Fundamenta Informaticae XX (2015) 1–19
DOI 10.3233/FI-2012-0000
IOS Press

Maximizing T-complexity

Gregory Clark
U.S. Department of Defense
gsclark@tycho.ncsc.mil

Jason Teutsch
Penn State University
teutsch@cse.psu.edu

Abstract. We investigate Mark Titchener's *T-complexity*, an algorithm which measures the information content of finite strings. After introducing the T-complexity algorithm, we turn our attention to a particular class of "simple" finite strings. By exploiting special properties of simple strings, we obtain a fast algorithm to compute the maximum T-complexity among strings of a given length, and our estimates of these maxima show that T-complexity differs asymptotically from Kolmogorov complexity. Finally, we examine how closely de Bruijn sequences resemble strings with high T-complexity.

1. Introduction

How efficiently can we generate random-looking strings? Random strings play an important role in data compression since such strings, characterized by their lack of patterns, do not compress well. For practical compression applications, compression must take place in a reasonable amount of time (typically linear or nearly linear time). The degree of success achieved with the classical compression algorithms of Lempel and Ziv [9, 21, 22] and Welch [19] depend on the complexity of the input string; the length of the compressed string gives a measure of the original string's complexity.

In the same monograph in which Lempel and Ziv introduced the first of these algorithms [9], they gave an example of a succinctly describable class of strings which yield optimally poor compression up to a logarithmic factor, namely the class of lexicographically least de Bruijn sequences for each length. On one hand, one can view the incompressibility of de Bruijn sequences as evidence of "randomness," but on the other hand one can also view this phenomenon as a failure of the Lempel-Ziv algorithm to detect patterns. Indeed, the Lempel-Ziv and Welch algorithms achieve compression by detecting repeating patterns whereas de Bruijn sequences minimize such pattern repetitions.

Do de Bruijn sequences defy compression under other general-purpose compression schemes? Do there exist other natural classes of "random" strings besides de Bruijn sequences? The present exposition investigates these questions. Titchener [17] introduced an algorithm called *T-complexity* which effectively measures string complexity using different principles from those employed by Lempel, Ziv, and Welch (Section 2), and so one may view T-complexity as the basis for an alternative compression scheme. In this paper, we show how to find and quickly calculate the maximum T-complexity among strings of a given length (Theorem 5.2) and then give a precise analytic estimate for these values (Theorem 6.3). Our asymptotic bounds confirm conjectures from [7] and [16].

We conclude with some illustrative applications. It has been conjectured in earlier works [4, 5] that T-complexity approximates Kolmogorov complexity. We show that T-complexity is neither an upper bound nor a lower bound for Kolmogorov complexity (Section 7). We also examine to what degree de Bruijn sequences exhibit incompressibility in the context of T-complexity (Section 8). There exists a close correlation between the substring distribution in Kolmogorov random strings and de Bruijn sequences [10, Theorem 2.6.1]. Since T-complexity uses substrings in its computation, one might expect that this statistical property would make the T-complexity of de Bruijn sequences relatively high, as was the case for the Lempel-Ziv algorithms (assuming a reasonable window size for [21]). But in reality this is not necessarily what happens.

2. How to compute T-complexity

T-complexity is an algorithm which maps finite strings to reals [17]. Like the well-known LZ-complexity algorithm [9], the T-complexity algorithm progressively looks at longer and more complex symbols. We introduce some notation here in order to describe the algorithm.

For any finite alphabet $\mathcal{A}$, $\mathcal{A}^*$ denotes the nonempty strings over $\mathcal{A}$. In order to describe the T-complexity algorithm, we must distinguish between strings over the alphabet $\mathcal{A}$ and symbols used to describe these strings. To this end, let $\mathcal{L} : \mathcal{A}^* \to \mathbb{N}$ be a canonical one-to-one encoding of the members of $\mathcal{A}^*$, and call the members of the image $\mathcal{L}(\mathcal{A}^*)$ the *symbols* of $\mathcal{A}^*$. A *word* is an element of $\mathcal{L}(\mathcal{A}^*)^*$. We define $|\sigma|$ to be the length of the string σ, and for a symbol p, $|p|$ denotes the length of the string $\mathcal{L}^{-1}(p)$. $\sigma \cdot \tau$ denotes the concatenation of strings (or symbols) σ and τ, and σ^k denotes σ concatenated with itself k times.

Definition 2.1. For $x_1, x_2, \ldots, x_n \in \mathcal{L}(\mathcal{A}^*)$, let

$$\mathsf{join}(x_1 x_2 \cdots x_n) = \mathcal{L}\left[\mathcal{L}^{-1}(x_1)\mathcal{L}^{-1}(x_2)\cdots\mathcal{L}^{-1}(x_n)\right]$$

denote the symbol of $\mathcal{L}(\mathcal{A}^*)$ which is the concatenation of the input symbols when interpreted as strings over $\mathcal{A}$.

Algorithm 1 describes the T-complexity algorithm. Note that at each stage of this algorithm, $\mathcal{L}^{-1}(\mathsf{join}\,\sigma') = \sigma$, however formally σ' and σ consist of different symbols in $\mathcal{L}$. Furthermore, the main loop must eventually terminate because $|\sigma'| < |\sigma|$. We will use $T(\sigma)$ to denote the T-complexity of a string $\sigma \in \mathcal{A}^*$. This notation is consistent with the function T in Algorithm 1 but with slight notational abuse because the input to this new function T may be a string rather than a word. Finally, we may view T-complexity as a function from strings to reals.

Algorithm 1 The T-complexity algorithm

Input: a finite string $x_0x_1\cdots x_n \in \mathcal{A}^*$.
Output: a real number $T[\mathcal{L}(x_0)\mathcal{L}(x_1)\cdots\mathcal{L}(x_n)]$.

As a preprocessing step, let $\sigma = \mathcal{L}(x_0)\mathcal{L}(x_1)\cdots\mathcal{L}(x_n)$.
function $T(\sigma)$
  if $|\sigma| \leq 1$ **then return** 0;
  else
    let $p \in \mathcal{L}(\mathcal{A}^*)$ be the *penultimate* (i.e. next to rightmost) symbol of σ.
    let k be the maximum number such that $\sigma = \tau \cdot p^k q$ for some (possibly empty) word $\tau \in \mathcal{L}(\mathcal{A}^*)^*$ and symbol $q \in \mathcal{L}(\mathcal{A}^*)$. k is called the *copy factor* of p.
    Starting from the left of σ, look for subwords contained in the set

$$\{\pi : \pi = p^j s \text{ for some } s \neq p \text{ and } j \leq k\} \cup p^{k+1}.$$

    When such a subword π is encountered, replace it with the singleton $\mathsf{join}\,\pi$. Continue this replacement procedure over the remainder of σ. The result of these replacements is σ'.
    return $T(\sigma') + \log_2(k+1)$.
  end if
end function

Definition 2.2. The *T-decomposition* of a string is the finite sequence $(p_1, k_1), (p_2, k_2), \ldots$ of penultimate symbols with respective copy factors generated by successive iterations of Algorithm 1.

Since any string can be reconstructed from its T-decomposition plus its final symbol, T-decomposition can be viewed as a compression algorithm.

