A Probabilistic Analysis of the Reduction Ratio in the Suffix-Array IS-Algorithm

Cyril Nicaud

LIGM, Université Paris-Est & CNRS, 77454 Marne-la-Vallée Cedex 2, France.
cyril.nicaud@u-pem.fr

Abstract. We show that there are asymptotically γn LMS-factors in a random word of length n, for some explicit γ that depends on the model of randomness under consideration. Our results hold for uniform distributions, memoryless sources and Markovian sources. From this analysis, we give new insight on the typical behavior of the IS-algorithm [9], which is one of the most efficient algorithms available for computing the suffix array.

1 Introduction

The suffix array of a word, is a permutation of its suffixes that orders them for the lexicographic order. Since their introduction by Manber and Meyers [8, 7] in 1990, suffix arrays have been intensively studied in the literature. Nowadays, they are a fundamental, space efficient, alternative to suffix trees. They are used in many applications such as pattern matching, plagiarism detection, data compression, etc.

The first linear suffix array algorithms that do not use the suffix tree construction were proposed by Ko and Aluru [5], Kim et al. [4] and Kärkkäinen and Sanders [3] in 2003. Since then, a lot of variations or heuristics have been developed [12], motivated by the various practical uses of this fundamental data structure.

A few years ago, Ge Nong, Sen Zhang and Wai Hong Chan proposed such a linear suffix array algorithm [9], which is particularly efficient in practice. This algorithm, called the *IS-algorithm*, is a recursive algorithm, where the suffix array of a word u is deduced from the suffix array of a shorter word v. This shorter word is built using the LMS-factors of u: an LMS-position i in u is an integer such that the suffix of u that starts at position i is smaller, for the lexicographic order, than both the one that starts at position $i-1$ and the one that starts at position $i+1$; LMS-factors are the factors of u delimited by two consecutive LMS-positions. Once the suffix array of v is recursively calculated, the suffix array of u can be computed in linear time.

In this article we are interested in the typical reduction ratio $\frac{|v|}{|u|}$ obtained when making this recursive call. We propose a probabilistic analysis of the number of LMS-factors in a random word of length n, for classical models of random words: uniform distributions, memoryless sources and Markovian sources. We prove that the reduction ratio is concentrated around a constant γ, which can be explicitly computed from the parameters that describe the source.

In this extended abstract, we chose to focus on memoryless sources. After recalling the basics on words and suffix arrays in Section 2, we explain in Section 3 the steps that lead to our main statement (Theorem 2). In Section 4, we briefly explain how this result can be generalized to Markovian sources, and give the explicit formula for the typical reduction ratio under this model (Theorem 3). We conclude this article with some experiments, that are just intended to illustrate our theoretical results, and with a short discussion in Section 5.

2 Preliminaries

2.1 Definitions and notations

Let A be a non-empty totally ordered finite alphabet. For given $n \geq 0$, we denote by A^n the set of words of length n on A. Let A^* be the set of all words on A.

If $u \in A^n$ is a word of length $n \geq 1$, let u_0 be its first letter, let u_1 be its second letter, ... and let u_{n-1} be its last letter. The *reverse* of a word $u = u_0 \cdots u_{n-1}$ is the word $\overline{u} = u_{n-1} \cdots u_0$. For given i and j such that $0 \leq i \leq j \leq n-1$, let $u[i,j]$ be the factor of u that starts at position i and ends at position j: it is the unique word w of length $j-i+1$ such that there exists a word v of length i such that vw is a prefix of u. For given i such that $0 \leq i \leq n-1$, let $\texttt{suff}(u,i) = u[i, n-1]$ be the suffix of u that starts at position i.

Recall that the *suffix array* of a word u of length $n \geq 1$ is the unique permutation σ of $\{0, \ldots, n-1\}$ such that, for the lexicographic order, we have

$$\texttt{suff}(u, \sigma(0)) < \texttt{suff}(u, \sigma(1)) < \ldots < \texttt{suff}(u, \sigma(n-1)).$$

See [12] for a more detailed account on suffix arrays and their applications.

2.2 LMS-factors of a word

The first step of the IS-algorithm [9] consists in marking every position in $v = u\$$, where $\$ \notin A$ is an added letter that is smaller than every letter of A. The mark of each position in v is either the letter S or the letter L. A position $i \in \{0, \ldots, n-1\}$ is marked by an S or by an L when $\texttt{suff}(v,i) < \texttt{suff}(v, i+1)$ or $\texttt{suff}(v,i) > \texttt{suff}(v,i+1)$, respectively. We also say that the position is of type S or L. By convention, the last position n of v always is of type S.

