Annales Academiæ Scientiarum Fennicæ
Mathematica
Volumen 26, 2001, 375–390

POLYNOMIAL COMPLEXITY OF THE GILMAN–MASKIT DISCRETENESS ALGORITHM

Yicheng Jiang

Rutgers University, Mathematics Department
Newark, NJ 07102, U.S.A.; ycjiang@andromeda.rutgers.edu

Abstract. The main result of this paper is a polynomial bound for the computational complexity of an algorithm to determine whether or not a non-elementary two-generator subgroup of $\mathrm{PSL}(2,\mathrm{R})$ is discrete, that is, an algorithm to determine whether such a subgroup is Fuchsian. The proof that there exists such a bound uses techniques from both hyperbolic geometry and symbolic computation.

1. Introduction

The main result of this paper is a polynomial bound for the computational complexity of the Gilman–Maskit discreteness algorithm. The Gilman–Maskit algorithm presented in [5] and [2] determines whether or not a non-elementary two-generator subgroup of $\mathrm{PSL}(2,\mathrm{R})$ is discrete, that is, it is an algorithm to determine whether such a subgroup is Fuchsian.

In [3] several different forms of the algorithm are distinguished and these are analyzed from the viewpoint of computational complexity. A polynomial bound is found on the complexity of all forms except for a Turing machine implementation. The goal of this paper is to produce a polynomial bound for the complexity of the Turing machine implementation.

The main results of this paper are an improved bound for the number of pairs of generators that any form of the algorithm must consider (Theorems 3.2 and 4.2) along with a polynomial bound on the maximal length of words in the two initial generators that any form of the algorithm must consider (Theorems 5.1 and 5.2) and the polynomial complexity follows from these results.

For any given pair of generators, the Gilman–Maskit algorithm either determines that the group is discrete or non-discrete and then stops or it determines a next pair of generators to consider. The next pair is a word in the two initial generators. A step in the algorithm which involves replacing the pair of generators by another pair of generators is called a *generator-step*. It is known that before it finally determines that the group is or is not discrete, the algorithm considers at most a polynomial number of pairs of generators [3] and these generators are given as words in the initial generators. At the n-th generator step, let $w(n)$ denote

2000 Mathematics Subject Classification: Primary 30F40, 32G15, 68Q25.

the word length of the current generators when given as words in the algorithm's two initial generators. The main result of this paper is that $w(n)$ is bounded by a polynomial function. The polynomial complexity of the full discreteness algorithm follows from this.

The algorithm given that appears in [5] and [2], proceeds by considering the geometric types of the pair of generators (e.g. hyperbolic-hyperbolic, hyperbolic-elliptic). A bound for the complexity of the algorithm for a Turing machine implementation in any finite extension field of the rational numbers was given in [3]. The complexity bound involved some exponential terms for two cases in the algorithm:

(1) Hyperbolic-hyperbolic with disjoint axes;
(2) Hyperbolic-hyperbolic with intersecting axes.

For each of these cases we will produce a polynomial bound.

A *generator step* in the algorithm is called a G-step for short and consists of replacing a pair of generators by a Nielsen equivalent pair. The exponential complexity resulted from the possibility at the n-th G-step of generators of exponential word length in the two initial generators. To understand this, assume that (g, h) is the initial pair of two generators, and count this pair as G-step 0. According to the algorithm, each of these hyperbolic-hyperbolic cases might give a sequence of generating pairs of the form:

$$(g,h) \to (gh, g) \to (ghg, gh) \to (ghg^2h, ghg) \to (ghg^2hghg, ghg^2h) \to \cdots .$$

When this happens, the word length in the two initial generators will increase as a Fibonacci sequence related to the number of generating pairs that remain in that type of hyperbolic-hyperbolic case. We will call such a sequence of pairs of generators a Fibonacci sequence. Denote by $F(n)$ the length of the longest word in the generators at G-step n in a Fibonacci sequence growth (i.e. $F(-2) = 0$, $F(-1) = 1$, $F(n) = F(n-1) + F(n-2)$, $n \geq 0$ so that $\left(\frac{3}{2}\right)^n < F(n) < 2^n$, $n > 1$).

There are also sequences where the word length grows linearly as the number of generating pairs. Such a sequence will be referred to as a non-Fibonacci sequence of generators. An example of a non-Fibonacci sequence would be,

$$(g,h) \to (g, gh) \to (g, g^2h) \to \cdots \to (g, g^nh).$$

When the algorithm repeats and returns to the same geometric type for either type of hyperbolic-hyperbolic case, the whole process could be the mixture of these two types of sequences. For the other cases the algorithm considers (e.g. where the pairs of generators are either hyperbolic-elliptic, hyperbolic-parabolic, parabolic-parabolic, parabolic-elliptic, or elliptic-elliptic) this does not happen. The word length does not grow as Fibonacci sequence for repeated cases of these types, see [3].

Since the total number of G-steps estimated before in [3] for all cases is polynomial in the initial trace (initial trace is defined more precisely below), the word length for the Fibonacci sequence might be exponential in terms of the maximal initial trace.