Example 2.3. (A sample T-decomposition)
We compute the T-complexity of the string $\sigma = 0100000100011$. In the computation below we use periods to separate distinct symbols in $\mathcal{L}(\mathcal{A}^*)$. Thus removing a period mark from between two adjacent symbols replaces these symbols with their join. The input to the function T in this case is:

$$\sigma = 0.1.0.0.0.0.0.1.0.0.0.1.1,$$

and the respective penultimate symbol and copy factor are $p_1 =$ '1' and $k_1 = 1$. Thus

$$\sigma' = 0.10.0.0.0.0.10.0.0.11.$$

In the next iteration, we get the penultimate symbol $p_2 =$ '0' and copy factor $k_2 = 2$, and

$$\sigma'' = 010.000.010.0011,$$

Next $p_3 =$ '010' and $k_3 = 1$, so

$$\sigma''' = 010000.0100011,$$

and finally $p_4 = 010000$ and $k_4 = 1$. Hence the corresponding T-decomposition is

$$\langle(\text{'1'}, 1), (\text{'0'}, 2), (\text{'010'}, 1), (\text{'010000'}, 1)\rangle\,.$$

and

$$T(\sigma) = \sum_{i=1}^{k} \log_2 k_i = \log_2 2 + \log_2 3 + \log_2 2 + \log_2 2 = 3 + \log_2 3.$$

3. The size of T-parsings

The T-complexity of a string may be viewed as the number of bits required to represent the string in a stagewise prefix-free "T-code" as defined by Titchener [17, 15]. Titchener, Nicolescu, Staiger, Gulliver, and Speidel defined generalized T-codes in terms of a finite set of words [18].

We will be interested in the union of T-codes over all stages of the T-complexity algorithm. We call this union a "T-parsing." The T-complexity algorithm always uses penultimate symbols to parse input strings, but in general one could parse with any sequence of symbols. We include an order structure on every T-parsing in order to keep track of the order of parsing symbols.

In the next definition, the set S_p^k is what is usually known as a *T-code*.

Definition 3.1. [18] Let $\mathcal{A}$ be a finite alphabet. If $S \subseteq \mathcal{L}(\mathcal{A}^*)$, and $p \in S$, the *T-augmentation of S by p with copy factor mapping $k : \mathcal{L}(\mathcal{A}^*) \to \{1, 2, 3, \dots\}$* is

$$S_p^k = \bigcup_{i=1}^{k(p)} \mathsf{join}\left[p^i \cdot (S \setminus \{p\})\right] \cup \left\{\mathsf{join}\ p^{k(p)+1}\right\}$$

where $\mathsf{join}(p^i \cdot (S \setminus \{p\}))$ is the set consisting of the joins of p^i with each element of $S \setminus \{p\}$. If $(p_1, k(p_1)), (p_2, k(p_2)), \dots$ is the T-decomposition of some string σ, then we say k is a copy factor mapping *induced* by the T-decomposition of σ.

Definition 3.2. Let $<$ be an well-ordering on $\mathcal{L}(\mathcal{A}^*)$ and $k : \mathcal{L}(\mathcal{A}^*) \to \mathbb{N}$. The *T-parsing generated by $<$ and k over $\mathcal{A}$* is the union

$$(<, k)_{\mathcal{A}} = \bigcup_{i=0}^{\infty} S_i$$

where $S_0 = \{\mathcal{L}(a) : a \in \mathcal{A}\}$, p_i is the i^{th} symbol of the ordering, and

$$S_{i+1} = (S_i)_{p_i}^{k(p_i)}.$$

An example of a natural ordering on a T-parsing would be $p_1 < p_2 < p_3 < \cdots$ where the p_i's are the penultimate symbols occurring in the T-decomposition of some given string and $k(p_i)$ is the copy factor of p_i in this T-decomposition, and the remaining non-penultimate symbols are assigned arbitrarily at the higher positions of the ordering. In this way, the T-decomposition of every string *induces* a T-parsing. The purpose of our definition is to separate the string being parsed from the list of symbols used to parse it; in a T-decomposition the latter is always the list of penultimate strings.

Definition 3.3. If $G = (<, k)_{\mathcal{A}}$ is a T-parsing and $x \in G$, we say that x is *periodic* if there exists $y \in G$ with $x = \mathsf{join}\left[\mathcal{L}^{-1}(y)^{k(y)+1}\right]$. If x is not periodic, it is *aperiodic*.

A string τ is said to be a *cyclic permutation* of a string σ if $|\sigma| = |\tau| = n$ and there exist an index $0 \leq k < n$ such that for all $i < n$, $\sigma(i + k \mod n) = \tau(i)$. The lexicographically least cyclic permutation of any finite string (or symbol) is called a *necklace*, and a necklace is called *aperiodic* if no two distinct cyclic permutations of it are equal. Gulliver, Makwakwa, and Speidel [6] demonstrated a correspondence between necklaces and T-codes. We use their result to count the number of aperiodic symbols of length n in a T-parsing (Proposition 3.5), and we give an unrestricted count in Theorem 3.7 below.

Definition 3.4. Let

$$e_a(n) = \frac{1}{n} \sum_{d|n} \mu(d) a^{n/d}, \tag{3.1}$$

where μ is the Möbius function [14].

It is possible to describe the size of a T-parsing in terms of $e_a(n)$, which for a prime happens to be the number of aperiodic symbols of length n over a T-parsing with alphabet size a (Proposition 3.5) as well as the number of irreducible polynomials of degree n over a finite field of size a [3], [8, p. 84].

Proposition 3.5. Let $G = (<, k)_{\mathcal{A}}$ be a T-parsing. The number of aperiodic symbols in G of length n is $e_{|\mathcal{A}|}(n)$.

Proof:
First we argue that the number of aperiodic necklaces of length n does not exceed the number of aperiodic symbols of length n in G. Let $\bigcup_{i=1}^{k} S_i$ be the stage k approximation of G, and let k_0 be sufficiently large so that all elements with length less than n occur within the first $k_0 - 1$ indices of G's ordering. From [6, Theorem 5.1, Case 3], we have that some cyclic permutation of each length n aperiodic necklace belongs to $\bigcup_{i=1}^{k_0} S_i$ and hence to G. Any cyclic permutation of any aperiodic necklace in G is also aperiodic in the T-parsing sense, and therefore the number of aperiodic necklaces of length n is at most the number of aperiodic symbols in G of length n.

[6, Theorem 4.1] states that for any i and n, no two length n members of S_i are cyclic permutations of each other. It follows that no two members of length n in G are cyclic permutations of each other because, for any sufficiently large index k, S_k contains all the length n members of G. In particular, each aperiodic element is a cyclic permutation of a distinct necklace. It follows that the number of aperiodic symbols of length n in G is at most the the number of necklaces of length n. Thus there is a one-one correspondence between the aperiodic necklaces of length n and the aperiodic elements of size n in G.

Moreau [13] showed that the number of aperiodic necklaces of length n over an alphabet of size a equals the quantity expressed in (3.1), whence the proposition follows. □

Remark 3.6. It follows from Proposition 3.5 that we may estimate the number of aperiodic strings of length n in a T-parsing over an alphabet of fixed size a as follows:

$$e_a(n) = \frac{a^n}{n} + \frac{1}{n} \cdot \sum_{\substack{d|n \\ d>1}} \left(\mu(d) \cdot a^{n/d} \right) = \Theta\left(\frac{a^n}{n} \right).$$

The hidden constant does not depend on a.

Using Proposition 3.5, we can show that the number of symbols in a T-parsing of a given length is independent of its underlying ordering.

Theorem 3.7. For any T-parsing $G = (<, k)_{\mathcal{A}}$, the number of symbols of G of length n is

$$e_{|\mathcal{A}|}(n) + \left| \left\{ x : k(x) = \frac{n}{|x|} - 1 \text{ and } |z| < |x| \implies x^{k(x)+1} \neq z^{k(z)+1} \right\} \right|.$$

Thus the number of symbols of a given length in a T-parsing depends only on the alphabet $\mathcal{A}$ and function k and not the underlying ordering $<$.