A leftmost type S position in $v = u\$$ (*LMS-position* for short) is a position $i \in \{1, \ldots, n\}$ such that i is of type S and $i-1$ is of type L. Note that with this definition, the last position of v is always an LMS-position, for a non-empty u. An *LMS-factor* of v is a factor $v[i,j]$ where $i < j$ are both LMS-positions and such that there is no LMS-position between i and j. By convention, the factor $v[n,n] = \$$ is also an LMS-factor of v.

The following notations and definitions will be used throughout this article. They are reformulations of what we just defined.

Definition 1. *Let A be a finite totally ordered non-empty alphabet. The alphabet* $\mathrm{LS}(A)$ *is defined by* $\mathrm{LS}(A) = (A \times \{L, S\}) \cup \{(\$, S)\}$*. For simplification, elements of* $\mathrm{LS}(A)$ *are written αX instead of (α, X). A letter αS of* $\mathrm{LS}(A)$ *is said to be of* type S*, and a letter αL is said to be of* type L*.*

Definition 2. *Let $u \in A^n$ for some $n \geq 1$. The* LS-extension $\mathbf{Ext}(u)$ *of u, is the word $v \in \mathrm{LS}(A)^{n+1}$ that ends with the letter $\$S$ and such that for every $i \in \{0, \dots, n-1\}$, $v_i = u_i X_i$ with $X_i = S$ if and only if $u_i < u_{i+1}$ or ($u_i = u_{i+1}$ and $X_{i+1} = S$), with the convention that $u_n = \$$.*

Observe that from its definition, $\mathbf{Ext}(u)$ is exactly the word u with an added \$ at its end, and whose positions have been marked. Thus, an LMS-position in $v = \mathbf{Ext}(u)$ is a position $i \geq 1$ such that $v_i = \alpha S$ and $v_{i-1} = \beta L$, for some $\alpha, \beta \in A \cup \{\$\}$. We extend this definition to all words of $LS(A)^*$.

Definition 3. *For any $u \in \mathrm{LS}(A)^n$, an* LMS-position *of u is a position $i \in \{1, \dots, n-1\}$ such that u_i is of type S and u_{i-1} is of type L.*

Example. Consider the word $u = bacbcaab$ on $A = \{a, b, c\}$. We have:

letter	b	a	c	b	c	a	a	b	\$
type	L	S	L	S	L	S	S	L	S

$$\mathbf{Ext}(u) = bL\, \underline{aS}\, cL\, \underline{bS}\, cL\, \underline{aS}\, aS\, bL\, \underline{\$S},$$

where the LMS-positions have been underlined.

2.3 Brief overview of the IS-algorithm

The IS-algorithm [9] first computes the type of each position. This can be done in linear time, by scanning the word once from right to left. From this, the LMS-positions can be directly computed.

The LMS-factors are then numbered in increasing order using a radix sort (the types are kept and used for the lexicographic comparisons of these factors). This yields an array of numbers, the numbers associated with the LMS-factors, which is viewed as a word and whose suffix array σ' is recursively calculated.

The key observation is that once σ' is known, the suffixes of type L can be sorted by scanning the word once from left to right, then the suffixes of type S can be sorted by a scan from right to left. Therefore, the suffix array can be computed in linear time, once σ' is given.

For the running time analysis, if $T(n)$ is the worst case cost of the algorithm applied to a word of length n, then we have the inequality $T(n) \leq T(m) + \Theta(n)$, where m denote the number of LMS-factors. Since we always have $m \leq \frac{n}{2}$, the running time of the IS-algorithm is $\Theta(n)$. The quotient m/n is called the *reduction ratio* and it is the main focus of this article.

2.4 Distributions on words

The *uniform distribution* on a finite set E is the probability p defined for all $e \in E$ by $p(e) = \frac{1}{|E|}$. By a slight abuse of notation, we will speak of the *uniform* *distribution on* A^* to denote the sequence $(p_n)_{n\geq 0}$ of uniform distributions on A^n. For instance, if $A = \{a, b, c\}$, then each element of A^n has probability 3^{-n} under this distribution.

An element $u \in A^n$ taken uniformly at random can also be seen as built letter by letter, from left to right or from right to left, by choosing each letter uniformly and independently in A. This is a suitable way to consider random words, which can easily be generalized to more interesting distributions. Indeed, if p is a probability on A, one can extend p to A^n by generating each letter independently following the probability p. This is called a *memoryless distribution of probability* p, and the probability of an element $u = u_0 \cdots u_{n-1} \in A^n$ is defined by $\mathbb{P}_p(u) = p(u_0)p(u_1)\cdots p(u_{n-1})$.

