Generalization of Repetitiveness Measures for Two-Dimensional Strings*

Lorenzo Carfagna[1][0009−0005−9591−057X],
Giovanni Manzini[1][0000−0002−5047−0196],
Giuseppe Romana[2][0000−0002−3489−0684]**,
Marinella Sciortino[2][0000−0001−6928−0168], and
Cristian Urbina[3,4][0000−0001−8979−9055]

[1] Dipartimento di Informatica, University of Pisa, Italy
lorenzo.carfagna@phd.unipi.it, giovanni.manzini@unipi.it
[2] Dipartimento di Matematica e Informatica, University of Palermo, Italy
{giuseppe.romana01, marinella.sciortino}@unipa.it
[3] Department of Computer Science, University of Chile, Chile
[4] Centre for Biotechnology and Bioengineering (CeBiB), Chile
crurbina@dcc.uchile.cl

Abstract. Detecting and measuring repetitiveness of strings is a problem that has been extensively studied in data compression and text indexing. When the data are structured in a non-linear way, as in two-dimensional strings, inherent redundancy offers a rich source for compression, yet systematic studies on repetitiveness measures are still lacking. In this paper, we extend to two dimensions the measures δ and γ, defined in terms of the submatrices of the input, as well as the measures g, g_{rl}, and b, which are based on copy-paste mechanisms. We study their properties and mutual relationships, and we show that the two classes of measures become incomparable when two-dimensional inputs are considered. We also compare our measures with the 2D Block Tree data structure [Brisaboa et al., Computer J., 2024], and provide some insights for the design of effective 2D compressors.

Keywords: Two–dimensional strings · Repetitiveness measures · Text compression

1 Introduction

In the latest decades, the amount of data generated in the world has become massive but it has been observed that, in many fields, most of this data is highly repetitive. For the study of highly repetitive one-dimensional data, an important role is played by the notions of *substring complexity*, *string attractor*,

* Postprint version. The final authenticated publication is available online at https://doi.org/10.1007/978-3-031-72200-4_5
** Corresponding author.

bidirectional macro scheme, and *grammar compression*, which lead to the definition of the repetitiveness measures δ, γ, b, g, and g_{rl} (see [18] for the definitions and their remarkable properties). Such measures provide the theoretical basis for the design and analysis of data compressors and compressed indexing data structures for highly repetitive data [17].

Two-dimensional data, ranging from images to matrices, often contains inherent redundancy, wherein identical or similar substructures recur throughout the dataset. This great source of redundancy can be exploited for compression. Very recently, Brisaboa et al. introduced the 2D Block Trees to compress images, graphs, and maps [3]. On the theoretical side in [4], the authors propose two generalizations of the measures δ and γ for square 2D input strings. Such generalized measures are based on properties of the *square* submatrices of the input string. The choice of considering only square submatrices was dictated by purely practical considerations: 2D submatrices can be efficiently handled using the 2D Suffix Tree data structure [12], and the 2D Block Tree is based on repeated occurrences of square submatrices in the input string. Indeed, [4] provides the first theoretical analysis of the space usage of the 2D Block Tree and shows that the generalized δ measure can be computed in linear time. More recently, in [5] the same authors have introduced a 2D generalization of the measure b considering the parsing of an input square matrix into (possibly overlapping) square phrases.

In this paper, we set aside any consideration about algorithmic efficiency and we generalize the 1D measures mentioned above (δ, γ, b, g and g_{rl}) to 2D strings, considering submatrices of any rectangular shape and not only square submatrices as in [4,5]. Our main results can be summarized as follows:

- we show that using rectangular submatrices the above 1D measures can be naturally generalized to 2D strings, and we compare their properties with those of the square-based 2D measures introduced in [4,5];
- we establish some relationships between the new 2D measures and we prove that some properties which are valid in 1D are no longer valid in 2D; other properties are still valid but the gap between some measures can be asymptotically much larger than in 1D;
- we show that the measures δ and γ, which have a simple definition in terms of the submatrices of the input, are not as expressive as in 1D, while the measures g, g_{rl} and b appear to retain their role of capturing the repetitiveness of the input even in 2D;
- we show that the 2D Block Tree data structure, being based on square submatrices, fails to capture the regularities of some inputs which are instead captured by the measures g, g_{rl} and b;
- we use our generalized measures to analyze a frequently used heuristics for 2D compression, namely *linearization* i.e. the transformation of the input matrix into a 1D string which is then compressed. We measure the effectiveness of this technique for the simple row-by-row linearization and the more complex linearization based on the Peano-Hilbert space-filling curve.

Overall our results shed some light on the difficulties of detecting and exploiting repetitiveness in the 2D setting, and show that some concepts/tools introduced in 1D are less effective in 2D. Our results on the 2D Block Tree suggest that it could be worthwhile to make this data structure more flexible by possibly using also non-square submatrices and that 2D compression algorithms based on grammar compression (approaching the measures g and g_{rl}) and "copy and paste" macro schemes (approaching the measure b) should be considered as alternatives.

For space reasons, in this paper, we focus only on 2D strings. The repetitiveness measures defined and studied in this paper can be naturally extended to d-dimensional strings with $d > 2$. In this context, all the results provided in the paper are still valid. More extensive and detailed results on higher dimensional strings are deferred to the full paper.

2 2D Strings and Measures δ and γ

Let $\Sigma = \{a_1, a_2, \ldots, a_\sigma\}$ be a finite ordered set of *symbols*, which we call an *alphabet*. A *2D string* $M_{m\times n}$ is a $(m \times n)$-matrix with m *rows* and n *columns* such that each element $M[i][j]$ belongs to Σ. The size of $M_{m\times n}$ is $N = mn$. Note that a position in $M_{m\times n}$ consists of a pair (i, j), with $1 \leqslant i \leqslant m$ and $1 \leqslant j \leqslant n$. Throughout the paper, we assume that for each 2D string $M_{m\times n}$ it holds that $m, n \geqslant 1$. Note that traditional one-dimensional strings are a special case of 2D strings with $m = 1$. We denote by $\Sigma^{m\times n}$ the set of all matrices with m rows and n columns over Σ. A 2D string in $\Sigma^{m\times n}$ is called *square* if $m = n$.

The concatenation between two matrices is a partial operation that can be performed horizontally ($\oplus$) or vertically ($\ominus$), with the constraint that the number of rows or columns coincide respectively. Such operations have been described in [11] where concepts and techniques of formal languages have been generalized to two dimensions.

