UNIVERSIDAD
DE CHILE

A Study on Repetitiveness Measures

PhD. Thesis Proposal

Author:
Cristian Urbina

Supervisor:
Gonzalo Navarro

Department of Computer Science
University of Chile

March 24, 2023

A Study on Repetitiveness Measures

CRISTIAN URBINA GALLEGOS,
Computer Science Department (DCC), University of Chile, Chile

Nowadays, high volumes of repetitive data are being generated every second. Traditional compressors based on Shannon's entropy fail to exploit the repetitiveness of data as a source of compressibility. Because of this limitation, repetition-aware compressors like the Lempel-Ziv parsing, the run-length encoding of the Burrows-Wheeler transform, and grammar-based compression via heuristics like Re-Pair have been devised.

From a theoretical point of view, the size of a compressed representation of a text defines a measure of compressibility. In the case of repetition-aware compressors, these values are, in fact, measures of repetitiveness. Recently, measures of repetitiveness without a compression scheme have been devised, building only on the combinatorial properties of strings. Repetitiveness measures based on compressors are essential because they help us study their properties in a more simple, elegant, and abstract way. On the other hand, measures based on the string's combinatorial properties help us understand what we must exploit when designing compressors or indexes to achieve some desired amount of space or functionality.

The goal of this thesis is to deepen our understanding of repetitiveness measures. We want to establish asymptotic bounds between them. We also want to assess their sensitivity to simple edit operations such as insertion, deletion, or substitution, as well as more complex ones like reverse and morphism application. Finally, we plan to introduce and explore other measures that focus on more specific aspects of repetitiveness.

Overall, this research will enhance our knowledge of repetitiveness measures and provide insights into their behavior under various scenarios. The findings will be useful in designing more efficient compressors and indexes that can handle repetitive texts.

Additional Key Words and Phrases: Repetitiveness measures, text compression, combinatorics on words

1 MOTIVATION

Over recent years, the amount of data being generated in the world has risen to enormous levels at a growth rate that even surpasses Moore's law [45]. There exist several sources of massive data, ranging from social networks to natural phenomenons. These sources generate extremely large amounts of data every year[1], which is of interest to businesses and scientific research.

A great part of the massive data generated every day happens to be very repetitive, what we understand intuitively as having a lot of copies of the same data in our collections of data. As examples, we have human genome codifications, where two distinct human genomes differ only in about 0.1% [43], so a collection of human genomes is highly repetitive. We also have versioned websites like Wikipedia and platforms of control of software versions like Github, among various other sources of repetitiveness.

To make this data useful, it needs to be processed. Even before this, the data has to be stored somewhere. Data compression deals with the problem of representing data using the least amount of space possible, or at least, space close to this optimal value. By compressing the data, we obtain easier transfer, upload, and download of files of large size (in compressed form). In areas like Bioinformatics, the amount of data is so big that it usually needs to be processed in compressed form. Hence, compressed data structures [35], and more specifically, repetition-aware compressed self-indexes [37] have been the object of intense research in the latest decades.

Usually, compression algorithms and compressed indexes are based on the so-called Shannon's entropy. The problem with this is that Shannon's entropy measures the compressibility of texts based only on the frequency of symbols, being completely blind to repetitiveness [26].

[1] http://groups.ischool.berkeley.edu/archive/how-much-info/how-much-info.pdf

Author's address: Cristian Urbina Gallegos, crurbina@dcc.uchile.cl,
Computer Science Department (DCC), University of Chile, Santiago, Chile.

Because of the impossibility of Shannon's entropy to capture repetitions, several other measures quantifying repetitiveness have been defined. A recent survey [36] covers repetitiveness measures and distinguishes between those that are reachable (i.e., we can compress any string to that space) from those that are not. Examples of reachable measures are the number z of phrases of the Lempel-Ziv parsing of a given string [48], the number b of phrases of the smallest valid bidirectional macro scheme that represents the string [46], the size g of the smallest context-free grammar generating only the string [22], and the size c of the smallest collage system generating only the string [21]. From all reachable measures to date, b is the most general and also the smallest of them.

There also exist measures based on combinatorial properties of strings, like γ, the size of the smallest string attractor for a given string [20], and δ, which is based upon the concept of string complexity [24]. Although probably both of them are unreachable (δ has been proved so), they are still useful as lower bounds for reachable measures.

Repetitiveness measures based on compressors are crucial because they enable us to study their properties in a more straightforward, elegant, and abstract manner. Similarly, measures based on a string's combinatorial properties can help us understand what we need to exploit when designing compressors or indexes to achieve a specific level of space or functionality.

The goal of this thesis is to deepen our understanding of repetitiveness measures. We want to establish asymptotic bounds between them. We also want to assess their sensitivity to simple edit operations such as insertion, deletion, or substitution, as well as more complex ones like reverse and morphism application. Finally, we plan to introduce and explore other measures that focus on more specific aspects of repetitiveness.

Overall, this research will enhance our knowledge of repetitiveness measures and provide insights into their behavior under various scenarios. The findings will be useful in designing more efficient compressors and indexes that can handle repetitive strings.

2 BACKGROUND AND RELATED WORK

In this section, we present some basic concepts needed to understand the thesis. We also mention some closely related results.

2.1 Basics

2.1.1 Strings. An *alphabet* is a finite set of *symbols*, and is usually denoted by Σ. A (finite) *string* w is a finite sequence $w[1]w[2]\dots w[n]$ of symbols where $w[i] \in \Sigma$ for $i \in [1, n]$, and its *length* is denoted by $|w| = n$. The *empty string* whose length is 0 is denoted by ε. The set of all possible finite strings over Σ is denoted by Σ^*, and the set of strings of length n is denoted Σ^n. If $x = x[1]\dots x[n]$ and $y = y[1]\dots y[m]$, the concatenation operation $x \cdot y$ (or just xy) yields the string $x[1]\dots x[n]y[1]\dots y[m]$. Let $w = xyz$. Then y is a *substring* (resp. x, z) of w (resp. *prefix*, *suffix*). It is *proper* if it is not equal to w, and *non-trivial* if its distinct from ε. The notation $w[i : j]$ stands for the substring $w[i]\dots w[j]$, if $1 \le i \le j \le |w|$, or ε otherwise.

A (right) *infinite string* $\mathbf{w}$ (we use boldface to emphasize them) over an alphabet Σ is a mapping from $\mathbb{Z}^+$ to Σ, and its length is ω, which is greater than n for any $n \in \mathbb{Z}^+$. It is possible to define the concatenation $x \cdot \mathbf{y}$ if x is finite and $\mathbf{y}$ infinite. The concepts of substring, prefixes, and suffixes also carry over, with proper prefixes always being finite and suffixes always being infinite. The notations $\mathbf{w}[i]$ and $\mathbf{w}[i : j]$ carry over to infinite strings. For both finite and infinite strings, the *factor complexity* function $P_w(n)$ denotes the number of distinct substrings of length n in the string.

As an observation, because the topic of our thesis lies in between the areas of Text compression, Combinatorics on words, and Stringology, we sometimes use the terms string, word, or text to refer to the same thing. Similarly, we use the words symbol, letter, and character interchangeably.

2.1.2 Morphisms. The set Σ^* together with the (associative) concatenation operator and the (identity) string ε form a *monoid* structure $(\Sigma^*, \cdot, \varepsilon)$. A *morphism* on strings is a function $\varphi : \Sigma_1^* \to \Sigma_2^*$ satisfying that $\varphi(\varepsilon) = \varepsilon$ and $\varphi(x \cdot y) = \varphi(x) \cdot \varphi(y)$ (i.e., a function preserving the monoid structure), where Σ_1 and Σ_2 are alphabets. To define a morphism of strings, it is sufficient to define how it acts over the symbols in its domain. The pairs $(a, \varphi(a))$ for $a \in \Sigma_1$, usually denoted $a \to \varphi(a)$, are called the *rules* of the morphism, and there are $|\Sigma_1|$ of them. If $\Sigma_1 = \Sigma_2$, then the morphism is called an *endomorphism*.