A further classical generalization consists in allowing some (limited) dependency from the past when generating the word. This leads to the notion of Markov chain, which we describe now. Let Q be a non-empty finite set, called the *set of states*. A sequence of Q-valued random variables $(X_n)_{n\geq 0}$ is a *homogeneous Markov chain* (or just *Markov chain* for short in this article) when for every n, every $\alpha, \beta \in Q$ and every $q_0, \dots q_{n-1} \in Q$,

$$\mathbb{P}(X_{n+1} = \alpha \mid X_n = \beta, X_{n-1} = q_{n-1}, \dots, X_0 = q_0) = \mathbb{P}(X_1 = \alpha \mid X_0 = \beta).$$

In the sequel, we will use the classical representation of a Markov chain by its *initial probability (row) vector* $\pi_0 \in [0, 1]^Q$ and its *transition matrix* $M \in [0, 1]^{Q\times Q}$, defined for every $i, j \in Q$ by $M(i, j) = \mathbb{P}(X_1 = j \mid X_0 = i)$. In this settings, the probability of a word $u = u_0 \cdots u_{n-1}$ on $A = Q$ is

$$\mathbb{P}_{M,\pi_0}(u) = \pi_0(u_0)\, M(u_0, u_1)\, M(u_1, u_2) \cdots M(u_{n-2}, u_{n-1}).$$

Such a Markov chain for generating words of A^* is also called a *first order* Markov chain, since the probability of a new letter only depends on the last letter. One can easily use Markov chains to allow larger dependencies from the past. For instance, a *second order* Markov chain can be defined by setting $Q = A \times A$. The probability of a word $u = u_0 \cdots u_{n-1}$, with $n \geq 2$, is now defined, for an initial probability vector $\pi_0 \in [0, 1]^Q$, by

$$\mathbb{P}_{M,\pi_0}(u) = \pi_0(u_0u_1)M(u_0u_1, u_1u_2)M(u_1u_2, u_2u_3) \cdots M(u_{n-3}u_{n-2}, u_{n-2}u_{n-1}).$$

Higher order Markov chain are defined similarly. More general sources, such as dynamical sources [13], are also considered in the literature, but they are beyond the scope of this article.

2.5 About the probabilistic analysis of the original article

In their article [9], the authors proposed a brief analysis of the expected reduction ratio. This analysis is done under the simplistic assumption that the marks of the positions are independent and of type S or L with probability $\frac{1}{2}$ each.

We first observe that if $A = \{a, b\}$ consists of exactly two letters and if we consider the uniform distribution on A^n, then, up to the very end of the word, every a is of type S and every b is of type L. Hence, we are mostly in the model proposed in [9]. Unfortunately, if there are three or more letters, then uniform distributions, memoryless distributions and Markovian distributions failed to produce types that are i.i.d. in $\{L, S\}$. It is also the case for a binary alphabet, when the distribution under consideration is not the uniform distribution.

Their result, Theorem 3.15 page 1477, also contains a miscalculation. The average reduction ratio when the types are i.i.d. S and L with probability $\frac{1}{2}$ tends to $\frac{1}{4}$ and not to $\frac{1}{3}$ as stated. This can easily be obtained the following way: in this model, a position $i \geq 1$ is such that i is of type S and $i-1$ is of type L with probability $\frac{1}{4}$. The result follows by linearity of the expectation.[1]

In the sequel we give formulas for the reduction ratio for alphabets of any size, and for uniform, memoryless and Markovian distributions.

3 Probabilistic analysis for memoryless sources

If instead of generating a word letter by letter from left to right, we choose to perform the generation from right to left, then it is easy to compute, on the fly, the type of each position. This is a direct consequence of Definition 1. In probabilistic terms, we just defined a Markov chain, built as an extension of our random source. This is the idea developed in this section, and we will use it to compute the typical reduction ratio of the IS-algorithm.

3.1 A Markov chain for the LS-extension

Let A be a totally ordered alphabet, with at least two letters, and let p be a probability on A such that for every $a \in A$, $p(a) > 0$. In this section we consider the memoryless distributions on A^* of probability p, as defined in Section 2.4. To simplify the writing, we will use p_a instead of $p(a)$ in the sequel.