The organization of this paper is as follows: In Section 3 we show that the number of G-steps that contribute to a Fibonacci sequence type growth for the hyperbolic-hyperbolic case with disjoint axes is not a polynomial function of the initial trace and that in fact it is a doubly logarithmic function, that is, the logarithm of a logarithm (Theorem 3.2) and in Section 4, we show that it is a logarithmic function for hyperbolics with intersecting axes (Theorem 4.2). In Section 5 we use the results of Sections 3 and 4 to obtain bounds on the worst word length that each case must consider (Theorems 5.1 and 5.2). Since in all forms the algorithm proceeds by considering successive pairs of generators, these results improve Corollary 5.4 of [3] and can be substituted into results of [4] to obtain the polynomial complexity bound for the Turing machine implementation. This is done in the last section, Section 6.

2. Notation and preliminaries

Here is some notation that will be used. We let $\mathrm{GL}(2,\mathrm{R})^+$ denote the general linear (2×2)-matrix group in R with positive determinant where R denotes the real numbers. If $g \in \mathrm{GL}(2,\mathrm{R})^+$ and

$$g = \begin{pmatrix} a & b \\ c & d \end{pmatrix},$$

denote the trace of the equivalent matrix with determinant one by $T(g)$. That is,

$$T(g) = \frac{a+b}{(ad-bc)^{1/2}}.$$

Here saying g is equivalent to h means

$$\frac{g}{\sqrt{\det(g)}} = \frac{h}{\sqrt{\det(h)}}.$$

We know both g and $-g$ project onto the same element in $\mathrm{PSL}(2,\mathrm{R})$. The algorithm picks the preimage with non-negative $T(g)$ when normalizing the initial generators in carrying out the calculation before each G-step in both hyperbolic-hyperbolic cases and the repetitions of that type of case.

If g and h are the initial two matrices with $T(g)$ and $T(h)$ non-negative, T will denote the maximal initial trace, which is the maximum of $|T(g)|$, $|T(h)|$, $|T(gh)|$ and $|T(g^{-1}h)|$. When the algorithm is implemented as a Turing machine algorithm, the input includes polynomials with rational coefficients and their degrees. That is, the entries in the matrices are assumed to be given by polynomials with rational coefficients, and the size of the entries is measured by the semi-norms of the polynomials and by their degrees (see [3]). S_I is used to denote the maximal initial semi-norm. (More details are supplied in Section 6.) We remind the reader that if

$$P(x) = \sum_{i=0}^{n} \frac{a_i}{b_i} x^i,$$

then the semi-norm of P is denoted by $\mathrm{SN}(P)$ and defined by

$$\mathrm{SN}(P) = \sum_{i=0}^{n} (|a_i| + |b_i|).$$

3. Hyperbolic-hyperbolic with disjoint axes

Suppose $g, h \in \mathrm{GL}(2,\mathrm{R})^+$ are two initial hyperbolic elements with disjoint axes, and $2 < T(g) \le T(h)$. Denote the ordered pair as (g,h).

For any hyperbolic element f, let a_f be the attracting fixed point of f, and r_f the repelling fixed point. Let the cross ratio

$$C(g,h) = \frac{(r_g - a_h)(a_g - r_h)}{(r_g - r_h)(a_g - a_h)}$$

if r_g, a_g, r_h, a_h are finite, and let the cross ratio be defined by continuity if any one of the fixed points is infinite. We follow the algorithm as given on pp. 15–17, steps I-7, I-8, I-9, and I-10 of [5] or equivalently pp. 182–183 of [2].

By the algorithm we may assume that the pairs are normalized so that the cross ratio, $C(g,h) < 1$, the Jørgensen number $\mu(g,h) = |T([g,h]) - 2| + |T(g)^2 - 4| > 1$ and if $T(gh) > 2$, then we repeat this hyperbolic-hyperbolic case with the pair (g, gh) or (gh, g), and we always have $T(gh) < T(h)$. That is, throughout this section we assume the transformations are normalized with $T(h) > T(g) > 2$ so that the hyperbolic-hyperbolic case is repeated precisely when $T(gh) > 2$. Otherwise, (if $T(gh) \le 2$) the algorithm will stop, saying either that G the group is discrete or that G is not discrete, or it will switch to another case, not a hyperbolic-hyperbolic case.

We first establish a variant on the lower bound obtained on p. 23 of [5].

Lemma 3.1. *With above assumption, we have*

$$T(h) - T(gh) > \tfrac{1}{9}.$$

Proof. Since $T(g)$ is invariant under conjugation, as in [5], we can normalize g and h so that the repelling fixed point of g is at 0 and the attracting fixed point of g is at ∞. Since the axes are disjoint and the cross ratio is < 1, we can normalize further so that the repelling fixed point of h is at 1 and the attracting fixed point of h is at a with $0 < a < 1$. Here a is just the cross-ratio. Then we can express g and h in $\mathrm{SL}(2, R)$ as

$$g = \begin{pmatrix} R & 0 \\ 0 & R^{-1} \end{pmatrix}, \qquad h = \frac{1}{a-1} \begin{pmatrix} aK - K^{-1} & a(K^{-1} - K) \\ K - K^{-1} & aK^{-1} - K \end{pmatrix}$$

with $0 < a < 1 < R < K$. Then following and extending the calculation on p. 20 in [5], we have

$$T(h) - T(gh) = \frac{R-1}{1-a}(R^{-1}K - K^{-1} + aK - aR^{-1}K^{-1}).$$

From [1], we know the Jørgensen number will be

$$\mu(g,h) = \frac{(R - R^{-1})^2 (K - K^{-1})^2}{\left(\sqrt{a} - \sqrt{a}^{-1}\right)^2} + (R - R^{-1})^2.$$

Let $q(x) = (x - x^{-1})^2$, here $q(x)$ is just the function $1/f(x^2)$ of [1] and K, R here are K^2, R^2 in that paper. Then similarly to [1],

$$\mu(g,h) = \frac{q(R)q(K)}{q\left(\sqrt{a}\right)} + q(R).$$

We have two cases to consider.