Let $\varphi : \Sigma_1^* \to \Sigma_2^*$ be a morphism on strings. Some useful definitions are $width(\varphi) = max_{a \in \Sigma_1} |\varphi(a)|$ and $size(\varphi) = \sum_{a \in \Sigma_1} |\varphi(a)|$. A morphism is *non-erasing* if $\forall a \in \Sigma_1, |\varphi(a)| > 0$, *expanding* if $\forall a \in \Sigma_1, |\varphi(a)| > 1$, *k-uniform* if $\forall a \in \Sigma_1, |\varphi(a)| = k > 2$, and it is a *coding* if $\forall a \in \Sigma_1, |\varphi(a)| = 1$ (sometimes called a 1-uniform morphism).

Let $\varphi : \Sigma^* \to \Sigma^*$ be an endomorphism. Then φ is *prolongable* on a symbol a if $\varphi(a) = ax$ for some string $x \neq \varepsilon$. If this is the case, then for each i, j with $0 \leq i \leq j$, it holds that $\varphi^i(a)$ is a prefix of $\varphi^j(a)$, and $\mathbf{x} = \varphi^\omega(a) = ax\varphi(x)\varphi^2(x)\dots$ is the unique infinite fixed-point of φ starting with the symbol a. An infinite string $\mathbf{w} = \varphi^\omega(a)$ that is the fixed-point of a morphism is called a *purely morphic word* [32], its image under a coding $\mathbf{x} = \tau(\mathbf{w})$ is called a *morphic word* [32], and if the morphism φ is k-uniform, then $\mathbf{x}$ is said to be *k-automatic* [2].

A remarkable family of words generated by morphisms is the one composed by the finite *Fibonacci words* defined as $D^i(\mathsf{a})$ for $i \geq 0$, where D is the morphism with rules $\mathsf{a} \to \mathsf{ab}$ and $\mathsf{b} \to \mathsf{a}$. Another relevant family are the *Thue-Morse words*, defined as $T^i(\mathsf{a})$ for $i \geq 0$, where T is the morphism with rules $\mathsf{a} \to \mathsf{ab}$ and $\mathsf{b} \to \mathsf{ba}$.

2.1.3 Sturmian words. *Sturmian words* are defined as exactly those infinite words with factor complexity function $P_{\mathbf{w}}(n) = n+1$ for $n \geq 0$ [32]. As $P_{\mathbf{w}}(1) = 2$, all Sturmian words are binary words. The fixed point of the Fibonacci morphism is a Sturmian word. Moreover, the finite Fibonacci words are *Standard Sturmian words*, that is, words that can be constructed by defining a directive sequence $(d_0, d_1, \dots)$ where $d_0 \geq 0$ and $d_i > 0$ for $i > 0$, and using the recursive formula $s_0 = \mathsf{b}, s_1 = \mathsf{a}$, $s_{i+1} = s_i^{d_{i-1}} s_{i-1}$ [8].

2.2 Repetitiveness measures

A *repetitiveness measure* μ is a function that arguably captures the degree of *repetitiveness* of strings. Repetitiveness is an intuitive and elemental concept, yet is still subject to debate. In general, a repetitive string is understood as one containing many copies of the same substrings on itself. Ideally, the most repetitive is a string w, the smallest the value $\mu(w)$ should be.

If, for all strings, we can represent them using $O(\mu(w))$ space, then we say the measure $\mu(w)$ is reachable. Space is usually measured in $\Theta(\log n)$-bit words following the conventions of the *Transdichotomous RAM model of computation*. Hence, when we say that a family can be represented in $O(\mu)$ space, we mean also that it can be represented using $O(\mu \log(n))$ bits. We can represent any symbol in the alphabet of $w[1:n]$ using a constant number of words, as long as $|\Sigma| = O(n^d)$ for some $d \geq 0$.

Two repetitiveness measures u_1 and u_2 are *uncomparable* if they are not asymptotically equivalent, and none of them asymptotically dominates the other on every string family, i.e., if there are infinite string families $\mathcal{F}_1$ and $\mathcal{F}_2$ where $u = o(v)$ in $\mathcal{F}_1$ and $v = o(u)$ in $\mathcal{F}_2$.

In the following, we explain the repetitiveness measures to be considered in the thesis. For an in-depth review of repetitiveness measures, see [36].

2.2.1 Parsing-based measures. A *parsing* produces a factorization of a string w into non-empty *phrases*, i.e., $w = w_1 w_2 \dots w_k$ where $w_i \in \Sigma^+$ for $i \in [1, k]$. Several compressors work by parsing w in a way that storing summary information about the phrases enables recovering w.

The *Lempel-Ziv* (LZ) parsing processes a string greedily from left to right, always forming the longest phrase that has a copy starting inside some previous phrase or forming a *explicit* phrase of length 1 in other case [29]. The LZ-no parsing always forms the longest phrase with a copy fully contained in the concatenation of previous phrases or a explicit phrase of length 1 if not possible. Another variation is the LZ-end parsing, which forms the longest phrase with an occurrence as a suffix of a previous phrase or a explicit phrase of length 1 if not possible [25]. All of these parsings can be constructed in linear time, and their number of phrases are denoted by z, z_{no}, and z_e, respectively. While z and z_{no} are optimal among parsing satisfying their respective conditions, this is not always the case for z_e. The optimal factorization where each phrase w_{i+1} appears as a suffix of $w_1 \dots w_j$ for some $j \leq i$ is denoted by z_{end}. Because of the optimality of z, z_{no}, and z_{end}, it holds that $z \leq z_{no} \leq z_{end} \leq z_e$.

A *bidirectional macro-scheme* (BMS) [46] is any parsing where each phrase of length greater than 1 has a copy starting at a different position in such a way that the original string can be recovered following these pointers. The measure b is defined as the size of the smallest BMS for w. It strictly lower bounds asymptotically all the other reachable repetitiveness measures [38]. On the other hand, b is NP-hard to compute [13].

Finally, another interesting parsing-based measure is the size of the greedy *lex-parse* of w, denoted as $v(w)$. This parsing, greedily from left to right, always forms as a phrase the longest common prefix between the unprocessed part of the string and the suffix that lexicographically precedes it (a unique \$ symbol smallest than any other symbol is assumed to exist at the end of w). It forms an explicit phrase of length one if the longest common prefix is empty or no predecessor exists [38]. It has been proven that sometimes $v = o(z)$ [38]. It is unknown if $v = O(z)$ in the general case.

2.2.2 Grammars-based measures. There exist several compressors which build upon context-free grammars, and hence, there also exist many measures of repetitiveness associated with their size.

A *straight-line program* (SLP) is a deterministic context-free grammar whose language is a singleton $\{w\}$. The measure g is defined as the size of the smallest SLP G generating w. Finding the smallest SLP is an NP-complete problem [7], although, there exist algorithms providing log-approximations [18, 44]. In practice, an heuristic called Re-Pair usually works better [28], yet no strong guarantees are known about its approximation ratio [7].

Another measure based on context-free grammars is g_{rl}, the size of the smallest *run-length* SLP (RLSLP) generating w [42]. RLSLPs allow constant-size rules of the form $A \to B^n$ for $n > 1$, which can make a noticeable difference in some string families like $\{\mathtt{a}^n \mid n \geq 0\}$, where $g = \Theta(\log n)$, but $g_{rl} = O(1)$.

An even stronger (in terms of space) measure based on context-free grammars is the size c of the smallest *collage system* [21]. A collage system is an RLSLP, which in addition, supports rules of the form $A \to B[i : |exp(B)|]$ and $A \to B[1 : i]$ for some $i \in [1, |exp(B)|]$. While g and g_{rl} are both $\Omega(z)$, it has been proven that $c = O(z)$ [38]. It is unknown if $c = o(z)$ for some string family.

