City University of New York (CUNY)
CUNY Academic Works

Publications and Research

Kingsborough Community College

2018

Two-Dimensional Maximal Repetitions

Amihood Amir
Bar Ilan University

Gad M. Landau
University of Haifa

Shoshana Marcus
CUNY Kingsborough Community College

Dina Sokol
CUNY Brooklyn College

How does access to this work benefit you? Let us know!

Follow this and additional works at: https://academicworks.cuny.edu/kb_pubs

Part of the Computer Sciences Commons

Recommended Citation

Amir, Amihood; Landau, Gad M.; Marcus, Shoshana; and Sokol, Dina, "Two-Dimensional Maximal Repetitions" (2018). *CUNY Academic Works.*
https://academicworks.cuny.edu/kb_pubs/170

This Article is brought to you for free and open access by the Kingsborough Community College at CUNY Academic Works. It has been accepted for inclusion in Publications and Research by an authorized administrator of CUNY Academic Works. For more information, please contact AcademicWorks@cuny.edu.

Two-Dimensional Maximal Repetitions

Amihood Amir[1]
Bar Ilan University, Ramat-Gan, 52900, Israel
amir@esc.biu.ac.il

Gad M. Landau[2]
University of Haifa, Haifa 31905, Israel, and
NYU Tandon School of Engineering, New York University,
Six MetroTech Center, Brooklyn, NY 11201, USA
landau@univ.haifa.ac.il

Shoshana Marcus
Kingsborough Community College of the City University of New York
2001 Oriental Boulevard, Brooklyn, NY 11235, USA
shoshana.marcus@kbcc.cuny.edu

Dina Sokol[3]
Brooklyn College of the City University of New York
2900 Bedford Avenue, Brooklyn, NY, 11210, USA
sokol@sci.brooklyn.cuny.edu

Abstract

Maximal repetitions or *runs* in strings have a wide array of applications and thus have been extensively studied. In this paper, we extend this notion to 2-dimensions, precisely defining a *maximal 2D repetition*. We provide initial bounds on the number of maximal 2D repetitions that can occur in a matrix. The main contribution of this paper is the presentation of the first algorithm for locating all maximal 2D repetitions in a matrix. The algorithm is efficient and straightforward, with runtime $O(n^2 \log n \log\log n + \rho \log n)$, where n^2 is the size of the input, and ρ is the number of 2D repetitions in the output.

2012 ACM Subject Classification Mathematics of computing → Combinatorics on words, Theory of computation → Design and analysis of algorithms

Keywords and phrases pattern matching algorithms, repetitions, periodicity, two-dimensional

Digital Object Identifier 10.4230/LIPIcs.ESA.2018.2

1 Introduction

Repetitions in strings constitute one of the most fundamental areas of string combinatorics. They are exploited in the design of efficient algorithms for string matching, data compression, and analysis of biological sequences. Maximal repetitions are important structures, as they encode all of the repetitions in the string in a concise way. Once the set of maximal repetitions is known, repetitions of any other type (such as squares and cubes) can be extracted from it.

[1] Partially supported by the Israel Science Foundation grant 571/14 and Grant No. 2014028 from the United States-Israel Binational Science Foundation (BSF).

[2] Partially supported by the Israel Science Foundation grant 571/14 and Grant No. 2014028 from the United States-Israel Binational Science Foundation (BSF).

[3] Partially supported by Grant No. 2014028 from the United States-Israel Binational Science Foundation (BSF).

© Amihood Amir, Gad M. Landau, Shoshana Marcus, and Dina Sokol;
licensed under Creative Commons License CC-BY

26th Annual European Symposium on Algorithms (ESA 2018).
Editors: Yossi Azar, Hannah Bast, and Grzegorz Herman; Article No. 2; pp. 2:1–2:14
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Driven by the many applications to pattern recognition, low level image processing, computer vision and multimedia, the past decades have seen the extension of clever string searching techniques and combinatorial properties to two-dimensional arrays. However, the notion of *maximal* two-dimensional repetitions has not been explored, neither from the combinatorial perspective nor from the algorithmic perspective. Thus, in this project we propose to fill this void. We define a maximal 2D repetition to be a submatrix that can be decomposed into repeating non-overlapping occurrences of the same subblock horizontally and vertically that is maximally extended in all directions.

A range of motivating applications exist that can spur the exploration of maximal repetitions in matrices. In one-dimension, algorithms that compute all the maximal repetitions in a text have application to data compression. The discovery of repetitive structures in the two-dimensional sense can lead to improvements in the compression schemes used for images and video. Just as properties of repetitions have enabled the speeding up of one-dimensional pattern searching algorithms and are relied on by space-efficient one-dimensional pattern matching algorithms, discovering properties of two-dimensional repetitions should create new possibilities and opportunities to speed up two-dimensional string matching algorithms and to design algorithms that use less working space in memory.

As Crochemore et al. have pointed out [9], "the difficulties in extending string-matching techniques to image pattern matching methods are essentially due to different and more complex structures of 2D-periodicities."

In this paper we define two-dimensional *maximal repetitions* for matrices, prove upper bounds on the number of maximal repetitions that can occur in a matrix, and develop an efficient algorithm for locating them. We begin by putting our work in context of related work in Section 2. In Section 3 we precisely define a 2D maximal repetition. Then, in Section 4, we prove that there are at most $O(n^3)$ maximal 2D repetitions in an $n \times n$ matrix. In Section 5 we develop an algorithm to find all the maximal 2D repetitions in an $n \times n$ matrix in close to linear time.

2 Related Work

A string r is *periodic* if its longest prefix that is also a suffix is at least half the length of r. A string s is *primitive* if it cannot be expressed in the form $s = u^j$, for some integer $j > 1$ and some prefix u of s. A periodic string r can be expressed as $u^j u'$ for one unique primitive u, which is called *the period* of r. Every non-primitive string is periodic but not every periodic string is non-primitive. For example, `abc, abcab` are both primitive and non-periodic, `abcabc` is non-primitive (and hence periodic), while `abcabca` is primitive and periodic with period `abc`.