Proof:
For a given n, the number of aperiodic symbols of length n in G is $e_{|\mathcal{A}|}(n)$ (Proposition 3.5), so it remains to count the periodic symbols. Let C_n denote the set on the right hand side of (3.7). First we show that the number of aperiodic symbols of length n is no greater than $|C_n|$. If $y \in G$ is a periodic symbol of length n, then by definition, $y = x^{k(x)+1}$ for some shortest symbol x. Now $k(x) = |y|/|x| - 1$, and moreover every periodic symbol of length n corresponds to a distinct such $x \in C_n$. Hence $|C_n|$ is at least as large as the set of periodic symbols of length n.

For the reverse injection, suppose $x \in C_n$, and let $y = x^{k(x)+1}$. Then no string z of length at most $|x|$ satisfies $z^{k(z)+1} = y$ except for x itself. Thus each $x \in C_n$ corresponds to a distinct periodic symbol, namely $x^{k(x)+1}$, and we conclude that the number of periodic symbols of length n is at most $|C_n|$. □

4. Bounds on copy factors for simple strings

Strings without periodic symbols in their T-decomposition will play an important role in our quest for maximal T-complexity, as will strings with penultimate symbols of nondecreasing length. We capture these classes with the following notions.

Definition 4.1. A string σ is called *periodic-free* if all the symbols in its T-decomposition are aperiodic. A string σ is called *length-ordered* if whenever p_i, p_j are penultimate symbols in the T-decomposition of σ with $i \leq j$, then $|p_i| \leq |p_j|$. A string σ is called *simple* if it is periodic-free and length-ordered.

The goal of this section is to obtain bounds on copy factors for simple strings σ whose T-complexity is maximal among all strings of length $|\sigma|$ (Theorem 4.4). We must also demonstrate that such strings exist (Theorem 4.7). These bounds will allow us to efficiently compute maximal T-complexity for each string length, both exactly (Theorem 5.2) and asymptotically (Theorem 6.3).

Lemma 4.2. Let σ be a string with maximal T-complexity among all strings of length $|\sigma|$, and assume that σ is simple. Let m be the last penultimate symbol in σ's T-decomposition (so that m is at least as long as any other symbol in σ's T-decomposition), and suppose that x is a symbol in the T-parsing induced by σ's T-decomposition with $|x| < |m|$. Then x is a penultimate symbol in the T-decomposition of σ.

Proof:
Suppose that x does not occur as a penultimate symbol in the T-decomposition of σ. It suffices to assume

that x is among the shortest symbols meeting the assumptions of this argument because we can contradict existence over all other lengths by induction. Replace σ's leftmost prefix $\mathcal{L}^{-1}(m^r)$ (i.e. m including copy factors) with a copy of $\mathcal{L}^{-1}(x^r)$. This replacement does not change the T-complexity of σ because the r copies of x cannot be joined prior to the final T-decomposition step as x is not a penultimate symbol, and by minimality x must already have found its way into σ's T-parsing by the stage in which it becomes a penultimate symbol.

On the other hand, this replacement shortens σ. Let q be the first penultimate symbol in σ's T-decomposition, and form a new string by adding $r(|m| - |x|)$ copies of $\mathcal{L}^{-1}(q)$ to the next-to-rightmost position of σ after the above replacement. Since σ was prefix-free, this new string has the same length and the same copy factors as σ except that the copy factor of q increased. So σ must not have had maximal T-complexity. □

Lemma 4.3. Suppose σ is has maximal T-complexity for its length and is simple, and let m be the length of the longest last penultimate symbol in the T-decomposition of σ. For any penultimate symbol p with copy factor k in the T-decomposition of σ,

$$k \leq \left\lceil \frac{2m+2}{|p|} \right\rceil - 1.$$

Proof:
Suppose to the contrary that $k > \left\lceil \frac{2m+2}{|p|} \right\rceil - 1$ for some penultimate symbol p with copy factor k. Form a new string $\tilde{\sigma}$ by removing $\left\lceil \frac{m+1}{|p|} \right\rceil$ copies of p from σ at the spot where p becomes the penultimate symbol in the T-decomposition of σ. Then the copy factor $\tilde{k}$ of p in $\tilde{\sigma}$ is

$$\tilde{k} = k - \left\lceil \frac{m+1}{|p|} \right\rceil > \frac{m}{|p|} - 1.$$

Since $\tilde{k} \cdot |p|$ exceeds the length of any penultimate symbol in σ, the copy factors and penultimate symbols for $\tilde{\sigma}$ are σ's except that k is replaced with $\tilde{k}$. Therefore

$$T(\tilde{\sigma}) = T(\sigma) - \log(k+1) + \log(\tilde{k}+1).$$

Furthermore, $\tilde{\sigma}$ is periodic-free and $|\sigma| - |\tilde{\sigma}| \geq m+1$.

By Proposition 3.5, there is at least one aperiodic symbol x of length $m+1$ in the T-parsing induced by the T-decomposition of $\tilde{\sigma}$. Let q be the first penultimate symbol in σ's T-decomposition, and form a new string τ by adding a copy of x to the beginning of $\tilde{\sigma}$ and padding the initial penultimate copies of q with $|\sigma| - |\tilde{\sigma}| - m - 1$ extra copies of q. Since $\tilde{\sigma}$ is periodic-free, the increased copy factor on q does not change the sequence of penultimate symbols in τ. Now $|\tau| = |\sigma|$, and

$$\begin{aligned} T(\tau) \geq T(\tilde{\sigma}) + 1 &= T(\sigma) + 1 - \log_2(k+1) + \log_2(\tilde{k}+1) \\ &= T(\sigma) + 1 + \log_2 \left(\frac{k - \left\lceil \frac{m+1}{|p|} \right\rceil + 1}{k+1} \right) \\ &> T(\sigma) + 1 + \log_2 \left(\frac{\left\lceil \frac{2m+2}{|p|} \right\rceil - 1 - \left\lceil \frac{m+1}{|p|} \right\rceil + 1}{\left\lceil \frac{2m+2}{|p|} \right\rceil - 1 + 1} \right) \geq T(\sigma) + 1 + \log_2 \left(\frac{1}{2} \right) = T(\sigma), \end{aligned}$$

contradicting that σ was maximal. □

Theorem 4.4. Let σ be a simple string with maximal T-complexity for its length, and let m be the length of the longest symbol in the T-decomposition of σ. Then for any copy factor k corresponding to a penultimate symbol p which occurs in the T-decomposition of σ, we have

$$\max\left\{\left\lceil\frac{m-|p|}{|p|}\right\rceil, 1\right\} \leq k \leq \left\lceil\frac{2m+2}{|p|}\right\rceil - 1.$$

Proof:
Note that p^{k+1} is a symbol of length $|p|\cdot(k+1)$ in the T-parsing induced by the T-decomposition of σ. By Lemma 4.2, if $m > |p|\cdot(k+1)$, then p^{k+1} must appear as a penultimate symbol in σ's T-decomposition. But σ is periodic-free, so $\frac{m-|p|}{|p|} \leq k$; since k is an integer, $\left\lceil\frac{m-|p|}{|p|}\right\rceil \leq k$.

Applying Lemma 4.3 gives $k \leq \left\lceil\frac{2m+2}{|p|}\right\rceil - 1$ directly. □

It remains to verify that simple strings can actually have maximal T-complexity. To this end, we introduce a "symbol-switching" lemma (Lemma 4.6) which allows us to permute the order of adjacent symbols without changing the length or T-complexity of a given string.