2.2.3 Burrows-Wheeler transform. The Burrows-Wheeler transform (BWT) [6] is a reversible transformation that usually makes a string more compressible. In the Combinatorics on words community, the BWT is obtained by concatenating the last symbols of the sorted rotations of w. In the Text compression field, it is obtained by concatenating the last symbols of the sorted suffixes of $w\$$ where \$ is a symbol smaller than any symbol appearing in w. This is essentially the same as sorting the rotations of $w\$$, hence we stick with the Combinatorics on words version. The BWT tends to produce long runs of the same symbol when a string is repetitive, and these runs can be

compressed into one symbol in the alphabet $\{(a,k) \mid a \in \Sigma, k \in [1,n]\}\}$ using *run-length encoding* (*rle*). A repetitiveness measure based on this idea is defined as $r(w) = |rle(BWT(w))|$. Although r is not as low as other reachable measures in practice, its size still can be bound in terms of z [19]. The selling point of the BWT is its multitude of practical applications representing sequences in Bioinformatics because of its indexing power [12].

It has been proven that, although deeply related, the measures r and $r_\$(w) = r(w\$)$ can behave wildly differently in some string families. For instance, in the Fibonacci words ending with the letter b, the number of BWT-runs is always 2 (in fact, this is true for all powers of conjugates of Standard Sturmian words [33]). On the other hand, if a \$ symbol is assumed to appear at the end, then the number of runs increases logarithmically [38]. Also, it has been proven that $v = O(r_\$)$ [38]. Because of these reasons, we are interested in studying both variants in this thesis.

2.2.4 String attractors. Kempa and Prezza introduced *string attractors* as a unifying framework and lower-bound for dictionary-based compressors [20].

Let w be a string of length n. A string attractor for w is a set of positions $\Gamma \subseteq [1,n]$ such that for each substring $w[i:j]$ of w, there exist integers $i', j' \in [1,n]$ and $k \in \Gamma$, such that $w[i:j] = w[i':j']$ and $i' \leq k \leq j'$. The measure $\gamma(w)$ is defined as the size of the smallest string attractor for w.

Computing $\gamma(w)$ is an NP-complete problem. The measure γ is a lower bound to b, and is also strictly smaller than b when considering the infinite family of Thue-Morse words [3]. On the other hand, it is unknown if γ space or even $o(\gamma \log(n/\gamma))$ space is reachable.

2.2.5 Substring complexity. Kociumaka et al. introduced a repetitiveness measure based on the complexity function, defined as $\delta(w) = max\{P_w(k)/k \mid k \in [1..|w|]\}$ [24].

The measure δ has several nice properties: it is computable in linear time, monotone, insensitive to reversals, resistant to small edits on w, and can be used to construct $O(\delta \log(n/\delta))$-space representations supporting efficient access and pattern matching queries [23, 24], and is a lower bound to almost every other theoretical or *ad-hoc* repetitiveness measure considered in the literature. On the other hand, $o(\delta \log(n/\delta))$ space has been proved to be unreachable [24].

2.3 Indexing highly-repetitive texts in compressed space

In many areas, like Bioinformatics, only compressing the text is not enough. There is also a need for processing the text in compressed form without ever decompressing it. The reason is that decompressing the data is unaffordable because of its size.

A *self-index* is a data structure that replaces the text (i.e., we can recover the original text from the data structure) and which allows us to solve some useful queries on the text. More ambitious are the so-called *compressed self-indexes*, which are self-indexes using substantially less space than the original text, generally $\Theta(u)$ where u is a compressibility measure [35]. With compressed self-indexes, we are not interested in decompressing the text. What we are interested in is on solving the queries using no more than $O(u)$ space at any step of the query.

Some of the most common queries to be supported in compressed space, from which many other queries are built, are the following:

- *Access:* Given a text $T[1:n]$ and a position $i \in [1,n]$, return the symbol $T[i]$.
- *Locate:* Given a text $T[1:n]$ and a pattern P, return a sorted list of all the (starting) positions where P appears in T.
- *Count:* Given a text $T[1:n]$ and a pattern P, return the number of (starting) positions where P appears in T. For this query, we expect a better time than the one used by the trivial solution using locate.

There exist many relevant queries other than access, locate, and count to be considered when designing a compressed self-index. For instance, in Bioinformatics are especially useful those queries that can be done by using a data structure called the *suffix-tree* [47]. Nevertheless, there is an implicit trade-off between space and query power: to solve more complex queries efficiently, usually more space is needed.

Compressors based on the Lempel-Ziv parsing are generally the best in terms of space for repetitive texts, yet they provide very few indexing capabilities. On the other hand, context-free grammars can provide efficient access, locate, count, and many other queries in $O(g)$ space. Run-length context-free grammar can solve direct access and some other queries like *range-minimum queries* efficiently in $O(g_{rl})$ space [39]. Moreover, some run-length grammars with *local-consistency* properties can also solve locate and count efficiently in $O(\delta \log(n/\delta))$ space [23, 24]. Although the $r_\$$ measure typically is greater than z and g, it has the advantage of enabling full suffix tree functionality in $O(r_\$ \log(n/r_\$))$ space [12].

2.4 Sensitivity to string operations

One of the most critical properties to study about repetitiveness measures is their *sensitivity* to simple *string operations*. The sensitivity of a measure to an operation is a function that calculates how badly the measure's value can increase after applying the operation to a string in a worst-case scenario. If a string operation is simple enough, one would like for a repetitiveness measure that its sensitivity to this operation was low. If this holds, then the measure cannot vary much when this specific operation is applied to any string, i.e., the measure is *resistant* to this operation. A measure that is resistant to many of the most common string operations is considered to be a *stable* measure, which is a desirable property to have in general.

Let $T[1, n]$ be a text. A *substitution* is an assignment of the form $T[i] \leftarrow a$ for $1 \leq i \leq n$ and $a \neq T[i]$. A *insertion* produces a string T' from T such that $T' = T[1 : i]aT[i + 1 : n]$ with $0 \leq i \leq n$ and $a \in \Sigma$. A *deletion* produces a string T' from T such that $T' = T[1 : i]T[i + 2 : n]$ with $0 \leq i \leq n - 1$. The *Levenshtein distance* (from now on *edit distance*) denoted $\mathsf{ed}(T, T')$, is a metric counting the minimum number of substitutions, insertions, and deletions, needed to obtain T' starting from T.

Other relevant string operations that are not as simple as edit operations are the reverse operation defined as $T^R = T[n] \dots T[1]$; the shift operation defined as $S(T) = T[2] \dots T[n]T[1]$; and morphism substitution, which given a morphism φ, yields the text $\varphi(T)$.

The *worst case multiplicative sensitivity* of a string operation with respect to a repetitiveness measure is the maximum ratio between the measure applied to the text after the operation and the measure applied to the original text. We follow the definitions of Akagi et al. [1] for the operations in scope:

$$
\begin{aligned}
\mathsf{MS}_{\mathsf{sub}}(C, n) &= \max_{T \in \Sigma^n} \{C(T')/C(T) : T' \in \Sigma^n \wedge \mathsf{ed}(T, T') = 1\},\\
\mathsf{MS}_{\mathsf{ins}}(C, n) &= \max_{T \in \Sigma^n} \{C(T')/C(T) : T' \in \Sigma^{n+1} \wedge \mathsf{ed}(T, T') = 1\},\\
\mathsf{MS}_{\mathsf{del}}(C, n) &= \max_{T \in \Sigma^n} \{C(T')/C(T) : T' \in \Sigma^{n-1} \wedge \mathsf{ed}(T, T') = 1\},\\
\mathsf{MS}_{\mathsf{rev}}(C, n) &= \max_{T \in \Sigma^n} \{C(T')/C(T) : T' \in \Sigma^n \wedge T' = T^R\},\\
\mathsf{MS}_{\varphi}(C, n) &= \max_{T \in \Sigma^n} \{C(T')/C(T) : T' \in \Sigma^* \wedge T' = \varphi(T)\}.
\end{aligned}
$$

