The central tree property and algorithmic problems on subgroups of free groups

Mallika Roy, Enric Ventura and Pascal Weil*

Communicated by Rachel Skipper

Abstract. We study the average case complexity of the Uniform Membership Problem for subgroups of free groups, and we show that it is orders of magnitude smaller than the worst case complexity of the best known algorithms. This applies to subgroups given by a fixed number of generators as well as to subgroups given by an exponential number of generators. The main idea behind this result is to exploit a generic property of tuples of words, called the central tree property. An application is given to the average case complexity of the Relative Primitivity Problem, using Shpilrain's recent algorithm to decide primitivity, whose average case complexity is a constant depending only on the rank of the ambient free group.

1 Introduction

Algorithmic problems have been prominent in the theory of infinite groups at least since Dehn formulated the word problem for finitely presented groups [7]: a finite group presentation $\langle A | R \rangle$ being fixed, the word problem asks whether a given element of the free group on A (seen as a reduced word on the alphabet $A \cup A^{-1}$) is equal to the identity in the group presented by $\langle A | R \rangle$.

A problem is called *decidable* if one can exhibit an algorithm that solves it. It is well known that the word problem is undecidable for certain finite group presentations (Novikov [16]). For decidable problems, it is natural to try to evaluate the complexity of an algorithm solving them, namely the amount of resources (time or space) required to run the algorithm, as a function of the size of the input.

The most common complexity evaluation for an algorithm $\mathcal{A}$ is the *worst case complexity*, which measures the maximum time required to run $\mathcal{A}$ on an input of size n. In certain cases, it may be relevant to consider the *generic complexity* of $\mathcal{A}$: $\mathcal{A}$ has generic complexity at most $f(n)$ if the ratio of inputs of size n on which $\mathcal{A}$ requires time at most $f(n)$ tends to 1 as n tends to infinity. This notion of complexity recognizes that the instances that are hard for $\mathcal{A}$ (the witnesses of

the worst case complexity) may be few, but it does not attempt to quantify the time required on the instances in a vanishing set.

Here we will be concerned with the more precise *average case complexity*, namely the expected time required to run $\mathcal{A}$ on size n instances taken uniformly at random. This measure of complexity takes into account the resources needed for every input.

The specific problems we consider in this paper are the Uniform Membership Problem and the Relative Primitivity Problem in finite rank free groups. Recall that an element w of $F(A)$, the free group on A, is *primitive* if $F(A)$ admits a free basis containing w. The *Uniform Membership Problem* (resp., the *Relative Primitivity Problem*) is the following: given elements $w_0, w_1, \ldots, w_k$ of $F(A)$, decide whether w_0 belongs to (resp., is primitive in) the subgroup H of $F(A)$ generated by $w_1, \ldots, w_k$. In this paper, the length k of the tuple $(w_1, \ldots, w_k)$ is not fixed, and the parameters we consider to gauge the size of an instance are $n = \max\{|w_i| \mid 1 \leq i \leq k\}$, k as a function of n, and $m = |w_0|$. In particular, the total length of an input $(w_0, w_1, \ldots, w_k)$ is at most $kn + m$.

It is well known that both these problems are decidable, and can be solved in polynomial time worst case complexity (see [20], [17, Fact 3.6], and Sections 4, 5 below). The most efficient solution (again, from the point of view of the worst case complexity) of the Uniform Membership Problem uses the concept of the Stallings graph of a subgroup, a finite A-labeled graph uniquely associated with a finitely generated subgroup of $F(A)$, which can be easily computed and then gives a linear time solution for the Uniform Membership Problem. More precisely, the Stallings graph of H has at most kn vertices and it is computed in time $\mathcal{O}(kn \log^*(kn))$ (see [21]). After this computation, deciding whether $w_0 \in H$ is done in linear time in m.

Our main result is an algorithm solving the Uniform Membership Problem, whose average case complexity is asymptotically a little o of the worst case complexity described above, at least when the number k grows at most polynomially with n. It is notable that the dependance of our algorithm's expected performance on m (the length of the word w_0 to be tested) is extremely low.

A specific instance of our main result shows, for instance, that if k is a constant, then the Uniform Membership Problem can be solved in expected time

$$\mathcal{O}(\log n + mn^{-\log(2r-1)}),$$

where $r = |A|$ is taken to be constant.

The fundamental ingredient in this result is the so-called *central tree property* (ctp) for a tuple $(w_1, \ldots, w_k)$. This property, formally introduced in [2, 3], holds when the w_i and their inverses have little initial cancellation (that is, they have short common prefixes). This property turns out to hold with high probability and,

when it holds, solving the Uniform Membership Problem is considerably simpler than in the general case.

We then apply our result to the Relative Primitivity Problem: we give an algorithm solving it whose average case complexity is much lower than its worst case complexity. Here we use an algorithm recently proposed by Shpilrain [19] to solve the Primitivity Problem (deciding whether a given word w_0 is primitive in $F(A)$). Shpilrain's algorithm has the remarkable property of having constant average case complexity, that is, its expected time does not depend on the length of w_0. As it turns out, this constant average case complexity depends on the rank of the ambient free group and this is important in the context of the Relative Primitivity Problem, where we need to test for primitivity in the subgroup generated by $w_1, \ldots, w_k$, whose rank may be as large as k.

The paper is organized as follows. Section 2 briefly discusses the fundamental notions on subgroups of free groups which we will use, especially the notions of the Stallings graph and the growth function of a subgroup, as well as the notion of average case complexity and the computational model which we rely upon. Section 3 is dedicated to the central tree property, applied to a tuple $\vec{w} = (w_1, \ldots, w_k)$, its definition, its consequences in terms of the rank and the growth function of the subgroup generated by $\vec{w}$, and the probability that it holds (in terms of the parameters k and n). Our main result on the average case complexity of the Uniform Membership Problem is presented in Section 4. Finally, we discuss Shpilrain's constant average case complexity algorithm for the Primitivity Problem, and our application to the Relative Primitivity Problem in Section 5.

2   Preliminaries

2.1   Subgroups of free groups

Throughout the paper, A is a finite non-empty set, called an *alphabet*. We say that a directed graph Γ is an A-graph if its edges are labeled with elements of A. A *rooted A-graph* is a pair (Γ, v), where Γ is a finite A-graph and v is a vertex of Γ. Finally, we say that a rooted A-graph (Γ, v) is *reduced* if Γ is finite and connected, distinct edges with the same start (resp., end) vertex always have distinct labels, and every vertex, except possibly the root v, is incident to at least two edges.

The set $\tilde{A} = \{a, a^{-1} \mid a \in A\}$ (with cardinality $2|A|$) is called the (*symmetrized*) *alphabet*; its elements are called *letters*. We denote by $\tilde{A}^*$ the set of *words* on $\tilde{A}$, that is, of finite sequences of letters. We also denote by A^* the set of words using only letters in A. A word in $\tilde{A}^*$ is said to be *reduced* if no letter $a \in A$ is immediately preceded or followed by the letter a^{-1}.

For convenience, if $A = \{a_1, \ldots, a_k\}$, we let $a_{-i} = a_i^{-1}$ for $1 \leq i \leq k$ so that $\tilde{A} = \{a_i \mid -k \leq i \leq k, i \neq 0\}$.

Suppose that $w = x_1 \cdots x_m$ is a word in $F(A)$ (with each $x_i \in \tilde{A}$) and that p, q are vertices of a reduced A-graph Γ. We say that w *labels a path in* Γ *from p to q* if there exists a sequence of vertices $p_0 = p, p_1, \ldots, p_m = q$ such that, for every $1 \leq i \leq m$, Γ has an x_i-labeled edge from p_{i-1} to p_i if $x_i \in A$, and an x_i^{-1}-labeled edge from p_i to p_{i-1} if $x_i^{-1} \in A$. If the start and end vertices of the path are equal (that is, if $p = q$), we say that the path is a *circuit at p*.

The free group on A is written $F(A)$, and we identify it with the set of reduced words in $\tilde{A}^*$. It is well known that every subgroup of $F(A)$ is free [15]. It is also well known that every finitely generated subgroup H of $F(A)$ can be associated with a uniquely defined reduced rooted A-graph $(\Gamma(H), 1)$, called the *Stallings graph* of H, with the following property: a reduced word is in H if and only if it labels a circuit in $\Gamma(H)$ at vertex 1. We refer the reader to the seminal works of Serre [18] and Stallings [20] who introduced this combinatorial tool, and to [8, 12–14, 17] for some of its many applications.

Of particular interest for this paper are the following facts.

- Given $\vec{w} = (w_1, \ldots, w_k)$ a tuple of reduced words in $F(A)$, one can effectively compute the Stallings graph $(\Gamma(H), 1)$ of the subgroup H generated by the w_i, and this graph has at most $n = \sum_i |w_i|$ vertices. Touikan [21] showed that $(\Gamma(H), 1)$ can be computed in time $\mathcal{O}(n \log^* n)$. Recall that $\log^*(n)$ is the least integer k such that the k-th iterate of the logarithmic function yields a result less than or equal to 1, that is, such that $\log^{(k)}(n) \leq 1 < \log^{(k-1)}(n)$. Equivalently, let $b_1 = 2$ and $b_{k+1} = 2^{b_k}$. Then $\log^* n = 1$ if $n \leq b_1$ and $\log^* n = k$ if $b_{k-1} < n \leq b_k$.

- Once $\Gamma(H)$ is constructed, deciding whether a word $w_0 \in F(A)$ is an element of H is done by checking whether w_0 labels a circuit in $\Gamma(H)$ at vertex 1. This can be done in time $\mathcal{O}(|w_0|)$.

- Let V and E be the sets of vertices and edges of $\Gamma(H)$, respectively. Let T be a spanning tree of $\Gamma(H)$ (that is, a subgraph of $\Gamma(H)$ which is a tree and contains every vertex of $\Gamma(H)$). Let E_T be the set of edges of T; then we have $|E_T| = |V| - 1$. For each vertex p of $\Gamma(H)$, let $u(p)$ be the only reduced word which labels a path in T from the root vertex 1 to vertex p (with $u(1)$ the empty word). For each edge e of $\Gamma(H)$ which is not in E_T, say, e is an edge from vertex p_e to vertex q_e with label $a(e) \in A$, let $b(e) = u(p_e)a(e)u(q_e)^{-1}$. Then $b(e)$ is a reduced word in $F(A)$ which labels a circuit at 1 so that $b(e) \in H$. Moreover, the set $\{b(e) \mid e \in E \setminus E_T\}$ is a basis of H.

- If T is a fixed spanning tree of $\Gamma(H)$ and B is the corresponding basis of H, in bijection with $E \setminus E_T$, the expression of an element of H in that basis is

obtained as follows. Given a reduced word w in H, consider the circuit at 1 labeled by H and the sequence of edges not in T traveled by this circuit, say $e_1^{\varepsilon_1}, \ldots, e_h^{\varepsilon_h}$, where the e_i are in $E \setminus E_T$, $\varepsilon_i = 1$ if e_i is traversed in the direct sense, and $\varepsilon_i = -1$ if e_i is traversed backwards. Then $w = b(e_1)^{\varepsilon_1} \cdots b(e_h)^{\varepsilon_h}$.

- One can compute a spanning tree of $\Gamma(H)$ in time $\mathcal{O}(|V| + |E|)$ (by classical depth-first search). Since $|E| \leq |V||A|$, it follows that one can compute a spanning tree of $\Gamma(H)$, and therefore a basis of H, in time $\mathcal{O}(|V||A|)$.