Lemma 4.5. Let $\sigma \in \mathcal{A}^*$ be a string with T-decomposition $(p_1, k_1), (p_2, k_2), \ldots$. Let $S_0 = \mathcal{A}$ and let $S_{i+1} = (S_i)^k_{p_i}$ where k is a copy factor mapping induced by σ's T-decomposition, as in Definition 3.2. Given $i < m$ such that $p_{i+1} \neq \mathsf{join}\,(p_i^r s)$ for any $r > 0$ and $s \in S_{i-1}$, let

$$S_{i+1}^{i \leftrightarrow i-1} = \left((S_{i-1})^k_{p_i}\right)^k_{p_{i-1}},$$

Then there is a length-preserving bijection $f : S_{i+1} \to S_{i+1}^{i \leftrightarrow i-1}$ such that f restricted to S_{i-1} is the identity.

Proof:
Since $p_{i+1} \neq \mathsf{join}\,(p_i^r s)$ for any $r > 0$ and $s \in S_{i-1}$, the T-augmentation $S_{i+1}^{i \leftrightarrow i-1}$ is well-defined. Given $x \in S_{i+1}$, we may find $0 \leq c_i \leq k(p_i)$ and $0 \leq c_{i+1} \leq k(p_{i+1})$ such that $x = \mathsf{join}(p_{i+1}^{c_{i+1}} \cdot p_i^{c_i} \cdot t)$ for some $t \in S_{i-1}$. We set

$$f(x) = \mathsf{join}\left(p_i^{c_i} \cdot p_{i+1}^{c_{i+1}} \cdot t\right).$$

If $c_i = c_{i+1} = 0$, then $f(x) = x$, so f restricted to S_{i-1} is the identity. Since

$$|f(x)| = c_i|p_i| + c_{i+1}|p_{i+1}| + |t| = |x|,$$

f is length-preserving. Finally, f has an obvious inverse:

$$f^{-1}\left[\mathsf{join}\left(p_i^{c_i} \cdot p_{i+1}^{c_{i+1}} \cdot t\right)\right] = \mathsf{join}\left(p_{i+1}^{c_{i+1}} \cdot p_i^{c_i} \cdot t\right).$$ □

Lemma 4.6. Let σ be a string over the alphabet $\mathcal{A}$ whose T-decomposition is $(p_1, k_1)\ldots,(p_r, k_r)$, and let S_i denote the i^{th} iteration of the T-augmentation for this decomposition, as in the notation of Lemma 4.5. Let p_i be a symbol such that $p_i \neq \mathsf{join}\left(p_{i-1}^d s\right)$ for any $0 < d \leq k_{i-1}$ and $s \in S_{i-2}$. Then there is a string $\tau \in \mathcal{A}^*$ with T-decomposition $(p'_1, k'_1), \ldots, (p'_n, k'_n)$ and $n \leq r$ such that:

(I) $|\tau| = |\sigma|$ and $T(\tau) = T(\sigma)$,

(II) $(p'_j, k'_j) = (p_j, k_j)$ for $j < i-2$,

(III) $\left|p'_{j+n-r}\right| = |p_j|$ and $k'_{j+n-r} = k_j$ for $j > i+1$, and

(IV) either p_i is not a penultimate symbol in the T-decomposition of τ or p_i occurs before p_{i-1} in the T-decomposition of τ.

Proof:
Let $q_1 \cdots q_R \cdot \mathsf{join}(p_i^{k_i} p_{i-1}^{k_{i-1}} x)$ be the word resulting after the $(i+1)^{\text{st}}$ stage of the T-complexity algorithm starting from σ. Here $x \in S_{i-1}$ and $q_1, \ldots, q_R \in S_{i+1}$. By Lemma 4.5, there is a length-preserving bijection $f : S_{i+1} \to S_{i+1}^{i \leftrightarrow i-1}$, so define

$$\tau = \mathcal{L}^{-1}\left(\mathsf{join}\left[f(q_1)\cdots f(q_R)\cdot p_{i-1}^{k_{i-1}} p_i^{k_i} x\right]\right)$$

to be the word with f applied to each q_j and the order of $p_i^{k_i}$ and $p_{i-1}^{k_{i-1}}$ reversed. We verify the desired properties of τ by cases.

Case 1: $p_{i+1} \neq \mathsf{join}\, p_{i-1}^{k_{i-1}+1}$.

In this case, $q_j \neq \mathsf{join}\, p_{i-1}^d$ and $f(q_j) \neq \mathsf{join}\, p_{i-1}^d$ for any j or d, and the copy factor of p_{i-1} in τ equals the copy factor of p_{i-1} in σ. Let $p_i = \mathsf{join}\, p_{i-2}^t u$ for some $0 \leq t \leq k_{i-2}$ and some $u \in S_{i-3}$. There are two mutually exclusive cases to consider.

Case (a): $u \neq p_{i-2}$.

In this case p_i is aperiodic, and the copy factor of p_{i-2} in τ is k_{i-2}. τ has the same number of symbols as σ, and $|p_j| = \left|p'_j\right|$ and $k_j = k'_j$ for all $1 \leq j \leq r$ except for $j \in \{i-1, i\}$.

Case (b): $p_i = \mathsf{join}\, p_{i-2}^{k_{i-2}+1}$.

Now p_{i-2} cannot be periodic in p_{i-3}, since that would violate the definition of copy factor. So τ_{i-3} (the string resulting from τ after copies of p_{i-3} have been merged) now has penultimate symbol p_{i-2} with copy factor $k_i(k_{i-2}+1)+k_{i-2}$. Consider the leading symbols $f(q_1), \ldots, f(q_R), p_{i-1}$ of τ_{i-3}. If some subword of $f(q_1), \ldots, f(q_R)$ is a repeated sequence of the symbol $p_i^{t_i} p_{i-2}^{t_{i-2}}$ with $0 \leq t_j \leq k_j$, then in τ_{i-2} that sequence is replaced by $p_{i-2}^{(k_{i-2}+1)t_i + t_{i-2}}$: since f is the identity on symbols p_j for $j < i-1$, the initial $i-3$ steps of the T-decomposition of τ are the same as the first $i-3$ steps of the T-decomposition of σ. Note that $k_{i-2}t_i + t_i + t_{i-2} \leq k_{i-2}k_i + k_i + k_{i-2}$, and therefore the former two T-complexity algorithm steps of merging p_{i-2} and p_i are now combined into a single step which preserves the merging of the repeated sequence of p_i and/or p_{i-2}. Thus while one step has been removed from the T-complexity algorithm in moving from σ to τ, all other penultimate symbol lengths and copy factors remain the same. The copy factor of p_{i-2} in τ is $k_i k_{i-2} + k_i + k_{i-2}$, so its new contribution to the T-complexity of τ is

$$\begin{aligned}\log_2(k_i k_{i-2} + k_i + k_{i-2} + 1) &= \log_2[(k_i+1)(k_{i-2}+1)] \\ &= \log_2(k_i+1) + \log_2(k_{i-2}+1),\end{aligned}$$

which is exactly the sum contribution of p_{i-2} and p_i to the T-complexity of σ.

In short: when p_i is not periodic in p_{i-2}, there is no change in copy factors when moving from σ to τ, so T-complexity is preserved. When p_i is periodic in p_{i-2}, the string τ no longer contains p_i as a penultimate symbol, but the copy factor of p_{i-2} ends up increasing such that T-complexity is preserved.