In a string s, a *maximal repetition*, or *run*, is a periodic substring r with period u in which an extension by one letter to the right or to the left yields a string with a longer period than $|u|$ [16]. The maximal repetitions in a string can overlap, be embedded one within another, or begin at the same position. Thus, it was remarkable when Kolpakov and Kucherov proved that a string of length n can contain only $O(n)$ runs [16]. More recently, Bannai et al. proved that the number of runs is strictly less than n [6].

A *square* is a particular type of repetition. In one-dimension, a square is a string which consists of precisely two consecutive occurrences of a substring. Apostolico and Brimkov [3] extend the notion of a square to two dimensions, to form a 2D tandem. They define a 2D tandem as a configuration consisting of two occurrences of the same primitive block that share a side or a corner. A primitive array is one that cannot be partitioned into non-overlapping replicas of some block W [3]. Apostolico and Brimkov prove combinatorially that an $n \times n$ matrix can contain $\Theta(n^4)$ corner-sharing tandems and $\Theta(n^3 \log n)$ side-sharing tandems [3]. They develop an $O(n^3 \log n)$ algorithm for finding side-sharing tandems in an $n \times n$ matrix, which can be used to derive an $O(n^4)$ algorithm for locating all corner-sharing tandems [4].[4] In this paper we extend Apostolico and Brimkov's concept of a side-sharing 2D tandem to many copies to form maximal tandems horizontally and vertically.

A combinatoric construct that is related to repetitions is that of periodicities, i.e. highly repetetive subblocks. The different kinds of two-dimensional periodicities in matrices have been studied by Amir and Benson [1] in terms of self-overlap. Their definition of line and radiant periodicity do not result in 2D repetitions since only the overlapping portion repeats. The lattice periodicity of Amir and Benson is most similar to a 2D repetition. It is also similar in concept to the bi-periodic infinite pictures studied by Bacquey [5]. Bacquey provides interesting combinatoric properties of the primitive roots of bi-periodic infinite pictures. The current paper is more restrictive in terms of lattice periodicity in that the primitive root always has to occur immediately adjacent to its neighbor to the right or beneath it, forming a lattice with all right angles. Apostolico and Brimkov [3], at the beginning of the above-mentioned paper on tandems, define exactly this kind of repetition.

The right-angle lattice periodicity is also used by Gamard and Richomme [11] where the primitive roots of 2D arrays are studied. A matrix is defined as *primitive* if it cannot be broken down to a repeating factor vertically and/or horizontally. Gamard et al. [12] show that every matrix has one unique primitive root. They present several 2D generalizations of the Lyndon-Schutzenberger periodicity theorem for words. However, all exponents in their periodic matrices are integers, i.e. only whole copies of the primitive root are allowed in a repetitive matrix.

In this paper we discuss periodicity where partial copies are allowed at the ends of the matrix, i.e. we use *real* exponents. Our goal is to find *maximal* rectangular submatrices that are repetitions in a given matrix. In the next section we precisely define a 2D repetition and a *maximal 2D repetition* in a matrix.

3 Definition of 2D Maximal Repetition

3.1 1D Maximal Repetitions

In one-dimensional data, a maximal repetition is a substring that is a repetition such that its extension by one character to the right or to the left yields a word with a larger period [16].

▶ **Definition 1.** Let T be a 1D repetition of length t with period U of length u. The exponent e of T is the rational number that satisfies $e = \frac{t}{u}$.

▶ **Lemma 2.** *Let T be a 1D repetition of length t with period U of size u. Let the exponent e be the number of adjacent times U occurs in T such that $U^e = T$ and $u \cdot e = t$. Then T is* maximal *iff it is a substring in which extending one character to the right or left yields a string T' of size $t + 1$ with period U' of size u' and exponent e' such that $e' < e$.*

Proof. The proof has been omitted due to lack of space. ◀

[4] They consider this optimal based on the largest number of such repetitions that can occur in a matrix. However, this is not optimal for a matrix with few 2D tandems. A truly optimal algorithm would find all 2D tandems in $O(n^2 + occ)$ time, where occ is the number of 2D tandems in the matrix.

Table 1 Non-primitive matrices.

$$\begin{array}{|c|}\hline \text{X} \\ \hline \text{X} \\ \hline \end{array} \qquad \begin{array}{|c|c|}\hline \text{X} & \text{X} \\ \hline \end{array} \qquad \begin{array}{|c|c|}\hline \text{X} & \text{X} \\ \hline \text{X} & \text{X} \\ \hline \end{array}$$

3.2 2D Maximal Repetitions

We say that U is a horizontal prefix (resp. suffix) in matrix M if U is an initial (resp. ending) sequence of contiguous columns in M. A *horizontal border* of matrix M is a proper horizontal prefix that is also a horizontal suffix of M. We say that B is the *longest horizontal border* of M if it is the horizontal border of M that spans the largest number of columns among the horizontal borders of M.

▶ **Definition 3.** The *horizontal period*, or *h-period*, of an $m \times n$ matrix M is $n - b$ where b is the number of columns contained in the longest horizontal border of M.

▶ **Definition 4.** [8, 17] An $m \times n$ matrix M with *h-period* p is *horizontally periodic*, or *h-periodic*, if $p \leq \lfloor \frac{n}{2} \rfloor$.

The vertical period of a matrix and vertical periodicity are defined analogously.

▶ **Definition 5.** [3] A matrix M is a *two-dimensional repetition* if M is h-periodic and v-periodic.

Consider an $m \times n$ matrix M and rational numbers $x > 0$, $y > 0$. $M^{x,y}$ is the matrix constructed by repeating M x times vertically and y times horizontally, yielding an $\lfloor xm \rfloor \times \lfloor yn \rfloor$ matrix.

