRegion(x,y)$ then
8        $Q = \lceil R \rceil$;
9        call Expand($Q, (x,y), T$);
10   end for
```

Fig. 3. A polynomial space polynomial delay enumeration algorithm for maximal two-dimensional patterns $\mathcal{M}$ in an input text matrix.

Corollary 1. *The maxmal pattern enumeration problem is enumerable in polynomial space and polynomial delay in the input size for the following classes*:

- *the class of two-dimensional square patterns with wild cards.*
- *the class of two-dimensional patterns of arbitrary shape.*

Proof: The second result follows from the first result since patterns of the second class can be simulated by the patterns of the first class. □

```
Procedure Expand($P$: motif, $(p_x,p_y)$: core position, $T$: input matrix)
  output $P$;
  Initialize the hash $\mathcal{L} \subseteq [1..n,1..n] \times \Sigma$;
  foreach location $(i,j) \in \mathcal{L}(P)$ do
    foreach $(x,y) \in [1..n,1..n]$ s.t. $(x,y) >_{\rm ord} (p_x,p_y)$ do
      $\mathcal{L}(x,y,c) = \mathcal{L}(x,y,c) \cup \{(i+x, j+y)\}$, where $c = T[i+x, j+y]$;
  foreach $(x,y,c) \in \mathcal{L}$ do
    $\mathcal{L}(P) = \mathcal{L}(x,y,c)$;
    Compute the ppc-extension $Q$ of $P$ with subsutitution $[\,(x,y) \leftarrow c\,]$ as line 5 to line 8 of the original;
    Expand($Q, (x,y), T$);
  end for
```

Fig. 4. A modified version of the recursive procedure *Extend*

Removing the dependency on the alphabet size. We show that the dependency of the delay time on $|\Sigma|$ of the algorithm MaxMatrix in Theorem 7 is remove by a modification based on the *occurrence-deliver* technique in Uno *et al.* [20].

Corollary 2 (modified algorithm). *The algorithm* MaxMatrixModified *of Fig. 4 enumerates all maximal two-dimensional square patterns P with wild cards within $\mathcal{M}$ in $O(N\ell)$ amortized time per maximal pattern with $O(M\ell)$ space and $O(N\ell)$ delay, where the parameters N, M, and ℓ are the same as Theorem 7.*

5 Conclusion

In this paper, we study the problem of enumerating all maximal square patterns with wild cards in a given $n \times n$ input matrix of symbols without duplicates. We presented a polynomial space and polynomial delay algorithm, MaxMatrix, for the problem by adopting the framework of reverse search [10].

We considered a version of pattern discovery problem where only parallel motion is allowed as geometrical transformation. For two-dimensional string matching, there are a number of variations such matching with rotation, enlargement, transpose, and with mismatch and approximate match [3, 12]. Thus, it is a future research to consider two-dimensional pattern discovery problems with the above extensions. In this paper, we used a class of matrix suffix trees [13] for maximal constant pattern problem. The speedup of the presented algorithm incorporating matrix suffix trees will be another research issue.

Boros *et al.* [11], Apostolico, Comin, and Parida [6], Uno and Arimura [8, 9, 20] studied efficient solutions of maximal pattern enumeration problems for some classes of structured objects other than two-dimensinal matrices. It would be an interesting problem to develop a uniform algorithmic scheme to construct polynomial space polynomial delay enumeration algorithms for a class of such structured objects.

We are presently implementing the algorithm MaxMatrix developed in this paper. We plan to make experimental study about the practical performance and the utility for applications, and will include some results of preliminary experiments in the complete paper.

Acknowledgments

The authors would like to thank Matteo Comin for providing the materials on their output-polynomial time algorithm for maximal motif discovery for sequences. The first author is grateful to Shin-ichi Shimozono for discussion on geometric discovery, and to Tatsuya Asai, Takuya Kida, Seishi Okamoto, Masayuki Takeda, Ayumi Shinohara, Shin-ichi Minato, and Thomas Zeugmann for discussions on pattern matching and pattern discovery in non-standard texts and semi-structured data mining.

References

1. A. V. Aho, J. E. Hopcroft, J. D. Ullman, The Design and Analysis of Computer Algorithms, Addison-Wesley, 1974.
2. A. Amir, M. Farach, Two-dimensional dictionary matching, *Information Processing Letters*, 44, 233–239, 1992.
3. A. Amir, M. Farach, Two Dimensional Matching, In *A. Apostolico, Z. Galil (eds), Pattern Matching Algorithms*, Chapter 9, Oxford University Press, 1997.