We denote by $M_{m\times n}[i_1 \,..\, i_2][j_1 \,..\, j_2]$ the submatrix starting at position (i_1, j_1) and ending at position (i_2, j_2). We say that a matrix F is a *factor* or *substring* of $M_{m\times n}$ if there exist two positions (i_1, j_1) and (i_2, j_2) such that $F = M_{m\times n}[i_1 \,..\, i_2][j_1 \,..\, j_2]$. Given a 2D string $M_{m\times n}$, the *2D substring complexity* function P_M counts for each pair of positive integers (k_1, k_2) the number of distinct $(k_1 \times k_2)$-factors in $M_{m\times n}$. The following definition extends the notion of δ measure [7] to 2D strings.

Definition 1. *Let $M_{m\times n}$ be a 2D string and P_M be the 2D substring complexity of $M_{m\times n}$. Then, $\delta(M_{m\times n}) = \max\{P_M(k_1, k_2)/k_1k_2, 1 \leqslant k_1 \leqslant m, 1 \leqslant k_2 \leqslant n\}$.*

Note that for 1D strings (i.e. $m = 1$) the above definition coincides with the one used in the literature [14,18]. Recently, in [4] Carfagna and Manzini introduced an alternative extension of δ, here denoted by $\delta_\square$, limited to square 2D input strings and using only *square* factors for computing the substring complexity. Below we report the definition of such a measure, applied to a generic two-dimensional string.

Definition 2. *Let $M_{m\times n}$ be a 2D string and P_M be the 2D substring complexity of $M_{m\times n}$. Then, $\delta_\square(M_{m\times n}) = \max\{P_M(k,k)/k^2, 1 \leqslant k \leqslant \min\{m,n\}\}$.*

From the definitions of $\delta_\square$ and δ, the following lemma easily follows.

Lemma 1. *For every 2D string $M_{m\times n}$ it holds that $\delta(M_{m\times n}) \geqslant \delta_\square(M_{m\times n})$.*

Although the two measures δ and $\delta_\square$ may seem similar, considering square factors instead of rectangular ones may result in very different values. Example 1 shows how different the two measures can be when applied to one-dimensional strings, while Example 2 shows that there exist families of square 2D strings for which $\delta_\square = o(\delta)$.

Example 1. Given a 1D string $S \in \Sigma^n$, let $M_{1\times n} \in \Sigma^{1\times n}$ be the matrix such that $M_{1\times n}[1][1\,..\,n] = S[1\,..\,n]$. Since the only squares that occur in $M_{1\times n}$ are the factors of size 1×1, it is $\delta_\square(M_{1\times n}) = P_M(1,1)/1^2 \leqslant |\Sigma|$. On the other hand, $\delta(M_{1\times n}) = \delta(S)$.

Example 2. Let $M_{n\times n}$ be the square 2D string in [4, Lemma 4]. Assuming n is a perfect square, the first row of $M_{n\times n}$ is the string $S = B_1B_2\,..\,B_{\sqrt{n}/2}$ composed by $\sqrt{n}/2$ blocks, each one of size $2\sqrt{n}$, with $B_i = \mathtt{1}^i\mathtt{0}^{(2\sqrt{n}-i)}$. The remaining rows of $M_{n\times n}$ are all $\#^n$. In [4, Lemma 4] it is shown that $\delta_\square(M_{n\times n}) = O(1)$. On the other hand, notice that for $i \in [2\,..\,\sqrt{n}/2]$ and $j \in [0\,..\,\sqrt{n}-i]$, the strings $\mathtt{0}^j\mathtt{1}^i\mathtt{0}^{\sqrt{n}-j-i}$ are all different substrings of length $\sqrt{n}$ of S. Since these substrings are in total $\Omega(n)$, it is $\delta(M_{n\times n}) = \Omega(\sqrt{n})$.

The following definition generalizes to 2D strings the notion of attractor [13].

Definition 3. *An* attractor *for a 2D string $M_{m\times n}$ is a set $\Gamma \subseteq [1\,..\,m]\times[1\,..\,n]$ with the property that any substring $M[i\,..\,j][k\,..\,l]$ of $M_{m\times n}$ has an occurrence $M[i'\,..\,j'][k'\,..\,l']$ such that $\exists(x,y) \in \Gamma$ with $i' \leqslant x \leqslant j'$ and $k' \leqslant y \leqslant l'$. The size of the smallest attractor for $M_{m\times n}$ is denoted by $\gamma(M_{m\times n})$.*

When $m = 1$ the above definition coincides with the one for 1D strings, hence the measure γ inherits the properties for the one-dimensional case [16,13]. In particular: γ is not monotone and computing $\gamma(M_{m\times n})$ is NP-hard. In addition, the following property holds.

Proposition 1. *For every 2D string $M_{m\times n}$, it is $\delta(M_{m\times n}) \leqslant \gamma(M_{m\times n})$.*

Proof. Reasoning as in the 1D case, we get that for any 2D string M it is $P_M(k_1,k_2) \leqslant k_1k_2\gamma(M)$. □

The next proposition shows that in the 2D context, the gap between δ and γ can be larger than the one-dimensional case, where it is logarithmic [14].

Proposition 2. *For all $m,n \geqslant 1$ there exists a 2D string $M_{m\times n}$ such that $\delta(M_{m\times n}) = O(1)$ and $\gamma(M_{m\times n}) = \Omega(\min(m,n))$.*

Proof. Let I_k be the $k \times k$ identity matrix. For all $m, n \geqslant 1$, let us consider the 2D string $M_{m\times n}$ such that $M_{m\times n}[1\,..\,\min(m,n)][1\,..\,\min(m,n)] = I_{\min(m,n)}$, and all the remaining symbols are 0's. When either $m = 1$ or $n = 1$ the proof is trivial from known results on 1D strings, so let us assume $m, n \geqslant 2$. Let us further assume $m < n$, next we show that $\Gamma = \{(2,2)\,..\,(m-1, m-1)\} \cup \{(1,m),(m,1),(m,m+1)\}$ is an attractor for $M_{m\times n}$. All the substrings of $M_{m\times n}$ that contain at least two occurrences of 1's have an occurrence crossing the position (i,i), for some $1 < i < m$, while all the substrings that consist of only 0's have an occurrence aligned with one of the 0's at position $(1,m)$, $(m,1)$, or $(m,m+1)$. The factors left contain only one occurrence of 1's that do not cross any position in Γ. These factors of size $k_1 \times k_2$ have to cross either the 1 in position $(1,1)$ or in position (m,m), and therefore it is either $k_1 = 1$ and $k_2 < m$, or vice versa. Observe that all these factors have another occurrence either starting in $(2,2)$ or ending in $(m-1, m-1)$, and therefore Γ is an attractor of $M_{m\times n}$. Since $M_{m\times n}$ has $m+1$ distinct columns the above attractor has minimum size, i.e. $\gamma(M_{m\times n}) = m+1$. On the other hand, there exist at most $k_1 + k_2$ distinct substrings of size $k_1 \times k_2$ in $M_{m\times n}$: $k_1 + k_2 - 1$ correspond to substrings where the diagonal of $M_{m\times n}$ touches ones of the positions in the left or upper borders of the factor; the last one is the string of only 0's. Hence $\delta(M_{m\times n}) \leqslant 2$. The case $m > n$ is treated symmetrically by considering the attractor $\Gamma' = \{(2,2)\,..\,(n-1, n-1)\} \cup \{(1,n),(n,1),(n+1,n)\}$. For the case $n = m$ it is $M_{m\times n} = I_n$ and reasoning as above it is easy to see that $\Gamma'' = \{(2,2)\,..\,(n-1, n-1)\} \cup \{(1,n),(n,1)\}$ of size n is a minimal attractor for I_n and that $\delta(I_n) \leqslant 2$. □