Similarly, the *worst case additive sensitivity* of a string operation with respect to a repetitiveness measure is the maximum difference between the measure applied to the text after the operation and the measure applied to the original text.

$$\begin{aligned}
\mathsf{AS}_{\mathtt{sub}}(C,n) &= \max_{T\in\Sigma^n}\{C(T') - C(T) : T' \in \Sigma^n \wedge \mathsf{ed}(T,T') = 1\},\\
\mathsf{AS}_{\mathtt{ins}}(C,n) &= \max_{T\in\Sigma^n}\{C(T') - C(T) : T' \in \Sigma^{n+1} \wedge \mathsf{ed}(T,T') = 1\},\\
\mathsf{AS}_{\mathtt{del}}(C,n) &= \max_{T\in\Sigma^n}\{C(T') - C(T) : T' \in \Sigma^{n-1} \wedge \mathsf{ed}(T,T') = 1\},\\
\mathsf{AS}_{\mathtt{rev}}(C,n) &= \max_{T\in\Sigma^n}\{C(T') - C(T) : T' \in \Sigma^{n} \wedge T' = T^R\},\\
\mathsf{AS}_{\varphi}(C,n) &= \max_{T\in\Sigma^n}\{C(T') - C(T) : T' \in \Sigma^{*} \wedge T' = \varphi(T)\}.
\end{aligned}$$

A recent work by Akagi et al. exhibits lower bounds and upper bounds for the multiplicative and additive sensitivity to edit operations of many repetitiveness measures [1]. For some measures like r and $r_{\$}$, there is still plenty of work to do as lower and upper bounds are very far between them. For some other measures like v, no results have been proved.

3 PROBLEM STATEMENT

While several advances have been made in understanding repetitiveness measures in recent years, there are still too many open questions that need to be solved.

From a practical point of view, a deep understanding of the properties and relationships between repetitiveness measures is crucial to making better choices when selecting a compressor or index for some specific domain and task. By improving the body of knowledge, we can provide stronger guarantees about these compressors and indexes.

From a theoretical point of view, the open questions regarding repetitiveness measures are interesting combinatorial problems, and solving them allows us to build bridges between the areas of Combinatorics on words and Text compression.

Finally, concerning repetitiveness as a whole, a possibility is that there exist other sources of repetitiveness worth considering besides simple copy-paste, which at the moment, is essentially the only underlying idea under study. We believe addressing this problem is crucial to reach lower space on repetitive texts.

4 HYPOTHESIS

Our current knowledge of start-of-the-art repetitiveness measures can be substantially improved in several departments. Moreover, there exist forms of repetitiveness not being explored yet, which can help us to design compressors using less space on highly repetitive texts.

5 RESEARCH GOALS

In the following, we present the general goal we want to accomplish with this thesis.

5.1 General Goal

To understand better the phenomenon of repetitiveness as a source of compressibility.

5.2 Specific Goals

We focus on three specific goals contributing to the fulfillment of our general goal:

- **Goal 1:** Understand combinatorial properties of repetitiveness measures, like reachability and sensitivity to simple edit operations like insertion, deletion, and substitution, or even more complex transformations like the reverse operation and morphism application.
- **Goal 2:** Understand how measures relate between them asymptotically in terms of space.
- **Goal 3:** Design and study measures exploiting unexplored forms of repetitiveness. Also, compare them with the existing state-of-the-art measures.

6 METHODOLOGY

There are many repetitiveness measures that need to be further studied. In this thesis, we focus deeply only on some of them. Nevertheless, almost all repetitiveness measures appear in this thesis in at least one of our results.

We selected some topics that we believe that improving our understanding of them contribute to the completion of the goal. For each goal, we present the concepts and related work needed to understand these topics, then we explain the work we have already done, and finally, we present the pending work and give an idea of how we plan to accomplish it.

6.1 Understanding properties of repetitiveness measures (Goal 1)

Within this goal, we are mainly interested in providing some results about BWT-runs (both variants) and the size v of the lex-parse. The reason is that these are the state-of-the-art measures for which fewer results about their sensitivity to string operations are known [1]. In the case of the lex-parse, it has not been studied too much, even though it has some nice properties [38] that, in our opinion, make it worthy of more attention.

6.1.1 Sensitivity of r and $r_\$$ to edit operations. Advances have been made in the last years regarding the sensitivity of BWT to string operations, yet it is still not enough. Giuliani et al. showed that $\mathtt{MS}_{\mathtt{rev}}(r, n) = \Omega(\log n)$ by using *Fibonacci-plus* words [14]. On the other hand, Kempa and Kociumaka showed that $\mathtt{MS}_{\mathtt{rev}}(r, n) = O(\log^2 n)$. In an unpublished work, Giuliani et al. showed that $\mathtt{MS}_{\mathtt{ins}}(r, n), \mathtt{MS}_{\mathtt{ins}}(r, n)$ and $\mathtt{MS}_{\mathtt{ins}}(r, n)$ are $\Omega(\log n)$ [15]. There are still many open questions about lower and upper bounds for the additive sensitivity of r, and about the same operations but considering $r_\$$ instead of r.

We have made advances in the study of the sensitivity of the number of runs for both variants of the BWT to many simple edit operations [15]. More specifically, we showed that there exists a string family where the number r of BWT-runs can increase by a $\Theta(\sqrt{n})$ additive factor with respect to the length of the original string, after one insertion, deletion, substitution, or appending a \$ symbol at the end.

Let $s_i = \mathsf{ab}^i\mathsf{aa}$ and $e_i = \mathsf{ab}^i\mathsf{aba}^{i-2}$ for $i \in [2, k-1]$, and also $q_k = \mathsf{ab}^k\mathsf{a}$. Then, for $k > 2$ we define the word $w_k = (\prod_{i=2}^{k-1} s_i e_i) q_k$. This family of words satisfies that each factor $\mathsf{ab}^i\mathsf{a}$ for $i \in [2, k-1]$ appears exactly two times in the word (circularly), and the left-shifts of the only two rotations starting with this prefix are the smallest and greatest rotations starting with $\mathsf{b}^i\mathsf{a}$, respectively. Moreover, both of these rotations end with an a. All the rotations in between end with the letter b.

The main idea we used to prove that $\mathtt{AS}_{\mathtt{ins}}(r, n)$ (even if the inserted symbol is a \$), $\mathtt{AS}_{\mathtt{del}}(r, n)$ and $\mathtt{AS}_{\mathtt{sub}}(r, n)$ are $\Omega(\sqrt{n})$, is to show that after inserting, deleting, or substituting a symbol in some specific position of w_k, it happens that in between the rotations ending with the letter a, delimiting the blocks of rotations starting with $\mathsf{b}^i\mathsf{a}$ and $\mathsf{b}^{i+1}\mathsf{a}$, a letter b is inserted, yielding two new BWT-runs, for each $i \in [2, k-1]$. Then we show that in the remaining blocks of the BWT of w_k, the number of runs does not change too much. As $|w_k| = \Theta(k^2)$, we obtain the desired results. We use a similar approach to show analogous results for $r_\$$. The tightness of these results is still an open question.

6.1.2 Sensitivity of the BWT to morphism application. Surprisingly, the impact of (one) morphism application on the number r of runs of the BWT of a string had not been studied before.

In a very recent work [9], we have studied the different kinds of impact that the application of a morphism can have in the value r for all binary words containing both letters a and b.

Let φ be any fixed morphism. While for other repetitiveness measures like z and g it holds that $\mathtt{AS}_\varphi(z,n) = c_\varphi$ and $\mathtt{AS}_\varphi(g,n) = c'_\varphi$, where c_φ and c'_φ are constants depending only on the morphism φ, this is not the case for $\mathtt{AS}_\varphi(r,n)$. In fact, when φ is a binary morphism, it could be the case that $\mathtt{AS}_\varphi(r,n) = 0$, or that $\mathtt{AS}_\varphi(r,n) = c_\varphi$, or even that $\mathtt{AS}_\varphi(r,n) = \Omega(\sqrt{n})$ (recall that here our universe of strings contains only those where both letters are present).