For example,

$$M = \begin{bmatrix} a & b & c & d \\ e & f & g & h \end{bmatrix}$$

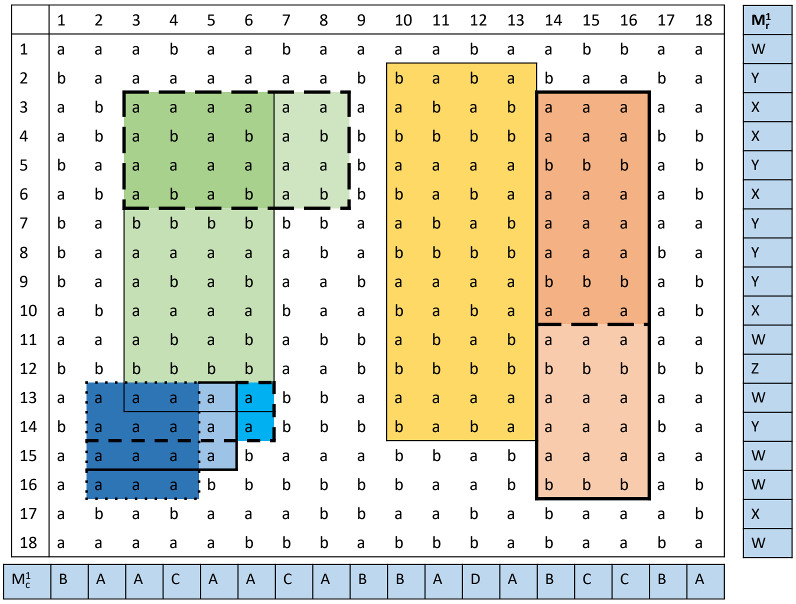
$$M^{1.5,2.25} = \begin{bmatrix} a & b & c & d & a & b & c & d & a \\ e & f & g & h & e & f & g & h & e \\ a & b & c & d & a & b & c & d & a \end{bmatrix} \quad M^{2,1.5} = \begin{bmatrix} a & b & c & d & a & b \\ e & f & g & h & e & f \\ a & b & c & d & a & b \\ e & f & g & h & e & f \end{bmatrix}$$

▶ **Definition 6.** [3, 12] A matrix M is *primitive* if it cannot be partitioned into more than one non-overlapping complete occurrences of some block W. M is *non-primitive* if M can be expressed as $M = W^{r,s}$ for integers r, s such that either $r > 1$ or $s > 1$ or both r and s are strictly greater than 1.

Table 1 shows the different basic configurations of a non-primitive matrix. As in the string terminology, a periodic matrix can be either primitive or non-primitive. In the example, $M^{1.5,2.25}$ is both periodic and primitive, while $M^{2,1.5}$ is periodic and non-primitive.

▶ **Definition 7.** The *primitive root* W of a matrix M is a primitive submatrix such that $M = W^{r,s}$ for rational numbers r, s. M begins with W at its upper left corner and can be partitioned into non-overlapping replicas of W, possibly including partial occurrences of W at its right and / or lower ends.

▶ **Lemma 8.** *Every matrix M has a unique primitive root W such that $M = W^{r,s}$ for rational numbers r, s.*



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	M_r^1
1	a	a	a	b	a	a	b	a	a	a	a	b	a	a	b	b	a	a	W
2	b	a	a	a	a	a	a	a	b	b	a	b	a	b	a	a	b	a	Y
3	a	b	a	a	a	a	a	a	a	a	b	a	b	a	a	a	a	a	X
4	a	b	a	b	a	b	a	b	b	b	b	b	b	a	a	a	b	b	X
5	b	a	a	a	a	a	a	a	b	a	a	a	a	b	b	b	a	b	Y
6	a	b	a	b	a	b	a	b	b	b	a	b	a	a	a	a	a	b	X
7	b	a	b	b	b	b	b	b	a	a	b	a	b	a	a	a	a	a	Y
8	b	a	a	a	a	a	a	b	a	b	b	b	b	a	a	a	b	a	Y
9	b	a	a	b	a	b	a	a	b	a	a	a	a	b	b	b	a	b	Y
10	a	b	a	a	a	a	b	a	a	b	a	b	a	a	a	a	a	b	X
11	a	a	a	b	a	b	a	b	b	a	b	a	b	a	a	a	a	a	W
12	b	b	b	b	b	b	a	a	b	b	b	b	b	b	b	b	b	b	Z
13	a	a	a	a	a	a	b	b	a	a	a	a	a	a	a	a	a	a	W
14	b	a	a	a	a	a	b	b	b	b	a	b	a	a	a	a	b	a	Y
15	a	a	a	a	a	b	a	a	a	b	b	a	b	a	a	a	a	a	W
16	a	a	a	a	b	b	b	b	b	b	a	a	b	b	b	b	a	b	W
17	a	b	a	b	a	a	a	b	b	a	a	b	a	b	a	a	a	b	X
18	a	a	a	a	a	b	b	b	a	b	b	b	a	b	a	a	b	a	W
M_c^1	B	A	A	C	A	A	C	A	B	B	A	D	A	B	C	C	B	A	

Figure 1 A matrix M with many maximal 2D repetitions highlighted. The first row of M_c^1 is depicted below M and the first column of M_r^1 is depicted on the right.

Proof. The proof has been omitted due to lack of space. ◀

▶ **Definition 9.** Let R be a 2D repetition of size $r_1 \times r_2$ with primitive root W of size $w_1 \times w_2$. The *exponent* of R is a tuple (e_1, e_2) in which e_1 and e_2 are rational numbers that satisfy $e_1 = \frac{r_1}{w_1}$ and $e_2 = \frac{r_2}{w_2}$.

In a 2D repetition $R =$

$$\begin{array}{|ccc|c|}
\hline
W & \dots & W & W' \\
\dots & \dots & \dots & \dots \\
W & \dots & W & W' \\
\hline
W'' & \dots & W'' & W''' \\
\hline
\end{array}$$

there are at least two W-blocks horizontally and vertically. That is, the primitive root W repeats both to the right and underneath its initial occurrence in R.

We introduce the idea of a maximal two-dimensional repetition. A 2D repetition R with root W is *maximal* if it cannot be extended by one row or one column to obtain a 2D repetition with the same primitive root W. Figure 1 depicts a matrix with many 2D maximal repetitions highlighted.