In [4] the authors introduced an alternative definition of string attractors for square 2D input strings in which they consider only square factors. We can define such a measure, denoted by $\gamma_{\square}$, also for generic 2D strings, by simply considering only square substrings of $M_{m\times n}$ in Definition 3. From the definitions of γ and $\gamma_{\square}$ we immediately get the following relationship:

Lemma 2. *For every 2D string $M_{m\times n}$ it holds that $\gamma(M_{m\times n}) \geqslant \gamma_{\square}(M_{m\times n})$.*

The following example shows that γ and $\gamma_{\square}$ can be asymptotically different.

Example 3. Consider again the $m \times m$ identity matrix I_m. For each $k \leqslant m$, a $k \times k$ square factor of I_m either consists of i) all 0's, or ii) all 0's except only one diagonal composed by 1's. Hence, all square factors of type i) have an occurrence that includes position $(m,1)$ (i.e. the bottom left corner), while all those of type ii) have an occurrence that includes the position $(\lfloor m/2 \rfloor, \lfloor m/2 \rfloor)$ (i.e. the 1 at the center). It follows that $\gamma_{\square}(I_m) = 2 \in O(1)$, while from the proof of Proposition 2 it can be deduced that $\gamma(I_m) = \Theta(m)$.

The measures δ and γ inherit from the 1D case the property that δ is unreachable and γ is unknown to be reachable [18]. In the next sections, we show how to generalize to the 2D case the measures g, g_{rl}, and b which are reachable both in 1D and 2D (details in the full paper).

3 (Run-Length) Straight-Line Programs for 2D Strings

In this section we consider a generalization of SLPs for the two-dimensional space introduced in [2] and use it to generalize the measures g and g_{rl} to 2D strings.

Definition 4. *Let $M_{m\times n}$ be a 2D string. A* 2-dimensional straight-line program *(2D SLP) for $M_{m\times n}$ is a context-free grammar (V, Σ, R, S) that uniquely generates $M_{m\times n}$ and where the definition of the right-hand side of a variable can have the form*

$$A \to a,\ A \to B \oslash C,\ \text{ or } A \to B \ominus C,$$

where $a \in \Sigma$, $B, C \in V$. We call these definitions terminal rules*,* horizontal rules*, and* vertical rules*, respectively. The expansion of a variable is defined as*

$$\mathtt{exp}(A) = a,\ \mathtt{exp}(A) = \mathtt{exp}(B) \oslash \mathtt{exp}(C),\ \text{ or } \mathtt{exp}(A) = \mathtt{exp}(B) \ominus \mathtt{exp}(C),$$

respectively. The size of a 2D-SLP is the sum of the sizes of all right-hand sides of the rules of the grammar, where we assume the terminal rules have size 1*, and the horizontal and vertical rules have size* 2*. The measure $g(M_{m\times n})$ is defined as the size of the smallest 2D SLP generating $M_{m\times n}$.*

It is easy to see that the above definition coincides with that of the measure g for one-dimensional strings if only horizontal concatenations are considered.

Proposition 3. *It always holds that $g(M_{m\times n}) = \Omega(\log(mn))$.*

Proof. From the starting variable S, each substitution step can double the size of the current 2D string of variables. Hence, a 2D SLP of size g can produce a 2D string of size at most $2^{\lfloor g/2 \rfloor}$. Therefore, a string of size $N = mn$ needs a grammar of size $\Omega(\log_2 N)$. □

Proposition 4. *The problem of determining if there exists a 2D SLP of size at most k generating a text $M_{m\times n}$ is NP-complete.*

Proof. Clearly, the problem belongs to NP. Observe that the 1D version of the problem, known to be NP-complete [6], reduces to the 2D version by considering 1D strings as matrices of size $1 \times n$. □

Given any 2D SLP, we can support direct access to an arbitrary cell of the input in $O(h)$ time, where h is the height of the parse tree[5] of the 2D SLP. Note that the balancing procedure of Ganardi et al. [10] can be easily adapted in the 2D setting, so $h = O(\log N)$, where N is the size of the 2D string.

As in the 1D case, we can extend 2D SLPs with *run-length rules* obtaining more powerful grammars.

[5] Analogously to the 1D setting, the *parse tree* of 2D SLP is an ordered labeled tree where S is the root, and the children of a variable A are the variables in its right-hand side (possibly repeated).

Definition 5. *A* 2-dimensional run-length straight-line program *(2D RLSLP) is a 2D SLP that in addition allows special rules, which are assumed to be of size 2, of the form*

$$A \to ①^k B \text{ and } A \to ⊖^k B$$

for $k > 1$, with their expansions defined as

$$\texttt{exp}(A) = \texttt{exp}(\underbrace{B ① B ① \cdots ① B}_{k \text{ times}}) \qquad \text{and} \qquad \texttt{exp}(A) = \texttt{exp}(\underbrace{B ⊖ B ⊖ \cdots ⊖ B}_{k \text{ times}})$$

respectively. The measure $g_{rl}(M_{m\times n})$ is defined as the size of the smallest 2D RLSLP generating $M_{m\times n}$.

Proposition 5. *For every 2D string $M_{m\times n}$ it holds that $g_{rl}(M_{n\times n}) \leqslant g(M_{m\times n})$. Moreover, there are infinite string families where $g_{rl} = o(g)$.*

Proof. The first claim is trivial by definition. The second claim is proven by considering the family of $1 \times n$ matrices $M_{1\times n} = \texttt{1}^n$ for which $g = \Omega(\log n)$ and $g_{rl} = O(1)$. □

4 Macro Schemes for 2D Strings

The notion of macro scheme and the corresponding measure b can be naturally generalized to 2D strings with the following definition.

