Combinatorics, Probability and Computing (2004) 13, 499–536. © 2004 Cambridge University Press
DOI: 10.1017/S0963548304006261 Printed in the United Kingdom

Dynamical Analysis of the Parametrized Lehmer–Euclid Algorithm

BENOÎT DAIREAUX and BRIGITTE VALLÉE

GREYC, Université de Caen, F-14032 Caen, France
(e-mail: daireaux@info.unicaen.fr)

Received 13 January 2003; revised 31 December 2003

The Lehmer–Euclid Algorithm is an improvement of the Euclid Algorithm when applied to large integers. The original Lehmer–Euclid Algorithm replaces divisions on multi-precision integers by divisions on single-precision integers. Here we study a slightly different algorithm that replaces computations on n-bit integers by computations on μn-bit integers. This algorithm depends on the truncation degree $\mu \in]0,1[$ and is denoted as the $\mathcal{LE}_\mu$ algorithm. The original Lehmer–Euclid Algorithm can be viewed as the limit of the $\mathcal{LE}_\mu$ algorithms for $\mu \to 0$. We provide here a precise analysis of the $\mathcal{LE}_\mu$ algorithm. For this purpose, we are led to study what we call the *Interrupted Euclid Algorithm*. This algorithm depends on some parameter $\alpha \in [0,1]$ and is denoted by $\mathcal{E}_\alpha$. When running with an input (a,b), it performs the same steps as the usual Euclid Algorithm, but it stops as soon as the current integer is smaller than a^α, so that $\mathcal{E}_0$ is the classical Euclid Algorithm. We obtain a very precise analysis of the algorithm $\mathcal{E}_\alpha$, and describe the behaviour of main parameters (number of iterations, bit complexity) as a function of parameter α. Since the Lehmer–Euclid Algorithm $\mathcal{LE}_\mu$ when running on n-bit integers can be viewed as a sequence of executions of the Interrupted Euclid Algorithm $\mathcal{E}_{1/2}$ on μn-bit integers, we then come back to the analysis of the $\mathcal{LE}_\mu$ algorithm and obtain our results.

1. Introduction

The computation of the gcd is a major subroutine for most computations on long integers; it is widely used for expressing rational numbers in 'lowest terms', finding modular inverses, and so on. It is one of the most time-consuming basic operations on long integers. For instance, during the computation of Gröbner bases, it was noticed that 80% of computing time is spent in long integer arithmetic, and notably in gcd computations. When applied to long integers, the Euclid Algorithm is not very attractive, since it performs a sequence of multiple-precision divisions that are quite time-consuming. Indeed, although theoretically division has the same time complexity as multiplication [16], a division algorithm that will be designed along the lines explained by Knuth will be about 30 times slower than multiplication [14]. With some improvements due to Krandick and Johnson [17], one may hope to reduce the gap to 15 times. For very long integers, Jebelean [15] proposed a

https://doi.org/10.1017/S0963548304006261 Published online by Cambridge University Press

division algorithm which is about twice as slow as Karatsuba multiplication – in most of the cases, since there is a small probability of failure.

A significant improvement in the speed of the Euclid Algorithm when high-precision numbers are involved can be achieved with the so-called Lehmer–Euclid Algorithm, which uses a method due to Lehmer [18]. Working only with the leading digits of large integers, it is possible to simulate most of the multiple-precision divisions by single-precision divisions, which leads to a significant speed-up of the algorithm. The first version of this algorithm appeared in [18]; then some variants were described in Knuth [16], and finally Collins [7] and Jebelean [14] provided various improvements to the algorithm. Nowadays, most of computer algebra systems and multi-precision libraries use many of these variants. However, there exist very few analyses of the Lehmer–Euclid Algorithm. Sorenson [23] obtained a worst-case analysis of this algorithm, but, to the best of our knowledge, there does not exist any precise average-case analysis of the Lehmer–Euclid Algorithm. It is the purpose of this paper to provide such an analysis.