(i) $q(R) > \frac{1}{2}$. Since $R > 1$, from $q(R) > \frac{1}{2}$ we can get $R > \sqrt{2}$ and $R - 1 > R/\sqrt{2}\,(R+1)$. With $0 < a < 1 < R < K$, we have

$$\begin{aligned} T(h) - T(gh) &> \frac{R(R^{-1}K - K^{-1})}{\sqrt{2}\,(R+1)} > \frac{R-1}{\sqrt{2}\,(R+1)} \\ &= \frac{\left((R+1) - 2\right)}{\sqrt{2}\,(R+1)} = \frac{1}{\sqrt{2}} - \frac{2}{\sqrt{2}\,(R+1)} > \frac{1}{9}. \end{aligned}$$

(ii) $q(R) \le \frac{1}{2}$. Since the Jørgensen number $\mu(g,h)$ is greater than 1, we have

$$1 < \mu(g,h) = q(R)\left(\frac{q(K)}{q\left(\sqrt{a}\right)} + 1\right) \le \frac{1}{2}\left(\frac{q(K)}{q\left(\sqrt{a}\right)} + 1\right).$$

Then $q(K) > q\left(\sqrt{a}\right) = q\left(\sqrt{a^{-1}}\right)$. Since as in [1], $q'(x) = 2x^{-3}(x^4 - 1) > 0$ when $x > 1$, we can get $K > \sqrt{a^{-1}} > 1$, so $K^2 a > 1$. Similarly to case (i), we know $1 < R \le \sqrt{2}$. Then

$$R - 1 > \frac{R}{R+1} \cdot \frac{1-a}{\sqrt{(a-1)^2 + a(K - K^{-1})^2}}.$$

With this inequality, we can get, for this case,

$$\begin{aligned} T(h) - T(gh) &> \frac{K - K^{-1} + K^{-1}(1-a) + RK^{-1}(aK^2 - 1)}{(R+1)\sqrt{(a-1)^2 + a(K - K^{-1})^2}} \\ &> \frac{1}{(R+1)\sqrt{\big(aq(\sqrt{a}\,)/q(K)\big) + a}} \\ &> \frac{1}{(R+1)\sqrt{a+a}} > \frac{1}{(\sqrt{2}\,+1)\sqrt{2}} > \frac{1}{9}. \end{aligned}$$

By (i) and (ii), the lemma is proved. □

Now consider the case when the word length grows as Fibonacci sequence. We observe that a G-step results in the word length growing as Fibonacci sequence only if

$$2 < T(gh) < T(g) \leq T(h).$$

This condition dictates the next pair of generators so at the next G-step, the pair will be (gh, g) and at the following G-step the pair will be (gh, ghg) or (ghg, gh), i.e., the word length will increase as Fibonacci sequence.

So we look at that situation, to find a bound for $T(gh)$ in terms of $T(h)$ and we then use this to get the bound for the number of G-steps in such growth. Namely, we will show

$$T(gh) < \sqrt{2T(h)}\, + 1.$$

From this we will show the bound of number of such G-steps will not be a polynomial function in the initial trace.

From the paper [5], we know if $2 < T(gh) < T(g) \leq T(h)$, we can normalize the pair to be

$$g = \begin{pmatrix} y^2 & 0 \\ 0 & 1 \end{pmatrix} \qquad \text{and} \qquad h = \begin{pmatrix} 2 & -(z-w) \\ z-w & 2zw \end{pmatrix}$$

with

$$-1 < -y < -w < 0 < z < y < 1.$$

See Figure 1, where L is the common perpendicular geodesic of Axis$\,(g)$ and Axis$\,(h)$, let r, r_g, r_h denote the reflections about L, L_g, L_h, then $g = r_g r$, and $h = r r_h$. (Note a slight difference in notation: here the w of [5] is replaced by $-w$.)

Then

$$gh = \begin{pmatrix} 2y^2 & -y^2(z-w) \\ (z-w) & 2zw \end{pmatrix}.$$

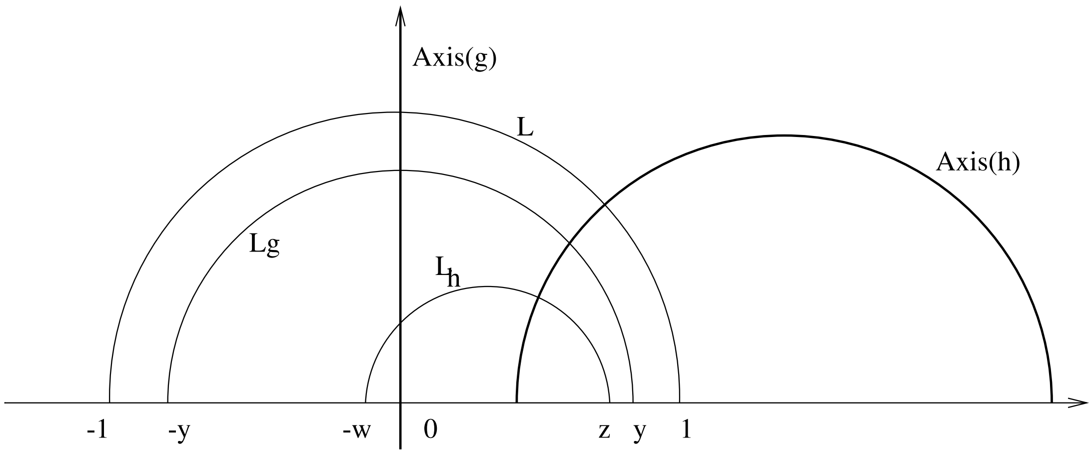

Figure 1.