Definition 6. *A* 2D macro scheme *for a string $M_{m\times n}$ is any factorization of $M_{m\times n}$ into a set of disjoint phrases such that any phrase is either a square of dimension 1×1 called an* explicit symbol/phrase*, or is a copied phrase with source in $M_{m\times n}$ starting at a different position. For a 2D macro scheme to be* valid *or* decodable*, there must exist a function* $\texttt{map} : ([1\,..\,m] \times [1\,..\,n]) \cup \{\perp\} \to ([1\,..\,m] \times [1\,..\,n]) \cup \{\perp\}$ *such that: i)* $\texttt{map}(\perp) = \perp$*, and if $M[i][j]$ is an explicit symbol, then* $\texttt{map}(i, j) = \perp$*; ii) for each copied phrase $M[i_1\,..\,j_1][i_2\,..\,j_2]$, it must hold that* $\texttt{map}(i_1 + t_1, i_2 + t_2) = \texttt{map}(i_1, i_2) + (t_1, t_2)$ *for $(t_1, t_2) \in [0\,..\,j_1 - i_1] \times [0\,..\,j_2 - i_2]$, where* $\texttt{map}(i_1, i_2)$ *is the upper left corner of the source for $M[i_1\,..\,j_1][i_2\,..\,j_2]$; iii) for each $(i, j) \in [1\,..\,m] \times [1\,..\,n]$ there exists $k > 0$ such that* $\texttt{map}^k(i, j) = \perp$*. We define $b(M_{m\times n})$ as the size of the smallest valid 2D macro scheme for $M_{m\times n}$.*

Example 4. Let I_n be the $n \times n$ identity matrix. A macro scheme for I_n consists of the phrases $\{X_1, X_2, X_3, X_4, X_5, X_6\}$ where: i) $X_1 = I_n[1][1]$ is an explicit symbol (the 1 in the top-left corner); ii) $X_2 = I_n[1][2]$ is an explicit symbol; $X_3 = I_n[2][1]$ is an explicit symbol; $X_4 = I_n[1][3\,..\,n]$ is a phrase with source $(1, 2)$; $X_5 = I_n[3\,..\,n][1]$ is a phrase with source $(2, 1)$; and $X_6 = I_n[2\,..\,n][2\,..\,n]$ is a phrase with source $(1, 1)$. The underlying function $\texttt{map}$ is defined as $\texttt{map}(1, 1) = \texttt{map}(1, 2) = \texttt{map}(2, 1) = \perp$, $\texttt{map}(1, j) = (1, j - 1)$ for $j \in [3\,..\,n]$, $\texttt{map}(i, 1) = (i - 1, 1)$ for $i \in [3\,..\,n]$, and $\texttt{map}(i, j) = (i - 1, j - 1)$ for $i, j \in [2\,..\,n] \times [2\,..\,n]$. One can see that $\texttt{map}^n(i, j) = \perp$ for each i and j. Hence, the macro scheme is valid and $b(I_n) \leqslant 6$. Figure 1 shows this macro scheme for I_7.

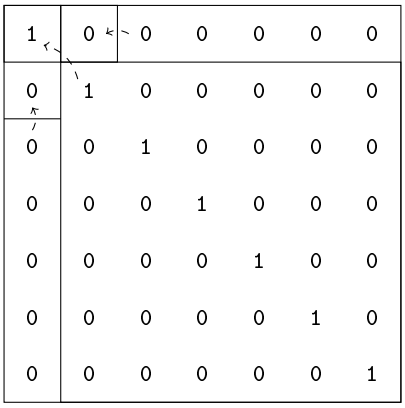

Fig. 1. Macro scheme with 6 phrases for I_7. The entries $(1,1)$, $(1,2)$, and $(2,1)$ are explicit symbols. The remaining phrases point to the source from where they are copied.

The following two propositions show that the computability properties of b and its relationship with the measures g_{rl} and g are preserved in the 2D context.

Proposition 6. *The problem of determining if there exists a valid 2D macro scheme of size at most k for a text $M_{m\times n}$ is NP-complete.*

Proof. The 1D version of the problem, which is known to be NP-complete [9], reduces to the 2D version of the problem in constant time. □

Proposition 7. *For every 2D string $M_{m\times n}$ it holds that $b(M_{m\times n}) \leqslant g_{rl}(M_{m\times n})$. Moreover, there are string families where $b = o(g_{rl})$.*

Proof. We show how to construct a macro scheme from a 2D RLSLP, representing the same 2D string and having the same asymptotic size. Let G be a 2D RLSLP generating $M_{m\times n}$ and consider its parse tree. The first occurrence of each variable B that expands to a single symbol at position $M[i][j]$, becomes an explicit phrase of the parsing at that position. For each occurrence of a variable B that is not the first one in this tree, we create a phrase—exactly where this occurrence of $\texttt{exp}(B)$ should be in M—that maps to the expansion of the first occurrence of B in M. For a rule $A \to \textcircled{1}^k B$, we construct at most two phrases: one phrase for the leftmost B of $\textcircled{1}^k B$ pointing to the first occurrence of B, or if the expansion of this B is a single symbol (or no phrase if this B is the first occurrence but its expansion is not a single symbol), and another phrase for $\textcircled{1}^{k-1} B$ pointing to the expansion of the occurrence of the B before in M. We do analogously, for 2D vertical run-length rules. It is easy to see that this parsing is decodable and that its size is bounded by the size g_{rl} of the grammar. For a family where $b = o(g_{rl})$, this holds for the 1D version [19]. □

In [5] the authors introduce a notion of 2D macro scheme for square strings that differs from the one in Definition 6 in that 1) phrases must all be square

and 2) phrases are allowed to overlap (see [5, Section 4] for details). Given a 2D string M, in the following we call $b_{\square}(M)$ the minimum number of phrases in a macro scheme with square, possibly overlapping, phrases. We now show that the use of square phrases can limit significantly the power of a macro scheme since there are string families for which $b = o(b_{\square})$. This result will have important consequences also for other measures.

For any $k > 0$, let D_k denote a binary de Bruijn sequence of length $n = 2^k + k - 1$ containing all the possible binary substrings of length k exactly once. We define the $n \times n$ matrix B_k over the alphabet $\Delta = \{\langle \mathtt{0},\mathtt{0}\rangle, \langle \mathtt{0},\mathtt{1}\rangle, \langle \mathtt{1},\mathtt{0}\rangle, \langle \mathtt{1},\mathtt{1}\rangle\}$ by the relationship

$$B_k[i][j] = \langle D_k[i], D_k[j]\rangle.$$

Notice that each row and each column of B_k is a de Bruijn sequence over a binary alphabet which is a subset of Δ. For example, if $D_k[i] = \mathtt{1}$, then row i of B_k is a de Bruijn sequence over the alphabet $\{\langle \mathtt{1},\mathtt{0}\rangle, \langle \mathtt{1},\mathtt{1}\rangle\}$. Similarly, if $D_k[j] = \mathtt{0}$, then column j of B_k is a de Bruijn sequence over the alphabet $\{\langle \mathtt{0},\mathtt{0}\rangle, \langle \mathtt{1},\mathtt{0}\rangle\}$. Notice that B_k contains only two distinct rows/columns since rows/columns i and j are different if and only if $D_k[i] \neq D_k[j]$.