Main results

The original Lehmer–Euclid Algorithm replaces divisions on multi-precision integers by divisions on single-precision integers (sometimes double precision is used). Here, we study a slightly different algorithm that replaces computations on n-bit integers by computations on μn-bit integers. This algorithm depends on the truncation degree $\mu \in]0,1[$ and is denoted as the $\mathscr{LE}_\mu$ algorithm. This Lehmer–Euclid Algorithm can be viewed as a sequence of executions of the so-called *Interrupted Euclid Algorithm*. Generally speaking, this Interrupted Algorithm depends on some parameter $\alpha \in [0,1]$, and is denoted by $\mathscr{E}_\alpha$. It performs exactly the same steps as the Euclid Algorithm but, when running on some input $(a,b), a \geqslant b$, it stops as soon as the current integer is smaller than a^α. We first provide a complete analysis of this Interrupted Algorithm, and we carefully study (in the average case) its main parameters – number of iterations and bit complexity – as a function of the parameter α (Theorem 2.1, proved in Section 4, and Theorem 2.2, proved in Section 5).

Then, we come back to the initial algorithm, the $\mathscr{LE}_\mu$ algorithm and we carefully compare its average bit complexity with the average bit complexity of the classical Euclid Algorithm. We first prove that the $\mathscr{LE}_\mu$ algorithm, when running on n-bit integers, performs (almost surely) a sequence of executions of the Interrupted Euclid Algorithm $\mathscr{E}_{1/2}$ on μn-bit integers. We then use the previous analysis of the $\mathscr{E}_\alpha$ algorithm, and obtain a precise asymptotical value for the average bit complexity of the $\mathscr{LE}_\mu$ algorithm when it runs on n-bit integers. This value involves, together with the parameter μ, the constants M, D, resulting from the cost of a multiplication or a division, together with some constants L_1, L_2 that appear in the average bit complexity of the classical Euclid Algorithm. More precisely, the main result of the paper is the following (Theorem 2.3, proved in Section 6):

When the Lehmer–Euclid Algorithm deals with a truncation degree μ, its average bit complexity on pairs of length n is asymptotically equal to

$$\left[\frac{3}{2}(L_1 M + L_2 D)\mu + (L_1 + L_2)M\mu + (2-\mu)M\right] n^2,$$

https://doi.org/10.1017/S0963548304006261 Published online by Cambridge University Press

while the average bit complexity of the plain Euclid Algorithm on pairs of length n is asymptotically equal to $(L_1 M + L_2 D)n^2$. *Here,* L_1 *and* L_2 *are the two constants*

$$L_1 = \frac{12 \log^2 2}{\pi^2} \sim 0.58, \qquad L_2 = \frac{6 \log 2}{\pi^2} \log \prod_{k=0}^{\infty} \left(1 + \frac{1}{2^k}\right) \sim 0.66,$$

and M, D *are the constants resulting from the costs of a multiplication or a division.*

At this point, some important remarks have to be made. The Lehmer–Euclid Algorithm $\mathscr{LE}_\mu$ is useful only in the case when a large division is more expensive than a large multiplication. More precisely, the Lehmer–Euclid Algorithm may be less expensive than the Euclid Algorithm only if $2M < L_1 M + L_2 D$, which is always the case on most of the computers, since we usually have $D \geqslant 5M$. Suppose that it is the case. Then, all the values of μ are not convenient. For instance, in the case when $D = 5M$, we have to choose a truncation degree μ at most equal to 0.3. Generally speaking, if $D/M := \rho$, the maximal value μ_0 of μ is:

$$\mu_0 = 2 \frac{L_1 + L_2 \rho - 2}{5L_1 + L_2(3\rho + 2) - 2}.$$

For $\rho = 15$, and $\mu = 1/3$, the ratio between the bit complexity of the two algorithms (the Lehmer–Euclid Algorithm and the Euclid Algorithm) is close to 0.7. For $\rho = 30$, and $\mu = 1/10$, this ratio is close to 1/4.

Consider now a situation that may be found in 'real life'. When we replace long integers used in the RSA algorithm ($n = 1024$) by single-precision integers (32 bits), we use a truncation degree μ equal to 1/32. Suppose also that we work with $\rho = 30$. If we apply our (asymptotical) results (only true for $n \to \infty \ldots$), we find that the ratio between the bit complexity of the two algorithms (the Lehmer–Euclid Algorithm and the Euclid Algorithm) is now close to 1/6.