The case $\mathtt{AS}_\varphi(r,n) = 0$ was completely characterized in [9]. The binary morphisms satisfying this condition are exactly the so-called *Sturmian morphisms*, i.e., morphisms mapping Sturmian words to Sturmian words [4], which have been proved in [34] to be compositions of the three morphisms

$$E : \begin{cases} \mathtt{a} & \mapsto & \mathtt{b} \\ \mathtt{b} & \mapsto & \mathtt{a} \end{cases} \qquad D : \begin{cases} \mathtt{a} & \mapsto & \mathtt{ab} \\ \mathtt{b} & \mapsto & \mathtt{a} \end{cases} \qquad G : \begin{cases} \mathtt{a} & \mapsto & \mathtt{ba} \\ \mathtt{b} & \mapsto & \mathtt{a} \end{cases}$$

Moreover, we proved that *Thue–Morse-like* morphisms, i.e. those of the form

$$\tau_{p,q} : \begin{cases} \mathtt{a} & \mapsto & \mathtt{ab}^p \\ \mathtt{b} & \mapsto & \mathtt{ba}^q \end{cases}$$

satisfy that $\mathtt{AS}_\varphi(r,n) = 2$. We do not know if these are the only morphisms with this behavior. Naturally, we can compose Thue-Morse-like morphisms with Sturmian morphisms to construct other morphisms such that $\mathtt{AS}_\varphi(r,n) = 2k$ for any $k \in \mathbb{N}$.

On the other hand, we observed the opposite behavior in the *period-doubling* morphism pd. In particular, we showed that $\mathtt{AS}_{pd}(r,n) = \Theta(\sqrt{n})$, using a string family specially designed for this task and very reminiscent of the one of the previous subgoal.

There are many open questions regarding this topic. We are interested in characterizing morphisms according to their functions $\mathtt{MS}_\varphi(r,n)$ instead of $\mathtt{AS}_\varphi(r,n)$. We also plan to generalize our results to bigger alphabets.

6.1.3 BWT-runs of morphic words on general alphabets. Understanding the value r for prefixes of words generated by morphisms has raised a lot of interest in the latest years [5, 10]. The reason is that because of the nature of these words, they make good theoretical representatives for repetitive sequences. Hence, by studying the number of BWT-runs on these words, we better understand the performance of r in highly repetitive settings.

Brlek et al. studied the number of runs of the BWT of Thue-Morse words, and also for some generalizations of the Fibonacci words [5]. In recent work, Frosini et al. studied the value r for families of (some) prefixes of binary morphic words [10]. More specifically, they showed that $r = O(\log n)$ in the family of words $\{\varphi^i(c) \mid i \geq 0\}$ when $c \in \{\mathtt{a}, \mathtt{b}\}$ and φ is a binary morphism prolongable on c. Moreover, they showed that for a wide class of binary morphism, this upper-bound is tight, i.e., it also holds that $r = \Omega(\log n)$.

One of the reasons why we started to study the impact of morphism application on r (on [9]) is that this knowledge could be helpful to find lower and upper bounds for the number of runs of purely morphic words on alphabets of size bigger than 2, or even the more general class morphic words. This would be a generalization of the work of Frosini et al. [10].

One idea we had was to use the observation that, in the hypothetical case that the additive sensitivity of r to a fixed morphism φ prolongable on c was a constant k depending only on φ, then we could easily derive that $r \leq ki + 1 = O(i)$ on the family $\{\varphi^i(c) \mid i \geq 0\}$. Yet, this hypothesis is not true in general, as we can see with the period-doubling morphism [9].

Even if for the period-doubling morphism the additive sensitivity on r is not constant, it holds that $r = O(i)$ on the family of iterations of this morphism, which can be derived from the work by Frosini et al. [10]. We believe that an analogous result is true for general alphabets, but we need to find another approach to handle it.

We plan to study bounds for the impact of applying a morphism over itself. We want to show that successive applications of the same morphism do not increase the number of runs considerably, even if the morphism has unbounded $\mathsf{AS}_\varphi(r, n)$ function. To do so, we will start by running more brute-force experiments to see if they support our conjecture. If that is the case, we will start by studying small classes of morphisms satisfying this property and try to generalize from there.

Finally, we want to prove similar results, but for the measure $r_\$$, which is more commonly used in the Text compression and Bioinformatics communities.

6.1.4 Sensitivity of lexicographic parsings. Some years ago, Navarro et al. introduced ordered parsings [38]. In particular, they introduced lexicographic parsings, which are ordered parsings, where the order relation used is the suffix order. The left-to-right greedy lexicographic parsing, called lex-parse, has been proven to be optimal, and its size v is considered a measure of repetitiveness.

Not many advances in lexicographic parsing have been made since their introduction [38], other than a work from Dominik Köppl dealing with their construction [27]. We believe that the lex-parse deserves further study, as it is easy to compute in linear time, it always holds that $v = O(r_\$)$, and it is an open question if $v = o(z)$ [38], which if true would make the lex-parse even more interesting.

Experiments suggest that the measure v behaves similarly to $r_\$$ in terms of its additive and multiplicative sensitivity to insertion, deletion, substitution, and even to the reverse operation. We have some ideas of what families we could use to find lower bounds on the sensitivity of v to string operations: Fibonacci words, Fibonacci-plus words, and the same words used to find lower bounds for $\mathsf{AS}_{\mathtt{ins}}(r_\$, n)$, $\mathsf{AS}_{\mathtt{del}}(r_\$, n)$ and $\mathsf{AS}_{\mathtt{sub}}(r_\$, n)$ [15].

6.2 Understanding relationships between repetitiveness measures (Goal 2)

There exist many interesting open questions regarding the asymptotic relations between the state-of-the-art repetitiveness measures [36]. Most of the pending work on this thesis is, in fact, related to this specific goal.

We will focus mainly on improving our understanding of the relation between the size v of the smallest lexicographic parsing with the size z of the Lempel-Ziv parsing, and finding an approximation ratio for the Re-Pair heuristic with respect to the size g of the smallest context-free grammar, or at least, with respect to $r_\$$. The reason why we focus on these topics and not others is that obtaining results on them would have important practical implications that could be useful immediately. If $v = o(z)$, then we have a reachable measure lower than z that we can easily compute in linear time [38]. If we find an approximation ratio for Re-Pair, then we can provide some degree of guarantee on the space achieved by this heuristic. Although Re-Pair performs well in practice, currently, we do not have strong guarantees on it [7].

If there is time left after studying the questions above, we plan to use it to relate the size c of the smallest collage system to the size z of the Lempel-Ziv parsing. This is an interesting open question, but we do not know how to compute c efficiently nor have heuristics to produce collage systems. Hence, we give lower priority to this task.

6.2.1 Lempel-Ziv with overlaps vs. lexicographic parsings. It always holds that $v = O(r_\$)$, and in the family of Fibonacci words ending with a, it holds that $z = \Omega(v \log n)$ [38]. Nevertheless, it is still unknown if, in general, $v = O(z)$ or if z can be $o(v)$ for some string family.

There are experiments suggesting that v performs worse than z on strings that are the concatenation of several copies of a text, where each copy of the text is modified by applying an edit

operation to the previous copy [38]. This gives an intuition of the properties that a string family where $z = o(v)$ should satisfy if it exists. On the other hand, the difference in favor of z in these experiments is unlikely to be asymptotic.

Another observation that we think could be helpful to solve this question is the difference in sensitivity to string operations between z and v. While z has been proven to be resistant to simple edit operations [1], the measure v is likely sensitive to all these operations. We think we can exploit these differences to find an example where $z = o(v)$.

Finally, there is a deep relationship between v and $r_\$$, and it is known that z is uncomparable to $r_\$$. We think this relationship can be useful to establish a similar result between z and v.