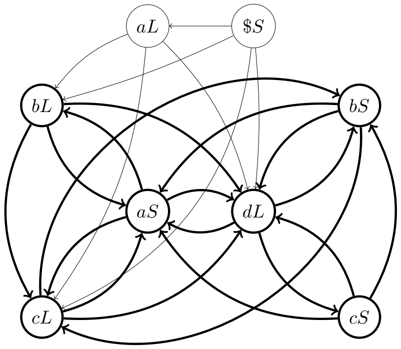
Recall that if P is a property, then $[\![P]\!]$ is equal to 1 if P is true and to 0 if it is false. Let π_0 be the row vector of $[0,1]^{\mathrm{LS}(A)}$ defined by $\pi_0(\alpha X) = [\![\alpha X = \$S]\!]$. Let M_p be the matrix of $[0,1]^{\mathrm{LS}(A)\times\mathrm{LS}(A)}$ defined for every $\alpha, \beta \in A$ by

$$\begin{aligned} M_p(\alpha S, \$S) = M_p(\alpha L, \$S) = M_p(\$S, \beta S) = 0; &\qquad M_p(\$S, \beta L) = p_\beta; \\ M_p(\alpha S, \beta S) = p_\beta \cdot [\![\beta \leq \alpha]\!]; &\qquad M_p(\alpha S, \beta L) = p_\beta \cdot [\![\beta > \alpha]\!]; \\ M_p(\alpha L, \beta S) = p_\beta \cdot [\![\beta < \alpha]\!]; &\qquad M_p(\alpha L, \beta L) = p_\beta \cdot [\![\beta \geq \alpha]\!]. \end{aligned}$$

We first establish that the reverse of the LS-extension of a random word generated by a memoryless source is Markov (M_p, π_0):

Proposition 1. *Let A be a totally ordered alphabet, with at least two letters, and let p be a probability on A such that for every $a \in A$, $p_a > 0$. If u is a word on $\mathrm{LS}(A)$ such that $\mathbb{P}_{M_p,\pi_0}(u) \neq 0$, then the reverse of u is the LS-extension of a word v of A^* and $\mathbb{P}_{M_p,\pi_0}(u) = \mathbb{P}_p(v)$.*

[1] In their proof, they compute the mean length of an LMS-factor. The types of such a factor form a word of SS^*LL^*S. For the considered model, the mean length of an element of S^* (and of L^*) is one. Hence, the average length of an LMS-factor is 5 (and not the announced 4).



$$\begin{array}{c|cccccc} & aS & bS & bL & cS & cL & dL \\ \hline aS & p_a & 0 & p_b & 0 & p_c & p_d \\ bS & p_a & p_b & 0 & 0 & p_c & p_d \\ bL & p_a & 0 & p_b & 0 & p_c & p_d \\ cS & p_a & p_b & 0 & p_c & 0 & p_d \\ cL & p_a & p_b & 0 & 0 & p_c & p_d \\ dL & p_a & p_b & 0 & p_c & 0 & p_d \end{array}$$

The matrix $\underline{M}_p$

Fig. 1. On the left, the underlying graph of the Markov chain for $A = \{a, b, c, d\}$. The state dS is not depicted, as it is not reachable. Every state but $\$S$ also has a loop on itself, which is not depicted for readability. The thin states are the transient states, and the bold states are the recurrent states. For the memoryless source of probability p, the probability of each edge $\alpha X \to \beta Y$ is p_β. If we start on $\$S$ with probability 1, then this chain generates the marked words *from right to left*. On the right is presented the matrix $\underline{M}_p$, which is the restriction of M_p to its recurrent part.

In other words, generating v using a memoryless source of probability p is the same as generating the reverse of $\mathrm{LS}(v)$ using the Markov chain (M_p, π_0).

Proposition 1 is the key observation of this article. It described a purely probabilistic way to work on LS-extensions of random words: the deterministic algorithm used to mark each position with its type is encoded into the Markov chain (M_p, π_0). We now aim at using the classical results on Markov chains to obtain some information on the number of LMS-factors.

3.2 Properties of the Markov chain

From now on, except for the examples, we fix $A = \{a_1, \ldots, a_k\}$, with $k \geq 2$, and we consider the total strict order $<$ on A defined by $a_1 < a_2 \ldots < a_k$.

The *underlying graph* G_M of a Markov chain (M, π_0) of set of states Q is the directed graph whose vertices are the elements of Q and with an edge $s \to t$ whenever $M(s,t) > 0$. A state $q \in Q$ is *transient* when it is not in a terminal strongly connected component of G_M: if we start in state q, there is a non-zero probability that we will never return to q. Transient states play a minor role in our settings as with high probability they are only used during the generation of the very first letters. A state that is not transient is called *recurrent*.