- The subgroup H has finite index if and only if every vertex of $\Gamma(H)$ is the origin (and so the terminus) of an a-labeled edge for every letter $a \in A$ or, equivalently, if $|E| = |V||A|$. In that case, the index of H is $|V|$.

2.2 Growth modulus

The *growth function* of a set L of words over an alphabet A is the function $s_L(n)$ counting the words of length n in L. The following result belongs to the folklore of combinatorial automata theory.

Fact 2.1. If L is a regular language, then its growth function (restricted to its support, which is the complement of an ultimately periodic sequence) is asymptotically equivalent to an expression of the form $Cn^k\lambda^n$, with $C > 0$, $k \in \mathbb{N}$ and $\lambda \geq 1$. $\qquad\square$

The real number λ in Fact 2.1 is called the *growth modulus* of L.

Proof. A result usually attributed to Chomsky and Schützenberger [6] (see [9, Proposition I.3] for a quick proof) states that the generating function of L,

$$S_L(z) = \sum_{n \geq 0} s_L(n) z^n,$$

is a rational fraction. The partial fraction decomposition (over the reals) of this rational fraction yields the stated asymptotic equivalent for the coefficients $s_L(n)$ of $S_L(z)$. $\qquad\square$

We record the following elementary remark.

Remark 2.2. If a language L has at most $C\lambda^n$ words of length n ($C > 0$, $\lambda > 1$), then the number of its words of length less than or equal to n is at most $C\frac{\lambda}{\lambda-1}\lambda^n$, which is $\Theta(\lambda^n)$. $\qquad\square$

The Perron–Frobenius theorem (see, e.g., [11, Theorem 8.4.4]) yields a more precise characterization of the growth modulus of a regular language. If $\mathcal{A}$ is a finite state automaton over alphabet A with state set Q, we denote by $G_{\mathcal{A}}$ the

graph with vertex set Q and with an edge from vertex p to vertex q for every letter $a \in A$ labeling a transition from p to q. We also let $M_{\mathcal{A}}$ be the associated incidence $(Q \times Q)$-matrix. Note that $M_{\mathcal{A}}$ has only non-negative integer coefficients. We say that $\mathcal{A}$ is *irreducible* if $M_{\mathcal{A}}$ is, that is, if $G_{\mathcal{A}}$ is strongly connected.

The *period* of $\mathcal{A}$ is defined to be the largest positive integer d such that Q can be partitioned as $Q = Q_1 \sqcup \cdots \sqcup Q_d$ in such a way that every transition from a state in Q_i leads to a state in Q_{i+1} (indices are taken modulo d). For instance, the minimal automaton of the language of words of length a multiple of d has period d. Finally, we say that $\mathcal{A}$ is aperiodic if its period is 1.

The following result also belongs to the folklore; see [9, Proposition V.7].

Proposition 2.3. *Let L be a regular language, accepted by a deterministic finite state automaton $\mathcal{A}$ which is irreducible and aperiodic. The growth modulus of L is equal to the dominant eigenvalue of the transition matrix of $\mathcal{A}$.*

The growth modulus of the free group $F(A)$ is easily computed.

Example 2.4. It is clear that, for every integer $n \geq 1$, the number R_n of reduced words of length n is $2r(2r-1)^{n-1}$, where $r = |A|$.

Recall that we identify $F(A)$ with the set of reduced words over the alphabet $\tilde{A}$. Thus the growth modulus of $F(A)$ is $2r - 1$.   □

2.3   Algorithmic problems: Average case complexity

In evaluating the complexity of an algorithm, one needs to specify the model of computation and the input space. The input space is usually equipped with a notion of size (a positive integer) such that there are finitely many inputs of any given size. It is important, in particular, to make clear which parameters of the problem are taken to be constants. Unless otherwise indicated, we consider in this paper that the rank $r = |A|$ of the ambient free group is a constant.

The model of computation we adopt is the standard RAM model. Concretely, this means that if a word w is part of the input, it takes unit time to move the reading head to a position i (where i has been computed before), and unit time to read the letter of w in position i. Arithmetic operations on integers (addition, multiplication) are also considered as taking unit time.

Remark 2.5. In the standard RAM model, figuring out the length of an input word w (in order, for instance, to read its last letter) takes time $\mathcal{O}(\log|w|)$, using an instance of the so-called exponentiation search method [4]: one reads letters in positions 1, 2, 4, 8, 16, etc., until one exceeds the length of the word after, say, b steps. At that point, we know that $2^{b-1} \leq |w| < 2^b$, that is, we know the leading

bit of the binary expansion of $|w|$. The next bits are established by a classical dichotomy method. Concretely, probing position $c = 2^{b-1} + 2^{b-2}$ allows us to know the second bit: 1 if position c is still in the word, 0 if c exceeds its length. This is repeated for the successive bits of the binary expansion of $|w|$.

Since we are going to work with algorithms with very low, even constant, average case complexity, we do not want to have to add the logarithmic time needed to compute the length of input words, and will therefore accompany every input word with its length. □

Remark 2.6. When handling "very large" integers, or words on a "very large" alphabet, it may be more appropriate to use the bitcost model: adding integers takes time linear in the length of their binary representations (that is, in their logarithm), and reading or comparing letters from a large alphabet A takes time proportional to $\log|A|$ (since each letter can be encoded in a bit string of length $\lceil \log|A| \rceil$). □

The *worst case complexity of an algorithm* $\mathcal{A}$ is the function $f(n)$, defined on $\mathbb{N}$, which accounts for the maximum time required to run algorithm $\mathcal{A}$ on an input of size n. If a distribution is specified on the set of size n inputs (in this paper, the uniform distribution), the *average case complexity* of $\mathcal{A}$ is the function $g(n)$ which computes the expected time required to run $\mathcal{A}$ on inputs of size n. Such complexity functions are usually considered up to asymptotic equivalence.

The average case complexity of an algorithm is obviously bounded above by the worst case complexity, and it may sometimes be much lower. The usual idea in discussing average case complexity is to distinguish, within the input space, between a subset of high probability where the algorithm performs very fast, and its low-probability complement, containing all the hard instances (those which witness the worst case complexity).

Finally, the worst case (resp., average case) *complexity of a problem* is the lowest worst case (resp., average case) complexity of an algorithm solving this problem.

A well-known example which will be useful in the sequel, is the *Proper Prefix Problem* (PPP) on alphabet A: given two words u, v on alphabet A, decide whether u is a proper prefix of v (that is, $v = uu'$ for some non-empty word u'). We also consider the *Prefix Problem* (PP) (deciding whether u is a prefix of v) and the *Equality Problem* (EqP) (deciding whether $u = v$). The following observation is elementary: it is just an instance of the fact that the expected value of a geometric distribution is constant.

Lemma 2.7. *Let A be a finite alphabet with $|A| \geq 2$ and suppose that each set A^n (the words of length n) is equipped with the uniform distribution. Problems PPP,*

PP and EqP can be solved in constant expected time. The same holds for the set R_n of (reduced) words of length n in $F(A)$.

Proof. We prove the statement for PPP. The proofs for PP and EqP are entirely similar. Here is a simple and natural algorithm solving the PPP.

Algorithm $\mathcal{PPP}$. On input u, v, read u and v from left to right one letter at a time, comparing each letter of u with the corresponding letter of v, and stopping when (i) a difference is detected (that is, the i-th letters of u and v are different for some i); (ii) the end of u is reached but not the end of v; or (iii) the end of v is reached.

It is clear that u is a proper prefix of v in case (ii), and not a proper prefix of v in every other case. That is, Algorithm $\mathcal{PPP}$ solves the PPP. We now show that it has constant average case complexity. In effect, this is due to the fact that, with high probability, we will detect a difference between the words u and v without having to read either word to its end.

Let u_i (resp., v_i) denote the i-th letter in u (resp., v). At each step, verifying whether $u_i = v_i$ is done in constant time, and it is the case with probability $p = \frac{1}{|A|}$. Therefore, the probability that the algorithm stops after exactly k steps (with $k \le |u|, |v|$) is $p^{k-1}(1 - p)$. Detecting whether we reached the end of u or v is also done in constant time. It follows that the average case complexity of Algorithm $\mathcal{PPP}$ is bounded above, up to a multiplicative constant, by

$$1 + p + \cdots + p^{|u|-1} \le \frac{1}{1 - p} = \frac{|A|}{|A| - 1} \le 2,$$

and this concludes the proof relative to A^*.

To transfer this result to $F(A)$, one can for instance rewrite the words in $F(A)$ as follows. Let $X = \{x_1, \ldots, x_{2r-1}\}$ and $\tilde{A} = \{a_{-r}, \ldots, a_{-1}, a_1, \ldots, a_r\}$ be ordered in the natural way. For each $b \in \tilde{A}$, we let ζ_b be the order isomorphism from $\tilde{A} \setminus \{b\}$ to X.

Given a word $w = b_1 \cdots b_n \in F(A)$ of length $n \ge 2$, we let $\tau(w) = b_1 c_2 \cdots c_n$ be the word in $\tilde{A} X^{n-1}$, where $c_{i+1} = \zeta_{b_i^{-1}}(b_{i+1})$ for every $1 \le i < n$. It is clear that τ is a bijection between R_n and $\tilde{A} X^{n-1}$, and hence it preserves the uniform distribution.

The result follows if we modify Algorithm $\mathcal{PPP}$ as follows: on input u and v, the algorithm first compares u_1 and v_1, and then it compares $\zeta_{u_i^{-1}}(u_{i+1})$ and $\zeta_{v_i^{-1}}(v_{i+1})$ until a difference is detected.   □

Remark 2.8. Lemma 2.7 establishes that the expected running time of Algorithm $\mathcal{PPP}$ is bounded by a constant. This relies on our choice of the RAM model

of computation, according to which it takes constant time to compare two letters in A. As mentioned in Remark 2.6, if A is very large, it may take non-trivial time to perform such a comparison (namely time $\mathcal{O}(\log|A|)$) and, in that case, the expected running time of Algorithm $\mathcal{PPP}$ is $\mathcal{O}(\log|A|)$. □

3   The central tree property: A generic property of tuples of words

If d is a positive integer, we say that the k-tuple $\vec{w} = (w_1, \ldots, w_k)$ of words in $F(A)$ has the *central tree property of depth d* (the *d-ctp* for short) if the w_i have length greater than $2d$ and the prefixes of length d of the w_i and the w_i^{-1} are pairwise distinct. We also say that $\vec{w}$ *has the ctp* if it has the d-ctp for some $d < \frac{1}{2}\min\{|w_i| \mid 1 \le i \le k\}$. The central tree property was formally introduced in [2] (see also [3]), but it was implicit in the literature, especially on the (exponential genericity of the) small cancellation property since the ctp can be viewed as a *small initial cancellation property*.

Let $\vec{w} = (w_1, \ldots, w_k)$ be a k-tuple of words in $F(A)$. For convenience, we let $\min|\vec{w}| = \min\{|w_i| \mid 1 \le i \le k\}$ and $\max|\vec{w}| = \max\{|w_i| \mid 1 \le i \le k\}$, and we write w_{-i} for w_i^{-1} ($1 \le i \le k$).

Suppose that $\vec{w}$ has the d-ctp. Then we let pr_i be the length d prefix of w_{-i} and let $\mathsf{mf}_d(w_i)$ be the *middle factor* of w_i of length $|w_i| - 2d$. In particular, we have $\mathsf{mf}_d(w_{-i}) = \mathsf{mf}_d(w_i)^{-1}$ and

$$w_i = \mathsf{pr}_{-i} \cdot \mathsf{mf}_d(w_i) \cdot \mathsf{pr}_i^{-1} \tag{3.1}$$

for $-k \le i \le k, i \ne 0$.