▶ **Lemma 10.** *A 2D repetition R of size $r_1 \times r_2$ with root W of size $w_1 \times w_2$ and exponent (e_1, e_2) is maximal iff extending R by one row or column in either direction yields a matrix R' of size $r'_1 \times r'_2$ with primitive root W' of size $w'_1 \times w'_2$ and exponent (e'_1, e'_2) such that $e'_2 < e_2$ or $e'_1 < e_1$.*

Proof. The proof has been omitted due to lack of space. ◀

4 Bounds on the Number of 2D Maximal Repetitions

▶ **Lemma 11.** *There are $O(n^3)$ maximal 2D repetitions in an $n \times n$ matrix.*

Proof. In each row there are $O(n)$ maximal 1D repetitions [16]. For each possible height $0 < h \leq n$, we can linearize the 2D submatrix beginning in each row with height h and width n, by naming metacharacters of subcolumns of height h. This linearization yields a string of length n with $O(n)$ runs. Thus, beginning in each row, for each height, we have $O(n)$ 2D h-periodic horizontally maximal repetitions, resulting in $O(n^3)$ over all rows. The number of 2D maximal repetitions is no more than the number of h-periodic submatrices, since all 2D maximal repetitions are h-periodic. ◀

5 Algorithm to Find 2D Maximal Repetitions

In this section we develop an efficient algorithm to identify all maximal 2D repetitions in an $n \times n$ matrix M. The naive algorithm can examine each of the $O(n^4)$ submatrices in M. For each submatrix S, we can check whether S can possibly be the primitive root of a 2D repetition by attempting to extend it as far as possible. This would take $O(n^6)$ time for all submatrices S. Using LCA queries within each row or column to extend S would speed up the algorithm to $O(n^5)$ time. The last step that remains is to filter out repetitions that were located more than once, which can complete the process in $O(n^5)$ time. The remainder of this section presents a more efficient $O(n^2 \log n \log\log n + \rho \log n)$ algorithm for finding all ρ maximal 2D repetitions that occur in M.

Algorithm Overview:

Step 1 Preprocess the matrix and set up data structures that are used later on by algorithm.

Step 2 Search in each row of the matrix for h-periodic submatrices of height 2^i, for every $1 \leq i \leq \log n$, that begin in that row.

Step 3 Locate all maximal 2D repetitions of height $2^i \leq r < 2^{i+1}$, for every $1 \leq i \leq \log n$, whose prefix 2^i rows are v-periodic.

Step 4 Identify the maximal 2D repetitions of height $2^i < r < 2^{i+1}$, for every $1 \leq i \leq \log n$, whose v-period is not apparent in the first 2^i rows.

5.1 Step 1: Preprocessing the matrix

There are three steps to the preprocessing stage of our algorithm:

1. Naming
   We use Karp-Miller-Rosenberg (KMR) naming [14] on matrix M. One-dimensional KMR naming works with a string, naming each substring whose length is a power of 2. In two-dimensions, we name subcolumns of an $n \times n$ matrix M spanning a number of rows that are powers of 2, i.e., $r = 2^i, 1 \leq i \leq \log n$. We construct $\log n$ matrices of names called M_c^i, for each $1 \leq i \leq \log n$, by naming subcolumns of heigth 2^i. Similarly, we construct a second set of $\log n$ matrices of names which we call M_r^j, for each $1 \leq j \leq \log n$, by naming subrows of M whose widths are 2^j. Throughout the rest of this paper, i is used as the exponent when denoting a number of *rows*, e.g. height 2^i, while j is used in reference to columns, e.g. width 2^j.
2. Substring Periodicity Queries
   A Substring Periodicity Query (SPQ) is as follows: given a string T of length n and two indices, $1 \leq i < j \leq n$, return the period length of $T[i..j]$, when $T[i..j]$ is a repetition. Kociumaka et al. [15] presented an algorithm that processes a string in linear time and space to support $O(1)$ time Substring Periodicity Queries, which they call 2-Period Queries. (Similar time and space complexities are presented by Bannai et al. [7].) We preprocess each column of M_r^j, for each $1 \leq j \leq \log n$, in linear time following the algorithm of Kociumaka et al. [15] to support $O(1)$ time SPQ.

3. Vertical Squares Preprocessing
   We build and decorate suffix trees of each column in each of M_r^j, $1 \leq j \leq \log n$, using the approach of Gusfield and Stoye [13]. The decorated suffix tree marks the endpoints of tandem repeats, either at a node or along an edge. In each decorated suffix tree, we add a link at each node that points to its closest ancestor that is marked or has a marked edge leading into it. We also add links from each marked node to its closest marked ancestor. We then preprocess each decorated suffix tree to admit $O(\log \log n)$ time weighted ancestor queries, where the weight of a node corresponds to the string length it encodes in the suffix tree [2].

Time Complexity for Preprocessing: Subcolumns and subrows can be named with generalized suffix trees of the matrix columns and of the matrix rows for each of the $\log n$ matrices of names. This takes $O(n^2 \log n)$ time, since the naming can be done during Ukkonen's suffix tree construction process [18]. We build a suffix tree of the matrix and preprocess it in linear time to admit $O(1)$ time LCA queries later on. The preprocessing of Kociumaka et al. [15] for SPQ runs in linear time per column of each matrix of names, a total of $O(n^2 \log n)$ time and space. Suffix trees of each column in each of M_r^j, $1 \leq j \leq \log n$, are constructed in linear $O(n^2)$ time and space for each column, overall $O(n^2 \log n)$, and the preprocessing of Amir et al. [2] for weighted ancestor queries is also linear in time and space. In total, the complexity of preprocessing is $O(n^2 \log n)$ time and space.

5.1.1 Queries Used in Algorithm

Once the preprocessing is performed, we can make use of three kinds of efficient queries later on in our algorithm.

Query 1: Vertical Periodicity Query. Given an h-periodic submatrix S within matrix M, what is the vertical period of S?