Lemma 1. *The Markov chain (M_p, π_0) has three transient states: $\$S$, a_1L and a_kS. All other states are in the same terminal strongly connected component of its underlying graph.*

Remark 1. From the definition of M_p and π_0, a path of positive probability in the chain always starts on the state $\$S$, may pass through the state a_1L, but never reaches the state a_kS. This property is obvious if one remember that a word generated by the chain is the reverse of the LS-extension of a word on A.

Recall that a Markov chain is *irreducible* when its underlying graph is strongly connected, and that it is *aperiodic* when the gcd of its cycles is equal to 1. Most useful theorems are stated for Markov chains that are either irreducible, or both irreducible and aperiodic. Since the chain (M_p, π_0) is not irreducible, we propose to "approximate" it with an irreducible and aperiodic one. This new Markov chain produces reversed LS-extensions where some of the types can be wrong, for a limited number of positions at the beginning. However, we will see that it does not change significantly the number of LMS-factors, the statistic we are interested in.

3.3 An irreducible and aperiodic Markov chain

Let $\underline{\mathrm{LS}}(A)$ denote the alphabet $\mathrm{LS}(A)$ restricted to the recurrent states of M_p: $\underline{\mathrm{LS}}(A) = \mathrm{LS}(A) \setminus \{\$S, a_1L, a_kS\}$. We first formalize the notion of LS-extension with errors.

Definition 4. *Let u be a word of A^n, with $n \geq 1$, and let $w = \mathbf{Ext}(u)$ be the LS-extension of u. The* pseudo LS-extension $\underline{\mathbf{Ext}}(u)$ *of u is the word $v \in \underline{\mathrm{LS}}(A)^n$ defined by $v_i = a_1S$ if $w_j = a_1L$ for all $j \in \{i, \dots n-1\}$, and $v_i = w_i$ otherwise.*

The pseudo LS-extension of u is therefore obtained from the LS-extension w of u by first removing the last character $\$S$, and then by changing the (possibly empty) sequence of a_1L's at the end into a sequence of a_1S. For instance, if $u = a_3a_1a_2a_1a_1a_1$, then we have $\mathbf{Ext}(u) = a_3L\, a_1S\, a_2L\, a_1L\, a_1L\, a_1L\, \$S$ and $\underline{\mathbf{Ext}}(u) = a_3L\, a_1S\, a_2L\, a_1S\, a_1S\, a_1S$.

Lemma 2. *Let $u \in A^n$, with $n \geq 1$. If u contains at least two different letters and ends with the letter a_1, then $\mathbf{Ext}(u)$ and $\underline{\mathbf{Ext}}(u)$ have the same number of LMS-positions. Otherwise, there is exactly one more LMS-position in $\mathbf{Ext}(u)$.*

Let $\underline{M}_p$ denote the restriction of the matrix M_p to $[0,1]^{\underline{\mathrm{LS}}(A) \times \underline{\mathrm{LS}}(A)}$. This defines a stochastic matrix, since by Lemma 1, the states of $\underline{\mathrm{LS}}(A)$ form a stable subset. By construction, $\underline{M}_p$ is irreducible. It is also aperiodic, as there is a loop on every vertex of $\underline{M}_p$. Let $\underline{\pi}_0$ be the probability row vector on $\underline{\mathrm{LS}}(A)$ defined for every $\alpha \in A \setminus \{a_1\}$ by $\underline{\pi}(\alpha L) = p_\alpha$ and $\underline{\pi}(\alpha S) = 0$, and by $\underline{\pi}(a_1S) = p_{a_1}$. We now restate Proposition 1 using the Markov chain $(\underline{M}_p, \underline{\pi}_0)$.

Proposition 2. *Let A be a totally ordered alphabet, with at least two letters, and let p be a probability on A such that for every $a \in A$, $p_a > 0$. If u is a non-empty word on $\underline{\mathrm{LS}}(A)$ such that $\mathbb{P}_{\underline{M}_p,\underline{\pi}_0}(u) \neq 0$, then the reverse of u is the pseudo LS-extension of a word v of A^* and $\mathbb{P}_{\underline{M}_p,\underline{\pi}_0}(u) = \mathbb{P}_p(v)$.*

Recall that a *stationary vector* of a Markov chain (M, π_0) is a probability row vector π that satisfies the equation $\pi \times M = \pi$. If the chain is irreducible and aperiodic, a classical theorem [10] states that there exists a unique stationary vector. Moreover, after t steps, the probability that we are on a given state q is $\pi(q) + \mathcal{O}(\lambda^t)$, for some $\lambda \in (0,1)$ and for any choice of π_0.