Denote by $L(\vec{w})$ the set of all the pr_i. By definition of the d-ctp, $|L(\vec{w})| = 2k$. Let also $\Gamma_d(\vec{w})$ be the tree of prefixes of the w_i and w_i^{-1} (rooted at the empty word): this is the graph with vertices all the prefixes of the words w_i and w_i^{-1} ($1 \le i \le k$), including the empty word, and with an edge from a word v to a word w exactly if $w = va$ ($a \in \tilde{A}$). We identify the words in $L(\vec{w})$ with the corresponding leaves of $\Gamma_d(\vec{w})$. If $H = \langle \vec{w} \rangle$ (i.e., $H = \langle w_1, \ldots, w_k \rangle$), then $\Gamma(H)$ consists of the *central tree* $\Gamma_d(\vec{w})$, together with k disjoint paths: for every $1 \le i \le k$, there is such a path from vertex pr_{-i} to vertex pr_i, labeled $\mathsf{mf}_d(w_i)$. In view of equation (3.1), the word w_i labels a circuit at the root in $\Gamma(H)$, going first through $\Gamma_d(\vec{w})$ to the leaf pr_{-i}, and returning to the root through the leaf pr_i.

Remark 3.1. It follows directly from the definition that one can decide whether a k-tuple $\vec{w}$ has the d-ctp, and construct $\Gamma_d(\vec{w})$, in time $\mathcal{O}(kd)$. □

We record the following property of k-tuples with the ctp, the first of which is [2, Lemma 1.2].

Proposition 3.2. *Let $d \geq 1$, let $\vec{w}$ be a tuple of words in $F(A)$ with the d-ctp and let $H = \langle \vec{w} \rangle$. Then H has infinite index and $\vec{w}$ is a basis of H.*

Let $\mu = \min|\vec{w}|$ and $\nu = \max|\vec{w}|$. If w_0 is a word in $F(A)$ which belongs to H, then the length ℓ of the expression of w_0 in the basis $\vec{w}$ satisfies

$$(\mu - 2d)\ell \leq |w_0| \leq \nu\ell.$$

Proof. The infinite index property is immediately verified since $\Gamma(H)$ has vertices of degree 2, namely the leaves of $\Gamma_d(\vec{w})$; see Section 2.1.

Let $X = \{x_1, \ldots, x_k\}$ be a k-letter alphabet and let $\varphi \colon F(X) \to F(A)$ be the morphism given by $\varphi(x_i) = w_i$. It is obvious that the image of φ is H. Recall that we let $x_{-i} = x_i^{-1}$ for every $1 \leq i \leq k$. Let $x = x_{i_1} \cdots x_{i_\ell}$ be a non-empty word in $F(X)$. Then $\varphi(x)$ is the word obtained by reducing $w_{i_1} \cdots w_{i_\ell}$. Because of the ctp, reduction occurs only in segments of length at most $2d$ around the boundary between w_{i_h} and $w_{i_{h+1}}$ for each $1 \leq h < \ell$. In particular,

$$\ell\mu - 2d(\ell - 1) \leq |\varphi(x)| \leq \ell\nu.$$

It follows that $\varphi(x) \neq 1$, and hence φ is injective and $\vec{w}$ is a basis of H. □

It is well known that an infinite index subgroup H of $F(A)$ has growth modulus smaller than $2r - 1$. In the case where H is generated by a tuple with the ctp, its growth modulus is greatly constrained.

Proposition 3.3. *Let $\vec{w}$ be a k-tuple of words with the d-ctp and let $\mu = \min|\vec{w}|$. Let $H = \langle \vec{w} \rangle$. Then the growth modulus of H is at most $(2k - 1)^{\frac{1}{\mu - 2d}}$.*

Proof. Let X and φ be as in the proof of Proposition 3.2, let $x \in F(X)$ and $w = \varphi(x) \in H$. Then $(\mu - 2d)|x| \leq |w|$. If $|w| = m$, then $|x| \leq \frac{m}{\mu - 2d}$. In particular, the set of words of H of length m is contained in the φ-image of the words in $F(X)$ of length at most $\frac{m}{\mu - 2d}$. This set has cardinality $\Theta((2k - 1)^{\frac{m}{\mu - 2d}})$; see Remark 2.2. The stated inequality follows since φ is a bijection between $F(X)$ and H. □

We also record the following fact, which is elementarily verified and is well known (see, e.g., [1]).

Proposition 3.4. *Let $k \geq 1$. The probability for a k-tuple $\vec{w}$ of words in $F(A)$ of length at most n to satisfy $\min|\vec{w}| \leq \frac{n}{2}$ is $\mathcal{O}(k(2r - 1)^{-n/2})$.*

Proof. The set of words of length h is $2r(2r - 1)^{h-1}$, so the set of words of length at most h is

$$1 + 2r + 2r(2r - 1) + \cdots + 2r(2r - 1)^{h-1} = \frac{r}{r - 1}(2r - 1)^h - \frac{1}{r - 1}.$$

The probability that a word in $F(A)$ of length at most n actually has minimal length at most $\frac{n}{2}$ is therefore asymptotically equivalent to

$$C(2r-1)^{n/2-n} = C(2r-1)^{-n/2}$$

for some constant C, and the result follows.   □

We are interested in the $d(n)$-ctp, where $d(n)$ is an increasing function of n. The following statement is derived from [2].

Proposition 3.5. *Let* $r = |A| \geq 2$ *and let* $k \geq 2$ *be an integer. Let* $d(n)$ *be a non-decreasing function of n such that* $d(n) < \frac{n}{2}$. *A random k-tuple of words in $F(A)$ of length at most n fails the $d(n)$-ctp with probability* $\mathcal{O}(k^2(2r-1)^{-d(n/2)})$.

Proof. Let $\eta_n = k^2(2r-1)^{-d(n/2)}$. It is shown in [2, proof of Proposition 3.17][1] that the probability that a k-tuple $\vec{w} = (w_1, \dots, w_k)$ of words in $F(A)$ of length at most n fails to have the $d(n)$-ctp is bounded above by the sum of $5\eta_n$ and the probability that $k^2(2r-1)^{-d(\min|\vec{w}|)} > \eta_n$.

We note that, if $\min|\vec{w}| > \frac{n}{2}$, then for all n, we have

$$k^2(2r-1)^{-d(\min|\vec{w}|)} \leq k^2(2r-1)^{-d(n/2)} = \eta_n.$$

Therefore, the probability that $\vec{w}$ fails to have the $d(n)$-ctp is bounded above by the sum of the probability that $\min|\vec{w}| \leq \frac{n}{2}$, which is $\mathcal{O}(k(2r-1)^{-n/2})$ by Proposition 3.4, and $5\eta_n = 5k^2(2r-1)^{-d(n/2)}$. This concludes the proof since $k < k^2$ and $d(\frac{n}{2}) < \frac{n}{4}$.   □

If the size k of the tuple of words is itself a function of n, Proposition 3.5 directly yields the following statement.

Corollary 3.6. *Let* $r = |A| \geq 2$ *and let* $k(n)$ *be an integer function such that* $k(n) \leq (2r-1)^{n/2}$.

(i) *If* $k(n)$ *is a constant function, then a random $k(n)$-tuple of words in $F(A)$ of length at most n fails the $\log n$-ctp with probability* $\mathcal{O}(n^{-\log(2r-1)})$.

(ii) *If* $\theta > 0$, $k(n) = n^\theta$ *and* $0 < \gamma < 1$, *then a random $k(n)$-tuple of words in $F(A)$ of length at most n fails the $(2n)^\gamma$-ctp with probability*

$$\mathcal{O}(n^{2\theta}(2r-1)^{-n^\gamma}).$$

---

[1] Proposition 3.17 in [2] is formulated in the case of a so-called prefix-heavy distribution on words of fixed length. This is the case for the distribution used here, namely the uniform distribution on reduced words of fixed length. The parameters C and α in [2, Proposition 3.17] are, respectively, 1 and $(2r-1)^{-1}$; see [2, Example 3.2].

(iii) *At density β (that is, if $k(n) = (2r-1)^{\beta n}$) for $0 < \beta < \frac{1}{8}$, and if $4\beta < \gamma < \frac{1}{2}$, then a random $k(n)$-tuple of words in $F(A)$ of length at most n fails the γn-ctp with probability $\mathcal{O}((2r-1)^{(4\beta-\gamma)n/2})$.*

4 The Uniform Membership Problem

The *Uniform Membership Problem* (UMP) on alphabet A and for an integer $k \geq 1$ is the following: given w_0, a word in $F(A)$, and $\vec{w} = (w_1, \ldots, w_k)$, a k-tuple of words in $F(A)$, decide whether w_0 belongs to the subgroup H generated by $\vec{w}$.

The notion of Stallings graphs provides a nice algorithmic solution for the UMP, and to its extension where we also ask for the expression of w_0 in a basis of H, if $w_0 \in H$ (see [20]).

Algorithm $\mathcal{MP}$. On input a pair $(w_0, \vec{w})$ of a reduced word and a k-tuple of words in $F(A)$,

(1) compute the Stallings graph $\Gamma(H)$ of $H = \langle \vec{w} \rangle$.

(2) Compute a spanning tree T of $\Gamma(H)$ (which specifies a basis B of H; see Section 2.1).

(3) Try reading w_0 as a label of a path in $\Gamma(H)$ starting at the root vertex, keeping track of the sequence of edges traversed in the complement of T. If one can indeed read w_0 in this fashion and the resulting path is a circuit, then $w_0 \in H$ and the sequence of edges not in T yields the reduced expression of w_0 in basis B (see Section 2.1); otherwise $w_0 \notin H$.

Remark 4.1. We do not assume the length k of the tuple $\vec{w}$ to be a constant. We will see it instead as a function of $n = \max|\vec{w}|$. $\square$

Proposition 4.2. *The (worst case) complexity of Algorithm $\mathcal{MP}$ is*

$$\mathcal{O}(kn \log^*(kn) + rkn + m),$$

where $n = \max|\vec{w}|$, $m = |w_0|$, and $r = |A|$.

Proof. This is a direct application of Section 2.1. $\square$

Remark 4.3. If we only want to know whether $w_0 \in H$, step (2) can be skipped, and the complexity is $\mathcal{O}(kn \log^*(kn) + m)$. $\square$

Let us now consider Algorithm $\mathcal{MP}$ in more detail, in the case when $\vec{w}$ has the d-ctp for some d. In this situation, we can exploit the shape of $\Gamma(H)$ described in Section 3. In particular, we get a spanning tree by removing one edge from the

$\mathsf{mf}_d(w_i)$-labeled path from pr_{-i} to pr_i for each $1 \leq i \leq k$, and the corresponding basis is $\vec{w}$. It does not, actually, matter which edge is removed. As we will see, we do not even need to compute explicitly the full picture of $\Gamma(H)$.

Let X be the k-letter alphabet $X = \{x_1, \ldots, x_k\}$ and let $\varphi \colon F(X) \to F(A)$ be the morphism which maps letter x_i to w_i, as in the proof of Proposition 3.2.