And

$$T(g) = y + \frac{1}{y}, \qquad T(h) = \frac{2(1+zw)}{z+w}, \qquad T(gh) = \frac{2(y^2+zw)}{y(z+w)}.$$

So we have

$$\frac{2(y^2+zw)}{y(z+w)} < \frac{y^2+1}{y} \leq \frac{2(1+zw)}{z+w}.$$

Because $0 < z < y < 1$ and $0 < w < y < 1$, then

$$0 < z + w < 2y < 2.$$

Here we have two cases:

(1) $T(h) \leq 4$.

We know from Lemma 3.1, in this case,

$$T(h) - T(gh) > \tfrac{1}{9}.$$

If $T(h) \leq 4$, to stay in the same case, the number of G-steps we have will be bounded by a constant number $C_1 - 7$.

(2) $T(h) > 4$.

$$T(h) = \frac{2(1+zw)}{z+w} > 4z + w < \frac{1+zw}{2} < 1.$$

Since $T(gh) < T(g)$, then

$$\frac{2(y^2+zw)}{y(z+w)} < \frac{y^2+1}{y},$$

and we have

$$
\begin{aligned}
2(y^2 + zw) &< (y^2+1)(z+w), \\
y^2\big(2-(z+w)\big) &< (z+w) - 2zw, \\
y^2 &< \frac{(z+w)-2zw}{2-(z+w)} < \frac{z+w}{2-(z+w)} < z+w, \\
y &< \sqrt{z+w}\,.
\end{aligned}
$$

Then,

$$T(gh) = \frac{2y}{z+w} + \frac{2zw}{y(z+w)} < \frac{2}{\sqrt{z+w}} + \frac{2}{(y/w)+(y/z)} < \frac{2}{\sqrt{z+w}} + 1$$

and

$$T(h) = \frac{2(1+zw)}{z+w} > \frac{2}{z+w}$$

so that we have

$$T(gh) < \sqrt{\frac{4}{z+w}} + 1 < \sqrt{2T(h)} + 1.$$

Suppose we stay in this case for n G-steps. Here we do not require these n G-steps to be consecutive, and the initial step is labeled as G-step 0. Let $t(i)$ denote the maximal trace of the pair at i-th G-step. We have

$$
\begin{aligned}
t(i+1) &\le t(i) \qquad \text{since } T(g) \le T(h), \text{ and } T(gh) < T(h), \\
t(i+2) &< \sqrt{2t(i)} + 1 \quad \text{since } T(gh) < \sqrt{2T(h)} + 1 \\
&< \sqrt{2t(i)} + \tfrac{1}{2}\sqrt{t(i)} \quad \text{since } t(i) \ge 4 \\
&= a\sqrt{t(i)} \text{ where } a = \sqrt{2} + \tfrac{1}{2}.
\end{aligned}
$$

Let $k = 2^{[n/2]}$, if $t(n) \ge 4$, we have $4 \le t(n) < a^2 \sqrt[k]{t(0)}$, then $\sqrt[k]{t(0)} > 4/a^2$, i.e.,

$$2^{[n/2]} < \frac{\log_2 t(0)}{\log_2 (4/a^2)} < 8 \log_2 t(0),$$

we can get $n < 2\log_2\log_2 t(0) + 7$. We can thus conclude:

Theorem 3.2. *The total number of Fibonacci type G-steps for a sequence that remains in the disjoint axes case, is bounded by*

$$2\log_2\log_2 t(0) + C_1$$

where C_1 is a constant number and $t(0)$ is the maximal initial trace. The bound holds whether or not the Fibonacci type G-steps are consecutive.

Proof. From (1) and (2), we can say the number of G-steps for Fibonacci sequence is at most

$$n < 2\log_2\log_2 t(0) + 7 + C_1 - 7 = 2\log_2\log_2 t(0) + C_1. \qquad \square$$

4. Hyperbolics with intersecting axes

In this section we use the algorithm for intersecting axes given in [2] and the repetition criteria given there. We let g, h be two initial elements with intersecting axes, and assume by the repetition criteria that $2 < T(g) \leq T(h)$. As in the previous section, $t(i)$ denotes the maximal trace at G-step i. With the current notation we know

Lemma 4.1. *The number of G-steps that stay in this case beginning at G-step i is bounded by $169 \cdot t(i)^2$.*

Proof. See Theorem 5.2 in [3]. □

Now consider the G-steps with word length growing as Fibonacci sequence. From Section 2.5 and also step 2 on p. 181 in [2], we can see that the word length increases as a Fibonacci sequence only if

$$2 < T(gh) < T(g) \leq T(h) \leq T(gh^{-1}),$$

i.e. this is what is called a "turn a corner step" in Section 2.5 of [2].