There is a general agreement between our theoretical results and the experimental curves obtained by Lercier in his thesis [19]. We reproduce these curves in Figure 1. The curves on the left part of the figure represent the execution times of the Euclid Algorithm, the Lehmer–Euclid Algorithm and the Binary Algorithm on two different workstations: DEC (on top) and SUN (bottom). The curves on the right part represent the speed-up between the algorithms: it is the ratio between the execution times of two algorithms. Here, the curves represent the speed-up of the Lehmer–Euclid and Binary Algorithms with respect to the Euclid Algorithm. Lercier insists on the fact that the implementation of divisions is quite different on DEC processors or on SUN processors. With our notation, the parameter ρ has thus two distinct values: its DEC value, and its SUN value. If the parameter μ could tend to 0, the limit ratio between the two algorithms should be equal to

$$L(\rho) = \frac{2}{L_1 + L_2 \rho}.$$

On the curves obtained by Lercier, this ratio is about 1/10 for DEC, and about 1/5 for SUN. This makes it possible to obtain (somewhat!) indirect values for parameter ρ; its DEC value is near 30, and its SUN value near 15. Moreover, for $n = 1024$, the DEC curve exhibits a ratio of 1/12, while the SUN curve shows a ratio of about 1/6. Finally, in [22],

https://doi.org/10.1017/S0963548304006261 Published online by Cambridge University Press

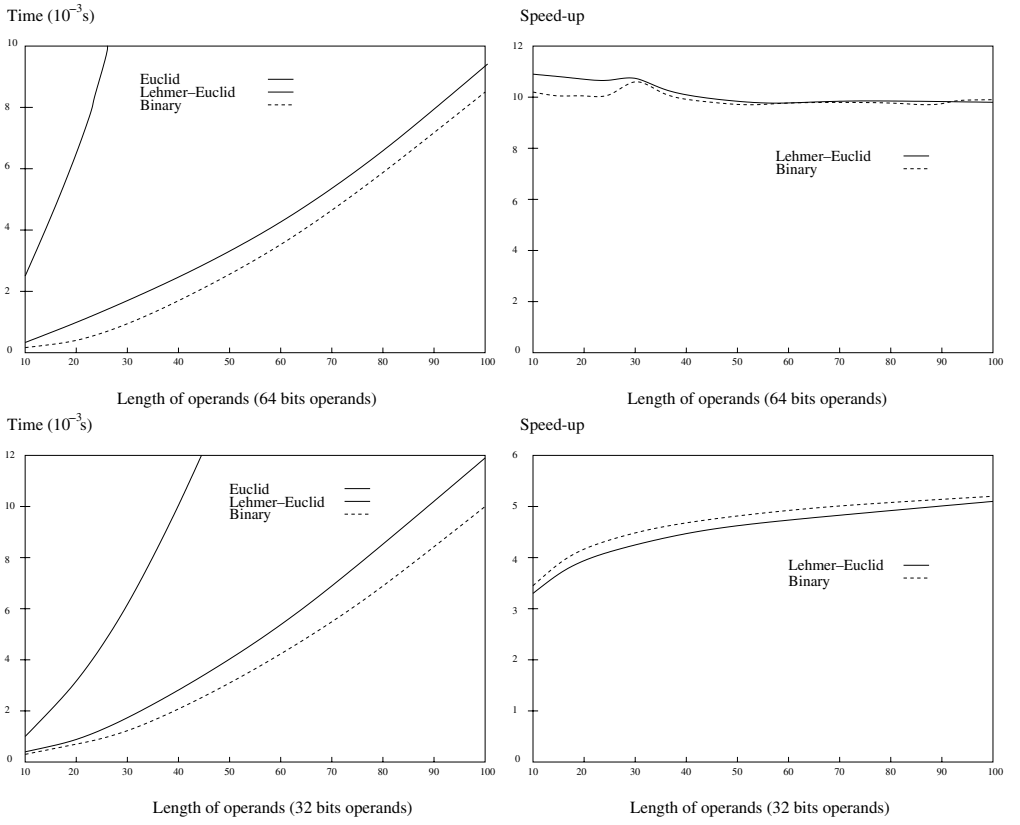


Figure 1. Comparison between three algorithms (from [19]): the plain Euclid Algorithm, the Lehmer–Euclid Algorithm and the Binary Algorithm. DEC workstation (top), SUN workstation (bottom); execution time (left) and speed-up (right)

Schönhage's implementation of the algorithms on a multitape Turing machine exhibits a speed-up of 2 for the Binary Algorithm and of 5/3 for the Lehmer–Euclid Algorithm, this speed-up still being with respect to the Euclid Algorithm.