6.2.2 An upper bound for Re-Pair in terms of the size of smallest grammar or BWT-runs. Since its introduction, the Re-Pair heuristic [28] has been one of the most successful grammar compressors. There exist other grammar-based compressors providing log-approximations of g [18, 44], yet they do not perform as well as Re-Pair in most scenarios. On the other hand, only a few weak theoretical lower and upper bounds on the size of the resulting grammar from Re-Pair have been proven [7], which are far from tight.

Upper bounding the size of Re-Pair would be a nice result, even if it is in terms of a measure other than g. There have been some tries to show that Re-Pair provides a grammar of size $O(r_\$ \log(n/r_\$))$ [16, 17]. Nevertheless, Gagie et al. found that these proofs were not correct [11].

We believe that we should be able to prove at least a poly-logarithmic upper bound on the size of the grammar produced by Re-Pair with respect to $r_\$$.

6.2.3 Lempel-Ziv with overlaps vs. collage systems. An open question regarding collage systems is how they relate to the Lempel-Ziv parsing. It has been shown that the size c of the smallest collage system is $O(z)$ [38]. On the other hand, it is unknown if there exists a family where $c = o(z)$. We believe this is the case and plan to prove it in the final period of this thesis.

6.3 Defining new repetitiveness measures (Goal 3)

Before the initial proposal of this thesis, the smallest reachable measure of repetitiveness was the size b of the smallest bidirectional macro scheme. In order to break this lower bound and, hopefully, approach the size γ of the smallest string attractor (a measure unknown to be reachable), we proposed a measure exploiting the description power of iterated morphisms, which we called ℓ [40, 41]. Iterated morphisms are commonly used to generate repetitive sequences known as purely morphic words. They are also used to simulate the growth of plants and algae when extended with some other features [30, 31]. We went even further and combined these mechanisms with traditional copy-paste techniques, yielding a new reachable measure called ν that is asymptotically strictly better than b in terms of space [40, 41]. In the following, we describe our advances in this matter.

6.3.1 L-systems and the measure ℓ. We introduced a mechanism for generating repetitive sequences based on Lindenmayer systems (L-systems) and a measure based on this mechanism called ℓ [41]. Then, we proved some results about this measure showing that it is a reasonable and competitive measure that can go lower than b in some scenarios.

We build on the CPD0L variant of L-systems [30, 31]. Formally, a CPD0L-system is a 4-tuple $L = (\Sigma, \varphi, \tau, s)$, where Σ is the *alphabet*, φ is the set of *rules* (a non-erasing endomorphism on Σ^*), τ is a coding on Σ^*, and $s \in \Sigma^+$ is the *axiom*. The system generates the sequence $(\tau(\varphi^d(s)))_{d\in\mathbb{N}}$. The "D0L" stands for *deterministic L-system with 0 interactions*. The "P" stands for *propagating*, which means that it has no ε-rules. The "C" stands for *coding*, which means that the system is extended with a coding. To define a compressor based on CPD0L-system, we extend them to 6-tuples by fixing

a level d and using another parameter n, so we can uniquely determine the string $\tau(\varphi^d(s))[1:n]$. For simplicity, we refer to these extended CPD0L-systems just as L-systems.

The *size* of an L-system $L = (\Sigma, \varphi, \tau, s, d, n)$ is $size(L) = size(\varphi) + |s| + |\Sigma| + 2$, which accounts for the size of the morphism φ, the length of the axiom s, the coding τ, and the values d and n, so we can effectively represent the system using $size(L)$ space. The measure ℓ is defined as the size of the smallest L-system generating a string w, and it is clearly reachable.

We showed that the measure ℓ is always $O(g)$ using a simple simulation argument [41]. It also holds (by definition) that the measure ℓ is constant in the family of finite morphic words generated by any fixed morphism and coding. This includes, among many others, the family of Fibonacci words and the family of Thue-Morse words. Hence, it holds that ℓ can be strictly smaller than b, as Bannai et al. showed that $b = \Omega(\log n)$ on the family of Thue-Morse words [3].

More surprisingly, we showed that $\delta = \Omega(\ell \log n)$ on a family of words with quadratic factor complexity constructed by iterating a morphism [41]. We improved this result later to $\delta = \Omega(\ell\sqrt{n})$ by considering another (similar) string family [40]. This observation, together with a result of Kociumaka et al., showing a string family where any reachable measure has to be $\Omega(\delta \log n)$ [24], immediately implies that ℓ and δ are uncomparable as measures of repetitiveness.

Not only the measure ℓ is uncomparable to δ. It is also uncomparable to almost any other repetitiveness measure to date [40]. This shows that ℓ, although reachable and competitive as a repetitiveness measure, captures the regularities in strings in a form that is largely orthogonal to other repetitiveness measures. As the underlying regularities being captured by ℓ and the other measures are apparently different, we tried to combine them to obtain more powerful measures/compressors.

6.3.2 NU-systems and the measure ν. We proposed another mechanism for generating repetitive sequences, called a NU-system, which combines L-systems with bidirectional macro schemes [41].

A *NU-system* is a 6-tuple $N = (V, R, \Gamma, s, d, n)$ that generates a unique string in a similar way to an L-system. The key difference is that on the right-hand side of its rules, a NU-system is permitted to have special symbols of the form $a(k)[i:j]$, whose meaning is to generate the k-th level from a, then extract the substring starting at position i and ending at position j, and finally apply the coding τ to the resulting substring. A NU-system is valid if it correctly defines a string, which is decidable to detect.

The size of a NU-system is defined analogously to the size of L-systems, with the extraction symbols $a(k)[i:j]$ being symbols of length 4. The measure ν (do not confuse with the size v of the lex-parse) is defined as the size of the smallest NU-system generating a string w.

Few things are known about ν other than some asymptotic relations. It holds almost by definition that $\nu = O(\ell)$ and $\nu = O(b)$ [41]. There also exist families where both asymptotic bounds are strict. Recently, we showed a string family where $\nu = o(min(\ell, b))$, which means that ν is most powerful than just the sum of its parts [40]. This makes ν the unique smallest reachable measure to date and also an example of what can be achieved by combining copy-paste with morphism application.

We summarize our current knowledge of the measures ℓ and ν with respect to other state-of-the-art repetitiveness measures in Figure 1.

6.3.3 Future work. There is still much work to do regarding L-systems and NU-systems. While both mechanisms yield reachable measures, we do not know what kind of indexing power we can obtain using $\Theta(\ell)$ or $\Theta(\nu)$ space. In the case of NU-systems, we do not even know how to decompress them in a reasonable amount of time. Both of these measures are probably NP-hard to compute (they are probably hard even to approximate), but this also has to be proved. While we will try to answer some of these questions, —the direct access for L-systems and decompression for NU-systems—, L-systems are difficult mechanisms to work with. For instance, the advances in

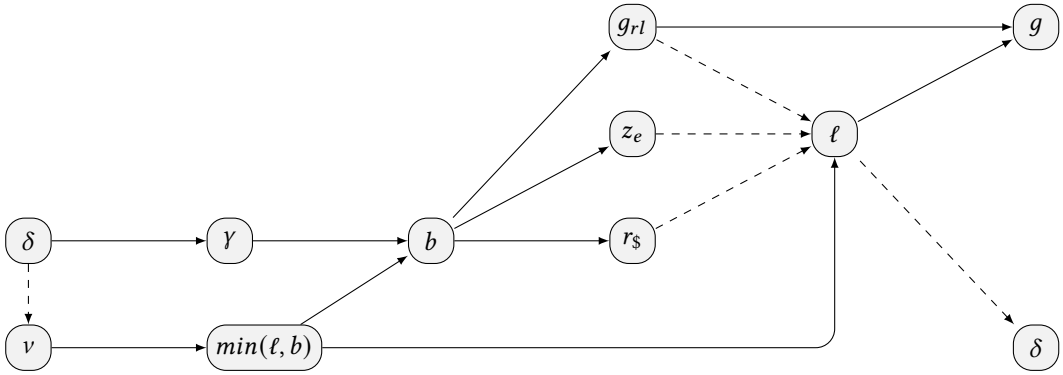

Fig. 1. Asymptotic relations between ℓ, ν and other repetitiveness measures. A black arrow from v_1 to v_2 means that it always holds that $v_1 = O(v_2)$, and there exists a string family where $v_1 = o(v_2)$. A dashed arrow from v_1 to v_2 means that there exists a family where $v_1 = o(v_2)$. The measure δ appears two times for clarity.

algorithms for inferring L-systems generating specific strings or sequences of strings have been minimal since their introduction. Hence, our expectations in this regard are low.