In this case we will show

$$\frac{T(gh)}{T(gh^{-1})} < \frac{1}{T(h) - 1}.$$

Since this is the intersecting axes case, we can normalize g to

$$g = \begin{pmatrix} k^2 & 0 \\ 0 & 1 \end{pmatrix} \qquad \text{with } k > 0,\ k \neq 1,$$

and let the attracting fixed point of h be x with $x > 0$, and the repelling fixed point of h be -1,

$$h = \begin{pmatrix} xR^2 + 1 & xR^2 - x \\ R^2 - 1 & R^2 + x \end{pmatrix} \qquad \text{with } R > 1.$$

Then

$$gh = \begin{pmatrix} xk^2R^2 + k^2 & xk^2R^2 - k^2x \\ R^2 - 1 & R^2 + x \end{pmatrix}$$

and

$$T(g) = k + \frac{1}{k}, \quad T(h) = R + \frac{1}{R}, \quad T(gh) = \frac{xk^2R^2 + k^2 + R^2 + x}{kR(x+1)}.$$

To stay in this case, we must have

$$T(gh^{-1}) > T(gh),$$

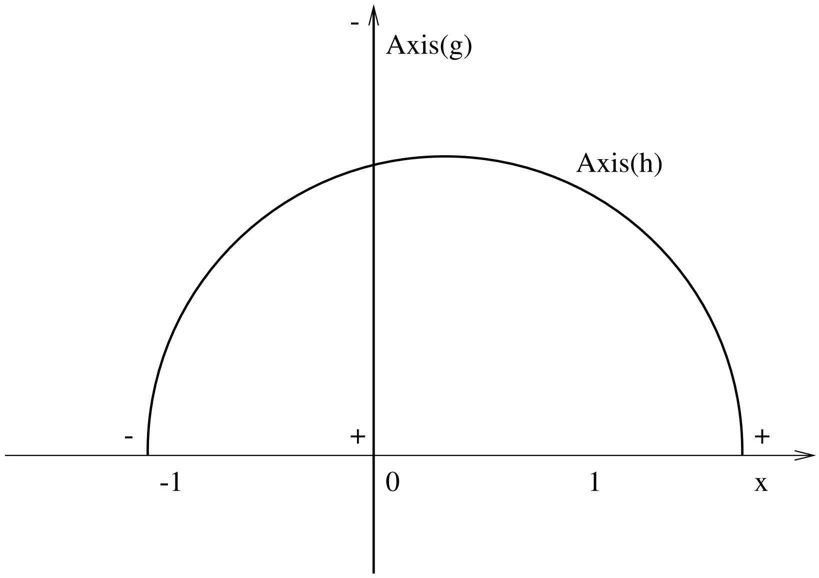

Figure 2.

otherwise, the algorithm will stop at the acute triangle step. From

$$T(gh^{-1}) = \frac{k^2R^2 + k^2x + xR^2 + 1}{kR(x+1)} > \frac{xk^2R^2 + k^2 + R^2 + x}{kR(x+1)} = T(gh),$$

we get $(k^2-1)(1-x)(R^2-1) > 0$. By $R > 1$, we have $(k^2-1)(1-x) > 0$. There are two cases to consider:

(1) $x > 1$ and $k < 1$. (See Figure 2 for this situation.)

Let $l = 1/k > 1$, then $1 < l \le R$. Since $T(gh) < T(g)$, we can get $l^2(R-1)(R-x) < (R-1)(1-xR)$. By $x > 1$, and $R > 1$, we get $R < x$ and $l^2 > (xR-1)/(x-R)$.

Let $f(t) = (a+bt)/(at+b)$, where $a = xR^2+1$, $b = R^2+x$, and $a-b = (x-1)(R^2-1) > 0$, we have $f'(t) = (b^2-a^2)/(at+b)^2 < 0$. Then

$$\frac{T(gh)}{T(gh^{-1})} = f(l^2) < f\left(\frac{xR-1}{x-R}\right) = \frac{R}{R^2-R+1} = \frac{1}{T(h)-1}.$$

(2) $k > 1$ and $0 < x < 1$.

This time we have $0 < x < 1 < k \le R$. Then by $T(gh) < T(g)$, we can get $x(k^2R-1) < k^2-R$, so $k^2 > R$ and $x < (k^2-R)/(k^2R-1) < 1$. Consider $T(gh)/T(gh^{-1}) = (xa+b)/(a+xb)$, where $a = k^2R^2+1$, $b = k^2+R^2$ and $a-b = (k^2-1)(R^2-1) > 0$. As in the first case, let $f(t) = (ta+b)/(a+tb)$. Then $f'(t) = (a^2-b^2)/(a+tb)^2 > 0$.

$$\frac{T(gh)}{T(gh^{-1})} = f(x) < f\left(\frac{k^2-R}{k^2R-1}\right) = \frac{1}{R-1+(1/R)} = \frac{1}{T(h)-1}.$$

From (1), (2), if $T(gh) < T(g) \leq T(h) \leq T(gh^{-1})$, we always have

$$\frac{T(gh)}{T(gh^{-1})} < \frac{1}{T(h) - 1}.$$

Denote the pair of i-th G-step as (g_i, h_i). Since the algorithm is trace minimizing, to estimate the maximal number of G-steps of Fibonacci sequence, we can suppose all G-steps satisfy $T(g_i h_i) < T(g_i) \leq T(h_i) \leq T(g_i h_i^{-1})$, i.e. that all will contribute to the word length increasing as Fibonacci sequence. We have following relations:

$$\begin{aligned} g_{i+1} &= g_i h_i, \\ h_{i+1} &= g_i, \\ g_{i+1} h_{i+1}^{-1} &= g_i h_i g_i^{-1}, \\ T(g_{i+1} h_{i+1}^{-1}) &= T(h_i). \end{aligned}$$