A Vertical Periodicity Query can be answered in constant time by a SPQ in a column of one of the matrices of names $M_r^j, 1 \leq j \leq \log n$. If S has width 2^j, we use M_r^j. Suppose S begins in row α and ends in row β of M. We ask a SPQ in the column of M_r^j in which S begins with indices α and β that indicate the starting and ending rows of S in M. If S has width c such that $2^j < c < 2^{j+1}$, it is sufficient to ask a SPQ on the first 2^j columns of S. Since S is h-periodic, all of its remaining columns must appear in the first 2^j columns, and they do not affect the vertical periodicity of S. For example, in Figure 1, the Vertical Periodicity Query (on 2 columns) will answer 4 for the 8x3 highlighted submatrix at position (3, 14) and the Vertical Periodicity Query will answer 7 for the 14x3 highlighted submatrix at position (3, 14).

Query 2: Vertical Extension Query. Given a submatrix R that is a 2D repetition of height r, and an integer $x < r$, can R be extended vertically by x rows?

A Vertical Extension Query can be answered in $O(1)$ time as follows. We can compute the v-period v of R using Query 1. Let R^a be the submatrix R extended above by x rows. We do not know if R^a is h-periodic so we do not use Query 1. Let the width c of R satisfy $2^j \leq c < 2^{j+1}$. To compute the v-periodic of R^a, we use two SPQs in $O(1)$ time: one query for a prefix of size 2^j and another query for a suffix of size 2^j in a column of M_r^j. If both answers to the SPQs are equivalent to v, then the answer to the Vertical Extension Query is *yes*, meaning that the repetition R can be extended above by x rows. Otherwise, we perform the same computation for R^b, the submatrix R extended below by x rows. If the answers to both SPQs for R^b are equivalent to v, the answer to the Vertical Extension Query is *yes*. Otherwise, the answer to the vertical extension query is *no*, meaning that the repetition R cannot be extended by x rows up or down. For example, in Figure 1, a Vertical Extension Query by 6 rows on the 8x3 2D repetition beginning at position (3, 14) answers *no* even though it results in a 2D repetition since the vertical period grows with the vertical extension.

Query 3: Vertical Squares Query. Given a column c in M_r^j, $1 \leq j \leq \log n$, a position $1 \leq p \leq n$ within the column, and $1 \leq i \leq \log n$, locate each vertical square beginning at position (c, p) with height $2^i < r < 2^{i+1}$.

We use the data structure of the Vertical Squares Preprocessing described in Step 3 of the preprocessing. To answer the query we ask an $O(\log\log n)$ time weighted ancestor query on suffix p of column c in M_r^j with weight $2^{i+1} - 1$. The returned node's link to the closest marked ancestor yields such a square if one exists. Later, in Lemma 20 we prove that there are at most two answers to this query, hence, one additional link may need to be followed.

5.2 Step 2: Populate the Set $\mathbb{H}$

In this section, we find h-periodic submatrices of height 2^i in the input matrix. A 1D search, e.g. [16, 6], for runs across each row in M_c^i yields a set of h-periodic submatrices. These submatrices are necessarily maximal in their widths but not their heights since the height is fixed at 2^i for some i. Since a 1D row of length n can contain $O(n)$ repetitions [16], each row in M_c^i can contain $O(n)$ h-periodic submatrices. Thus, each matrix of names can contain $O(n^2)$ h-periodic submatrices, yielding a total of $O(n^2 \log n)$ h-periodic submatrices over the $\log n$ matrices of names. These submatrices may or may not be v-periodic. However, we will use them as a starting point for our search.

▶ **Definition 12.** Let $\mathbb{H}$ denote the set of all horizontally maximal h-periodic submatrices of height 2^i, for all $1 \leq i \leq \log n$.

▶ **Lemma 13.** *The procedure described in the previous paragraph finds every horizontally maximal h-periodic submatrix with height 2^i, for each $1 \leq i \leq \log n$, in input matrix M, i.e. we can find the complete set $\mathbb{H}$, in $O(n^2 \log n)$ time.*

Proof. Let I be a horizontally maximal h-periodic submatrix of height 2^i, $1 \leq i \leq \log n$, in M. There must be a subrow of M_c^i that corresponds exactly to I. By the correctness of the 1D search algorithm for runs across the rows of M_c^i, I will be found as a maximal 1D run. The algorithms of [16, 6] run in linear time on each of the $O(n)$ rows of length n in the $O(\log n)$ texts, resulting in $O(n^2 \log n)$ time overall. ◀

In Step 3 and and Step 4 we use the set $\mathbb{H}$ as the starting point for our search for all 2D maximal repetitions. We prove that each 2D maximal repetition in the desired output has a representative in the set $\mathbb{H}$ that shares its h-period. Thus, our algorithm processes each element in $\mathbb{H}$, extending it possibly in several ways, yielding different size repetitions.

▶ **Lemma 14.** *Each maximal 2D repetition R of height $2^i \leq r < 2^{i+1}, 1 \leq i \leq \log n$, has a representative $R' \in \mathbb{H}$ that overlaps R by 2^i rows and shares a corner on the left with R. R also has a representative $R'' \in \mathbb{H}$ that overlaps R by 2^i rows and shares a corner on the right with R.*

Proof. Let R be a maximal 2D repetition of height r. If $r = 2^i, 1 \leq i \leq \log n$, then $R \in \mathbb{H}$ and R is its own representative. Now suppose $2^i < r < 2^{i+1}$. Let $\hat{R}$ denote the prefix 2^i rows of R. Let $\check{R}$ denote the suffix 2^i rows of R. If either $\hat{R} \in \mathbb{H}$ or $\check{R} \in \mathbb{H}$ then a corner on each side is shared with a member of $\mathbb{H}$ and we do not need to consider other scenarios.