Lemma 3. *For any $k > 0$ it is $g(B_k) = O(n \log\log n / \log n)$.*

Proof. In [8] the authors show that $g(D_k) = O(n \log\log n / \log n)$, that is, there exists a (one dimensional) SLP G of size $O(n \log\log n / \log n)$ generating the de Bruijn sequence D_k. If in G we replace the terminal symbols $\{\mathtt{0},\mathtt{1}\}$ with $\{\langle \mathtt{0},\mathtt{0}\rangle, \langle \mathtt{0},\mathtt{1}\rangle\}$ we obtain a SLP G_0 that generates all rows i in B_k such that $D_k[i] = \mathtt{0}$. Similarly, if in G we replace the terminals $\{\mathtt{0},\mathtt{1}\}$ with $\{\langle \mathtt{1},\mathtt{0}\rangle, \langle \mathtt{1},\mathtt{1}\rangle\}$ we obtain a SLP G_1 that generates all rows i in B_k such that $D_k[i] = \mathtt{1}$. Finally, if in G we make all rules vertical and we replace the terminal $\mathtt{0}$ with the starting symbol of G_0 and the terminal $\mathtt{1}$ with the starting symbol of G_1, we obtain a 2D SLP G_2 that combined with G_0 and G_1 generates the matrix B_k. The thesis follows since the size of $G_2 \cup G_1 \cup G_0$ is $O(n \log\log n / \log n)$. □

Lemma 4. *Every square submatrix of size k or more appears in B_k at most once.*

Proof. It suffices to prove the result for the square submatrices of size k. Assume that the $k \times k$ submatrix with upper left corner in (i, j) is identical to the submatrix with upper left corner in (u, v). The crucial observation is that $B_k[i][j] = B_k[u][v]$ implies $D_k[i] = D_k[u]$ and $D_k[j] = D_k[v]$. Considering also the other entries, we get that if the two submatrices are equal then we must have $D_k[i..i+k-1] = D_k[u..u+k-1]$ and $D_k[j..j+k-1] = D_k[v..v+k-1]$. Since D_k is a de Bruijn sequence, this implies $i = u$ and $j = v$ as claimed. □

Proposition 8. *For the 2D string B_k of size $N = n \times n$, with $n = 2^k + k - 1$, it is $b_{\square} = \Omega(g\sqrt{N}/(\log N \log\log N)) = \Omega(b\sqrt{N}/(\log N \log\log N))$.*

Proof. Since $b \leqslant g$, by Lemma 3 $b = O(\sqrt{N} \log\log N / \log N)$. Lemma 4 implies that there cannot be square phrases of size $k \times k$ or larger. Hence, the number of phrases is at least n^2/k^2, so $b_{\square} = \Omega(N/\log^2 N)$ and the lemma follows. □

5 On the Relative Power of 2D Measures

A remarkable property of the measures considered in this paper is that, for 1D strings, they can be totally ordered in terms of their relative power at capturing regularities in the input; indeed, for any 1D string S it is $\delta(S) \leqslant \gamma(S) \leqslant b(S) \leqslant g_{rl}(S) \leqslant g(S)$ (see [13,18]). In the previous sections, we have shown that when we consider also 2D strings the relationships $\delta \leqslant \gamma$ and $b \leqslant g_{rl} \leqslant g$ still hold. In this section, however, we prove that the two classes of measures, i.e. δ and γ from one side, based on the counting and distribution of distinct factors, and b, g_{rl}, and g from the other side, based on copy-paste mechanisms, become incomparable when also 2D strings are considered. In particular g can be asymptotically smaller than δ:

Proposition 9. *There exists an infinite family of 2D strings of size N with $\delta = \Omega(gN/\log^3 N)$.*

Proof. For $k \geqslant 1$, consider the 2D binary string E_k of size $N = k \times 2^k$ such that, for all $1 \leqslant i \leqslant 2^k$, the ith column of E_k is the binary representation of $i-1$ in k digits (with the top row containing the least significant bits). Since all columns of E_k are distinct, E_k contains 2^k distinct factors of size $k \times 1$ and therefore it is $\delta(E_k) \geqslant 2^k/k = kN/k^3$. We prove that $g(E_k) = O(k)$ by exhibiting a 2D SLP for E_k of size $O(k)$ and the result immediately follows because $k < \log N$.

Consider the 2D SLP $G_k = (V_k, \{\mathtt{0}, \mathtt{1}\}, R'_k, S_k)$ having the following rules R'_k:

- $X_0 \to \mathtt{0}$ and $X_h \to X_{h-1} \oslash X_{h-1}$ for all $1 \leqslant h \leqslant k-1$;
- $Y_0 \to \mathtt{1}$ and $Y_h \to Y_{h-1} \oslash Y_{h-1}$ for all $1 \leqslant h \leqslant k-1$;
- $C_h \to X_{h-1} \oslash Y_{h-1}$ for all $2 \leqslant h \leqslant k$;
- $S_1 = X_0 \oslash Y_0$ and $S_h \to R_h \ominus C_h$ for all $2 \leqslant h \leqslant k$.
- $R_h \to S_{h-1} \oslash S_{h-1}$ for all $2 \leqslant h \leqslant k$;

G_k has size $\Theta(k)$ and it is easy to see that $\mathtt{exp}(X_h) = \mathtt{0}^{(2^h)}$, $\mathtt{exp}(Y_h) = \mathtt{1}^{(2^h)}$ and therefore $\mathtt{exp}(C_h) = \mathtt{0}^{(2^{h-1})}\mathtt{1}^{(2^{h-1})}$. In the following, we show by induction on k that G_k is a 2D SLP for E_k, that is $\mathtt{exp}(S_k) = E_k$. For the base case $k = 1$ it is $\mathtt{exp}(S_1) = \mathtt{exp}(X_0) \oslash \mathtt{exp}(Y_0) = \mathtt{0} \oslash \mathtt{1} = E_1$. For the inductive step we assume that $\mathtt{exp}(S_k) = E_k$ and we note that for $k \geqslant 1$ the bottom row $E_{k+1}[k+1][1..2^{k+1}]$ of E_{k+1} is the string $\mathtt{0}^{(2^k)}\mathtt{1}^{(2^k)} = \mathtt{exp}(C_{k+1})$ and the remaining rectangular submatrix $E_{k+1}[1..k][1..2^{k+1}]$ is $E_k \oslash E_k$.