If $w_0 \in H$, then step (3) of Algorithm $\mathcal{MP}$ starts with an *initialization step*, identifying the first letter x_{i_1} of the expression x_0 of w_0 in basis $\vec{w}$ and reading $|w_{i_1}| - d$ letters of w_0, followed by a potentially *iterated step*, which identifies the next letter in x_0. The important observation is that, along each of these steps, a long factor of w_0 must be read (of length at least $\min|\vec{w}| - 2d$), and this factor must match one of a fixed collection of at most $2k$ words.

This leads to the family of Algorithms $\mathcal{MP}_d$ below, indexed by functions $d \colon \mathbb{N} \to \mathbb{N}$, $n \mapsto d(n)$, each of which solves the UMP. We then prove that, for well-chosen d, the average case complexity of Algorithm $\mathcal{MP}_d$ is much lower than the worst case complexity of Algorithm $\mathcal{MP}$.

Let $d(n)$ be a non-decreasing function of n such that $d(n) < \frac{n}{2}$.

Algorithm $\mathcal{MP}_d$. The input is the $(k + 1)$-tuple $(w_0, \ldots, w_k)$ of words in $F(A)$. We let $\vec{w} = (w_1, \ldots, w_k)$ and $H = \langle \vec{w} \rangle$. No assumption (in particular, no ctp assumption) is made about the input. For convenience, we assume that we are also given the k-tuple of lengths $(|w_1|, \ldots, |w_k|)$ (see Remark 2.5), and we let $n = \max|\vec{w}|$. We again let $X = \{x_1, \ldots, x_k\}$ and $\varphi \colon F(X) \to F(A)$ be given by $\varphi(x_i) = w_i$ $(1 \leq i \leq k)$.

(1) Decide whether $\vec{w}$ has the $d(n)$-ctp and $\min|\vec{w}| > \frac{n}{2}$. This decision requires computing the set $L(\vec{w})$ of length $d(n)$ prefixes of the w_i and w_i^{-1}. This set is recorded in the form of the tree $\Gamma_{d(n)}(\vec{w})$, which has at most $2kd(n)$ vertices and edges. There are two cases.

  (a) If $\vec{w}$ has the $d(n)$-ctp and $\min|\vec{w}| > \frac{n}{2}$, go to step (2).

  (b) Otherwise, run Algorithm $\mathcal{MP}$ to decide whether $w_0 \in H$, and find an expression of w_0 in a basis of H if it does.

(2) Start reading w_0 in $\Gamma_{d(n)}(\vec{w})$ from the root vertex. There are two cases.

  (a) We reach a leaf of $\Gamma_{d(n)}(\vec{w})$, say pr_{-i} $(-k \leq i \leq k, i \neq 0$, necessarily after reading exactly $d(n)$ letters from w_0), and the middle factor $\mathsf{mf}_{d(n)}(w_i)$ is a proper prefix of the suffix of w_0 starting in position $d(n) + 1$, that is, $\mathsf{pr}_{-i}\mathsf{mf}_{d(n)}(w_i)$ is a proper prefix of w_0. In this case, move the reading head to position $d(n) + |\mathsf{mf}_{d(n)}(w_i)| + 1 = |w_i| - d(n) + 1$ in w_0, record pr_i as the last-leaf-visited, output letter $x_i \in \check{X}$ and go to step (3).

  (b) Otherwise, stop the algorithm and conclude that $w_0 \notin H$.

(3) Suppose that the reading head on w_0 is in position j and that the last-leaf-visited is pr_i. Resume reading w_0 (from position j) in $\Gamma_{d(n)}(\vec{w})$, starting at pr_i. There are three cases.

   (a) We reach the end of w_0 while reading it inside $\Gamma_{d(n)}(\vec{w})$, landing at the root vertex. In this case, stop the algorithm and conclude that $w_0 \in H$.

   (b) After reading d' letters from w_0, we reach a leaf of $\Gamma_{d(n)}(\vec{w})$, say $\mathsf{pr}_{-i'}$ (necessarily $2 \leq d' \leq 2d(n)$ and $-i' \neq i$), and the word $\mathsf{mf}_{d(n)}(w_{i'})$ is a proper prefix of the suffix of w_0 starting in position $j + d' + 1$. In this case, move the reading head to position $j + d' + (|w_{i'}| - 2d(n)) + 1$ in w_0, record $\mathsf{pr}_{i'}$ as the last-leaf-visited, output letter $x_{i'} \in \tilde{X}$ and repeat step (3).

   (c) Otherwise, stop the algorithm and conclude that $w_0 \notin H$.

Theorem 4.4. *Let $d(n)$ be a non-decreasing function of n such that $d(n) < \frac{n}{2}$. Algorithm $\mathcal{MP}_d$ solves the Uniform Membership Problem in $F(A)$ and, if $w_0 \in H$, finds an expression of w_0 in a basis of H.*

Let $r = |A| \geq 2$, $0 < \delta' < \frac{1}{4}$ and $0 < \beta' < \frac{1}{2} - 2\delta'$, and suppose $d(n) \leq \delta'n$. If we restrict the input space to pairs of the form $(w_0, \vec{w})$, where $\max|\vec{w}| = n$ and $\vec{w}$ is a tuple of length $k \leq (2r-1)^{\beta'n}$, then the average case complexity of Algorithm $\mathcal{MP}_d$ is

$$\mathcal{O}\left(kd(n) + k^3n(2r-1)^{-d(n/2)}(r + \log^*(kn)) + k^2(2r-1)^{-d(n/2)}m\right),$$

where $m = |w_0|$. If the space of inputs is further restricted to those inputs where $\vec{w}$ has the $d(n)$-ctp and $\min|\vec{w}| > \frac{n}{2}$, then the expected running time is $\mathcal{O}(kd(n))$, independent of $|w_0|$.

Proof. Algorithm $\mathcal{MP}_d$ always stops because every one of its steps takes a finite amount of time and the only repeated step (step (3)) reads a positive number of letters of w_0. If Algorithm $\mathcal{MP}_d$ stops at step (1), then it answers the question whether $w_0 \in H$ and, in the affirmative case, finds an expression for it in a basis of H. If it stops at step (2), then $w_0 \notin H$. And if it stops at step (3), then Algorithm $\mathcal{MP}_d$ outputs a word x_0 on alphabet $\tilde{X}$, one letter at a time (one at the completion of step (2) and one at the completion of each iteration of step (3) except for the last one). As observed in the description of step (3), each new letter cannot be the inverse of the preceding one (because $\Gamma_{d(n)}(\vec{w})$ is a tree) so that word x_0 is always reduced, that is, $x_0 \in F(X)$. Moreover, the last iteration of step (3) concludes either that $w_0 \notin H$, or that $w_0 \in H$ and $x_0 = \varphi^{-1}(w_0)$ (that is, x_0 is the expression of w_0 in basis $\vec{w}$).

We now proceed with bounding the expected running time of Algorithm $\mathcal{MP}_d$. We use the following notation:

$$\mu = \min|\vec{w}|,$$
$$\mathfrak{p} = 2k(2r-1)^{-n/2+d(n)},$$
$$\mathfrak{q} = 2k(2r-1)^{-2-n/2+2d(n)} = (2r-1)^{d(n)-2}\mathfrak{p}.$$

Our hypotheses on k and d imply that both $\mathfrak{p}$ and $\mathfrak{q}$ tend to 0 as n tends to infinity.

Step (1) first requires comparing the lengths of $w_1, \ldots, w_k$ with $\frac{n}{2}$, deciding whether $\vec{w}$ has the $d(n)$-ctp and, if so, computing $\Gamma_{d(n)}(\vec{w})$. This takes time $\mathcal{O}(kd(n))$ (see Remark 3.1). If $\vec{w}$ does not have the $d(n)$-ctp or if $\mu \leq \frac{n}{2}$, step (1) runs Algorithm $\mathcal{MP}$, in time $\mathcal{O}(kn\log^*(kn) + rkn + m)$ (see Proposition 4.2). By Propositions 3.4 and 3.5, this happens with probability $\mathcal{O}(k^2(2r-1)^{-d(n/2)})$ (since we assumed that $d(n) < \frac{n}{2}$, and hence $k(2r-1)^{-n/2} < k^2(2r-1)^{-d(n/2)}$).

With the complementary probability, $\vec{w}$ has the $d(n)$-ctp, $\mu > \frac{n}{2}$ and Algorithm $\mathcal{MP}_d$ proceeds to step (2).

We now need to decide in which of the two cases of step (2) we are, that is, we need to solve the PPP (Proper Prefix Problem) $2k$ times: for every pair of input words (u, w_0), where $u = \text{pr}_{-i}\text{mf}_{d(n)}(w_i)$ for some $-k \leq i \leq k, i \neq 0$. The expected time for this is $\mathcal{O}(k)$ (see Lemma 2.7). Note that, by the ctp, the output will be positive for at most one of these u, thus uniquely identifying the leaf pr_{-i} of $\Gamma_{d(n)}(\vec{w})$ which is first visited when reading w_0. Moreover, the probability that the algorithm does not stop here – and therefore moves to step (3) –, that is, the probability that one of these words u is indeed a proper prefix of w_0 is[2] $\mathcal{O}(2k(2r-1)^{-\mu+d(n)})$. As $\mu > \frac{n}{2}$, this probability is $\mathcal{O}(\mathfrak{p})$.

Thus, with probability $\mathcal{O}(\mathfrak{p})$, we enter a loop where step (3) is repeated. Consider one such iteration of step (3), starting with the reading head in position j on w_0 and vertex pr_i as the last-leaf-visited. Let w_0' be the suffix of w_0 starting at position j. To decide in which of the cases of step (3) we are, we first consider whether $|w_0| = j + d(n)$, and if so, we solve the EqP (Equality Problem) on input (pr_i^{-1}, w_0'). If indeed $w_0' = \text{pr}_i^{-1}$, the algorithm stops and concludes that $w_0 \in H$. This is done in constant expected time. If $w_0' \neq \text{pr}_i^{-1}$, we solve the PPP $2k - 1$ times for every input pair (u, w_0'), where $u = \text{pr}_i^{-1}\text{pr}_{-i'}\text{mf}_{d(n)}(w_{i'})$ and $-k \leq i' \leq k, i' \neq 0, -i$. By Lemma 2.7, this is done in expected time $\mathcal{O}(kd(n))$ (the factor $d(n)$ corresponds to the work needed to reduce $\text{pr}_i^{-1}\text{pr}_{-i'}$ before solving the PPP). The probability that the algorithm continues to a new iteration of step (3), namely the probability that one of these words u is a proper prefix of w_0',

---

[2] The probability that u (of length ℓ) is a prefix of w_0 (of length m) is $\mathcal{O}((2r-1)^{-\ell})$. In the present situation, $\ell = |w_i| - d(n) \geq \mu - d(n)$.

is $\mathcal{O}((2k-1)(2r-1)^{-(2+\mu-2d(n))})$. Since $\mu > \frac{n}{2}$, we have

$$(2k-1)(2r-1)^{-(2+\mu-2d(n))} \leq \mathfrak{q}.$$

The expected time required for running Algorithm $\mathcal{MP}_d$ can be analyzed as follows. Step (1) runs in expected time

$$\mathcal{O}(kd(n)) + \mathcal{O}(k^2(2r-1)^{-d(n/2)}(kn\log^*(kn) + rkn + m))$$
$$= \mathcal{O}(kd(n) + k^3n(2r-1)^{-d(n/2)}(r + \log^*(kn)) + k^2(2r-1)^{-d(n/2)}m).$$

With probability $1 - \mathcal{O}(k^2(2r-1)^{-d(n/2)})$, Algorithm $\mathcal{MP}_d$ proceeds to step (2).