Now suppose $\hat{R} \notin \mathbb{H}$ and $\check{R} \notin \mathbb{H}$. This implies that both $\hat{R}$ and $\check{R}$ are not horizontally maximal. Suppose both $\hat{R}$ and $\check{R}$ need to be extended on the left to attain horizontal maximality. This implies that R needs to be extended on the left to attain horizontal maximality. This contradicts the fact that R is a maximal 2D repetition. Thus, R shares either its upper or lower left corner with a member of $\mathbb{H}$. The same argument can be used for the existence of a representative in $\mathbb{H}$ that shares a right corner with R. ◀

▶ **Corollary 15.** *Each maximal 2D repetition R shares either its upper left corner or its lower left corner with some element of $\mathbb{H}$.*

▶ **Corollary 16.** *Let $\mathbb{A}$ be the set of all horizontal prefixes in $\mathbb{H}$. Every maximal 2D repetition in M is the result of a vertical extension on some element of $\mathbb{A}$.*

▶ **Lemma 17.** *Let R be a maximal 2D repetition of height $2^i \leq r < 2^{i+1}, 1 \leq i \leq \log n$, with representatives $R' \in \mathbb{H}$ and $R'' \in \mathbb{H}$. R' and R'' both have the same h-period as R.*

Proof. Let h be the h-period of R. Let h' be the h-period of R'. Suppose $h' > h$. This is impossible since R cannot include fewer rows than R'. Suppose $h' < h$. This means that the Least Common Multiple (LCM) of the periods of the rows in R is larger than the LCM of the periods of the rows in R'. This is only possible if some row in R that is not part of R' has a period larger than that of any row in R'. This implies that some row in R does not occur in R'. This is impossible since we know that more than half of the vertical repetition in R occurs in R', and all the rows of R must occur in R' as well. Thus $h' = h$. The same argument can be made for the h-period of R''. ◀

Following Corollary 16, our algorithm will iterate through the elements in $\mathbb{H}$ and attempt to extend them downward and upward[5]. Since we know that the repetitions in $\mathbb{H}$ are maximal in width, it is not necessary to check horizontal maximality. However, it is possible that by reducing the width, an element in $\mathbb{H}$ can be extended to a taller height. (See the three 2D maximal repetitions beginning at position (13, 2) in Figure 1.) Hence, the task of extension is non-trivial. It is further complicated by the fact that we do not know the v-period of the elements in $\mathbb{H}$. In fact, some elements in $\mathbb{H}$ may not be v-periodic and yet may possibly be extendable into v-periodic matrices. (See the 14x3 2D maximal repetition at position (3, 14) and the 11x4 2D maximal repetition at (3, 3) in Figure 1.) Thus, we consider these two cases separately in the following two subsections. In Section 5.3, we consider the representatives in $\mathbb{H}$ that are v-periodic and identify all 2D maximal repetitions of height $2^i \leq r < 2^{i+1}$ whose prefix 2^i rows are v-periodic. Then, in Section 5.4, we locate the maximal 2D repetitions of height $2^i < r < 2^{i+1}$ whose prefix 2^i rows do not contain two complete copies of their v-periods.

5.3 Step 3: Extending 2D Repetitions Vertically

We begin by performing a Vertical Periodicity Query on each element in $\mathbb{H}$. If the element is v-periodic then it is processed in Step 3.

[5] The rest of the paper discusses the downward direction, the upward direction is analogous.

Algorithm 1 Find Maximal Height.

```
Input: 2D repetition R of height $2^i \leq r < 2^{i+1}$
Output: maximal height $r$ with width of R
                                              ▷ perform binary search to extend R
$j \leftarrow i - 1$                            ▷ we will try to extend by $2^j$ rows
while $j > -1$ do                              ▷ last extension should be by 1 row
    if Vertical Extension Query(R,$2^j$) then
        $r \leftarrow r + 2^j$                      ▷ R is extended by $2^j$ rows
    end if
    $j \leftarrow j - 1$        ▷ decrementing $j$ by 1 in effect halves the size of the extension
end while
```

Let R' denote an element in $\mathbb{H}$ that is both h-periodic and v-periodic. Algorithm 1 attempts to add rows to R' while maintaining its full width. A binary search procedure performs this extension by performing a sequence of Vertical Extension Queries. Once we have determined the maximal height for the full width, it is necessary to attempt to extend narrower widths. The following lemma shows that when there are several 2D maximal repetitions with the same primitive root that share a corner we get a progression of increasing heights and corresponding decreasing widths. For example, position (13, 2) of Figure 1 depicts several 2D repetitions with the same primitive root all starting at the same position.

▶ **Lemma 18.** *Let R be a maximal 2D repetition beginning at position (i, j) with dimensions $r_1 \times r_2$ and primitive root w of size $w_1 \times w_2$. For any other maximal 2D repetition R' beginning at (i, j) with dimensions $r'_1 \times r'_2$ and the same primitive root w, $r'_1 > r_1$ if and only if $r'_2 < r_2$.*

Proof. The proof follows from the definition of maximality. ◀

This monotonicity property gives us the ability to use the modified binary search presented in Algorithm 1 on the potential heights of a 2D repetition. To illustrate Algorithm 1, we can consider the 13x4 maximal 2D repetition at position (2, 10) in Figure 1. We begin with the representative in $\mathbb{H}$ which has $2^3 = 8$ rows. We first try to extend downwards by 4 rows and succeed. Then we try to extend downwards by 2 more rows and fail. Finally, we try to extend downwards by 1 additional row and succeed, resulting in a maximal 2D repetition of height 13.

Algorithm 2 is the outer loop; its job is to compute the widths for which it needs Algorithm 1 to compute the corresponding maximal heights. For each repetition R of height $2^i < r < 2^{i+1}$, Algorithm 2 first checks whether it can be extended to the full height of 2^{i+1}. If it can, then this particular output will be found in another iteration, and the representative has been completed being processed. Otherwise, Algorithm 1 is called. Let r' denote the maximal height returned by Algorithm 1 for some repetition R. The full width of R cannot be extended even one row past r' since the last Vertical Extension Query failed at the end of Algorithm 1. Hence, a Longest Common Prefix (LCP) query in M between the substring of row $r' + 1$ directly below R and the row that is a v-period above it will determine the next width to extend. If the LCP suffices to admit two horizontal copies of the primitive root, we attempt to extend further downwards with Algorithm 1, passing a 2D repetition whose width is the answer to the LCP query and whose height is $r' + 1$. This process continues until either the width is too narrow or the height becomes 2^{i+1}.