Methods

Most of the variants of the classical Euclid algorithms have already been analysed, in the worst case as well as the average case. Heilbronn [12] and Dixon [9] in 1969 provided the first average-case analysis of the Euclid Algorithm. All further average-case analyses of Euclidean algorithms are instances of what we now call dynamical analysis. This method, due to Vallée, consists in viewing the algorithm as a dynamical system, where each step corresponds to an iteration of the algorithm. More precisely, this method relies on a description of relevant parameters by means of generating functions, now a common tool in the average-case analysis of algorithms [11]. As is usual in number theory contexts, the generating functions are Dirichlet series. They are first proved to be algebraically related to the so-called transfer operators that encapsulate all the important informations relative

https://doi.org/10.1017/S0963548304006261 Published online by Cambridge University Press

to the 'dynamics' of the algorithm. The analytical properties of Dirichlet series depend on spectral properties of the transfer operators, most notably the existence of a 'spectral gap' that separates the dominant eigenvalue from the remainder of the spectrum. This determines the singularities of the Dirichlet series of costs. The asymptotic extraction of coefficients is then achieved by means of Tauberian theorems [8], so that average complexity estimates finally result. The main thread of the method is thus adequately summarized by the chain:

$$\begin{aligned}\text{Euclidean algorithm} &\rightsquigarrow \text{associated transformations}\\ &\rightsquigarrow \text{transfer operator} \rightsquigarrow \text{Dirichlet series of costs}\\ &\rightsquigarrow \text{Tauberian inversion} \rightsquigarrow \text{average-case complexity.}\end{aligned}$$

In this way, Vallée studied a whole class of Euclidean algorithms, and this analysis leads to a classification into two subclasses [28]: the first one is formed of slow algorithms of log-squared average complexity, whereas the other class is formed of fast algorithms, of log average complexity. The same method provided the complete analysis of another widely used algorithm, the Binary Algorithm [26]. These methods are also suitable for performing bit complexity analyses: see [1, 27].

However, all the previous dynamical analyses dealt with algorithms that exhibit a simple structure, so that it is easy to relate the algorithm to the underlying dynamical system. Here, the structure of the Lehmer–Euclid Algorithm is more intricate, since the algorithm can be described as a sequence of internal loops. This is why this analysis is also a kind of test for the dynamical analysis methodology.

Plan of the paper

Section 2 is an introductory section that explains the framework, and describes the main algorithms, the Interrupted Euclid Algorithm $\mathscr{E}_\alpha$ with interruption degree α, and the Lehmer–Euclid Algorithm $\mathscr{LE}_\mu$ with truncation degree μ. Here, the main cost parameters are defined and the theorems are stated. Section 3 describes the general framework of dynamical analysis methodology, and each following section is devoted to the proof of one of the main three theorems: Section 4 for Theorem 2.1, which describes the average number of iterations of the $\mathscr{E}_\alpha$ algorithm; Section 5 for Theorem 2.2, which describes the average bit complexity of the $\mathscr{E}_\alpha$ algorithm; finally, Section 6 for Theorem 2.3, which states the main result of this paper, *i.e.*, the average bit complexity of the $\mathscr{LE}_\mu$ algorithm.

2. The Lehmer–Euclid Algorithm

This introductory section presents the main algorithms to be studied: the plain Euclid Algorithm, the Interrupted Euclid Algorithm and finally the Lehmer–Euclid Algorithm. We describe here the parameters of interest, and the probabilistic models. We state the three main theorems.