Step (2) runs in expected time $\mathcal{O}(k)$. With probability $\mathcal{O}(\mathfrak{p})$, the algorithm proceeds to step (3) and stops with the complementary probability.

Each iteration of step (3) runs in expected time $\mathcal{O}(kd(n))$. Step (3) is repeated with probability $\mathcal{O}(\mathfrak{q})$, and the algorithm stops with the complementary probability.

It follows that the expected running time of step (2) and the ensuing iterations of step (3) is $\mathcal{O}(k(1 + \mathfrak{p}d(n)(1 + \mathfrak{q} + \mathfrak{q}^2 + \mathfrak{q}^3 + \cdots)))$, which is

$$\mathcal{O}\left(kd(n)\left(1 + \frac{\mathfrak{p}}{1-\mathfrak{q}}\right)\right).$$

Since $\mathfrak{p}$ and $\mathfrak{q}$ tend to 0, this is $\mathcal{O}(kd(n))$, independently of how many times step (3) is iterated. The expected running time of Algorithm $\mathcal{MP}_d$ is therefore at most

$$\mathcal{O}\left(kd(n) + k^3n(2r-1)^{-d(n/2)}\log^*(kn) + k^2(2r-1)^{-d(n/2)}m\right).$$

Finally, suppose that the input $(w_0, \vec{w})$ is such that $\min|\vec{w}| > \frac{n}{2}$ and $\vec{w}$ has the $d(n)$-ctp. Then step (1) consists only in computing $\Gamma_{d(n)}(\vec{w})$. The expected running time of Algorithm $\mathcal{MP}_d$ is therefore, on this smaller set of inputs, $\mathcal{O}(kd(n))$.

□

As a corollary, we get upper bounds on the average case complexity of the Uniform Membership Problem.

Corollary 4.5. *The Uniform Membership Problem (UMP) for $F(A)$, with input a $k(n)$-tuple of words of length at most n, and an additional word of length m, can be solved in expected time $C(n, m)$ as follows (where $r = |A|$ is taken to be constant).*

(1) *If k is constant, then $C(n, m) = \mathcal{O}(\log n + mn^{-\log(2r-1)})$, improving on its worst case complexity, namely $\mathcal{O}(n\log^* n + m)$.*

(2) *Let $\beta > 0$, $0 < \gamma < 1$. If $k = n^\beta$, then*

$$C(n,m) = \mathcal{O}(n^{\beta+\gamma} + mn^{2\beta}(2r-1)^{-n^\gamma}),$$

improving on its worst case complexity, namely $\mathcal{O}(n^{\beta+1}\log^* n + m)$.

(3) *If $k = n^\beta$ for some $\beta > 0$, we also have*

$$C(n,m) = \mathcal{O}(n^\beta \log n + n\log^* n + mn^{-\beta}).$$

(4) *For any $0 < \beta < \frac{1}{18}$, if $k = (2r-1)^{\beta n}$, then for every $0 < \varepsilon < \frac{1}{8} - \frac{9}{4}\beta$,*

$$C(n,m) = \mathcal{O}(n(2r-1)^{\beta n} + m(2r-1)^{(\frac{9}{4}\beta - \frac{1}{8} + \varepsilon)n}),$$

improving on its worst case complexity, namely $\mathcal{O}(n(2r-1)^{\beta n}\log^* n + m)$.

Proof. The worst case complexities mentioned in each item follow from Proposition 4.2. For every item, we apply Theorem 4.4 for an appropriate choice of the function $d(n)$.

(1) Suppose that k is a constant function and let $d(n) = \log n$. Note that

$$(2r-1)^{\log n} = n^{\log(2r-1)} \quad \text{and} \quad \lim_{n\to\infty} n^{1-\log(2r-1)}(r + \log^*(kn)) = 0.$$

Theorem 4.4 then shows that the average case complexity of Algorithm $\mathcal{MP}_d$ is $\mathcal{O}(\log n + mn^{-\log(2r-1)})$, as stated.

(2) Suppose now that $k = n^\beta$ and let $d(n) = (2n)^\gamma$. We can, again, apply Theorem 4.4. Since $\lim_{n\to\infty} n^{3\beta+1}(2r-1)^{-n^\gamma}(r + \log^*(n^{\beta+1})) = 0$, the average case complexity of Algorithm $\mathcal{MP}_d$ is $\mathcal{O}(n^{\beta+\gamma} + mn^{2\beta}(2r-1)^{-n^\gamma})$, as stated.

(3) Suppose, again, that $k = n^\beta$ and let

$$d(n) = \frac{3\beta}{\log(2r-1)}\log(2n).$$

Then $(2r-1)^{-d(n/2)} = n^{-3\beta}$. Then Theorem 4.4 shows that, in this case, the average case complexity of Algorithm $\mathcal{MP}_d$ is $\mathcal{O}(n^\beta \log n + n\log^* n + mn^{-\beta})$.

(4) Suppose that $k = (2r-1)^{\beta n}$ with $0 < \beta < \frac{1}{18}$. This inequality guarantees that $4\beta < \frac{1}{4} - \frac{\beta}{2}$. Let $0 < \varepsilon < \frac{1}{8} - \frac{9}{4}\beta$ and $\delta = \frac{1}{4} - \frac{\beta}{2} - 2\varepsilon$ (so that $\delta > 4\beta$), and let $d(n) = \delta n$. Then the hypotheses of Theorem 4.4 are satisfied with $\beta' = \beta$ and $\delta' = \delta$. As a result, the average case complexity of Algorithm $\mathcal{MP}_d$ is

$$\mathcal{O}\big(n(2r-1)^{\beta n} + n(2r-1)^{(3\beta - \frac{\delta}{2})n}\log^* n + m(2r-1)^{(2\beta - \frac{\delta}{2})n}\big).$$

Since $4\beta < \delta$, we have $3\beta - \frac{\delta}{2} < \beta$, and the second summand is less than the first. Moreover, $2\beta - \frac{\delta}{2} = \frac{9}{4}\beta - \frac{1}{8} + \varepsilon$, and the stated result follows. $\qquad\square$

5   The Primitivity and the Relative Primitivity Problems

An element w of a free group $F(A)$ is said to be *primitive* (in $F(A)$) if $F(A)$ admits a basis containing w. Equivalently, w is primitive if the cyclic subgroup $\langle w \rangle$ is a free factor of $F(A)$. The *Primitivity Problem* (PrimP) on alphabet A consists in deciding, given a word $w \in F(A)$, whether w is primitive in $F(A)$.

The Primitivity Problem is closely related to the following Whitehead problem: given two words $v, w \in F(A)$, decide whether there exists an automorphism φ of $F(A)$ such that $\varphi(w) = v$. The first step in Whitehead's classical solution to this problem [23] identifies the minimal length of the automorphic images of v and w. The classical solution of PrimP is a by-product of this first step: a word is primitive if and only if its orbit under the action of $\mathsf{Aut}(F(A))$ contains a word (and so all words) of length 1. This solution of PrimP is linear in $m = |w|$, but exponential in $r = |A|$ (relying, as it does, on an exploration of the action of the Whitehead automorphisms, whose number is exponential in r).

Roig, Ventura and Weil [17, Fact 3.6] modified Whitehead's algorithm, resulting in Algorithm $\mathcal{P}$ which solves the Primitivity Problem in time $\mathcal{O}(m^2 r^3)$, where $m = |w|$. We do not describe Algorithm $\mathcal{P}$ in this paper as we will use it as a black box.

The *Relative Primitivity Problem* (RPrimP) on alphabet A and for an integer $k \geq 1$ is the following: given a word w_0 in $F(A)$ and a k-tuple $\vec{w} = (w_1, \ldots, w_k)$ of words in $F(A)$, decide whether w_0 belongs to $H = \langle \vec{w} \rangle$ and, if it does, whether it is primitive in H.

Solving RPrimP is done naturally by the combination of an algorithm solving the Uniform Membership Problem and, in the case of affirmative answer, computing the expression x_0 of w_0 in a basis B of H, and applying an algorithm for solving PrimP in $F(B)$ (for example, Algorithm $\mathcal{P}$ mentioned above, with worst case complexity $\mathcal{O}(m^2 r^3)$). By Proposition 4.2, this results in a worst case complexity of

$$\mathcal{O}\big(kn \log^*(kn) + rkn + m^2 k^3\big)$$

(since the rank of H is at most k).

Recently, Shpilrain [19] gave an algorithm solving PrimP in $F(A)$ with constant average case complexity. This constant average case complexity assumes, as we have done so far, that the rank r of the ambient free group $F(A)$ is fixed. However, we cannot make this assumption anymore since we need to solve PrimP in free subgroups of $F(A)$, whose rank may be as large as k. We therefore revisit this algorithm in detail in Section 5.1, and we recompute its average case complexity to ascertain its dependency in r. The average case complexity of the combination of this algorithm with Algorithm $\mathcal{MP}_d$ (Section 4) is discussed in Section 5.2.

5.1   Shpilrain's primitivity algorithm

Recall that a word u in $F(A)$ is *cyclically reduced* if its last letter is not the inverse of its first letter, that is, if u^2 is reduced. It is clear that any word u factors in a unique fashion as $u = vwv^{-1}$ with w cyclically reduced, and we call w the *cyclic core* of u, written $\kappa(u)$. It is immediate that u is primitive if and only if $\kappa(u)$ is.

If $u = x_1 \cdots x_n$ is a reduced word of length at least 2, let $W(u)$ be the *Whitehead graph of u*, namely the simple (undirected) graph on vertex set $\tilde{A}$, with an edge from vertex x to vertex y if there exists $1 \le i \le n$ such that $x_i x_{i+1} = xy^{-1}$ or yx^{-1} (here x_{n+1} stands for x_1). Observe that $W(u)$ can be constructed one edge at a time when reading u from left to right, in time $\mathcal{O}(|u|)$.

Recall finally that a vertex p of a connected graph G is a *cut vertex* if deleting p from G (and all the edges adjacent to p) results in a disconnected graph. Whitehead showed the following [22].

Proposition 5.1. *Let u be a cyclically reduced word of length at least 2 in $F(A)$. If u is primitive, then either $W(u)$ is disconnected, or $W(u)$ admits a cut vertex.*

Shpilrain's Algorithm $\mathscr{S}$ (slightly modified) is as follows [19]. We let

$$ g(n) = n - \frac{1}{\log(2r - 1)} \log(n^4 r^6). $$

Algorithm $\mathscr{S}$. On input a reduced word $u \in F(A)$ (together with its length n),

(1) compute $\kappa(u)$, the cyclic core of u, say $\kappa(u) = x_1 \cdots x_h$. If $h = |\kappa(u)| \le g(n)$, go to step (4). Otherwise, let $i = 2$, let W be the graph with vertex set $\tilde{A}$ and no edges (so that every vertex is its own connected component) and go to step (2).

(2) Read x_i, add the edge (x_{i-1}, x_i^{-1}) to W and update the list of connected components of W. If W is connected and has no cut vertex, stop the algorithm: u is not primitive in $F(A)$. Otherwise, if $i < h$, increment i by a unit and repeat step (2), and if $i = h$, go to step (3).

(3) Add to W the edge (x_h, x_1^{-1}) and update the list of connected components of W. If W is connected and has no cut vertex, stop the algorithm: u is not primitive in $F(A)$. Otherwise, go to step (4).

(4) Run Algorithm $\mathscr{P}$ on $\kappa(u)$ to decide whether u is primitive in $F(A)$.