For every $a \in A$, let $p_{<a} = \sum_{\alpha<a} p_\alpha$ et $p_{>a} = \sum_{\alpha>a} p_\alpha$. The following theorem gives an explicit expression for the stationary vector of $\underline{M}_p$.

Theorem 1. *Let A be a totally ordered alphabet, with at least two letters, and let p be a probability on A such that for every $a \in A$, $p_a > 0$. The unique stationary vector of $\underline{M}_p$ is the vector $\underline{\pi}$ defined on $\underline{\mathrm{LS}}(A)$ by*

$$\underline{\pi}(\alpha S) = \frac{p_\alpha\, p_{>\alpha}}{1-p_\alpha} \qquad \textit{and} \qquad \underline{\pi}(\alpha L) = \frac{p_\alpha\, p_{<\alpha}}{1-p_\alpha}.$$

3.4 Main statements

Using Theorem 1 and the classical Ergodic Theorem for Markov chains (Theorem 4.16 page 58 of [6]), we get a precise estimation of the number of LMS-factors, which is also the number of LMS-positions, in a random word for the memoryless distribution of probability p. It is obtained by analyzing the number of LMS-positions in $\underline{\mathbf{Ext}}(u)$. Indeed, by Lemma 2, counting the number of LMS-positions in u is almost the same as counting the number of LMS-positions in $\underline{\mathbf{Ext}}(u)$.

Theorem 2. *Let A be a totally ordered alphabet, with at least two letters, and let p be a probability on A such that for every $a \in A$, $p_a > 0$. Let F_n be the random variable that counts the number of LMS-factors in a random word of length n, generated by the memoryless source of probability p. There exists a sequence $(\varepsilon_n)_{n\geq 0}$ that tends to 0 such that:*

$$\mathbb{P}_p\left(\left|\frac{1}{n}F_n - \gamma_p\right| > \varepsilon_n\right) \xrightarrow[n\to\infty]{} 0, \ \textit{with } \gamma_p = \sum_{a\in A} \frac{p_a}{1-p_a} p_{>a}^2. \tag{1}$$

Corollary 1. *When the input of the IS-algorithm is a random word of length n generated by the memoryless source of probability p, the expected reduction ratio tends to γ_p.*

Remark 2. The statement of Theorem 2 is more precise than a result for the expectation of F_n (as in Corollary 1). For instance, Equation (1) also implies that the random variable $\frac{1}{n}F_n$ is concentrated around its mean.

Remark 3. It is not completely obvious from its definition, but one can rewrite γ_p as $\sum_{a\in A} \frac{p_a}{1-p_a} p_{<a}^2$. As a consequence, if p' is the reverse of p, that is, $p'_{a_i} = p_{a_{k+1-i}}$ for every $1 \leq i \leq k$, then $\gamma_p = \gamma_{p'}$.

We conclude this section by the analysis of some specific cases. First, we simplify the formula of γ_p for uniform distributions.

Lemma 3. *If p is the uniform probability on A, i.e., $p_a = \frac{1}{k}$ for every $a \in A$, then $\gamma_p = \frac{2k-1}{6k}$. In particular, $\gamma_p \to \frac{1}{3}$ as the size of the alphabet tends to infinity.*

Observe also that if p is not uniform, then γ_p may change when one reorders the probabilities values. For instance, if $A = \{a,b,c\}$, we obtain that $\gamma_p = \frac{13}{48}$ for $(p_a, p_b, p_c) = (\frac{1}{4}, \frac{1}{4}, \frac{1}{2})$ and $\gamma_{p'} = \frac{1}{4}$ for $(p'_a, p'_b, p'_c) = (\frac{1}{4}, \frac{1}{2}, \frac{1}{4})$.

For a binary alphabet $A = \{a,b\}$, we have $p_b = 1 - p_a$ and $\gamma_p = p_a(1-p_a)$.