By taking two expansion steps starting from S_{k+1}, we obtain $\mathtt{exp}(S_{k+1}) = (\mathtt{exp}(S_k) \oslash \mathtt{exp}(S_k)) \ominus \mathtt{exp}(C_{k+1})$ which by inductive hypotheses and the definition of C_{k+1} expands to $(E_k \oslash E_k) \ominus E_{k+1}[k+1][1..2^{k+1}] = E_{k+1}$. □

From the above proposition, we get that the measure g (and therefore also g_{rl} and b) can be much smaller than both δ and γ. Intuitively, the reason is that the matrix E_k is hard to compress by columns (they are all distinct) but easily compressible by rows. The measure g is defined in terms of the best grammar compressor, so it fully exploits row compressibility. In contrast, the measure δ is based on occurrences of factors of any shape and is therefore "hindered" by the

difficulty of compressing columns. For 1D strings it is always $\delta \leqslant g$, but in the 2D setting, because of the greater freedom in choosing the shape of the copied patterns, the measures b, g_{rl} and g become mutually incomparable with both γ and δ.

Given the above observation, it is worthwhile to compare g with the measures $\delta_\square$ and $\gamma_\square$ whose definitions are based on square factors. In some sense, using square factors can be seen as a method to capture both horizontal and vertical compressibility. Unfortunately, the following proposition shows that g can be asymptotically smaller than $\delta_\square$ even for input square matrices. Note that the same result also holds for $\gamma_\square$, δ, and γ, since $\delta_\square \leqslant \gamma_\square$ and $\delta_\square \leqslant \delta \leqslant \gamma$.

Proposition 10. *There exists an infinite family of square 2D strings with* $\delta_\square = \Omega(g\sqrt{N}/(\log N \log\log N))$*, where* N *is the size of the input string.*

Proof. Consider the matrix B_k defined in Section 4. By Lemma 3 it is $g(B_k) = O(n \log\log n/\log n) = O(\sqrt{N}\log\log N/\log N)$. By Lemma 4 B_k contains $\Theta(n^2) = \Theta(N)$ distinct $k \times k$ factors, hence $\delta_\square = \Omega(N/\log^2 N)$ and the bound follows. □

Note that Propositions 8, 9 and 10 show that in the 2D setting there can be a significant gap between b, g_{rl} and g from one side, and $b_\square$, $\delta_\square$, $\gamma_\square$, δ, γ from the other. Furthermore, since the square matrix $\mathtt{0}_{n\times n}$ consisting of only zeros has constant measures $b_\square$, $\delta_\square$, $\gamma_\square$ but it is $g(\mathtt{0}_{n\times n}) = \Omega(\log n)$ (see Proposition 3), Propositions 8 and 10 imply that g is also mutually incomparable with each of the above square measures, even if we consider only square input matrices.

We now show that even the recently introduced 2D Block Tree data structure [3], which is also based on square factors, can fail to capture the regularity of certain two-dimensional strings. The 2D Block Tree is a tree-like compressed representation of a square matrix supporting random access to individual entries in logarithmic time. Given an $n \times n$ input matrix M, an integer parameter $c > 1$, and assuming that n is a power of c, the root of the 2D Block tree at level $\ell = 0$ represents the whole matrix M. To build the level $\ell \geqslant 1$ of the tree we recursively partition (some of) the submatrices represented at level $\ell - 1$ into c^2 smaller non-overlapping submatrices of size $n/c^\ell \times n/c^\ell$ called *blocks*; for each of these blocks, the tree stores a corresponding descending node at level ℓ. The 2D Block Tree attempts to compress the input matrix by avoiding the storage of redundant submatrices: if a block has a prior occurrence in row-major order (RMO), its corresponding subtree is candidate to be pruned and replaced with $O(1)$ pointers to the nodes overlapping its first occurrence in RMO. The pruned blocks are not partitioned into smaller matrices, and their corresponding nodes are leaves in the 2D block tree. See [3,5] for details.

Unfortunately, the following example exhibits a family of 2D strings that are significantly more compressible when represented as an SLP compared to their 2D Block Tree representation: for these matrices, the 2D Block Tree fails to achieve a compression close to the measure g (and therefore to g_{rl} and b).

Proposition 11. *There exists a family of square 2D strings such that the number of nodes of the corresponding 2D Block Tree is* $\Omega(g\sqrt{N}/(\log N \log\log N))$*, where* N *is the size of the input string.*

Proof. Consider again the matrix B_k defined in Section 4. By Lemma 4 until we reach the tree level in which the blocks are smaller than $k \times k$, all blocks are first occurrences, and therefore the corresponding tree nodes cannot be pruned. Hence, the 2D Block Tree nodes are $\Omega(n^2/k^2) = \Omega(N/\log^2 N)$. □

6 Effectiveness of Linearization Techniques

A classical heuristic for compressing 2D strings is to transform a matrix $M_{m\times n}$ into a 1D string S and use a one-dimensional compressor on S. Having generalized 1D measures to 2D strings, it is natural to measure the effectiveness of linearization techniques by comparing, for a given measure μ, the values $\mu(M_{m\times n})$ and $\mu(S)$. Clearly, for each matrix, there exists a linearization that makes the 2D string highly compressible: we can visit in order from left to right and from top to bottom all the occurrences of $a_1 \in \Sigma$, followed by all the occurrences of $a_2 \in \Sigma$, and so on, obtaining a string consisting in $|\Sigma|$ equal-letter runs. However, this method requires an ad-hoc linearization for each matrix which may require substantial additional information to retrieve the original input. It is therefore customary in the literature to consider linearization techniques that can be inverted efficiently in terms of both time and space.

The simplest linearization technique consists in mapping $M_{m\times n}$ to the string $\texttt{rlin}(M_{m\times n}) = \bigodot_{i=1}^{i=m} M[i][1..n] = M[1][1..n]\cdots M[m][1..n]$, obtained by concatenating its rows. The (lack of) effectiveness of this simple technique with respect to grammar compression has been already shown in [2, Theorem 2] with an example of a matrix T_n of size $(2^{n+1}-1)\times(2^n+1)2^n$ such that $g(T_n) = O(n)$, while $g(\texttt{rlin}(T_n)) = \Omega(2^n)$. The following example shows a similar result for the measure δ.