Case 2: $p_{i+1} = \mathsf{join}\, p_{i-1}^{k_{i-1}+1}$.

By an analogous argument p_{i+1} is removed as a penultimate symbol, but the copy factor of p_{i-1} ends up increasing such that T-complexity is preserved. By assumption, neither p_i nor p_{i-1} is periodic in the other, so we may treat the periodic case in each symbol separately.

Thus $|\tau| = |\sigma|$ and $T(\tau) = T(\sigma)$. In Case 1(a), the penultimate symbols in τ actually have the same length and copy factors as the penultimate symbols in σ, but in any case T-complexity and length are preserved. □

Theorem 4.7. For any string σ, there is a simple string τ such that $|\sigma| = |\tau|$ and $T(\sigma) = T(\tau)$.

Proof:
Let σ be as in the hypothesis, with T-decomposition $(p_1, k_1), \ldots, (p_r, k_r)$. Suppose that $p_i = \mathsf{join}\, p_j^{k_j+1}$. Then $i \neq j+1$, so i satisfies the conditions of Lemma 4.6. The result of Lemma 4.6 is a new string $\tilde{\sigma}$ with the same length and T-complexity such that either p_i is not a symbol in the T-decomposition of $\tilde{\sigma}$, or p_i occurs earlier in the T-decomposition of $\tilde{\sigma}$ than in σ. In the latter case, we may apply Lemma 4.6 again, repeating until p_i is no longer a symbol in the T-decomposition of the result. Iterating this process, we obtain a periodic-free string τ with $|\sigma| = |\tau|$ and $T(\sigma) = T(\tau)$ because Lemma 4.6 cannot introduce new periodic symbols.

Suppose that $|p_i| < |p_{i-1}|$. Then i satisfies the conditions of Lemma 4.6. Set τ as the result. Repeat until τ is length-ordered. Thus the final string is both periodic-free and length-ordered. □

Finally, our estimates in the next two sections require a bound on the length of the longest penultimate symbol, which we give in Theorem 4.9.

Lemma 4.8. Fix $a > 1$. Then for $m \geq 1$,

$$\sum_{i=1}^{m} \frac{a^i}{i} = \Theta\left(\frac{a^m}{m}\right).$$

The hidden constant does not depend on a.

Proof:
By splitting the sum and approximating one of the halves by partial sums of a geometric series, we obtain:

$$\sum_{i=1}^{m} \frac{a^i}{i} = \sum_{i=1}^{m/2} \frac{a^i}{i} + \sum_{i=m/2+1}^{m} \frac{a^i}{i} \leq \frac{m}{2} \cdot a^{m/2} + \frac{2}{m} \cdot \sum_{m/2+1}^{m} a^i = \mathcal{O}\left(\frac{a^m}{m}\right).$$

Since a^m/m is obviously a lower bound for the sum in question, we obtain the desired bound. □

Theorem 4.9. Let σ be a string with maximal T-complexity for its length over an alphabet of fixed size a, and assume that σ is simple. Then the longest penultimate symbol in σ's T-decomposition has length $m = \Theta(\log|\sigma|/\log a)$. Moreover, $|\sigma| = \Theta(a^m)$. The hidden constants do not depend on a.

Proof:
The length of σ is the number of penultimate symbols of each length in the T-decomposition times the length times the copy factor for the length, plus one for the leftmost symbol. By Proposition 3.5, Theorem 4.4, Remark 3.6 and Lemma 4.8, we obtain the following sequence of inequalities.

$$|\sigma| \leq 1 + \sum_{i=1}^{m} \frac{2m+2}{i} \cdot i \cdot e_a(i) = \mathcal{O}\left(m \sum_{i=1}^{m} \frac{a^i}{i}\right) = \mathcal{O}\left(a^m\right).$$

An equivalent lower bound follows by noting that each string of length m must have copy factor at least 1, and so

$$|\sigma| \geq 1 \cdot m \cdot e_a(m) \geq m \cdot \Omega\left(\frac{a^m}{m}\right) = \Omega\left(a^m\right).$$

Hence $|\sigma| = \Theta\left(a^m\right)$. □

5. Finding the T-complexity of maximal strings

Before we show how to quickly compute maximal T-complexity among strings of a given length, we introduce an additional copy factor bound which will greatly reduce the search space of the algorithm in Theorem 5.2.

Lemma 5.1. There is a simple string $\sigma \in \mathcal{A}^*$ with maximal T-complexity among all strings of length $|\sigma|$, such that if p_i, p_j are penultimate symbols in the T-decomposition of σ with $|p_i| = |p_j|$, then $|k(p_i) - k(p_j)| \leq 1$.

Proof:
Let σ be a simple string with maximal T-complexity for its length and T-decomposition $(p_1, k(p_1)), \ldots, (p_r, k(p_r))$. Suppose $|k(p_i) - k(p_j)| > 1$, and using Lemma 4.6 we may assume without loss of generality that $k(p_j) > k(p_i)$ and $j = i + 1$. We shall construct a string τ with the same length as σ which has greater T-complexity than σ.

Let $S_{i+2} = ((S_i)_{p_i}^{k})_{p_{i+1}}^{k}$ as in Definition 3.2, where k is a copy factor mapping induced by σ's T-decomposition. Let k' be the same mapping as k, except with $k'(p_i) = k(p_i) + 1$ and $k'(p_{i+1}) = k(p_i) - 1$, and let

$$S'_{i+2} = \left((S_i)_{p_i}^{k'}\right)_{p_{i+1}}^{k'}$$

so that the repetitions for p_i increases by one and the repetitions for p_{i+1} decreases by one compared to S_{i+2}. Suppose that $\sigma = \mathcal{L}^{-1}(\mathsf{join}\, q_1 \cdots q_R t)$ for some $q_1, \ldots, q_R, t \in S_{i+2}$, so that $\mathsf{join}\, q_1 \cdots q_R t$ is the result of the $(i+2)^{\text{th}}$ step of the T-decomposition of σ. Since each q_j must be a substring of some penultimate symbol during the T-decomposition of σ, and since σ is simple, we have $|q_j| \leq |p_r|$.

Given q_j, we may write $q_j = \mathsf{join}\left(p_{i+1}^{c_{i+1}} \cdot p_i^{c_i} \cdot s\right)$ for some $s \in S_i$ and $c_i, c_{i+1} \geq 0$. By Lemma 4.2, if x is a symbol with $|x| < |p_{i+1}|$, then $x = p_h$ for some $h \leq i + 1$. All occurrences of p_h were merged

with other symbols during the h^{th} step of the T-decomposition of σ, so either $s = p_{i+1}$ or $|s| > |p_i|$. By Theorem 4.4, $(k(p_i)+1)|p_i| \geq |p_r|$. Then

$$\begin{aligned}[k(p_{i+1})+1] \cdot |p_{i+1}| &> (k(p_i)+1)|p_i| \geq |p_r| \geq |q_j| = \left|\mathsf{join}\, p_{i+1}^{c_{i+1}} \cdot p_i^{c_i} \cdot s\right| \\ &= c_{i+1}|p_{i+1}| + c_i|p_i| + s = (c_{i+1}+c_i)|p_{i+1}| + s \geq (c_{i+1}+c_i+1) \cdot |p_{i+1}|.\end{aligned}$$

Dividing through by $|p_{i+1}|$ gives $k(p_{i+1}) > c_{i+1} + c_i$, and since $c_i \geq 0$, we have $k(p_{i+1}) > c_{i+1}$. Thus $q_j \in S'_{i+2}$, so modifying the copy factors does not affect the later T-decomposition steps.