Algorithm $\mathscr{S}$ certainly solves the Primitivity Problem (using Proposition 5.1) since the graph W constructed in steps (2) and (3) is an increasingly larger fragment of $W(\kappa(u))$. The algorithm stops when either W is connected and has no

cut vertex, in which case $W(\kappa(u))$ has the same property and u is therefore not primitive in $F(A)$, or when Proposition 5.1 has failed to give us an answer and Algorithm $\mathcal{P}$ has been called to settle the issue.

Shpilrain showed in [19] that the average case complexity of Algorithm $\mathcal{S}$ is bounded above by a constant, independent of the length of the input word. This constant does however depend on the ambient rank r, and we specify this dependency in Proposition 5.5 below. Before we state this proposition, we need to record a few results.

First recall that the number R_n of reduced words of length n in $F(A)$ is

$$2r(2r-1)^{n-1}.$$

The number CR_n of cyclically reduced words of length n satisfies

$$2r(2r-1)^{n-2}(2r-2) \leq \mathrm{CR}_n \leq 2r(2r-1)^{n-1} = R_n.$$

It follows that the probability that a reduced word is not cyclically reduced is at most $1 - \frac{2r-2}{2r-1} = \frac{1}{2r-1}$ (not exactly $\frac{1}{2r}$, as asserted in [19], because the first and last letter of a reduced word are random variables that are close to but not exactly independent from each other).

The following (obvious!) algorithm computes the cyclic core of a word in $F(A)$.

Algorithm $\mathcal{CR}$. On input a reduced word $u = a_1 \cdots a_n$ of length n and as long as $n \geq 3$, compare a_n with a_1^{-1}; if they are equal, delete the first and last letter of u and repeat this step; if they are different, return the word u.

Lemma 5.2. *The average case complexity of Algorithm $\mathcal{CR}$ is $\mathcal{O}(1)$, independent of the size r of the alphabet.*

Proof. Let $\mathfrak{p}_n$ be the probability that a length n reduced word is not cyclically reduced. As observed before, $\mathfrak{p}_n \leq \frac{1}{2r-1}$.

Every step of Algorithm $\mathcal{CR}$ compares two letters from $\tilde{A}$, and hence takes constant time C. On input u, of length n, Algorithm $\mathcal{CR}$ concludes in 1 step (that is, the case where u is cyclically reduced) with probability $1 - \mathfrak{p}_n$, and otherwise repeats its single step, on a length $n - 2$ input. Thus the expected time is bounded above by

$$(1 + \mathfrak{p}_n + \mathfrak{p}_n\mathfrak{p}_{n-2} + \mathfrak{p}_n\mathfrak{p}_{n-2}\mathfrak{p}_{n-4} + \cdots)\, C \leq \left(\sum_{i \geq 0}(2r-1)^{-i}\right) C.$$

Since $\sum_{i \geq 0}(2r-1)^{-i} = \frac{2r-1}{2r-2} \leq \frac{3}{2}$, this concludes the proof. $\quad\square$

We will also use the following fact.

Lemma 5.3. *The probability that the cyclic core of a length n element of $F(A)$ has length less than or equal to $n - 2\ell$ is $\mathcal{O}((2r - 1)^{-\ell})$.*

Proof. This is a side product of the proof of Lemma 5.2: as observed there, Algorithm $\mathcal{CR}$ concludes in 1 step with probability $1 - p_n \leq 1$. It concludes in exactly 2 steps with probability $p_n(1 - p_{n-2}) \leq (2r - 1)^{-1}$, and it concludes in $h + 1$ steps with probability $p_n p_{n-2} \cdots p_{n-2h+2}(1 - p_{n-2h}) \leq (2r - 1)^{-h}$. Now, $\kappa(u)$ has length $n - 2h$ if and only if Algorithm $\mathcal{CR}$ terminates in $h + 1$ steps. So $|\kappa(u)| \leq n - 2\ell$ if Algorithm $\mathcal{CR}$ terminates in at least $\ell + 1$ steps, and this happens with probability at most $(2r - 1)^{-\ell} \sum_i (2r - 1)^{-i}$. This quantity is

$$\frac{2r - 1}{2r - 2}(2r - 1)^{-\ell} \leq \frac{3}{2}(2r - 1)^{-\ell}. \qquad \square$$

An important observation is that if u is a random word in $F(A)$ of length n, then with high probability, $W(u)$ is connected and has no cut vertex. More precisely, the following holds. If $u = x_1 \cdots x_n$ is a reduced word of length at least 2, let $W'(u)$ be the simple graph with vertex set $\tilde{A}$, and with an edge from vertex x to vertex y if there exists $1 \leq i < n$ such that $x_i x_{i+1} = xy^{-1}$ or yx^{-1}. Note that this is almost identical to the definition of the Whitehead graph $W(u)$, except that we do not consider the case where $i = n$. In particular, $W'(u)$ is a subgraph of $W(u)$, and if $W'(u)$ is connected and has no cut vertex, then the same property holds for $W(u)$.

Proposition 5.4. *Let $r \geq 2$ and let $F = F(A)$, with $|A| = r$. There exists a positive number $\alpha(r) < 1 - \frac{1}{2}r^{-2}$ with the following property: the probability for a word u of length n in $F(A)$ that $W'(u)$ is disconnected, or is connected and has a cut vertex, is $\Theta(n^k \alpha(r)^n)$ for some $k \in \mathbb{N}$.*

Proof. Let $\mathcal{G}$ be the set of simple graphs on vertex set $\tilde{A}$ (that is, undirected loop-free graphs without multiple edges). If $G \in \mathcal{G}$, let $\mathcal{A}(G)$ be the $\tilde{A}$-automaton with the same vertex set, whose edges are as follows: for every edge of G connecting vertices a and b ($a, b \in \tilde{A}$), $\mathcal{A}(G)$ has a b^{-1}-labeled edge from state a to state b^{-1} and an a^{-1}-labeled edge from state b to state a^{-1}. Let also $L(G)$ be the set of all words in $\tilde{A}^*$ which label a path in $\mathcal{A}(G)$, with no condition on its starting and ending points; in particular, $L(G)$ is a regular language. Finally, let $M(G)$ be the transition matrix of $\mathcal{A}(G)$, that is, the order $2r$ matrix whose (a, b)-entry is 1 if $\mathcal{A}(G)$ has an edge from vertex a to vertex b, and 0 otherwise.

For $G, G' \in \mathcal{G}$, say that $G \leq G'$ if every edge of G is also an edge of G'. It is clear that if $G \leq G'$ and G is connected and has no cut vertex, then the same holds for G'. Let $G_1, \ldots, G_h$ be the $\leq$-maximal elements of $\mathcal{G}$ which are either

disconnected, or connected and with a cut vertex. Let λ_i be the growth modulus of $L(G_i)$; then the union of the $L(G_i)$ has growth modulus $\lambda_0 = \max_{1 \leq i \leq k} \lambda_i$.

Now let X denote the set of reduced words u such that $W'(u)$ is disconnected, or is connected and has a cut vertex. Observe that if $u \in L(G_i)$, then $W'(u) \leq G_i$, and hence $L(G_i)$ is contained in X. That is, X contains the union of the $L(G_i)$ ($i \in [1, h]$). Moreover, for every word u with first letter a, we have $u \in aL(W'(u))$ so that X is contained in the union of the $aL(G_i)$ ($a \in \tilde{A}$ and $i \in [1, h]$). It is easily verified that $aL(G_i)$ and G_i have the same growth rate, so both unions have growth rate λ_0, and hence so does X.

Thus the number of length n words in X is asymptotically equivalent to $Cn^k \lambda_0^n$, where $C > 0$ and $k \in \mathbb{N}$ (see Fact 2.1). As the growth modulus of the language of all reduced words is $2r - 1$ and $\alpha(r)$ was defined to be equal to $\frac{\lambda_0}{2r-1}$, it follows that the probability that a length n word is in X is asymptotically equivalent to $Cn^k \alpha(r)^n$.

In order to conclude the proof, we need to establish an explicit upper bound for $\alpha(r)$, as a function of r. This is done rather abruptly (following reasoning similar to that in [5]): for each $a, b \in \tilde{A}$ with $b \neq a, a^{-1}$, let $G_{a,a}$ be obtained from the maximum element of $\mathscr{G}$ (which has an edge between every pair of distinct vertices in $\tilde{A}$) by deleting the edge between a and a^{-1}, and let $G_{a,b}$ be obtained from the same maximum element by deleting the edge between a and b (Figure 1). Then every G_i satisfies either $G_i \leq G_{a,a}$ or $G_i \leq G_{a,b}$ for some $a, b \in \tilde{A}$. In particular, $L(G_i)$ is contained in $L(G_{a,a})$ or $L(G_{a,b})$, and hence λ_i is less than or equal to the growth modulus of $L(G_{a,a})$ or $L(G_{a,b})$. Since $G_{a,a}$ and $G_{a,b}$ are irreducible and aperiodic, Proposition 2.3 shows that these growth moduli are the leading eigenvalues of $M(G_{a,a})$ or $M(G_{a,b})$, respectively.

It should be clear that these growth moduli do not depend on the choice of $a, b \in \tilde{A}$. Facts A.1 and A.2 from the appendix show that both are at most

$$(2r - 1)\left(1 - \frac{1}{2}r^{-2}\right),$$

which completes the proof.   □

Proposition 5.5. *Let $r \geq 2$. There exists a positive number $\beta(r) < 1 - \frac{1}{2}r^{-2}$ such that the average case complexity of Algorithm 8 is $\mathcal{O}\left(\left(\frac{r}{1-\beta(r)}\right)^2 + r^3\right)$. In particular, this average case complexity is $\mathcal{O}(r^6)$.*

Proof. Let k and $\alpha(r)$ be given by Proposition 5.4, and let $\beta(r)$ be a number such that $\alpha(r) < \beta(r) < 1 - \frac{1}{2}r^{-2}$. Note that $n^k \alpha(r)^n = \mathcal{O}(\beta(r)^n)$.

Step (1) of Algorithm 8 takes constant expected time; see Lemma 5.2. It is a classical result (usually referred to [10]) that connectedness and the presence of a cut vertex in a graph with V vertices and E edges can be decided in time

$$
\begin{pmatrix}
0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\
1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\
1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 0 & 1
\end{pmatrix}
\quad \text{and} \quad
\begin{pmatrix}
1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\
1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 0 & 1
\end{pmatrix}
$$

Figure 1. The transition matrices of $\mathcal{A}(G_{a_1,a_1})$ (on the left) and $\mathcal{A}(G_{a_1,a_2})$ (on the right) for $r = 4$, where the vertex set is $\{a_1, a_{-1}, a_2, a_{-2}, \ldots, a_r, a_{-r}\}$, in that order.

$\mathcal{O}(V + E)$. For the graphs occurring in the algorithm, which are subgraphs of the Whitehead graph $W(u)$, we have

$$
V = 2r \quad \text{and} \quad E \leq 2r(2r - 1),
$$

so $\mathcal{O}(V + E) = \mathcal{O}(r^2)$. It follows that each iteration of step (2) takes time $\mathcal{O}(r^2)$ since W has $2r$ vertices, and the same holds for step (3). Finally, step (4) takes time $\mathcal{O}(m^2 r^3)$, where m is the length of the input word.

By Lemma 5.3, the probability that step (1) directly leads to step (4), that is, the probability that $|\kappa(u)| \leq g(n)$, is

$$
\mathcal{O}((2r - 1)^{-\frac{n - g(n)}{2}}) = \mathcal{O}(n^{-2} r^{-3}).
$$

So the contribution of this configuration to the average case complexity of Algorithm $\mathcal{S}$ is $\mathcal{O}(n^{-2} r^{-3}) \mathcal{O}(g(n)^2 r^3) = \mathcal{O}(1)$.