Example 5. Let $M_{n\times n}$ be obtained by appending to the identity matrix I_{n-1} a row of 0's at the bottom, and then a column of 1's at the right. For each k_1, k_2, $P_M(k_1,k_2)$ is at most $3(k_1+k_2)$. We can see this by considering three cases: the submatrices that do not intersect the last row or column, the submatrices intersecting the last row, and the submatrices intersecting the last column. In each case, the distinct submatrices are associated to where the diagonal of 1's intersects a submatrix (if it does so). This can happen in at most k_1+k_2 different ways. As $3(k_1+k_2)/k_1k_2 \leqslant 6$, we obtain $\delta(M_{n\times n}) = O(1)$. On the other hand, for each $k \in [0\,..\,n-2]$ and $i \in [0\,..\,n-k-2]$, each factor $\texttt{0}^i\texttt{1}\texttt{0}^k\texttt{1}\texttt{0}^{n-k-i-2}$ appears in $\texttt{rlin}(M_{n\times n})$. There are $n-k-1$ of these factors for each k. Summing over all k, we obtain $P_M(n)/n = (n-1)/2 = \Omega(n)$. Thus, $\delta(\texttt{rlin}(M_{n\times n})) = \Omega(n)$.

Somewhat surprisingly, in some settings, the linearized matrix has a smaller measure. The following example shows a family of 2D strings E_k for which $\gamma(\texttt{rlin}(E_k))$ is asymptotically smaller than $\gamma(E_k)$.

Example 6. Consider the matrix E_k having size $N = k \times 2^k$ of Proposition 9. We note that the i-th row of E_k is the periodic string $(\texttt{0}^{(2^{i-1})}\texttt{1}^{(2^{i-1})})^{(2^{k-i})}$ and therefore $\texttt{rlin}(E_k) = \bigodot_{i=1}^{i=k}(\texttt{0}^{(2^{i-1})}\texttt{1}^{(2^{i-1})})^{(2^{k-i})}$. We define the set $A =$

$\bigcup_{i=1}^{i=k}\{(i-1)2^k+1, (i-1)2^k+1+2^{i-1}, (i-1)2^k+2^i\}$, that is the set of positions of $\texttt{rlin}(E_k)$ where respectively the first $\texttt{0}$ and the first/last $\texttt{1}$ of the leftmost occurrence of $\texttt{0}^{(2^{i-1})}\texttt{1}^{(2^{i-1})}$ in row i are mapped during the linearization. We claim that A is an attractor for $\texttt{rlin}(E_k)$. If a substring S of $\texttt{rlin}(E_k)$ spans more than one row of E_k, it includes a $\texttt{0}$ from the first column of E_k, and therefore it crosses a position of A. Otherwise S is a substring of the ith row and since the rows of E_k are periodic, the leftmost occurrence S' of S starts inside the first group of $\texttt{0}^{(2^{i-1})}\texttt{1}^{(2^{i-1})}$ i.e. in $E_k[i][1..2^i]$. Suppose that S' does not include any attractor position. Then, S' has to be shorter than the maximum distance between two adjacent attractor positions in the same row i.e. it must be $l = |S'| \leqslant 2^{i-1}-1$, and therefore $S' = \texttt{0}^a\texttt{1}^{l-a}$ or $S' = \texttt{1}^a\texttt{0}^{l-a}$ for some $0 \leqslant a < l$ because S' cannot overlap two distinct groups of $\texttt{1}$'s or $\texttt{0}$'s. If $a = 0$ then S' must include respectively the first $\texttt{1}$ or the first $\texttt{0}$ in the run, otherwise it must include respectively the first or the last $\texttt{1}$, therefore we conclude that A is an attractor for $\texttt{rlin}(E_k)$.

Since A has size $3k-1$, it is $\gamma(\texttt{rlin}(E_k)) = O(k)$, on the other hand since each column of E_k is a distinct non-overlapping $k \times 1$ factor it is $\gamma(E_k) \geqslant 2^k$.

Another well-known linearization technique is the use of a plane-filling curve, such as the Peano-Hilbert curve. Lempel and Ziv [15, Lemma 1] showed that this technique, here denoted by $\texttt{phlin}$, is effective for compressing 2D strings using a finite-state encoder. We show that this is not necessarily true for the measure b, as shown in the following proposition. We postpone the formal definition of the Peano-Hilbert linearization $\texttt{phlin}$ to the full paper, as well as the proof the following characterization of the string $\texttt{phlin}(I_{2^k})$ obtained by applying $\texttt{phlin}$ to the identity matrix I_{2^k}: it is $\texttt{phlin}(I_2) = \texttt{1010}$, and, for $k \geqslant 1$,

$$\texttt{phlin}(I_{2^{k+1}}) \;=\; \texttt{phlin}(I_{2^k})\texttt{0}^{4^k}\,\texttt{phlin}(I_{2^k})\texttt{0}^{4^k}.$$

Proposition 12. *It is* $b(I_{2^k}) = O(1)$ *and* $b(\texttt{phlin}(I_{2^k})) = \Omega(k)$.

Proof. We have already observed in Example 4 that $b(I_n) = O(1)$ for any $n > 0$. To prove the second bound, for $\ell \geqslant 1$ define $t_\ell = \sum_{i=0}^{\ell-1} 4^i$. We preliminary prove by induction on k that: *a)* $\texttt{phlin}(I_{2^k})$ starts with $\texttt{1}$, *b)* $\texttt{phlin}(I_{2^k})$ ends with $\texttt{10}^{t_k}$, *c)* $\texttt{phlin}(I_{2^k})$ contains the substrings $\texttt{10}^{t_\ell}\texttt{1}$ for $\ell = 1, \ldots, k$.

For $k = 1$ $\texttt{phlin}(I_2) = \texttt{1010}$ satisfies all conditions. For the inductive step *a)* is trivial. Since $\texttt{phlin}(I_{2^{k+1}})$ ends with $\texttt{phlin}(I_{2^k})\texttt{0}^{4^k}$ also *b)* is immediate. To prove *c)* observe that since $\texttt{phlin}(I_{2^{k+1}})$ contains $\texttt{phlin}(I_{2^k})$, by induction it contains all substrings $\texttt{10}^{t_\ell}\texttt{1}$ for $\ell = 1, \ldots, k$. To prove that it also contains $\texttt{10}^{t_{k+1}}\texttt{1}$ observe that by *a)* $\texttt{phlin}(I_{2^{k+1}})$ starts with $\texttt{phlin}(I_{2^k})\texttt{0}^{4^k}\texttt{1}$ and by *b)* this string ends with $\texttt{10}^{t_k}\texttt{0}^{4^k}\texttt{1} = \texttt{10}^{t_{k+1}}\texttt{1}$, and therefore is a substring of $\texttt{phlin}(I_{2^{k+1}})$.