To simplify notation in the last part of this argument, we let $k_i = k(p_i)$ and $k_{i-1} = k(p_{i-1})$. Set

$$\tau = \mathsf{join}\left(q_1 \cdots q_R \cdot p_{i+1}^{k_{i+1}-1} p_i^{k_i+1} x\right).$$

Note that by assumption $k_{i+1} > k_i$, so $k_{i+1}|p_{i+1}| \geq |p_r|$. Therefore τ is periodic-free and hence simple. Now the T-complexity of τ is

$$T(\tau) = T(\sigma) - \log(k_i+1) - \log(k_{i+1}+1) + \log(k_i+2) + \log(k_{i+1}),$$

so it suffices to show that $(k_i+2)k_{i+1} > (k_i+1)(k_{i+1}+1)$. We assumed that $k_{i+1} - k_i > 1$, so adding $k_i k_{i+1} + k_i + k_{i+1}$ to both sides gives the desired inequality. □

In order to find a string with maximum T-complexity over strings of a given length, it suffices to search over simple strings (Theorem 4.7) whose copy factors satisfy the bounds in Theorem 4.4, and where all penultimate symbols of fixed length in the T-decomposition differ by at most 1 (Lemma 5.1). The next theorem analyzes the computational complexity of calculating maximal T-complexity according to these constraints.

Theorem 5.2. The maximal T-complexity among all strings of length n over an alphabet of size a may be found in $\mathcal{O}\left[\left(\frac{n^{\mathcal{O}(1)}}{\Omega[\log_a(n)]}\right)^{\mathcal{O}[\log_a(n)]}\right]$ time. Moreover, the algorithm finds a list consisting of pairs of the form ⟨penultimate symbol length, copy factor⟩ which agree with the T-decomposition of a string which has maximal T-complexity for its length. The hidden constants in the running time do not depend on a.

Proof:
Algorithm 2 below implicitly computes the copy factors and the respective penultimate symbol lengths in the T-decomposition of some maximal string σ, however there is no attempt to compute the symbols themselves. We would like to search for pairs of the form ⟨penultimate symbol length, copy factor⟩ which agree with the T-decomposition of a string which has maximal T-complexity for its length, but for purposes of efficiency, we want to minimize the search space.

We claim that for any n, there is a string σ with maximal T-complexity among strings of length n and longest penultimate symbol of some length m such that:

(i) every penultimate symbol p in σ's T-decomposition has a copy factor between $\max\{\frac{m-|p|}{|p|}, 1\}$ and $\frac{2m+2}{|p|}$,

(ii) for all $j < m$, there are exactly $e_a(j)$ symbols of length j in σ's T-decomposition, and

(iii) the copy factor for each penultimate symbol of length j is either $k(j)$ or $k(j)+1$ for some function k.

By Theorem 4.7, we may assume we are searching for a simple string σ. Then (i) follows immediately from Theorem 4.4. By Lemma 4.2, all aperiodic symbols of length $j < m$ must appear as penultimate symbols in the T-decomposition of σ. So (ii) follows by Proposition 3.5. Finally, (iii) follows from Lemma 5.1.

We do not know *a priori* what the function k in (iii) should be, but we do know that it conforms to the bounds in (i). We also don't know how many of the $e_a(j)$ symbols of length j have copy factor $k(j)$ and how many have copy factor $k(j)+1$, so let $d(j)$ guess the number of penultimate symbols of length j which have the greater copy factor of $k(j)+1$. Given m, it suffices to test all possible k satisfying

$$\frac{m-j}{j} \leq k(j) \leq \frac{2m+2}{j}$$

and $0 \leq d(j) < e_a(j)$ and then find the maximal T-complexity among these choices. That is, to find maximal T-complexity over strings of length n, we need only compute the maximal choice of k and d over all possible m.

We now examine the time complexity of this search algorithm. For each j, there are

$$\frac{2m+2}{j} - \frac{m-j}{j} = \frac{m+2+j}{j}$$

possible values of $k(j)$, each of which has $e_a(j)$ possible values of $d(j)$. So by Remark 3.6, each $j \leq m$ contributes

$$e_a(j) \cdot \left(\frac{m+j+2}{j}\right) = \Theta\left(\frac{a^j}{j} \cdot \frac{m+j+2}{j}\right) = \mathcal{O}\left(\frac{ma^j}{j^2}\right)$$

possibilities. Thus for each m, the number of values for k and d that have to be checked is:

$$\mathcal{O}\left(\prod_{j=1}^{m} \frac{ma^j}{j^2}\right) \quad = \quad \mathcal{O}\left(\frac{m^m a^{m^2}}{(m!)^2}\right) \quad = \quad \mathcal{O}\left[\frac{m^m a^{m^2}}{\left(\sqrt{2\pi m} \cdot \frac{m^m}{e^m}\right)^2}\right] \quad = \quad \mathcal{O}\left(\frac{a^{m^2}}{m^m}\right),$$

where the penultimate approximation follows from Stirling's approximation [12]. Theorem 4.9 states that $m = \Theta(\log n / \log a)$, so the total number of values for m that must be tested is, for some constants $c, d > 0$, at most

$$\mathcal{O}\left(m \cdot \frac{a^{m^2}}{m^m}\right) = \mathcal{O}\left(\frac{a^{[c\log_a(n)]^2}}{d\log_a(n)^{d\log_a(n)}}\right) = \mathcal{O}\left(\frac{n^{c^2/d}}{d\log_a(n)}\right)^{d\log_a(n)}.$$

This proves the theorem, as each test runs in polynomial-time. □

6. The asymptotic T-complexity of maximal strings

We now estimate the maximal T-complexity for strings of length n. We begin with the tangential but intuitive observation that the T-complexity of strings strictly increases with length.

Algorithm 2 Finding a maximal string over an alphabet of fixed size a.

Input: An integer n.

Output: The maximal T-complexity among all strings of length n.

1: **let** $t = 0$.
2: **for** $m \in \{1, \ldots, \Theta(\log n / \log a)\}$ **do**
3: **for** $k_j \in \{(m-j)/j, \ldots, (2m+2)/j\} \quad (j = 1, \ldots, m)$ **do**
4: **for** $d_j \in \{0, 1, \ldots, e_a(j)\} \quad (j = 1, \ldots, m)$ **do**
5: **if** d and k describe a string of length n, that is,

$$n = 1 + \sum_{j=1}^{m} j \cdot [d_j(k_j + 1) + (e_a(j) - d_j)k_j],$$

then
6: **if** The T-complexity of the corresponding string is the highest seen thus far, i.e.

$$t < \sum_{j=1}^{m} [d_j \log(k_j + 2) + (e_a(j) - d_j) \log(k_j + 1)]$$

then
7: set t to be equal to the value of the sum in line 6.
8: **end if**
9: **end if**
10: **end for**
11: **end for**
12: **end for**
13: **return** t

Proposition 6.1. If σ and τ have maximal T-complexity for their lengths, and $|\sigma| > |\tau|$, then $T(\sigma) > T(\tau)$.

Proof:
By Theorem 4.7, we may assume τ is simple. Obtain a new string $\tilde{\tau}$ by padding $|\sigma| - |\tau|$ copies of $\mathcal{L}^{-1}(p_1)$, where p_1 is the first penultimate symbol in σ's T-decomposition, to the penultimate position of τ. Since $\tilde{\tau}$ and τ are both periodic-free, their T-decompositions are the same except for the copy factor in the first step. In particular, $|\tilde{\tau}| = |\sigma|$ and $T(\tau) < T(\tilde{\tau}) \leq T(\sigma)$. □

We now give an explicit bound on the maximal T-complexity among strings of a given length based on the estimates in Theorem 4.4. Theorem 6.3 confirms conjectures made in [16] and [7]. The proof requires the following variant of Lemma 4.8.