4 Markovian sources

Let (N, ν_0) be a Markov chain on A. We say that it is a *complete Markov chain* when for every $\alpha, \beta \in A$, $N(\alpha, \beta) > 0$. A complete Markov chain is always irreducible and aperiodic. The construction of Section 3 can readily be extended to words that are generated *backward*, i.e., from right to left, using a complete Markov chain (N, ν_0). Let π_0 be the probabilistic vector of $[0,1]^{\underline{\mathrm{LS}}(A)}$ such that $\underline{\pi}_0(a_1 S) = \nu_0(a_1)$ and for every $\alpha \neq a_1$, $\underline{\pi}_0(\alpha L) = \nu_0(\alpha)$ and $\underline{\pi}(\alpha S) = 0$. Let $\underline{M}_N$ be the matrix of $[0,1]^{\underline{\mathrm{LS}}(A) \times \underline{\mathrm{LS}}(A)}$ defined for every $\alpha, \beta \in A$ by[2]

$$\begin{aligned} \underline{M}_N(\alpha S, \beta S) &= N(\alpha, \beta) \cdot [\![\beta \leq \alpha]\!]; & \underline{M}_N(\alpha S, \beta L) &= N(\alpha, \beta) \cdot [\![\beta > \alpha]\!]; \\ \underline{M}_N(\alpha L, \beta S) &= N(\alpha, \beta) \cdot [\![\beta < \alpha]\!]; & \underline{M}_N(\alpha L, \beta L) &= N(\alpha, \beta) \cdot [\![\beta \geq \alpha]\!]. \end{aligned}$$

Proposition 2 can be generalized to first order complete Markov chains the following way:

Proposition 3. *Let A be a totally ordered alphabet, with at least two letters, and let (N, ν_0) be a complete Markov chain on A. If u is a word on $\underline{\mathrm{LS}}(A)$ such that $\mathbb{P}_{M_N, \pi_0}(u) \neq 0$, then the reverse of u is the pseudo LS-extension of a word v of A^* and $\mathbb{P}_{\underline{M}_N, \underline{\pi}_0}(u) = \mathbb{P}_{N, \nu_0}(v)$.*

Though more complicated than in the memoryless case, the stationary vector of $\underline{M}_N$ can be calculated explicitly. This yields a computable formula for the typical number of LMS-factors:

Theorem 3. *Let A be a totally ordered alphabet, with at least two letters, and let (N, ν_0) be a complete Markov chain on A of stationary vector ν. Let F_n be the random variable that counts the number of LMS-factors in a random word of length n generated backward by (N, ν_0). There exists a sequence $(\varepsilon_n)_{n \geq 0}$ that tends to 0 such that*

$$\mathbb{P}_{N, \nu_0}\left(\left| \frac{1}{n} F_n - \gamma_N \right| > \varepsilon_n \right) \xrightarrow[n \to \infty]{} 0, \text{ with } \gamma_N = \sum_{a \in A} \underline{\pi}(aS) \sum_{b > a} N(a, b),$$

where $\underline{\pi}$ is the stationary vector of $\underline{M}_N$, which satisfies

$$\underline{\pi}(\alpha S) = \frac{\sum_{\beta > \alpha} \nu(\beta) N(\beta, \alpha)}{1 - N(\alpha, \alpha)} \quad \text{and} \quad \underline{\pi}(\alpha L) = \frac{\sum_{\beta < \alpha} \nu(\beta) N(\beta, \alpha)}{1 - N(\alpha, \alpha)}.$$

As a consequence, the expected reduction ratio in the first recursive call of the IS-algorithm tends to γ_N, as n tends to infinity.

Remark 4. This can be generalized to Markov chains that are not complete Markov chains, but by lack of place, we cannot describe how it works in this extended abstract. The fact that the word is generated backward is usually not an issue: if the initial distribution is equal to the stationary distribution, then there exists a Markov chain that generates the words from left to right with the same probability (see [10]). It is natural to start with the stationary distribution, as it often coincides with the empirical frequencies of the letters.

[2] The formulas below hold when the extended letters are in $\underline{\mathrm{LS}}(A)$ only. For instance, $\alpha L = a_1 L$ is not part of the definition, since it is not in $\underline{\mathrm{LS}}(A)$.

File	$\lvert A\rvert$	**size**	**red. ratio**	**uniform**	**memoryless**	**Markov**
`bible.txt`	63	4047392	**0.3113**	0.3307	0.3230	0.3251
`world192.txt`	93	2408281	**0.2838**	0.3315	0.3256	0.2997
`Chr_22.fa`	4	35033745	**0.2717**	0.2917	0.2928	0.2715

Fig. 2. In these experiments we compare the real reduction ratio with the theoretical ratios obtained when approximating the distributions by one of the models proposed in this article. The first two files are from the Canterbury corpus [11], the last one is the human chromosome 22 [2]. The real reduction ratio of the first recursive call is indicated in the column "red. ratio". The three last columns were obtained after computing a model (either uniform, memoryless or Markovian) from the file. The different values are the γ's given by Lemma 3, Theorem 2 and Theorem 3. The Markov chains of the first two files are not complete, but our results still hold, as Theorem 3 can be generalized to irreducible and aperiodic chains (see Section 5).