4. A. Apostolico, Z. Galil, Pattern Matching Algorithms, Oxford University Press, 1997.
5. A. Apostolico and L. Parida, Compression and the wheel of fortune, In *Proc. the 2003 Data Compression Conference (DCC'03)*, IEEE, 2003.

6. A. Apostolico, M. Comin, L. Parida, Mining, Compressing and Classifying with Extensible Motifs Manuscript, 2006. (to appear in BMC Algorithms for Molecular Biology, 2006.)
7. A. Apostolico, M. Comin, L. Parida, Conservative extraction of over-represented extensible motifs, Bioinformatics, 21, Suppl. 1, 9-18, 2005.
8. H. Arimura, T. Uno, A polynomial space polynomial delay algorithm for enumeration of maximal motifs in a sequence, In *Proc. the 16th Annual International Symposium on Algorithms and Computation (ISAAC'05)*, LNCS 3827, Springer, 2005.
9. H. Arimura, T. Uno, An output-polynomial time algorithm for mining frequent closed attribute trees, In *Proc. ILP'05*, LNAI 3625, 1–19, August 2005.
10. D. Avis, K. Fukuda, Reverse search for enumeration, Discrete Applied Mathematics, 65(1–3), 21–46, 1996.
11. E. Boros V. Gurvich, L. Khachiyan, K. Makino, The complexity of generating maximal frequent and minimal infrequent sets, In *Proc. STACS '02*, LNCS, 133-141, 2002.
12. M. Crochemore and W. Rytter, jewels of Stringology: Text Algorithms, World Scientific, 2003.
13. R. Giancarlo, R. Grossi, Suffix Tree Data Structures for Matrices, In *A. Apostolico, Z. Galil (eds), Pattern Matching Algorithms*, Chapter 10, Oxford University Press, 1997.
14. L. A. Goldberg, Polynomial space polynomial delay algorithms for listing families of graphs, In *Proc. the 25th STOC*, ACM, 218–225, 1993.
15. L. Parida, I. Rigoutsos, A. Floratos, D. Platt, and Y. Gao, Pattern discovery on character sets and real-valued data: linear bound on irredundant motifs and effcient polynomial time algorithm, In *Proc. the 11th SIAM Symposium on Discrete Algorithms (SODA'00)*, 297–308, 2000.
16. L. Parida, I. Rigoutsos, D. E. Platt, An Output-Sensitive Flexible Pattern Discovery Algorithm. In *Proc. CPM'01*, LNCS 2089, 131–142, 2001.
17. J. Pelfrêne, S. Abdeddaïm, and J. Alexandre, Extending Approximate Patterns, In *Proc. CPM'03*, LNCS 2676, 328–347, 2003.
18. N. Pisanti, M. Crochemore, R. Grossi, and M.-F. Sagot, A basis of tiling motifs for generating repeated patterns and its complexity for higher quorum, In *Proc. MFCS'03*, LNCS 2747, 622–631, 2003.
19. T. Uno, Two general methods to reduce delay and change of enumeration algorithms, NII Technical Report, NII-2003-004E, April 2003.
20. T. Uno, T. Asai, Y. Uchida, H. Arimura, An efficient algorithm for enumerating closed patterns in transaction databases, In *Proc. DS'04*, LNAI 3245, 16-30, 2004.
21. G. Yang, The complexity of mining maximal frequent itemsets and maximal frequent patterns, In *Proc. the 10th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'04)*, 344 - 353, 2004.

Appendix

This appendix is not included in the submission and can be ignored for the review.

A Proof of Theorem 1

Theorem 1 (constant square patterns are easy to enumerate): There exists an algorithm for enumerating all of n^3 maximal patterns in $O(n^2)$ total time for the class of constant two-dimensinal patterns is solvable in input polynomial time.