2.1. The Euclid Algorithm

Let (A_0, A_1) be a pair of positive integers with $A_1 \leqslant A_0$. When running on the input (A_0, A_1), the Euclid Algorithm computes the remainder sequence (A_i) defined by

$$A_{i+2} = (A_i \bmod A_{i+1}), \qquad A_{p+1} = 0,$$

https://doi.org/10.1017/S0963548304006261 Published online by Cambridge University Press

and the last nonzero remainder A_p is the gcd of (A_0, A_1). This computation is then a succession of divisions of the form

$$A_i = Q_{i+1}A_{i+1} + A_{i+2} \quad \text{with} \quad Q_{i+1} = \left\lfloor \frac{A_i}{A_{i+1}} \right\rfloor. \tag{2.1}$$

The integer Q_i is called the ith quotient and the successive divisions can be written as

$$\mathscr{A}_i = \mathscr{Q}_{i+1}\mathscr{A}_{i+1}, \quad \text{with} \quad \mathscr{A}_i := \begin{pmatrix} A_{i+1} \\ A_i \end{pmatrix} \quad \text{and} \quad \mathscr{Q}_i := \begin{pmatrix} 0 & 1 \\ 1 & Q_i \end{pmatrix}, \tag{2.2}$$

so that $\mathscr{A}_0 = \mathscr{M}_i\mathscr{A}_i$ with $\mathscr{M}_i := \mathscr{Q}_1\mathscr{Q}_2\cdots\mathscr{Q}_i$.

In addition to the quotient sequence (Q_i), the Extended Euclid Algorithm computes step by step the matrix $\mathscr{M}_i$ and its inverse $\mathscr{M}_i^{-1}$,

$$\mathscr{M}_i^{-1} := \begin{pmatrix} V_{i+1} & U_{i+1} \\ V_i & U_i \end{pmatrix}, \qquad \mathscr{M}_i := (-1)^i \begin{pmatrix} U_i & -U_{i+1} \\ -V_i & V_{i+1} \end{pmatrix}, \tag{2.3}$$

which involve two other sequences (U_i) and (V_i) defined by

$$\begin{pmatrix} V_1 & U_1 \\ V_0 & U_0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} V_{i+1} & U_{i+1} \\ V_i & U_i \end{pmatrix} = \begin{pmatrix} -Q_i & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} V_i & U_i \\ V_{i-1} & U_{i-1} \end{pmatrix}.$$

Then, for any $i, 0 \leqslant i \leqslant p$, the equality $A_i = A_0\,U_i + A_1\,V_i$ holds, and the final coefficients U_p and V_p form the Bezout pair (U, V) that satisfies $UA_0 + VA_1 = \gcd(A_0, A_1)$.

2.2. Main principles of the Lehmer–Euclid Algorithm

In spite of its simplicity, the Euclid Algorithm is not well suited to large integers. Indeed, Euclidean divisions are quite time-consuming, especially for large integers. The cost of a division of two large integers may be higher than the cost of the multiplication of two large integers. Lehmer's idea is the following: *Replace a sequence of large divisions by a sequence of small divisions and small multiplications followed by some large multiplications.*

From here on, we let $\ell(x) := \lfloor \log_2 x \rfloor + 1$ denote the binary length of a positive integer x. We consider a valid input (A_0, A_1) of the Euclid Algorithm. It satisfies $A_1 \leqslant A_0$, and its length n is (by definition) the binary length of A_0. For some $m \leqslant n$, the truncated pair (a_0, a_1), defined by

$$a_0 := \lfloor A_0/2^{n-m} \rfloor, \qquad a_1 := \lfloor A_1/2^{n-m} \rfloor,$$

is built by erasing the $n-m$ least significant digits of A_0 and A_1: it is of length m.

In fact, Lehmer suggests using the first steps of the Extended Euclid Algorithm on the small pair (a_0, a_1) to simulate the first steps of the Euclid Algorithm on the large pair (A_0, A_1).

More precisely, the Extended Euclid Algorithm is applied to the pair (a_0, a_1), and provides the remainder sequence (a_i), the quotient sequence (q_i) and the two co-sequences (u_i) and (v_i). Since the two rationals A_1/A_0 and a_1/a_0 are close, one can expect that the two quotient sequences, the sequence q_i obtained from the Euclid Algorithm on (a_0, a_1) and the sequence Q_i obtained from the Euclid Algorithm on (A_0, A_1) are not too different, at least at the beginning of the process. There always exists some integer $j \geqslant 0$ for which the two quotient sequences q_i and Q_i are equal for $i \leqslant j$. Since the equalities $(u_i, v_i) = (U_i, V_i)$

https://doi.org/10.1017/S0963548304006261 Published online by Cambridge University Press