Lemma 6.2. For $a > 1$ and $m \geq 1$,

$$\sum_{i=1}^{m} \frac{a^i}{i} \cdot \log\left(\frac{m}{i}\right) = \mathcal{O}\left(\frac{a^m}{m}\right).$$

Proof:

$$\begin{aligned}
\sum_{i=1}^{m} \frac{a^i}{i} \cdot \log\left(\frac{m}{i}\right) &= \sum_{i=1}^{m/2} \frac{a^i}{i} \cdot \log\left(\frac{m}{i}\right) + \sum_{i=m/2+1}^{m} \frac{a^i}{i} \cdot \log\left(\frac{m}{i}\right) \\
&\leq \frac{m}{2} \cdot a^{m/2} \cdot \log m + \frac{2}{m} \cdot \sum_{m/2+1}^{m} a^i \cdot 1 \\
&= \mathcal{O}\left(\frac{a^m}{m}\right).
\end{aligned}$$

□

Theorem 6.3. If $\tau \in \mathcal{A}^*$ has maximal T-complexity among strings of length $|\sigma|$, then any simple string σ of length $|\tau|$ with maximal T-complexity among strings of length $|\tau|$ satisfies

$$\sum_{i=1}^{m} e_a(i) \log_2\left(\max\left\{\frac{m-i}{i}, 1\right\} + 1\right) \leq T(\sigma) \leq \sum_{i=1}^{m} e_a(i) \log_2\left(\frac{2m+2}{i} + 1\right), \tag{6.1}$$

where m is the length of the longest penultimate symbols in the T-decomposition of σ and $a = |\mathcal{A}|$. Consequently,

$$T(\tau) = \Theta\left(\frac{|\tau|}{\log|\tau|}\right). \tag{6.2}$$

Proof:
By Lemma 5.1, there exists a simple string σ with length τ. Proposition 3.5 combined with Lemma 4.2 tell us that the number of penultimate symbols of length i in σ's T-decomposition is $e_a(i)$ for all $i < m$, and Theorem 4.4 tells us that the copy factor for an aperiodic string of length i must be between $\max\{\frac{m-i}{i}, 1\}$ and $\frac{2m+2}{i}$. Inequality (6.1) now follows from setting the copy factor of each symbol in σ's T-decomposition to either its minimum or maximum possible value.

Using (6.1), we next establish (6.2). By Lemma 6.2,

$$\begin{aligned}
T(\sigma) \leq \sum_{i=1}^{m} e_a(i) \log_2\left(\frac{2m+2}{i} + 1\right) &= \mathcal{O}\left[\sum_{i=1}^{m} \frac{a^i}{i} \log_2\left(\frac{2m+2}{i} + 1\right)\right] \\
&= \mathcal{O}\left[\sum_{i=1}^{m} \frac{a^i}{i} \cdot \log_2\left(\frac{m}{i}\right)\right] = \mathcal{O}\left(\frac{a^m}{m}\right).
\end{aligned} \tag{6.3}$$

The lower bound follows from a similar calculation:

$$T(\sigma) \geq \sum_{i=1}^{m} e_a(i) = \Omega\left(\sum_{i=1}^{m} \frac{a^i}{i}\right) = \Omega\left(\frac{a^m}{m}\right). \tag{6.4}$$

It remains to find estimates for $T(\tau)$ in terms of $|\tau|$. By Theorem 4.9, we have $|\sigma| = \Theta(a^m)$, and since $|\tau| = |\sigma|$, inequality (6.2) follows immediately from (6.3) and (6.4). □

7. T-complexity and Kolmogorov complexity

The *Kolmogorov complexity* of a string is the length of the shortest computer program, or *description*, which computes it. In order to make the notion of Kolmogorov complexity precise, we fix a standard programming language U whose description lengths exceed the description lengths in any other fixed programming language by no more than a constant [10, Theorem 2.1.1]. Then we may write the Kolmogorov complexity of a string σ as $C(\sigma) = \min\{|p|\colon U(p) = \sigma\}$. A string σ is called *random* if $C(x) \geq |x|$. For every n and string alphabet of size a, there exists a random string of length n because there are a^n strings of length n but only $a^{n-1}+a^{n-2}+\ldots+a+1 < a^n$ descriptions of shorter length. A similar counting argument shows that 99.9% of strings σ of length n satisfy $C(\sigma) \geq n - 10$. The name "random" is thus appropriate since most strings are close to being random and because, by definition, strings which are close to random lack simple descriptions. It is unknown what fraction of strings of a given length have high T-complexity.

Eimann [5] wrote that "[T-complexity] permit[s] *estimations* of the Kolmogorov-Chaitin complexity." (see also [4]). Zvonkin and Levin [23, Theorem 1.6] showed that no unbounded computable function, for example any scalar multiple of T-complexity, can be a lower bound for a Kolmogorov complexity. We now argue that T-complexity is also not an upper bound for Kolmogorov complexity.

Corollary 7.1. No scalar multiple of T-complexity is an upper bound for Kolmogorov complexity.

Proof:
Suppose $c > 0$ is a scalar, and let d be the constant guaranteed in the asymptotic T-complexity upper bound (6.2). Furthermore let $\sigma_0, \sigma_1, \sigma_2, \ldots$ be a sequence of random strings so that $C(\sigma_k) \geq |\sigma_k|$ for all k. For all of the infinitely many k satisfying $2cd < \log|\sigma_k|$, we have by Theorem 6.3,

$$c \cdot T(\sigma_k) \leq cd \cdot \frac{|\sigma_k|}{\log|\sigma_k|} + cd < |\sigma_k| \leq C(\sigma_k). \qquad \square$$

8. De Bruijn sequences

Lempel and Ziv used de Bruijn sequences, to benchmark their complexity algorithm against random strings [9]. They found de Bruijn sequences to have maximal complexity for their lengths, up to a logarithmic factor. With this result in mind, we examine how closely a de Bruijn sequence comes to having maximal T-complexity when compared to other strings of the same length.

Definition 8.1. (van Aardenne-Ehrenfest and de Bruijn [1])
If σ is a string over alphabet $\mathcal{A}$ with $|\mathcal{A}| = a$, let $\sigma[i, j]$ be the substring of the i^{th} through j^{th} symbols of σ, inclusive. An $(\mathcal{A}, k)$*-de Bruijn sequence* σ is a string of length $n = a^k + k - 1$ such that the a^k substrings $\sigma[i, i + k - 1]$ form the set of length k strings over $\mathcal{A}$, and $\sigma[1, k - 1] = \sigma[a^k + 1, n]$.

An $(\mathcal{A}, k)$-de Bruijn sequence contains each substring of length k exactly once and represents a maximally concise catalog of such strings. Thus de Bruijn sequences exhibit incompressibility properties

which are indicative of randomness. Even more compelling are the statistical properties of de Bruijn sequences. Every string of length at most k appears in an $(\mathcal{A}, k)$-de Bruijn sequence exactly the number of times one would expect to find it in a perfectly random string. Kolmogorov random strings exhibit this same property as well, but only in the limit [10, Theorem 2.6.1], [11]. It should be emphasized that de Bruijn sequences only exhibit the "random" statistical properties mentioned above over the course of the entire string; for large n it is quite possible to construct $(\{0,1\}, n)$-de Bruijn sequences which have mostly 0's at the beginning of the sequence and mostly 1's at the end (see Xie's Algorithm Ic from [20] which enumerates all de Bruijn sequences of a given length).