A problem we are currently working on is simplifying the definitions of both measures as most as we can. For instance, in the case of ℓ, we would like to know if the coding is effectively adding power to the mechanism, as it was an ad-hoc addition to claim that the measure is constant on prefixes of morphic words. Similar questions can be raised about NU-systems, and other features of this measure.

Our main focus from here onward regarding ν is on relating this measure to the size γ of the smallest string attractor. If we show that $\nu = o(\gamma)$ (or at least $\nu = o(\gamma \log n)$), we have shown that γ (or at least $\gamma \log n$) space is reachable, which is a long time open question. To do so, we plan either to construct a NU-system from a string attractor of the needed size or show that it is not possible. In the latter case, another approach is needed to answer this open question, which is still useful to know.

7 WORK PLAN

First, we summarize the work we have done so far. Then, we define some tasks to be solved in order to fulfill each of the specific goals of this thesis.

7.1 Advanced Work

We have participated in the publication of the following two papers at international conferences:

(1) G. Navarro and C. Urbina. 2021. On Stricter Reachable Repetitiveness Measures. In *Proc. 28th International Symposium on String Processing and Information Retrieval (SPIRE) (LNCS 12944).* 193–206.
(2) G. Navarro, F. Olivares, and C. Urbina. 2022. Balancing Run-Length Straight-Line Programs. In *Proc. 29th International Symposium on String Processing and Information Retrieval (SPIRE) (LNCS 13617).* 117–131.

In our first published paper, we introduced the measures ℓ and ν and proved some results about them [41].

In our second published paper, we showed that every run-length grammar could be balanced. This allows us to argue that we can answer certain queries efficiently in $O(g_{rl})$ compressed space [39]; hence, it is loosely related to this thesis.

We have sent the following papers to be published at the international conference *34-th Annual Symposium on Combinatorial Pattern Matching* (CPM):

(3) G. Fici, G. Romana, M. Sciortino, and C. Urbina. [n. d.]. On the impact of morphisms on BWT-runs. ([n. d.]). Unpublished manuscript.
(4) G. Navarro and C. Urbina. [n. d.]. L-systems for measuring repetitiveness. ([n. d.]). Unpublished manuscript.

The third work (3) was the result of a collaboration carried out during a stay of two months and a half at the University of Palermo, Italy. In this work, we made considerable advances in the characterization of morphisms according to their impact on the BWT of binary words [9].

In the fourth work (4), we further studied the relations of ℓ and ν with respect to other repetitiveness measures [40].

Finally, we sent the following paper to be peer-reviewed at the international conference *Developments in Language Theory 2023* (DLT):

(5) S. Giuliani, S. Inenaga, Z. Lipták, G. Romana, M. Sciortino, and C. Urbina. [n. d.]. Bit catastrophes for the Burrows-Wheeler Transform. ([n. d.]). Unpublished manuscript.

In this work (5), our contribution was mainly to the study of the additive sensitivity of r and $r_\$$ to edit operations [15]. The paper contains many other interesting results regarding the multiplicative sensitivity of r to edit operations too.

7.2 Future Work

For the remaining part of this thesis, we have identified and plan to work on the following tasks:

Goal 1: Understanding properties of repetitiveness measures.
Task 1.1: Further study the sensitivity of BWT-runs to edit operations.
Task 1.2: Further study the effect of morphism application on BWT-runs.
Task 1.3: Study the number of BWT-runs of morphic words on general alphabets.
Task 1.4: Study the sensitivity of lexicographic parsings to edit operations.
Goal 2: Understanding relationships between repetitiveness measures.
Task 2.1: Find a family where $z = o(v)$, or prove that it is not possible.
Task 2.2: Try to find a non-trivial upper bound for Re-Pair in terms of g or $r_\$$.
Task 2.3: Find a family where $c = o(z)$ or prove that it is impossible (optional).
Goal 3: Defining new repetitiveness measures.
Task 3.1: Further study properties of L-systems and the measure ℓ.
Task 3.2: Further study properties of NU-systems and the measure ν.

The level of detail of each task depends on the uncertainty surrounding the approach to each goal. For instance, we have already done considerable work on Goal 1 and Goal 3, and we have given thought to these tasks. We have raised many more questions about them than we could possibly handle; hence we are not completely sure which of these questions we will answer at the end. Conversely, the tasks associated with Goal 2 are more specific, as there are fewer open question that comes to mind, and we know exactly what result we should aim to prove, even if we are not sure how to do it at the time.

In particular, Task 2.2 has been tried to be solved several times in the past. We believe we can make a contribution on the topic and will spend time accordingly, yet it is not realistic for us to say that we will indeed find an upper-bound, even if we will try to do so. Task 2.3 is considered optional because we will only work on it after all the other tasks are finished, and it does not have many implications like the other tasks within Goal 2.

Goal	Task	Year 2023										Year 2024						
		Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Jan	Feb	Mar	Apr	May	Jun	Jul
Thesis Proposal		○	×															
Goal 1	Task 1.1		○		○	○	○	○	○									
	Task 1.2		○		○	○	○	○	○									
	Task 1.3			○	○	○	○	○	○									
	Task 1.4	○	○	○	○	○												
Goal 2	Task 2.1	○	○	○	○	○												
	Task 2.2	○	○	○	○	○												
	Task 2.3														○	○	○	
Goal 3	Task 3.1								○	○	○	○	○	○				
	Task 3.2								○	○	○	○	○	○				
Thesis Writing						○	○					○	○	○	○	○	○	
Thesis Defense																		×

Table 1. Work plan for the remaining tasks of this thesis. A ○ symbol at the intersection of a task with a month means that we plan to work on that task in that month. The × symbol means that also an oral presentation is expected to happen in that month.

The overall work plan for the remaining part of this thesis is summarized in Table 1, including some estimates of the time needed to solve each task. If all the tasks are finished, and there is still time left, we plan to continue working on studying the Lempel-Ziv-end parsing.

REFERENCES

[1] T. Akagi, M. Funakoshi, and S. Inenaga. 2023. Sensitivity of string compressors and repetitiveness measures. *Information and Computation* 291 (2023), 104999.

[2] J.-P. Allouche and Jeffrey Shallit. 2003. *Automatic Sequences: Theory, Applications, Generalizations.* Cambridge University Press.

[3] H. Bannai, M. Funakoshi, T. I, D. Köppl, T. Mieno, and T. Nishimoto. 2021. A Separation of γ and b via Thue–Morse Words. In *Proc. 28th SPIRE (LNCS 12944).* 167–178.

[4] J. Berstel and P. Séébold. 1993. A Characterization of Sturmian Morphisms. In *MFCS (Lect. Notes Comput. Sci., Vol. 711).* Springer, 281–290.

[5] S. Brlek, A. Frosini, I. Mancini, E. Pergola, and S. Rinaldi. 2019. Burrows-Wheeler Transform of Words Defined by Morphisms. In *Proc. 30th IWOCA (LNCS 11638).* 393–404.

[6] M. Burrows and D. Wheeler. 1994. *A block sorting lossless data compression algorithm.* Technical Report 124. Digital Equipment Corporation.

[7] M. Charikar, E. Lehman, Ding Liu, R. Panigrahy, M. Prabhakaran, A. Sahai, and A. Shelat. 2005. The smallest grammar problem. *IEEE Transactions on Information Theory* 51, 7 (2005), 2554–2576.

[8] A. de Luca and A. De Luca. 2006. Some characterizations of finite Sturmian words. *Theoretical Computer Science* 356, 1 (2006), 118–125. In honour of Professor Christian Choffrut on the occasion of his 60th birthday.