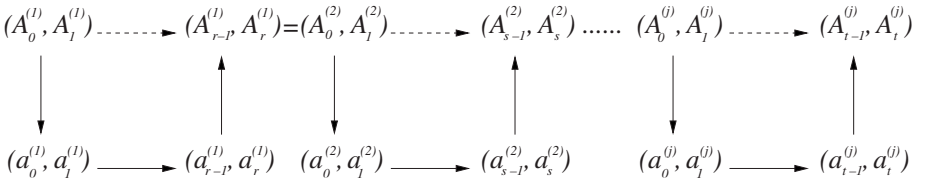


Figure 2. General principle of the Lehmer–Euclid Algorithm

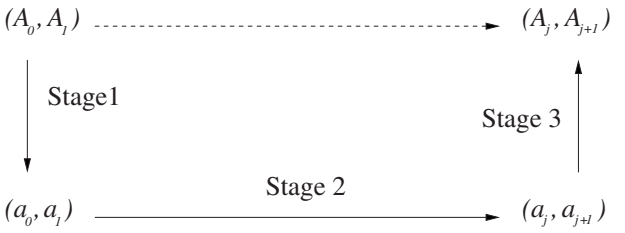


Figure 3. Principle of the Lehmer–Euclid Algorithm: one phase

hold for $i \leqslant j+1$, it is then possible to compute the large integers A_j and A_{j+1} that would have been obtained when performing j steps of the Euclid Algorithm on the input (A_0, A_1): the coefficients of matrix $\mathscr{M}_j$ are computed from the small co-sequences $(u_i), (v_i)$ so that, with (2.2) and (2.3),

$$A_j = v_j A_0 + u_j A_1, \qquad A_{j+1} = v_{j+1} A_0 + u_{j+1} A_1.$$

The main problem is now to evaluate a possible value of the index j without computing the quotients Q_i of the large sequence. There exist many different possible tests, due to Lehmer [18], Collins [7] or Jebelean [14], that are in fact very close; we choose the following one:

$$\text{If} \quad a_j > a_0^{1/2} \quad \text{then} \quad Q_i = q_i, \quad \text{for all } i \leqslant j-2.$$

Therefore, if r is the first index for which $a_r \leqslant a_0^{1/2}$, then the two sequences Q_i and q_i are the same until $i = r-3$, and it is thus possible to recover the value of the pair (A_{r-3}, A_{r-2}) without performing the large divisions.

2.3. The Lehmer–Euclid Algorithm

More precisely, the Lehmer–Euclid Algorithm can be viewed as a sequence of phases, each phase replacing a sequence of some steps of the Euclid Algorithm. Each phase is formed by three different stages (see Figure 3).

Stage 1 is a truncation step which replaces a large pair (A_0, A_1) by the small pair (a_0, a_1).

Stage 2 can be defined as an 'Interrupted Euclid Algorithm' that performs only the first iterations of the Euclid Algorithm on input (a_0, a_1). This algorithm stops, after r iterations, as soon as the current integer a_i becomes smaller than $\sqrt{a_0}$.

https://doi.org/10.1017/S0963548304006261 Published online by Cambridge University Press

Algorithm $\mathscr{LE}_\mu$.
Input. An integer pair (A, B) with $0 \leqslant B \leqslant A$.

Initialization $n := \ell(A)$; $m := \lfloor \mu n \rfloor$; $A_0 := A$; $A_1 := B$;
While $\ell(A_0) > m$ **do**
(1) $a_0 := T_m(A_0)$; $a_1 := T_m(A_1)$;
(2) $i := 1$; $u_0 := 1$; $u_1 := 0$; $v_0 := 0$; $v_1 := 1$;
While $a_i > a_0^{1/2}$ **do**
$q_i := a_{i-1} \operatorname{div} a_i$; $a_{i+1} := a_{i-1} \bmod a_i$;
$u_{i+1} := -u_i q_i + u_{i-1}$; $v_{i+1} := -v_i q_i + v_{i-1}$;
$i := i + 1$;
(3) $A' := u_{i-3} A_0 + v_{i-3} A_1$; $B' := u_{i-2} A_0 + v_{i-2} A_1$;
$A_0 := A'$; $A_1 := B'$;
$i := 1$;
While $A_i > 0$ **do**
$A_{i+1} := A_{i-1} \bmod A_i$;
$i := i + 1$;
Output. $\gcd(A, B) := A_{i-1}$.