How complex are de Bruijn sequences under T-complexity? For this discussion, we fix $\mathcal{B} = \{0,1\}$. Since $(\mathcal{B}, n)$-de Bruijn sequences are within $\log n$ of maximal for both Lempel-Ziv [9] and the more recent finite-state complexity due to Calude, Salomaa, and Roblot [2], we find the following question interesting for comparison: Do $(\mathcal{B}, n)$-de Bruijn sequences achieve T-complexity within $\log n$ of maximal over strings of length $2^n + n - 1$? Table 1 indicates that this is not always the case.

	T-COMPLEXITY $(\mathcal{B}, n)$-DE BRUIJN	T-COMPLEXITY LENGTH $2^n + n - 1$
$n = 2$	3.00000000000 to 3.16992500144	2.23192809489 to 3.16992500144
$n = 3$	4.58496250072 to 5.16992500144	3.16992500144 to 5.32192809489
$n = 4$	6.58496250072 to 8.16992500144	4.24792751344 to 8.22881869050
$n = 5$	9.00000000000 to 12.3398500029	5.16992500144 to 12.3923174228
$n = 6$	12.5849625007 to 19.0000000000	6.10852445677 to 19.0000000000

Table 1. Range for T-complexity over binary de Bruijn sequences and all strings of length n.

The relationship between maximal T-complexity and de Bruijn sequences is subtle. As Table 1 shows, for $n = 2$ and $n = 6$, $(\mathcal{B}, n)$-de Bruijn sequences can achieve maximal T-complexity over strings of length n while for $n = 3, 4, 5$ this cannot happen. Hence the existence of $(\mathcal{B}, n)$-de Bruijn sequences of maximal T-complexity depends on the length n. The following $(B, 6)$-de Bruijn sequences witness the extreme values of T-complexity for their lengths:

$$11001110100101111110110111001000110001010100000011110000100110101100 1, \tag{8.1}$$

$$10010011101100001000000111111010001101111000101110011001010110101001 0. \tag{8.2}$$

(8.1) has T-complexity 12.5849625007, while (8.2) has T-complexity 19.0.

9. Conclusion

Using simple strings, we have shown how to both efficiently calculate (Theorem 5.2) and analytically estimate (Theorem 6.3) the maximum T-complexity among strings of a given length. Our analytical estimate suffices to both verify conjectures of Hamano/Yamamoto [7] and Speidel [16] regarding the T-complexity of maximal strings and provide evidence against conjectures [4, 5] that T-complexity estimates Kolmogorov complexity (Section 7). Lastly, our empirical evidence suggests T-complexity may be useful for compression on some de Bruijn sequences where Lempel and Ziv's algorithms fail to compress significantly (Section 8).

Acknowledgments. We thank Roman Garnett and Richard Stong for useful discussions.

References

[1] van Aardenne-Ehrenfest, T., de Bruijn, N. G.: Circuits and trees in oriented linear graphs, *Simon Stevin*, **28**, 1951, 203–217, ISSN 0037-5454.

[2] Calude, C. S., Salomaa, K., Roblot, T. K.: *Finite-State Complexity and Randomness*, Technical Report 374, University of Auckland CDMTCS Research Report Series, December 2009.

[3] Chebolu, S. K., Mináč, J.: Counting Irreducible Polynomials over Finite Fields Using the Inclusion-Exclusion Principle, *Mathematics Magazine*, **84**(5), 2011, 369–371.

[4] Eimann, R., Speidel, U., Brownlee, N., Yang, J.: Network Event Detection with T-Entropy, *CDMTCS Research Report Series*, **266**, 2005.

[5] Eimann, R. E. A.: *Network event detection with entropy measures*, Ph.D. Thesis, The University of Auckland, 2008.

[6] Gulliver, T. A., Makwakwa, I., Speidel, U.: On the Generation of Aperiodic and Periodic Necklaces via T-augmentation, *Fundamenta Informaticae*, **83**(1-2), 2008, 91–107, ISSN 0169-2968.

[7] Hamano, K., Yamamoto, H.: A differential equation method to derive the formulas of the T-complexity and the LZ-complexity, *Proceedings of the 2009 IEEE international conference on Symposium on Information Theory - Volume 1*, ISIT 2009, IEEE Press, Piscataway, NJ, USA, 2009, ISBN 978-1-4244-4312-3.

[8] Ireland, K., Rosen, M.: *A Classical Introduction to Modern Number Theory*, Number 84 in Graduate Texts in Mathematics, second edition, Springer-Verlag, New York, 1990, ISBN 0-387-97329X.

[9] Lempel, A., Ziv, J.: On the complexity of finite sequences, *IEEE Transactions on Information Theory*, **IT-22**(1), 1976, 75–81, ISSN 0018-9448.

[10] Li, M., Vitányi, P.: *An introduction to Kolmogorov complexity and its applications*, Texts in Computer Science, third edition, Springer, New York, 2008, ISBN 978-0-387-33998-6.

[11] Li, M., Vitányi, P. M. B.: Statistical properties of finite sequences with high Kolmogorov complexity, *Mathematical Systems Theory*, **27**(4), 1994, 365–376, ISSN 0025-5661.

[12] Matoušek, J., Nešetřil, J.: *Invitation to discrete mathematics*, Second edition, Oxford University Press, Oxford, 2009, ISBN 978-0-19-857042-4.

[13] Moreau, C.: Sur les permutations circulaires distinctes, *Nouvelles annalles de mathématiques*, **2**(11), 1872, 309–314.

[14] Niven, I., Zuckerman, H. S., Montgomery, H. L.: *An introduction to the theory of numbers*, Fifth edition, John Wiley & Sons Inc., New York, 1991, ISBN 0-471-62546-9.

[15] Speidel, U.: *T-Complexity and T-Information Theory–an Executive Summary, 2nd revised version*, Technical Report 286, University of Auckland CDMTCS Research Report Series, October 2006.

[16] Speidel, U.: On the bounds of the Titchener T-complexity, *Proceedings of Communication Systems, Networks and Digital Signal Processing 2008*, CNSDSP 2008, July 2008.

[17] Titchener, M. R.: Digital encoding by means of new T-codes to provide improved data synchronization and message integrity, July 1984, Technichal note.

[18] Titchener, M. R., Nicolescu, R., Staiger, L., Gulliver, A., Speidel, U.: Deterministic Complexity and Entropy, *Fundamenta Informaticae*, **64**(1-4), 2004, 443–461, ISSN 0169-2968.

[19] Welch, T. A.: A Technique for High-Performance Data Compression, *Computer*, **17**(6), june 1984, 8–19, ISSN 0018-9162.

[20] Xie, S. Q.: Notes on de Bruijn sequences, *Discrete Applied Mathematics*, **16**(2), 1987, 157–177, ISSN 0166-218X.

[21] Ziv, J., Lempel, A.: A universal algorithm for sequential data compression, *IEEE Transactions on Information Theory*, **23**(3), may 1977, 337–343, ISSN 0018-9448.

[22] Ziv, J., Lempel, A.: Compression of individual sequences via variable-rate coding, *IEEE Transactions on Information Theory*, **24**(5), september 1978, 530–536, ISSN 0018-9448.

[23] Zvonkin, A. K., Levin, L. A.: The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms., *Russian Mathematical Surveys*, **25**(6), 1970, 83–124.