Having established that $\texttt{phlin}(I_{2^k})$ contains the *distinct* substrings $\texttt{10}^{t_\ell}\texttt{1}$ for $\ell = 1, \ldots, k$, since a single string position can be contained in at most two such substrings, we conclude that $b(\texttt{phlin}(I_{2^k})) \geqslant \gamma(\texttt{phlin}(I_{2^k})) = \Omega(k)$. □

7 Conclusions and Future Works

In this paper, we have proposed the first complete extension of repetitiveness measures previously used in the 1D context to generic two-dimensional strings. In particular, we have introduced extensions of the measures δ and γ to the two-dimensional case based on distinct factors of arbitrary rectangular shape, as well as the extensions of the measures g, g_{rl}, and b, which are based on copy-paste mechanisms.

We have studied the mutual relationships between these measures and we have shown that $\delta \leqslant \gamma$ and $b \leqslant g_{rl} \leqslant g$. We point out that, unlike in the 1D context where $\delta \leqslant \gamma \leqslant b \leqslant g_{rl} \leqslant g$, the two classes of measures become incomparable when 2D strings are considered. Indeed, we have shown that, depending on the 2D input, the measures g, g_{rl}, and b can be asymptotically smaller than δ and γ.

The results presented in the paper highlight that in the 2D case, the measures δ and γ (as well as their square-based versions introduced in [4,5]) are not satisfactory for capturing the regularities of a generic two-dimensional string, which are instead effectively detected by g, g_{rl}, and b measures. More importantly, the problem of finding a time-efficient 2D compression scheme approaching the measures g, g_{rl}, or b is still open. Indeed, in this paper we have proven that the 2D-Block Tree data structure, introduced in [3] to compress square matrices while supporting efficient random access to its elements, fails to achieve a compression close to g (and g_{rl} and b) for some 2D strings. At the same time, we have shown that the use of linearization strategies as preprocessing to compress two-dimensional input does not seem to be effective, even when considering approaches based on the Peano-Hilbert space-filling curve. For the above reasons, we believe it would be worthwhile to explore possible approximation strategies for b and g, as well as 2D versions of greedy grammar construction algorithms like the ones described in [1,19].

Acknowledgments. LC and GM are partially funded by the PNRR ECS00000017 Tuscany Health Ecosystem, Spoke 6, CUP I53C22000780001, funded by the NextGeneration EU programme, by the spoke "FutureHPC & BigData" of the ICSC — Centro Nazionale di Ricerca in High-Performance Computing, Big Data and Quantum Computing, funded by the NextGeneration EU programme.

GR and MS are partially funded by the MUR PRIN Project "PINC, Pangenome INformatiCs: from Theory to Applications" (Grant No. 2022YRB97K).

LC, GM, MS, and GR are partially funded by the INdAM-GNCS Project CUP E53C23001670001.

CU is partially funded by ANID-Subdirección de Capital Humano/Doctorado Nacional/2021-21210580, ANID, Chile, partially funded by Basal Funds FB0001, ANID, Chile, and partially funded by Fondecyt Grant 1-230755.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Bannai, H., Hirayama, M., Hucke, D., Inenaga, S., Jez, A., Lohrey, M., Reh, C.P.: The smallest grammar problem revisited. IEEE Transactions on Information Theory **67**(1), 317–328 (2021)
2. Berman, P., Karpinski, M., Larmore, L.L., Plandowski, W., Rytter, W.: On the complexity of pattern matching for highly compressed two-dimensional texts. J. Comput. Syst. Sci. **65**(2), 332–350 (2002)
3. Brisaboa, N.R., Gagie, T., Gómez-Brandón, A., Navarro, G.: Two-dimensional block trees. Comput. J. **67**(1), 391–406 (2024)
4. Carfagna, L., Manzini, G.: Compressibility measures for two-dimensional data. In: Proceedings of the 30th International Symposium on String Processing and Information Retrieval, SPIRE 2023. LNCS, vol. 14240, pp. 102–113. Springer (2023)
5. Carfagna, L., Manzini, G.: The landscape of compressibility measures for two-dimensional data. IEEE Access **12**, 87268–87283 (2024)
6. Charikar, M., Lehman, E., Liu, D., Panigrahy, R., Prabhakaran, M., Sahai, A., Shelat, A.: The smallest grammar problem. IEEE Trans. Inf. Theory **51**(7), 2554–2576 (2005)
7. Christiansen, A.R., Ettienne, M.B., Kociumaka, T., Navarro, G., Prezza, N.: Optimal-time dictionary-compressed indexes. ACM Trans. Algorithms **17**(1), 8:1–8:39 (2021)
8. Gagie, T., Navarro, G., Prezza, N.: On the approximation ratio of Lempel-Ziv parsing. In: Proceedings LATIN 2018. LNCS, vol. 10807, pp. 490–503. Springer (2018)
9. Gallant, J.K.: String Compression Algorithms. Ph.D. thesis, Princeton University (1982)
10. Ganardi, M., Jez, A., Lohrey, M.: Balancing straight-line programs. J. ACM **68**(4), 27:1–27:40 (2021)
11. Giammarresi, D., Restivo, A.: Two-dimensional languages. In: Handbook of Formal Languages (3), pp. 215–267. Springer (1997)
12. Giancarlo, R.: A generalization of the suffix tree to square matrices, with applications. SIAM J. Comput. **24**(3), 520–562 (1995)
13. Kempa, D., Prezza, N.: At the roots of dictionary compression: string attractors. In: STOC. pp. 827–840. ACM (2018)
14. Kociumaka, T., Navarro, G., Prezza, N.: Toward a definitive compressibility measure for repetitive sequences. IEEE Trans. Inf. Theory **69**(4), 2074–2092 (2023)
15. Lempel, A., Ziv, J.: Compression of two-dimensional data. IEEE Transactions on Information Theory **32**(1), 2–8 (1986)
16. Mantaci, S., Restivo, A., Romana, G., Rosone, G., Sciortino, M.: A combinatorial view on string attractors. Theor. Comput. Sci. **850**, 236–248 (2021)
17. Navarro, G.: Indexing highly repetitive string collections, part II: Compressed indexes. ACM Computing Surveys **54**(2), article 26 (2021)
18. Navarro, G.: Indexing highly repetitive string collections, part I: Repetitiveness measures. ACM Computing Surveys **54**(2), article 29 (2021)
19. Navarro, G., Ochoa, C., Prezza, N.: On the approximation ratio of ordered parsings. IEEE Trans. Inf. Theory **67**(2), 1008–1026 (2021)