Let $t(i)$ be the maximal trace of the pair (g_i, h_i), i.e $t(i) = T(h_i)$.

$$\frac{t(i+3)}{t(i)} = \frac{T(h_{i+3})}{T(h_i)} = \frac{T(g_{i+2})}{T(h_i)} = \frac{T(g_{i+1} h_{i+1})}{T(g_{i+1} h_{i+1}^{-1})} < \frac{1}{2} \qquad \text{when } t(i+1) \geq 3.$$

Suppose $t(n) \geq 3$ and $k = [\frac{1}{3} n]$, then

$$\begin{aligned} 3 \leq t(n) &< \left(\tfrac{1}{2}\right)^k t(0), \\ k + \log_2 3 &< \log_2 t(0), \\ n &< 3 \log_2 t(0) - 2. \end{aligned}$$

So if $T(h) \geq 3$, there are at most $3 \log_2 t(0) - 2$ G-steps.

We can now obtain one of our main results:

Theorem 4.2. *The number of Fibonacci growth G-steps for the intersecting axes case will be bounded by*

$$3 \log_2 t(0) + C_2$$

where C_2 is some constant number. This is true whether or not these Fibonacci type G-steps are consecutive.

Proof. If $T(h) < 3$, we know from Lemma 4.1, that the number of G-steps will be bounded by a constant number $C_2 + 2$. Then from the above discussion, the total number of G-steps for this case will be bounded by

$$3 \log_2 t(0) - 2 + C_2 + 2 = 3 \log_2 t(0) + C_2. \ \square$$

Thus for both cases, the number of G-steps of Fibonacci sequence length growth is bounded by a logarithmic function of the initial trace.

5. Worst word length estimates

Now we use the results of the last two sections to give new estimates for the word lengths of [3], [4]. The purpose here is to consider the mixture of Fibonacci G-steps and non-Fibonacci G-steps, so that we can get the worst word length in general. Recall that we are only looking at G-steps that remain in a hyperbolic-hyperbolic case, and before each G-step, the generator pair (g,h) will be normalized as needed so that $2<T(g)\leq T(h)$.

For an ordered generator pair (g,h) with $T(g)\leq T(h)$, let l_1 be the word length of h and l_2 be the word length of g, i.e. the length pair would be (l_2,l_1). Also let $d=l_1+l_2$, i.e. d is the word length of gh since from the algorithm, there is no possibility for word cancellation between g and h.

Denote n consecutive Fibonacci G-steps as F_n, i.e. F_n is

$$(g,h)\to(gh,g)\to(ghg,gh)\to(ghg^2h,ghg)\cdots$$

with $n+1$ pairs, and m consecutive non-Fibonacci G-steps as L_m, i.e. L_m is

$$(g,h)\to(g,gh)\to\cdots(g,g^mh)$$

with $m+1$ pairs.

If l_1, l_2 and d are the corresponding word lengths of h, g and gh before F_n or L_m, denote $l_1(F_n)$, $l_2(F_n)$ and $d(F_n)$ be the new value of the corresponding length after n consecutive Fibonacci steps, i.e., let F_n be $(g_0,h_0)\to(g_1,h_1)\to\cdots\to(g_n,h_n)$, then

$l_1=$ the word length of h_0,
$l_2=$ the word length of g_0,
$d=l_1+l_2$, the word length of g_0h_0,
$l_1(F_n)=$ the word length of h_n,
$l_2(F_n)=$ the word length of g_n,
$d(F_n)=l_1(F_n)+l_2(F_n)$, the word length of g_nh_n.

Similarly let $l_1(L_m)$, $l_2(L_m)$, $d(L_m)$ denote the corresponding length after L_m.

We also allow F_n and L_m to be composed with each other. For example, F_nL_m means we have n consecutive Fibonacci G-steps first, then followed by m consecutive non-Fibonacci G-steps, and the end pair of F_n will be the initial pair of L_m.

With this notation, we have $F_{n_1}F_{n_2}=F_{n_1+n_2}$, $L_{m_1}L_{m_2}=L_{m_1+m_2}$, and $l_i(F_{n_1}F_{n_2})=l_i(F_{n_1+n_2})$, $l_i(L_{m_1}L_{m_2})=l_i(L_{m_1+m_2})$, $i=1,2$.

For $n>0$ and $m>0$, we have

$$\begin{aligned}
l_1(F_n)&=F(n-3)l_1+F(n-2)l_2,\\
l_2(F_n)&=F(n-2)l_1+F(n-1)l_2,\\
l_1(L_m)&=m\cdot l_2+l_1,\\
l_2(L_m)&=l_2.
\end{aligned}$$

Then

$$d(F_n) = l_1(F_n) + l_2(F_n) = F(n-1)l_1 + F(n)l_2 < 2^n(l_1 + l_2) = 2^n d$$

and

$$d(L_m) = l_1(L_m) + l_2(L_m) = m \cdot l_2 + l_1 + l_2 < (m+1)(l_1 + l_2) = (m+1)d.$$

Denoting F_0 as the empty step, then $d(F_n) \le 2^n d$ for $n \ge 0$. Similarly denoting L_0 as the empty step, we have $d(L_m) \le (m+1)d$ for $m \ge 0$.