Proof: Let $N = n^2$. For each direction $\delta \in \{NE, NW, SE, SW\}$, we construct L-suffix tree $\mathcal{T}^\delta$ for an input text $T[1..n, 1..n]$ in $O(N \log n)$ time [2, 13]. For constant square patterns, we can show that if $P[m, m]$ is maximal in T with at least two locations in T then P has the corresponding explicit node in $\mathcal{T}^\delta$ for each δ. We enumerate all of such candidate patterns P by traversing $O(N)$ explicit nodes of one tree, say, $\mathcal{T}^{NE}$, and for each P, test if P has an explicit locus in $\mathcal{T}^\delta$ for each $\delta \in \{NW, SE, SW\}$ in $O(m^2) = O(N)$ per pattern P. Overall, this straightfoward method takes $T_{\text{total}} = O(N \log n + N^2) = O(n^4)$ time to solve the maximal pattern problem for constant square patterns. □

B Proof of Theorem 2

Theorem 2 (exponential lowerbound of maximal patterns): There is an infinite series of input texts $T_0, T_1, T_2, \ldots$, such that for every $k = 0, 1, 2, \ldots$, the number $|\mathcal{M}|$ of maximal two-dimensinal patterns with wild cards in T_i is bounded below by $2^{\Omega(n)}$, where the size of T_i is $n \times n$.

Proof: Let $k \geq 0$, $n = k^2$ and $T = T_k$ be an $n \times n$ matrix defined as follows. First, we partition T into k^2 square submatrices $B_1, \ldots, B_{k^2}$ of size $k \times k$, and for each B_i, we number the $k \times k$ locations in B_i as $p^i_1, \ldots, p^i_{k^2}$. Then, we define $B_i[p^i_1] = \$$, and for each $j > 1$, $B_i[p^i_j] = \texttt{1}$ if $i = j$ and $B_i[p^i_j] = \texttt{0}$ otherwise. Next, we define the set $\mathcal{X}$ consisting of the patterns $P : [1..k, 1..k]$ obtained from arbitrary $k \times k$ patterns over the alphabet $\{\texttt{0}, \circ\}$ by putting $P[1, 1] = \$$. Then, we can see that $\mathcal{L}(P) = \{\, p^i_j \mid 1 \leq i \leq k^2, P[p_j] = \circ \,\}$. Since all 2^{k^2-1} patterns have mutually distinct location lists in T, there at least the same number of maximal patterns in T. □

C Proof of Lemma 2

Lemma 2 (transformation from closed itemsets to maximal patterns): Let $\mathcal{I} = \{1, \ldots, n\}$ be items. For every transaction database $R = \{t_1, \ldots, t_m\}$ consisting of m transactions over n items of total size $N = mn$, there exists an $\tilde{n} \times \tilde{n}$ input matrix T with $\tilde{n}^2 = O(N + m^2)$ such that the number Q of closed itemsets in r is given by $Q = M + m - 1$, where M is the number of maximal patterns with wild cards in T.

Proof: We assume that $m \leq n$ and all members of R are mutually distinct. Suppose that $n = k^2$ and $m = \ell^2$ for some $k, \ell \geq 0$. Let $\tilde{n} = k\ell + 1/2\ell^2$. $\Sigma = \{\, \texttt{1}_j, \texttt{0}^i_j, \texttt{\#}^i_{i'} \mid \texttt{a} \in$

$\mathcal{I}, 1 \le j \le n, 1 \le i, i' \le \tilde{n}$ }. We define a text $T[\tilde{n} \times \tilde{n}]$ as follows. First, we number the positions in $[1.., k, 1..k]$ as $p_1, \ldots, p_{k^2}$. For each t_i, we define the $k \times k$ matrix A_i as $A_i[p_j] = \mathtt{1}_j$ if $x_j \in t_i$ and $A_i[p_j] = \mathtt{0}_j$. otherwise. We arrange k^2 matrices A_i's in T in k rows and k columns, where i-th column and $(i+1)$-th column of A's are separated by a #-column VD_i of width i and j-th row and $(j+1)$-th row HD_j of A's are separated by a #-row of height i. For all $(i, j) \in B = (VD_i)_{i=1}^m \cup (HD_j)_{j=1}^m$, we assign mutually distinct value $T[(i, j)] = \mathtt{\#}_j^i$. Then, we can show that for a square pattern P with $|\mathcal{L}(P)| \ge 2$, any of its occurrence is entirely contained in some submatrix A_i. We can show that the number Q_2 of closed itemsets appearing at least twice in r equals the number M_2 of maximal matrix appearing at least twice in T. Since $Q = Q_2 + m$ and $M = M_2 + 1$ and $\tilde{n}^2 = O(k^2\ell^2 + k\ell^3 + 1/4\ell^4)$, the result follows from $Q - m = M - 1$.
□