5 Experiments and conclusions

Though we provide a theoretical analysis of the IS-algorithm for classical distributions on words in this article, we thought it would be interesting to include some experiments on real data, even if we are not pretending to demonstrate anything with these few tests. These results are depicted in Fig. 2. It is also not our purpose to provide a statistical analysis of this information here, but we cannot help noticing that for the human chromosome 22, a Markov chain of order 1 seems to be an accurate model for analyzing the behavior of the IS-algorithm.[3]

The methodology presented in Section 4 can be extended to Markov chains (N, ν) that are only irreducible and aperiodic; the set of recurrent states may just be strictly included in $\underline{\mathrm{LS}}(A)$. It can also be extended to Markov chains of higher order, but the formulas become more and more complicated. Lets consider, say, a Markov chain of order 3 on $A = \{a, b, c, d\}$. Observe that in the recurrent part, a state adb is necessarily of type S since $b > d$. In fact, we always know the type of the last letter, except when the state is of the form $\alpha\alpha\alpha$. We need two different states for such words, one of type S and one of type L. Furthermore, $aaaL$ is transient and $dddS$ is not reachable. There are therefore $|A|^t + |A| - 2$ recurrent states in the Markov chain $\underline{M}_N$, where t is the order.

A continuation this work would be to analyze the whole behavior of the algorithm, when the reduction ratios of all the successive recursive calls are taken into account. This is technically challenging, as the letters of a given recursive call are the LMS-factors of the word at the previous stage. The precise analysis of other algorithms that compute suffix arrays is another natural direction for further investigations.

[3] This may be a consequence of the well-known fact that in a vertebrate genome, a C is very rarely followed by a G. This property is well captured by a Markov chain of order 1, but invisible to a memoryless model.

References

1. R. A. Baeza-Yates, E. Chávez, and M. Crochemore, editors. *Combinatorial Pattern Matching, 14th Annual Symposium, CPM 2003, Morelia, Michocán, Mexico, June 25-27, 2003, Proceedings*, volume 2676 of *Lecture Notes in Computer Science*. Springer, 2003.
2. I. Dunham, A. Hunt, J. Collins, R. Bruskiewich, D. Beare, M. Clamp, L. Smink, R. Ainscough, J. Almeida, A. Babbage, et al. The DNA sequence of human chromosome 22. *Nature*, 402(6761):489–495, 1999.
3. J. Kärkkäinen and P. Sanders. Simple linear work suffix array construction. In J. C. M. Baeten, J. K. Lenstra, J. Parrow, and G. J. Woeginger, editors, *Automata, Languages and Programming, 30th International Colloquium, ICALP 2003, Eindhoven, The Netherlands, June 30 - July 4, 2003. Proceedings*, volume 2719 of *Lecture Notes in Computer Science*, pages 943–955. Springer, 2003.
4. D. K. Kim, J. S. Sim, H. Park, and K. Park. Linear-time construction of suffix arrays. In Baeza-Yates et al. [1], pages 186–199.
5. P. Ko and S. Aluru. Space efficient linear time construction of suffix arrays. In Baeza-Yates et al. [1], pages 200–210.
6. D. A. Levin, Y. Peres, and E. L. Wilmer. *Markov chains and mixing times.* American Mathematical Soc., 2009.
7. U. Manber and E. W. Myers. Suffix Arrays: A New Method for On-Line String Searches. *SIAM J. Comput.*, 22(5):935–948, 1993.
8. U. Manber and G. Myers. Suffix Arrays: A New Method for On-Line String Searches. In D. S. Johnson, editor, *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms, 22-24 January 1990, San Francisco, California.*, pages 319–327. SIAM, 1990.
9. G. Nong, S. Zhang, and W. H. Chan. Two efficient algorithms for linear time suffix array construction. *IEEE Trans. Computers*, 60(10):1471–1484, 2011.
10. J. R. Norris. *Markov chains.* Cambridge series in statistical and probabilistic mathematics. Cambridge University Press, 1998.
11. M. Powell. The Canterbury Corpus. *Accessed April*, 25:2002, 2001. Available online at http://www.corpus.canterbury.ac.nz/.
12. S. J. Puglisi, W. F. Smyth, and A. Turpin. A taxonomy of suffix array construction algorithms. *ACM Comput. Surv.*, 39(2), 2007.
13. B. Vallée. Dynamical sources in information theory: Fundamental intervals and word prefixes. *Algorithmica*, 29(1-2):262–306, 2001.