Let $q < |\kappa(u)|$ and let p be the length $q - 1$ prefix of $\kappa(u)$. Step (2) is iterated at least q times if the graph $W'(p)$ is disconnected, or is connected and has a cut vertex. This happens with probability at most $C\beta(r)^{q-1}$ for some constant $C > 0$.

Thus the expected running time of Algorithm $\mathcal{S}$ is a big-$\mathcal{O}$ of

$$
\sum_q \beta(r)^{q-1} q r^2 + \beta(r)^{g(n)} n^2 r^3 \leq \left(\frac{r}{1 - \beta(r)} \right)^2 + \beta(r)^{g(n)} n^2 r^3.
$$

The inequality above is justified as follows: for $|s| < 1$, we have

$$
\sum_q q s^{q-1} = \frac{d}{ds} \left(\sum_q s^q \right) = \frac{d}{ds} \left(\frac{1}{1 - s} \right) = \frac{1}{(1 - s)^2}.
$$

Moreover, we have

$$\beta(r)^{g(n)}n^2 < \beta(r)^n n^2 < \left(1 - \frac{1}{2r^2}\right)^n n^2.$$

If n is large enough with respect to r, this quantity is less than 1. More precisely, suppose that $\frac{n}{\log n} > 4r^2$. Then

$$\log(\beta(r)^{g(n)}n^2) < 2\log n + n\log\left(1 - \frac{1}{2r^2}\right) < 2\log n - \frac{n}{2r^2} < 0.$$

This concludes the proof that the average case complexity of Algorithm $\mathcal{S}$ is

$$\mathcal{O}\left(\left(\frac{r}{1 - \alpha(r)}\right)^2 + r^3\right).$$

The last part of the statement follows from the observation that

$$\left(\frac{r}{1 - \alpha(r)}\right)^2 < 4r^6. \qquad \square$$

Remark 5.6. The Perron–Frobenius theorem can be invoked to show that the spectral radius of $M(G)$ is less than $2r - 1$ for each $G \in \mathcal{G}$ that is not the clique on $\tilde{A}$. As we saw, we need however an estimate of how much smaller than $2r - 1$ these spectral radii are. The method used in the proof of Proposition 5.4 is far from optimal: we estimate the spectral radius of $M(G)$ for the graphs obtained from the maximum element of $\mathcal{G}$ by removing a single edge. Such graphs are far from being disconnected or having a cut vertex. Any upper bound of the spectral radius of the $M(G)$ where G is disconnected or has a cut vertex would lead to an improvement in the expected running time of Algorithm $\mathcal{S}$. There is considerable scope for such an improvement. $\qquad \square$

5.2 The Relative Primitivity Problem

We finally get to the Relative Primitivity Problem, RPrimP: on input a k-tuple $\vec{w} = (w_1, \ldots, w_k)$ of words in $F(A)$ and a word w_0 in $F(A)$, along with their lengths, decide whether w_0 belongs to $H = \langle \vec{w} \rangle$ and is primitive in it. As indicated earlier, the idea is essentially to combine an Algorithm $\mathcal{MP}_d$, for a fast decision of the Uniform Membership Problem, with Shpilrain's Algorithm $\mathcal{S}$, for a fast decision of the Primitivity Problem, carefully distinguishing between the situations where $\vec{w}$ has good properties (the $d(n)$-ctp for a well-chosen function d, and the fact that $\min|\vec{w}| > \frac{1}{2}\max|\vec{w}|$), which will happen with high probability, and where it does not.

More precisely, consider the following algorithm, parametrized by the choice of a non-decreasing function $d(n)$ such that $d(n) < \frac{n}{2}$.

Algorithm $\mathcal{RP}_d$.

(1) Find out whether $\vec{w}$ has the $d(n)$-ctp and $\min|\vec{w}| > \frac{n}{2}$ (this is the first step of Algorithm $\mathcal{MP}_d$). If one of these properties does not hold, go to step (2). If both do, compute $\Gamma_{d(n)}(\vec{w})$ and go to step (3).

(2) Run Algorithm $\mathcal{MP}$ on input $(w_0, \vec{w})$ to decide whether $w_0 \in H = \langle \vec{w} \rangle$ and, if it does, to compute x_0, the expression of w_0 in a basis B of H. In the latter case, run Algorithm $\mathcal{S}$ on x_0 in $F(B)$, to decide whether x_0 is primitive in $F(B)$, equivalently, whether w_0 is primitive in H.

(3) Run steps (2) and (3), the latter iterated, of Algorithm $\mathcal{MP}_d$ to decide whether $w_0 \in H$ and, if it does, to compute x_0, the expression of w_0 in basis $\vec{w}$. If $w_0 \in H$, run Algorithm $\mathcal{S}$ on x_0 in the rank k free group $H = \langle \vec{w} \rangle$.

We prove the following theorem.

Theorem 5.7. *Let $d(n)$ be a non-decreasing function of n such that $d(n) < \frac{n}{2}$. Then Algorithm $\mathcal{RP}_d$ solves RPrimP.*

Let $r = |A| \geq 2$, $0 < \delta' < \frac{1}{4}$, $0 < \beta' < \frac{1}{2} - 2\delta'$ and $\gamma = (2r-1)^{\frac{2\beta'}{1-4\delta'}-1} < 1$. Suppose that $d(n) \leq \delta'n$ and $k(n) \leq (2r-1)^{\beta'n}$ for every n. If we restrict the input space to pairs of the form $(w_0, \vec{w})$, where $\max|\vec{w}| = n$ and $\vec{w}$ is a tuple of length $k(n)$, then the average case complexity of Algorithm $\mathcal{RP}_d$ is a big-$\mathcal{O}$ of

$$k(n)d(n) + k(n)^2(2r-1)^{-d(n/2)}\big(k(n)n\log^*(k(n)n) + m + k(n)^6\big) + \gamma^m k(n)^6,$$

where $m = |w_0|$.

If the input $(w_0, \vec{w})$ of RPrimP is limited to those pairs where $\vec{w}$ has the $d(n)$-ctp and $\min|\vec{w}| > \frac{n}{2}$, then the average case complexity of Algorithm $\mathcal{RP}_d$ is

$$\mathcal{O}\big(k(n)d(n) + k(n)^6\gamma^m\big).$$

Proof. It is clear that Algorithm $\mathcal{RP}_d$ solves RPrimP.

Step (1) of Algorithm $\mathcal{RP}_d$ takes time $\mathcal{O}(k(n)d(n))$; see the analysis of Algorithm $\mathcal{MP}_d$ in the proof of Theorem 4.4.

The algorithm moves to step (2) with probability $\mathcal{O}(k(n)^2(2r-1)^{-d(n/2)})$, and to step (3) with the complementary probability; see Propositions 3.4 and 3.5.

In case we reach step (2), as in the proof of Theorem 4.4, Algorithm $\mathcal{MP}$ takes time $\mathcal{O}(k(n)n\log^*(k(n)n) + rk(n)n + m) = \mathcal{O}(k(n)n\log^*(k(n)n) + m)$. If Algorithm $\mathcal{MP}$ concludes that $w_0 \in H$, then it also outputs a word $x_0 \in F(B)$, for a certain basis B of H; moreover, $|B| \leq k$ and $|x_0| \leq m$. Running Algorithm $\mathcal{S}$ takes time $\mathcal{O}(k(n)^6)$ on average (see Proposition 5.5).

Otherwise, we reach step (3); we are in the situation where $\vec{w}$ has the $d(n)$-ctp and $\min|\vec{w}| > \frac{n}{2}$. In particular, the expected running time of step (2) and all iterations of step (3) of Algorithm $\mathcal{MP}_d$ is $\mathcal{O}(k(n)d(n))$; see the proof of Theorem 4.4 (here is where we use the hypothesis on $k(n)$ and $d(n)$).

Moreover, the growth modulus γ_H of H is at most

$$(2k(n) - 1)^{\frac{2}{n-4d(n)}}$$

by Proposition 3.3. Then, for every n, we have that

$$\gamma_H \leq (2k(n) - 1)^{\frac{2}{n-4d(n)}} < (2k(n))^{\frac{2}{n-4d(n)}}$$

$$\leq (2(2r-1)^{\beta'n})^{\frac{2}{(1-4\delta')n}} = 2^{\frac{2}{(1-4\delta')n}} \cdot (2r-1)^{\frac{2\beta'}{1-4\delta'}}.$$

Taking the limit, we get

$$\gamma_H \leq (2r-1)^{\frac{2\beta'}{1-4\delta'}} < 2r-1$$

since, by hypothesis, $2\beta' < 1 - 4\delta'$. It follows that the probability that $w_0 \in H$ is $\mathcal{O}((\frac{\gamma_H}{2r-1})^m)$, and so $\mathcal{O}(\gamma^m)$. If indeed $w_0 \in H$, we run Algorithm $\mathcal{S}$ on the word $x_0 \in F(X)$, in expected time $\mathcal{O}(k(n)^6)$; see Proposition 5.5.

Thus the expected running time of this algorithm is bounded above by a big-$\mathcal{O}$ of

$$k(n)d(n) + k(n)^2(2r-1)^{-d(n/2)}\left(k(n)n\log^*(k(n)n) + m + k(n)^6\right) + \gamma^m k(n)^6,$$

giving the stated asymptotic estimate.

Finally, if the input $(w_0, \vec{w})$ is such that $\vec{w}$ has the $d(n)$-ctp and $\min|\vec{w}| > \frac{n}{2}$, then after step (1), we go directly to step (3), and the expected running time of the algorithm is $\mathcal{O}(k(n)d(n) + \gamma^m k(n)^6)$. □

As for the Uniform Membership Problem, this gives us upper bounds of the average complexity of the RPrimP for interesting functions $k(n)$.

Corollary 5.8. *The Relative Primitivity Problem (RPrimP) for $F(A)$, with input a $k(n)$-tuple of words of length at most n, and an additional word of length m, can be solved in expected time $C(n, m)$ as follows (where $r = |A|$ is taken to be constant).*

(1) *If k is constant, then $C(n, m) = \mathcal{O}(\log n + mn^{-\log(2r-1)})$.*

(2) *If $\theta > 0$ and $k(n) = n^\theta$,*

$$C(n, m) = \mathcal{O}\left(n^{\theta+\delta} + n^{2\theta}(2r-1)^{-n^\delta}m + n^{6\theta}\left(\frac{2}{2r-1}\right)^m\right)$$

for any $0 < \delta < 1$.

(3) *If* $0 < \beta < \frac{1}{58}$ *and* $k(n) = (2r-1)^{\beta n}$,

$$C(n,m) = \mathcal{O}\left(n(2r-1)^{\beta n} + (2r-1)^{-5\beta n}m + (2r-1)^{6\beta n - \frac{1-58\beta}{1-56\beta}m}\right).$$

Proof. For the case where k is constant, apply Theorem 5.7 with $d(n) = \log(2n)$ and arbitrary valid values of δ' and β'. This shows (1).

For $k(n) = n^\theta$, choose $d(n) = (2n)^\delta$. Then the hypotheses of Theorem 5.7 are satisfied with $\delta' = \frac{1}{8}$ and any β' such that $0 < \beta' < \frac{1}{4}$; the quantity γ is then $\gamma = (2r-1)^{4\beta'-1}$. Choosing β' such that $(2r-1)^{4\beta'} = 2$ yields the stated result. This shows (2).