As our purpose is to get a bound for the word length after $F_{n_1}L_{m_1}\cdots F_{n_k}L_{m_k}$, we need to use some additional notation. Given l_1, l_2, the initial word lengths of h and g, and $d = l_1 + l_2$, define

$$\begin{aligned} l_1(F'_n) &= 2^{n-1}(l_1 + l_2), \\ l_2(F'_n) &= 2^{n-1}(l_1 + l_2), \\ l_1(L'_m) &= ml_2 + (m+1)l_1, \\ l_2(L'_m) &= l_2. \end{aligned}$$

Then we have $l_i(F_n) \le l_i(F'_n)$ and $l_i(L_m) \le l_i(L'_m)$, $i = 1, 2$. By the fact that

$$d(F'_n) = l_1(F'_n) + l_2(F'_n) \qquad \text{and} \qquad d(L'_m) = l_1(L'_m) + l_2(L'_m),$$

we get $d(F'_n) = 2^n d$ with $n \ge 0$ and $d(L'_m) = (m+1)d$ with $m \ge 0$.

Just as we compose F_n and L_m, we have a similar composition between F'_n and L'_m. Then it is easy to check

$$l_i(F_{n_1}L_{m_1}\cdots F_{n_k}L_{m_k}) \le l_i(F'_{n_1}L'_{m_1}\cdots F'_{n_k}L'_{m_k}), \qquad i = 1, 2,$$

and

$$d(F'_n L'_m) = d(L'_m F'_n) = (m+1)2^n d.$$

Suppose we stay in one of the hyperpolic-hyperbolic cases for N G-steps, and with a sequence F_{n_1}, L_{m_1}, F_{n_2}, L_{m_2}, $\ldots$, F_{n_k}, L_{m_k}, where the n's and m's are non-negative integers. Let $w(N)$ be the longest word length in the initial pair after these G-steps, and let l_1, l_2 and d be the initial word length of h, g and gh, then

$$\begin{aligned} w(N) &< d(F_{n_1}L_{m_1}\cdots F_{n_k}L_{m_k}) = l_1(F_{n_1}L_{m_1}\cdots F_{n_k}L_{m_k}) + l_2(F_{n_1}L_{m_1}\cdots F_{n_k}L_{m_k}) \\ &< l_1(F'_{n_1}L'_{m_1}\cdots F'_{n_k}L'_{m_k}) + l_2(F'_{n_1}L'_{m_1}\cdots F'_{n_k}L'_{m_k}) = d(F'_{n_1}L'_{m_1}\cdots F'_{n_k}L'_{m_k}) \\ &= d(F'_{n_1}F'_{n_2}\cdots F'_{n_k}L'_{m_1}\cdots L'_{m_k}) = d(F'_n L'_m) = (m+1)2^n d, \end{aligned}$$

where $n = n_1 + \cdots + n_k$ is the total number of G-steps for the Fibonacci sequence, and $m = m_1 + \cdots + m_k$ is the total number of G-steps for the non-Fibonacci sequence, and $d = 2$, the sum of word lengths of the initial pair.

From the previous sections, we know for both cases, there are bounds for such n and m.

For the disjoint axes case, from Theorem 3.2, $n < 2\log_2\log_2 t(0) + C$ and from Lemma 3.1, $m < 9\big(t(0) - 2\big)$, we obtain

Theorem 5.1. *The worst word length for the disjoint axes case is*

$$w(N) < C_1 t(0)\big(\log_2 t(0)\big)^2$$

where C_1 is some constant.

For the intersecting axes case, by Theorem 4.2, $n < 3\log_2 t(0) + C$, and Lemma 4.1, $m < 169t(0)^2$, so

Theorem 5.2. *The worst word length for the intersecting axes case is*

$$w(N) < C_2 t(0)^5$$

where C_2 is some constant.

We observe that these bounds are actually independent of the number N of pairs of generators that we must consider.

6. Complexity estimate

A complete discussion of algorithms that provides a conceptual framework in which to discuss the real number algorithm and the Turing machine algorithms appears in [2] (see Chapter 14, Section 14.1 and pp. 167–173 or [3], p. 94, Section 3.2). The real number algorithm is considered as a BSS-machine (a Blum–Shub–Smale machine).

The complexity analysis needed for a number of different forms of the algorithm is carried out in [3]. In particular, it is shown that as a BSS machine the real number algorithm is of linear complexity. But only an exponential bound is found for the Turing machine algorithms, TM1 and TM2, where the algorithm is implemented using symbolic computation with minimal polynomials in finite extensions of the rationals.

The analysis in [3] takes into account bounds on (1) the number of pairs of generators the algorithm considers before it stops, (2) the length of the longest words the algorithm must process (denoted Length(T, D) in [3]), (3) the complexity of carrying out basic arithmetic operations using symbolic computation in a finite extension of the rationals, and (4) how the algorithm increases the size of the numbers (semi-norm of the polynomials) one is dealing with.

In particular Lemma 5.3 and Corollary 5.4 of [3] show that Length(T, D) is a product of factors, one for each geometric case. But $t(0) = T$. That is, the $t(0)$ that appears in Theorems 5.1 and 5.2 here is the same as T. Thus we are able to replace 2^{9T} and 2^{169T^2}, the exponential factors used in [3] for the hyperbolic-hyperbolic cases, by T^5 and $T(\log T)^2$, our two estimates for $w(n)$.

We remark that the estimates obtained here are better because they use more of the force of the trace decreasing property of the algorithm than [3].

We apply Theorems 5.1 and 5.2 to bound the complexity of the Turing machine implementations of the algorithm.