Algorithm 2 Find Maximal Repetitions with v-periodic prefix 2^i rows.

```
Input: set $\mathbb{H}$ of h-periodic submatrices, each with height $2^i$
Output: maximal 2D repetitions with height $2^i \leq r \leq 2^{i+1}$ that are v-periodic in their prefix $2^i$ rows
for all $R \in \mathbb{H}$ do          ▷ $R$ has height $r = 2^i$, width $c$, and h-period $h$
    Ask Vertical Periodicity Query on $R$ to get vertical period $v$ of $R$
    if $v \leq \frac{r}{2}$ then          ▷ $R$ is v-periodic
        repeat
            $d \leftarrow 2^{i+1} - r$          ▷ $d$ is distance to height $2^{i+1}$
            if Vertical Extension Query($R$, d) then
                break          ▷ Don't process $R$ if it extends to height $2^{i+1}$.
            end if
            Call Algorithm 1 to compute maximal height $r'$ for $R$
            $r \leftarrow r'$
            Output $R$ with new height
                          ▷ see if can extend $R$ further downwards by decreasing its width
            $\ell \leftarrow$ LCP between row $r - v + 1$ in $R$ and row $r + 1$ below $R$
            if $\ell \geq 2h$ then
                $c \leftarrow \ell$
                $r \leftarrow r + 1$
            end if
        until $\ell < 2h$ or $r \geq 2^{i+1}$
                    ▷ continue looking for taller and narrower 2D repetitions as long as width is h-periodic and height is less than $2^{i+1}$
    end if
end for
```

5.4 Step 4: Unknown Vertical Period

In this section we process *all elements* of $\mathbb{H}$ to discover v-periods that were previously unknown. This can happen in one of two ways in a 2D repetition of height $2^i < r < 2^{i+1}$, $1 \leq i \leq \log n$.

1. The first 2^i rows of a repetition are not v-periodic. For example, abaababa with $i = 2$, and each character of the string is a metacharacter representing a subrow in M. (See the 11x4 maximal repetition at position (3, 3) in Figure 1.)
2. The first 2^i rows are v-periodic, but there is another v-period that comes into existence when rows are added. For example, aabaaabaabaaab with $i = 3$ and each character of the string is a metacharacter representing a subrow in M. In the first 2^i rows of this submatrix, we have the periodic string aabaaaba with period aaba. The first 14 rows are also periodic with period aabaaab of size 7. (See the two maximal repetitions at position (3, 14) in Figure 1.)

The following two lemmas identify the key characteristics of a 2D repetition of height $2^i < r < 2^{i+1}$, $1 \leq i \leq \log n$, and v-period v such that v is not a vertical period in the first 2^i rows of R, i.e., $v > 2^{i-1}$.

▶ **Lemma 19.** *In a 2D repetition R of height $2^i < r < 2^{i+1}$ whose 2^i prefix rows are not v-periodic, the exponent e of the v-period v is between 2 and 4, i.e., $2 \leq e < 4$.*

Proof. Let $|v|$ be the size of v. We know $|v| > \frac{1}{2}2^i = 2^{i-1}$ since the first 2^i rows of R are not periodic in v. Thus, $|v| \geq 2^{i-1} + 1$. We know that the height of R is at most $2^{i+1} - 1$. The longest possible height of R divided by its shortest possible root yields the largest possible exponent for the v-period.

$$e \leq \frac{2^{i+1} - 1}{2^{i-1} + 1} < \frac{2^{i+1}}{2^{i-1}} = 4.$$

By the definition of an exponent for a period, $e \geq 2$. Overall, $2 \leq e < 4$. ◀

▶ **Lemma 20.** *Let I be an h-periodic matrix of height $2^i < r < 2^{i+1}$. No more than 2 v-periods v can occur at the beginning of I such that the first 2^i rows of I are not periodic in v.*

Proof. Suppose v_1 and v_2 are the smallest v-periods in I such that $v_1 > 2^{i-1}$ and $v_2 > 2^{i-1}$. Suppose I has a third v-period v_3 in which the first 2^i rows are not periodic in v_3. Then $v_3 \geq v_1 + v_2$ [10]. Thus, $v_3 > 2^i$. This implies that v_3 cannot occur twice in I of height $r < 2^{i+1}$ and I cannot be v-periodic in v_3. Thus, a third v-period of height larger than 2^{i-1} cannot exist in I. ◀

By Lemmas 19 and 20, we know that we are looking for 2D repetitions that contain either 2 or 3 complete copies of their v-periods and that each element of $\mathbb{H}$ will extend to at most 2 new v-periods. Hence, we are looking for at most two squares that begin with I and have height $2^i < r < 2^{i+1}$. Suppose I has width c such that $2^j \leq c < 2^{j+1}$ and that I begins in row α. We ask a Vertical Squares Query on the column at which I begins in M_r^j, α, and i. Once we identify the 1 or 2 v-periods, if they occur, we revert back to Step 3 of the algorithm (when the v-period is known) and use the procedure described in Algorithm 2 to find the set of maximal 2D repetitions corresponding to each v-period we have identified.