Finally, suppose that $k(n) = (2r-1)^{\beta n}$ with $0 < \beta < \frac{1}{58}$. Let $\delta = 14\beta$ and $d(n) = \delta n$. Note that $\delta < \frac{1}{4} - \frac{\beta}{2}$. Then the hypotheses of Theorem 5.7 are satisfied with $\beta' = \beta$ and $\delta' = \delta$. The asymptotic upper bound from that theorem now reads as a big-$\mathcal{O}$ of

$$(2r-1)^{\beta n}n + (2r-1)^{(3\beta - \frac{\delta}{2})n}\log^* n + (2r-1)^{(2\beta - \frac{\delta}{2})n}m$$

$$+ (2r-1)^{(8\beta - \frac{\delta}{2})n} + (2r-1)^{6\beta n}\gamma^m,$$

where $\gamma = (2r-1)^{-\frac{1-2\beta-4\delta}{1-4\delta}}$. Since $\delta = 14\beta$, we have $\gamma = (2r-1)^{-\frac{1-58\beta}{1-56\beta}}$, the second and fourth summands above are dominated by the first one, and the last summand becomes

$$(2r-1)^{6\beta n - \frac{1-58\beta}{1-56\beta}m}.$$

So our upper bound is a big-$\mathcal{O}$ of

$$(2r-1)^{\beta n}n + (2r-1)^{-5\beta n}m + (2r-1)^{6\beta n - \frac{1-58\beta}{1-56\beta}m},$$

as stated. This shows (3), concluding the proof. □

A Appendix

Fact A.1. The growth modulus of $L(G_{a,a})$ is

$$\frac{1}{2}\left(2r - 3 + \sqrt{(2r+1)^2 - 8}\right) = (2r-1)\left(1 - \frac{1}{2}r^{-2} - \frac{3}{8}r^{-4} + \mathcal{O}(r^{-5})\right). \quad \square$$

Proof. Let e_i $(-r \le i \le r, i \ne 0)$ be the column vectors with coordinate at vertex a_i equal to 1 and all other coordinates equal to 0, the standard basis of the dimension $2r$ vector space. Let $M_{a,a}$ be the transition matrix of $\mathcal{A}(G_{a,a})$; see Figure 1.

It is an elementary verification that $e_1 - e_{-1}$ is in the kernel of $M_{a,a}$, that each $e_i - e_{-i}$ $(2 \le i \le r)$ is an eigenvector for the eigenvalue 1, and that each

$e_i + e_{-i} - e_r - e_{-r}$ $(2 \leq i < r)$ is an eigenvector for the eigenvalue -1. These $2r - 2$ vectors, together with $v_1 = e_1 + e_{-1}$ and $v_2 = \sum_{i \geq 2}(e_i + e_{-i})$, form a basis of the full space.

Moreover, $M_{a,a} \cdot v_1 = 2v_2$ and $M_{a,a} \cdot v_2 = (2r - 2)v_1 + (2r - 3)v_2$. Therefore, the other eigenvalues of $M_{a,a}$ are the eigenvalues of $\left(\begin{smallmatrix} 0 & 2r-2 \\ 2 & 2r-3 \end{smallmatrix}\right)$, and the result follows. $\qquad \square$

Fact A.2. The growth modulus of $L(G_{a,b})$ is the maximum root of

$$X^3 - (2r - 1)X^2 + 4(r - 1),$$

and it is bounded above by $(2r - 1)(1 - \frac{1}{2}r^{-2})$. $\qquad \square$

Proof. Let e_i $(-r \leq i \leq r, i \neq 0)$ be as in the proof of Fact A.1 and let $M_{a,b}$ be the transition matrix of $\mathcal{A}(G_{a,b})$; see Figure 1.

One verifies easily that $e_1 - e_2$ is in the kernel of $M_{a,b}$ and that

$$M_{a,b} \cdot (e_{-1} - e_{-2}) = e_2 - e_1.$$

It also holds that each $e_i - e_{-i}$ $(3 \leq i \leq r)$ is an eigenvector for eigenvalue 1, and each $e_i + e_{-i} - e_r - e_{-r}$ $(3 \leq i < r)$ is an eigenvector for eigenvalue -1.

These $2r - 3$ vectors are linearly independent, and the vectors $v_1 = e_1 + e_2$, $v_2 = e_{-1} + e_{-2}$ and $v_3 = \sum_{i=3}^{r}(e_i + e_{-i})$ complete them to a basis of the full space.

In addition, we have $M_{a,b} \cdot v_1 = 2(v_1 + v_3)$, $M_{a,b} \cdot v_2 = v_1 + 2v_2 + 2v_3$ and $M_{a,b} \cdot v_3 = (2r - 4)v_1 + (2r - 4)v_2 + (2r - 5)v_3$.

Thus, in this basis (suitably ordered), the matrix of the linear transformation $M_{a,b}$ consists of the following diagonal blocks:

$$\begin{pmatrix} 0 & -1 \\ 0 & 0 \end{pmatrix}, \quad (1) \ (r - 2 \text{ times}), \quad (-1) \ (r - 3 \text{ times}), \quad \begin{pmatrix} 2 & 1 & 2r - 4 \\ 0 & 2 & 2r - 4 \\ 2 & 2 & 2r - 5 \end{pmatrix}.$$

In particular, the remaining eigenvalues are the roots of the characteristic polynomial of that $(3, 3)$-matrix, namely $P(X) = X^3 - (2r - 1)X^2 + 4(r - 1)$. We note that the local extrema of $P(X)$ are at 0 and $\frac{2}{3}(2r - 1)$, $P(0) = 4(r - 1) > 0$ and $P(r) = -r^3 + r^2 + 4r - 4$, which is negative for all $r \geq 2$. Therefore, $P(X)$ has three real roots.

Since $P(2r - 1) = 4(r - 1) > 0$, the leading eigenvalue of $M_{a,b}$ sits between $\frac{2}{3}(2r - 1)$ and $2r - 1$. For a closer estimate, let $\delta = (2r - 1)(1 - \frac{1}{2}r^{-2})$. Then

$$P(\delta) = 4(r - 1) + (2r - 1)^3 \left(1 - \frac{1}{2r^2}\right)^2 \left(1 - \frac{1}{2r^2} - 1\right)$$

$$= 4(r - 1) - \frac{(2r^2 - 1)^2}{8r^6}(2r - 1)^3$$

$$= \frac{16r^6 + 8r^5 - 44r^4 + 16r^3 + 8r^2 - 6r + 1}{8r^6},$$

which is positive when $r \geq 2$. Thus the leading eigenvalue of $M_{a,b}$, which is the growth modulus of $L(G_{a,b})$, is at most $\delta = (2r - 1)(1 - \frac{1}{2}r^{-2})$, as stated. □

Remark A.3. Facts A.1 and A.2, while mathematically elementary, would have been very difficult to establish without the help of a versatile computer algebra system. The authors are grateful to the developers of SageMath [24]. □

Bibliography

[1] G. N. Arzhantseva and A. Y. Ol'shanskiĭ, Generality of the class of groups in which subgroups with a lesser number of generators are free, *Mat. Zametki* **59** (1996), no. 4, 489–496, 638.

[2] F. Bassino, C. Nicaud and P. Weil, Generic properties of subgroups of free groups and finite presentations, in: *Algebra and Computer Science*, Contemp. Math. 677, American Mathematical Society, Providence (2016), 1–43.

[3] F. Bassino, C. Nicaud and P. Weil, Random presentations and random subgroups: A survey, in: *Complexity and Randomness in Group Theory – GAGTA Book 1*, De Gruyter, Berlin (2020), 45–76.

[4] J. L. Bentley and A. C. C. Yao, An almost optimal algorithm for unbounded searching, *Inform. Process. Lett.* **5** (1976), no. 3, 82–87.

[5] J. Burillo and E. Ventura, Counting primitive elements in free groups, *Geom. Dedicata* **93** (2002), 143–162.

[6] N. Chomsky and M. P. Schützenberger, The algebraic theory of context-free languages, in: *Computer Programming and Formal Systems*, North-Holland, Amsterdam (1963), 118–161.

[7] M. Dehn, Über unendliche diskontinuierliche Gruppen, *Math. Ann.* **71** (1911), no. 1, 116–144.

[8] J. Delgado and E. Ventura, A list of applications of Stallings automata, *Trans. Comb.* **11** (2022), no. 3, 181–235.

[9] P. Flajolet and R. Sedgewick, *Analytic Combinatorics*, Cambridge University, Cambridge, 2009.

[10] J. E. Hopcroft and R. E. Tarjan, Efficient algorithms for graph manipulation [H] (algorithm 447), *Commun. ACM* **16** (1973), no. 6, 372–378.

[11] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University, Cambridge, 1990.

[12] I. Kapovich and A. Myasnikov, Stallings foldings and subgroups of free groups, *J. Algebra* **248** (2002), no. 2, 608–668.

[13] S. Margolis, M. Sapir and P. Weil, Closed subgroups in pro-**V** topologies and the extension problem for inverse automata, *Internat. J. Algebra Comput.* **11** (2001), no. 4, 405–445.

[14] A. Miasnikov, E. Ventura and P. Weil, Algebraic extensions in free groups, in: *Geometric Group Theory*, Trends Math., Birkhäuser, Basel (2007), 225–253.

[15] J. Nielsen, Die Isomorphismen der allgemeinen, unendlichen Gruppe mit zwei Erzeugenden, *Math. Ann.* **78** (1917), no. 1, 385–397.

[16] P. Novikov, On the algorithmic unsolvability of the word problem in group theory, *Trudy Mat. Inst. Steklov.* **44** (1955), 1–43.

[17] A. Roig, E. Ventura and P. Weil, On the complexity of the Whitehead minimization problem, *Internat. J. Algebra Comput.* **17** (2007), no. 8, 1611–1634.

[18] J.-P. Serre, *Trees*, Springer, Berlin, 1980.

[19] V. Shpilrain, Average-case complexity of the Whitehead problem for free groups, *Comm. Algebra* **51** (2023), no. 2, 799–806.

[20] J. R. Stallings, Topology of finite graphs, *Invent. Math.* **71** (1983), no. 3, 551–565.

[21] N. W. M. Touikan, A fast algorithm for Stallings' folding process, *Internat. J. Algebra Comput.* **16** (2006), no. 6, 1031–1045.

[22] J. H. C. Whitehead, On certain cets of elements in a free group, *Proc. Londo. Math. Soc. (2)* **41** (1936), no. 1, 48–56.

[23] J. H. C. Whitehead, On equivalent sets of elements in a free group, *Ann. of Math. (2)* **37** (1936), no. 4, 782–800.

[24] The Sage Developers, SageMath, the Sage Mathematics Software System (Version 9.6), 2022, https://www.sagemath.org.

Author information

Corresponding author:
Pascal Weil, CNRS, ReLaX, IRL 2000, Siruseri, India; and Université de Bordeaux,
CNRS, Bordeaux INP, LaBRI, UMR 5800, 33400 Talence, France.
E-mail: pascal.weil@cnrs.fr

Mallika Roy, Departamento de Matemáticas, UPV/EHU, Bilbao, Spain.
E-mail: mallikaroy75@gmail.com

Enric Ventura, Department de Matemàtiques, and Institut de Matemàtiques de la
UPC-BarcelonaTech, Universitat Politècnica de Catalunya, Barcelona, Catalonia, Spain.
E-mail: enric.ventura@upc.edu