We recall that there are two different Turning machine implementations of the algorithm: for TM 1 the input is eight minimal polynomials one for each of the eight entries in the two input matrices, S_0 the maximal semi-norm of these eight polynomials and D the product of their degrees. For TM 2 the eight initial entries are assumed to lie in the same finite simple extension of the rationals, $Q(\gamma)$, an extension of degree D. The input for TM 2 includes the minimal polynomial for γ and the representing polynomials for the eight entries, that is, the polynomials in γ that give each of the eight matrix entries and S_I the maximal semi-norm of these nine polynomials.

In short, we let T be the maximal initial trace, D be the degree of the simple extension field of Q or the product of the eight degrees, S_0 be the maximal initial semi-norm of TM 1, S_I be the maximal initial semi-norm of TM 2, $L(\mathrm{SN})$ be the log of the semi-norm SN and Length (T, D) be the worst word length in initial generators of words considered by the algorithm. For more details of these definitions and notation, one can check [3] and [4].

First, from Theorem 5.1, for a hyperbolic pair with disjoint axes case, the word length $w(n)$ would be bounded by $w(n) < C_1 T(\log_2 T)^2$. Second, from Theorem 5.2, for a hyperbolic pair with intersecting axes case, the worst is $w(n) < C_2 T^5$.

Corollary 5.4 of [3], says that

$$\text{Length}\,(T, D) = (1 - \delta_{ih})(2^{\delta_{dh} 9T})2T(34D^2 + 1)^9 + \delta_{ih} 2^{(13T)^2},$$

where

$\delta_{dh} = 1$ when the initial generators are a pair of hyperbolics with disjoint axes and 0 otherwise, and

$\delta_{ih} = 1$ when the initial generators are a pair of hyperbolics with intersecting axes and 0 otherwise.

Substituting our bounds into the proof of Corollary 5.4 [3], we get

$$\text{Length}\,(T, D) = (1 - \delta_{ih})\big(\delta_{dh} C_1 T(\log_2 T)^2 + 1\big)2T(34D^2 + 1)^9 + \delta_{ih} C_2 T^5.$$

We next use Corollary 7.7 of [3]. It says that the complexity of TM 1 is at worst

$$O\left(D^8\big(L(S_0)\big)^2 \cdot \big[D\big(2(D-1)\text{Length}\,(T, D) + 1\big)\text{Length}\,(T, D)\big]^2 \cdot P(T, D)\right)$$

where $P(T, D) = 170T^2 + 32D^2 + 14$.

We substitute the new bound for Length (T, D) into that of Corollary 7.7 of [3] and also use Lemma 7.6 of [3], which says that $T \leq 4(S_0 D)^2$, to conclude

$$\text{Length}\,(T, D) \leq O\left(S_0^{10} D^{22}\big(\log_2(S_0 D)\big)^2\right) \qquad \text{and} \qquad P(T, D) \leq O(S_0^4 D^4).$$

Since $L(S) < S$ and $\log_2 S < S$, we obtain

Theorem 6.1. *The complexity of* TM 1 *at worst is*

$$O\left(\left(L(S_0)\right)^2 S_0^{44} D^{104} \left(\log_2(S_0 D)\right)^8\right) \qquad \text{or} \qquad O\left(S_0^{54} D^{112}\right).$$

For TM 2, from Corollary 3.3 in [4], we know the complexity is

$$O\left(D^{30}\left(L(S_I)\right)^2 \cdot \left[D\left(2(D-1)\text{Length}\,(T,D)+1\right)\text{Length}\,(T,D)\right]^2 \cdot P(T,D)\right),$$

and by Lemma 3.2 in [4], $T \le 4(1+S_I)^2$, we have

$$\text{Length}\,(T,D) \le O(S_I^{10} D^{18}) \qquad \text{and} \qquad P(T,D) \le O(S_I^4 + D^2).$$

We conclude

Theorem 6.2. *The complexity of* TM 2 *at worst is*

$$O\left(\left(L(S_I)\right)^2 S_I^{28} D^{108}\right) \qquad \text{or} \qquad O\left(S_I^{30} D^{108}\right).$$

Remark. The first completely correct discreteness algorithm was given by Rosenberger [7]. While the Gilman–Maskit algorithm is not exactly the same, it is likely that the complexity analysis given here could be modified to apply to a similar Turing machine implementation of the Rosenberger algorithm over a finite extension of the rationals and that the complexity would also be polynomial.

References

[1] Gilman, J.: A geometric approach to Jøgensen's inequality. - Adv. Math. 85, 1991, 193–197.

[2] Gilman, J.: Two-generator discrete subgroups of PSL(2, R). - Mem. Amer. Math. Soc. 117, 1995, 1–204.

[3] Gilman, J.: Algorithms, complexity and discreteness criteria in PSL(2, C). - J. Anal. Math. 73, 1997, 91–114.

[4] Gilman, J.: Complexity of a Turing machine discreteness algorithm. - Contemp. Math. 256, 2000, 165–171.

[5] Gilman, J., and B. Maskit: An algorithm for 2-generator Fuchsian groups. - Michigan Math. J. 38, 1991, 13–32.

[6] Maskit, B.: Kleinian Groups. - Springer-Verlag, 1988.

[7] Rosenberger, G.: All generating pairs of all two-generator Fuchsian groups. - Arch. Math. (Basel) 46, 1986, 198–204.

Received 18 January 2000
Received in revised form 27 November 2000