[9] G. Fici, G. Romana, M. Sciortino, and C. Urbina. [n. d.]. On the impact of morphisms on BWT-runs. ([n. d.]). Unpublished manuscript.

[10] A. Frosini, I. Mancini, S. Rinaldi, G. Romana, and M. Sciortino. 2022. Logarithmic Equal-Letter Runs for BWT of Purely Morphic Words. In *Developments in Language Theory.* Springer International Publishing, Cham, 139–151.

[11] T. Gagie, G. Navarro, and N. Prezza. 2017. Optimal-Time Text Indexing in BWT-runs Bounded Space. arXiv:1705.10382 [cs.DS]

[12] T. Gagie, G. Navarro, and N. Prezza. 2018. Optimal-Time Text Indexing in BWT-runs Bounded Space. In *Proc. 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 1459–1477.

[13] J. K. Gallant. 1982. *String Compression Algorithms*. Ph. D. Dissertation. Princeton University.

[14] S. Giuliani, S. Inenaga, Z. Lipták, N. Prezza, Ma. Sciortino, and A. Toffanello. 2021. Novel Results on the Number of Runs of the Burrows-Wheeler-Transform. In *SOFSEM 2021: Theory and Practice of Computer Science*. 249–262.

[15] S. Giuliani, S. Inenaga, Z. Lipták, G. Romana, M. Sciortino, and C. Urbina. [n. d.]. Bit catastrophes for the Burrows-Wheeler Transform. ([n. d.]). Unpublished manuscript.

[16] R. González and G. Navarro. 2007. Compressed Text Indexes with Fast Locate. In *Combinatorial Pattern Matching*. Springer Berlin Heidelberg, Berlin, Heidelberg, 216–227.

[17] R. González, G. Navarro, and H. Ferrada. 2015. Locally Compressed Suffix Arrays. *ACM J. Exp. Algorithmics* 19, Article 1.1 (jan 2015), 30 pages.

[18] A. Jeż. 2015. Approximation of grammar-based compression via recompression. *Theoretical Computer Science* 592 (2015), 115–134.

[19] D. Kempa and T. Kociumaka. 2022. Resolution of the Burrows-Wheeler Transform Conjecture. *Commun. ACM* 65, 6 (may 2022), 91–98.

[20] D. Kempa and N. Prezza. 2018. At the Roots of Dictionary Compression: String Attractors. In *Proc. 50th STOC* (Los Angeles, CA, USA). 827–840.

[21] T. Kida, T. Matsumoto, Y. Shibata, M. Takeda, A. Shinohara, and S. Arikawa. 2003. Collage system: a unifying framework for compressed pattern matching. *Theor. Comp. Sci.* 298, 1 (2003), 253–272.

[22] J. C. Kieffer and E.H. Yang. 2000. Grammar-based codes: A new class of universal lossless source codes. *IEEE Trans. Inf. Theory* 46, 3 (2000), 737–754.

[23] T. Kociumaka, G. Navarro, and F. Olivares. 2022. Near-Optimal Search Time in δ-Optimal Space. In *LATIN 2022: Theoretical Informatics*. Springer International Publishing, Cham, 88–103.

[24] T. Kociumaka, G. Navarro, and N. Prezza. 2020. Towards a Definitive Measure of Repetitiveness. In *Proc. 14th LATIN (LNCS 12118)*. 207–219.

[25] S. Kreft and G. Navarro. 2010. LZ77-Like Compression with Fast Random Access. In *2010 Data Compression Conference*. 239–248.

[26] S. Kreft and G. Navarro. 2013. On Compressing and Indexing Repetitive Sequences. *Theoretical Computer Science* 483 (2013), 115–133.

[27] D. Köppl. 2022. Computing Lexicographic Parsings. In *2022 Data Compression Conference (DCC)*. 232–241.

[28] N. J. Larsson and A. Moffat. 1999. Offline dictionary-based compression. In *Proceedings DCC'99 Data Compression Conference (Cat. No. PR00096)*. 296–305.

[29] A. Lempel and J. Ziv. 1976. On the Complexity of Finite Sequences. *IEEE Trans. Inf. Theory* 22, 1 (1976), 75–81.

[30] A. Lindenmayer. 1968. Mathematical models for cellular interactions in development I. Filaments with one-sided inputs. *J. Theor. Biol.* 18, 3 (1968), 280–299.

[31] A. Lindenmayer. 1968. Mathematical models for cellular interactions in development II. Simple and branching filaments with two-sided inputs. *J. Theor. Biol.* 18, 3 (1968), 300–315.

[32] M. Lothaire. 2002. *Algebraic Combinatorics on Words*. Cambridge University Press.

[33] S. Mantaci, A. Restivo, and M. Sciortino. 2003. Burrows–Wheeler transform and Sturmian words. *Inf. Process. Lett.* 86, 5 (2003), 241–246.

[34] F. Mignosi and P. Séébold. 1993. Morphismes sturmiens et règles de Rauzy. *Journal de théorie des nombres de Bordeaux* 5, 2 (1993), 221–233.

[35] G. Navarro. 2016. *Compact Data Structures – A practical approach*. Cambridge University Press. ISBN 978-1-107-15238-0. 536 pages..

[36] G. Navarro. 2021. Indexing Highly Repetitive String Collections, Part I: Repetitiveness Measures. *ACM Comp. Surv.* 54, 2 (2021), article 29.

[37] G. Navarro. 2021. Indexing Highly Repetitive String Collections, Part II: Compressed Indexes. *ACM Comput. Surv.* 54, 2, Article 26 (feb 2021), 32 pages.

[38] G. Navarro, C. Ochoa, and N. Prezza. 2021. On the Approximation Ratio of Ordered Parsings. *IEEE Trans. Inf. Theory* 67, 2 (2021), 1008–1026.

[39] G. Navarro, F. Olivares, and C. Urbina. 2022. Balancing Run-Length Straight-Line Programs. In *Proc. 29th International Symposium on String Processing and Information Retrieval (SPIRE) (LNCS 13617)*. 117–131.

[40] G. Navarro and C. Urbina. [n. d.]. L-systems for measuring repetitiveness. ([n. d.]). Unpublished manuscript.

[41] G. Navarro and C. Urbina. 2021. On Stricter Reachable Repetitiveness Measures. In *Proc. 28th International Symposium on String Processing and Information Retrieval (SPIRE) (LNCS 12944)*. 193–206.

[42] T. Nishimoto, T. I, S. Inenaga, H. Bannai, and M. Takeda. 2016. Fully Dynamic Data Structure for LCE Queries in Compressed Space. In *41st International Symposium on Mathematical Foundations of Computer Science (MFCS 2016)*

(Leibniz International Proceedings in Informatics (LIPIcs), Vol. 58). 72:1–72:15.

[43] M. Przeworski, R. R. Hudson, and A. Di Rienzo. 2000. Adjusting the focus on human variation. *Trends in genetics : TIG* 16, 7 (July 2000), 296—302.

[44] W. Rytter. 2003. Application of Lempel–Ziv factorization to the approximation of grammar-based compression. *Theoretical Computer Science* 302, 1 (2003), 211–222.

[45] Z. D. Stephens, S. Y. Lee, F. Faghri, R. H. Campbell, C. Zhai, M. J. Efron, R. Iyer, M.l C. Schatz, S. Sinha, and G. E. Robinson. 2015. Big Data: Astronomical or Genomical? *PLOS Biology* 13, 7 (07 2015), 1–11.

[46] J. A. Storer and T. G. Szymanski. 1982. Data Compression via Textual Substitution. *J. ACM* 29, 4 (1982), 928–951.

[47] P. Weiner. 1973. Linear pattern matching algorithms. In *14th Annual Symposium on Switching and Automata Theory (swat 1973)*. 1–11.

[48] J. Ziv and A. Lempel. 1977. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory* 23, 3 (1977), 337–343.