5.5 Algorithm Correctness and Time Complexity

▶ **Theorem 21.** *Let M be an $n \times n$ matrix. Our algorithm finds all maximal 2D repetitions that occur in M.*

Proof. By Lemma 14 every repetition in the output has a representative in $\mathbb{H}$. It remains to show that we hit upon every repetition R in the output with some vertical extension of an element $R' \in \mathbb{H}$ that is a prefix or suffix of R. By Corollary 15, we know that R' shares a corner on the left with R. If R and R' have the same primitive root (Step 3), then by Lemma 18 the successive binary searches will hit upon every output. Now suppose that R' has a different primitive root than R. R' and R must have the same h-period, by Lemma 17, so R' and R must have different v-periods. By Lemma 20, there can be no more than two possible v-periods to try extending with a binary search. One of the extensions must be R, by Corollary 16. Hence, our algorithm identifies all maximal 2D repetitions in M. ◀

▶ **Theorem 22.** *Let M be an $n \times n$ matrix. Our algorithm finds all maximal 2D repetitions in M in $O(n^2 \log n \log\log n + \rho \log n)$ time, where ρ is the number of maximal 2D repetitions that occur in M.*

Proof. Step 1 of the Algorithm Outline, the preprocessing, was shown in Section 5.1 to be done in $O(n^2 \log n)$ time. For Step 2, a linear time 1D search (e.g., [16, 6]) for runs across each row in M_c^i yields the set $\mathbb{H}$ in $O(n^2 \log n)$ time and space.

In Step 3, we iterate through the $O(n^2 \log n)$ elements in $\mathbb{H}$ and perform a constant time Vertical Periodicity Query on each element. Then, Algorithm 2 is called on each v-periodic element. Algorithm 2 performs a Vertical Extension Query and LCP query in constant time. It also calls Algorithm 1 for each representative for each width that is necessary to check. Algorithm 1 runs in $O(i) = O(\log n)$ time. However, the only widths that are checked are those that will certainly produce output. Therefore, we can charge the time spent running Algorithm 1 to the output it generates, yielding $O(\rho \log n)$ where ρ equals the number of repetitions reported. Each repetition is reported at most twice since it can be found once by the upward and downward extensions. For Step 4, we find the 1 or 2 v-periods of interest in $O(\log \log n)$ time using the Vertical Squares Query and again use Algorithm 2 to find the corresponding maximal 2D repetitions in $O(n^2 \log n + \rho \log n)$ time. Hence, the total time complexity of our algorithm is $O(n^2 \log n \log \log n + \rho \log n)$. ◀

References

1 A. Amir and G. Benson. Two-dimensional periodicity in rectangular arrays. *SIAM J. Comput.*, 27(1):90–106, 1998. doi:10.1137/S0097539795298321.

2 A. Amir, G. M. Landau, M. Lewenstein, and D. Sokol. Dynamic text and static pattern matching. *ACM Trans. Algorithms*, 3(2):19, 2007. doi:10.1145/1240233.1240242.

3 A. Apostolico and V. E. Brimkov. Fibonacci arrays and their two-dimensional repetitions. *Theor. Comput. Sci.*, 237(1-2):263–273, 2000. doi:10.1016/S0304-3975(98)00182-0.

4 A. Apostolico and V. E. Brimkov. Optimal discovery of repetitions in 2d. *Discrete Applied Mathematics*, 151(1-3):5–20, 2005. doi:10.1016/j.dam.2005.02.019.

5 N. Bacquey. Primitive roots of bi-periodic infinite pictures. In *Words 2015*, 2015.

6 H. Bannai, T. I, S. Inenaga, Y. Nakashima, M. Takeda, and K. Tsuruta. A new characterization of maximal repetitions by Lyndon trees. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 562–571, 2015. doi:10.1137/1.9781611973730.38.

7 H. Bannai, T. I, S. Inenaga, Y. Nakashima, M. Takeda, and K. Tsuruta. The "runs" theorem. *SIAM J. Comput.*, 46(5):1501–1514, 2017. doi:10.1137/15M1011032.

8 M. Crochemore, L. Gasieniec, R. Hariharan, S. Muthukrishnan, and W. Rytter. A constant time optimal parallel algorithm for two-dimensional pattern matching. *SIAM J. Comput.*, 27(3):668–681, 1998. doi:10.1137/S0097539795280068.

9 M. Crochemore, L. Ilie, and W. Rytter. Repetitions in strings: Algorithms and combinatorics. *Theor. Comput. Sci.*, 410(50):5227–5235, 2009. doi:10.1016/j.tcs.2009.08.024.

10 M. Crochemore and W. Rytter. Sqares, cubes, and time-space efficient string searching. *Algorithmica*, 13(5):405–425, 1995. doi:10.1007/BF01190846.

11 G. Gamard and G. Richomme. Coverability in two dimensions. In A. H. Dediu, E. Formenti, C. Martín-Vide, and B. Truthe, editors, *Language and Automata Theory and Applications - 9th International Conference, LATA 2015, Nice, France, March 2-6, 2015, Proceedings*, volume 8977 of *Lecture Notes in Computer Science*, pages 402–413. Springer, 2015. doi:10.1007/978-3-319-15579-1_31.

12 G. Gamard, G. Richomme, J. Shallit, and T. J. Smith. Periodicity in rectangular arrays. *Inf. Process. Lett.*, 118:58–63, 2017. doi:10.1016/j.ipl.2016.09.011.

13 D. Gusfield and J. Stoye. Linear time algorithms for finding and representing all the tandem repeats in a string. *J. Comput. Syst. Sci.*, 69(4):525–546, 2004. doi:10.1016/j.jcss.2004.03.004.

14 R. M. Karp, R. E. Miller, and A. L. Rosenberg. Rapid identification of repeated patterns in strings, trees and arrays. In *Proceedings of the 4th Annual ACM Symposium on Theory of Computing, May 1-3, 1972, Denver, Colorado, USA*, pages 125–136, 1972. doi:10.1145/800152.804905.

15 T. Kociumaka, J. Radoszewski, W. Rytter, and T. Walen. Internal pattern matching queries in a text and applications. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 532–551, 2015. doi:10.1137/1.9781611973730.36.

16 R. M. Kolpakov and G. Kucherov. Finding maximal repetitions in a word in linear time. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pages 596–604. IEEE Computer Society, 1999. doi:10.1109/SFFCS.1999.814634.

17 S. Marcus and D. Sokol. 2d Lyndon words and applications. *Algorithmica*, 77(1):116–133, 2017. doi:10.1007/s00453-015-0065-z.

18 E. Ukkonen. On-line construction of suffix trees. *Algorithmica*, 14(3):249–260, 1995. doi:10.1007/BF01206331.