Figure 4. The Lehmer–Euclid Algorithm with truncation degree μ

Stage 3 is a multiplication step that recovers the value of the large pair (A_{r-3}, A_{r-2}) which would have been computed with $r-3$ iterations of the Euclid Algorithm on the input pair (A_0, A_1).

The Lehmer–Euclid Algorithm mainly depends on the choice of the truncation. Here, we work with truncations that are proportional to the length of the input. When the input pair (A_0, A_1) has length n, each truncation step will produce a pair of length m, with $m := \lfloor \mu n \rfloor$ for some fixed parameter $\mu \in]0, 1]$. This parameter μ is called the truncation degree, and the corresponding Lehmer–Euclid Algorithm is denoted by $\mathscr{LE}_\mu$.

We work with the set $\widetilde{\Omega}$ of valid inputs and with the subsets $\widetilde{\Omega}_n$ of inputs of length n,

$$\widetilde{\Omega} = \{(A_0, A_1);\ 0 < A_1 \leqslant A_0\},$$
$$\widetilde{\Omega}_n := \{(A_0, A_1) \in \widetilde{\Omega};\ \ell(A_0) = n\} = \{(A_0, A_1) \in \widetilde{\Omega};\ 2^{n-1} \leqslant A_0 < 2^n\}.$$

If the input pair has length n, each truncation step uses the truncation mapping $T_m : \widetilde{\Omega}_n \to \widetilde{\Omega}_m$ with $m := \lfloor \mu n \rfloor$. This mapping creates, from a current pair (A', B') of length k, a pair (a', b') of length m obtained by erasing the $k - m$ least significant digits of A' and B',

$$T_m(A', B') := \begin{cases} \big(\lfloor A'/2^{k-m} \rfloor, \lfloor B'/2^{k-m} \rfloor\big), & \text{if } k \geqslant m, \\ (0, 0), & \text{else.} \end{cases}$$

The Lehmer–Euclid Algorithm $\mathscr{LE}_\mu$ is described in Figure 4.

https://doi.org/10.1017/S0963548304006261 Published online by Cambridge University Press

Input. An integer pair (a, b) with $b \leqslant a$.
$a_0 := a;\ a_1 := b;$
$u_0 := 1;\ u_1 := 0;\ v_0 := 0;\ v_1 := 1;$
$i := 1;$
While $a_i > a_0^{\alpha}$ **do**
$q_i := a_{i-1} \operatorname{div} a_i;\ a_{i+1} := a_{i-1} \bmod a_i;$
$u_{i+1} := -u_i q_i + u_{i-1};\ v_{i+1} := -v_i q_i + v_{i-1};$
$i := i + 1$
Output. An integer pair (a_i, a_{i+1}) that satisfies
$a_{i+1} \leqslant a_0^{\alpha} < a_i$ and $\gcd(a_0, a_1) = \gcd(a_{i+1}, a_i)$.

Figure 5. The Interrupted Euclid Algorithm $\mathscr{E}_{\alpha}$

2.4. The Interrupted Euclid Algorithm

We are mainly interested in the analysis of Stage 2, since it provides the value r that determines the length of a phase, *i.e.*, the number of large divisions that can be replaced by small ones. More generally, we wish to study a more general algorithm whose interruption depends on a real parameter $\alpha \in [0, 1]$ (*cf.* Figure 5). When running on an input (a_0, a_1), this algorithm, denoted by $\mathscr{E}_{\alpha}$, stops as soon as the current integer a_i becomes smaller than a_0^{α}.

Then, Stage 2 in the Lehmer–Euclid Algorithm is exactly the Interrupted Algorithm $\mathscr{E}_{1/2}$ applied on truncated integers. For the moment, we forget the Lehmer–Euclid Algorithm and focus on the analysis of the general Interrupted Algorithm $\mathscr{E}_{\alpha}$ for any value $\alpha \in [0, 1]$ and any input pair of integers. This algorithm is described in Figure 5.

We work with sets $\Omega, \widetilde{\Omega}$ of valid inputs and with sets $\Omega_n, \widetilde{\Omega}_n$ of valid inputs of binary length n,

$$\Omega = \{(a_0, a_1);\ 0 < a_1 < a_0, \gcd(a_0, a_1) = 1\}, \qquad \widetilde{\Omega} = \{(a_0, a_1);\ 0 < a_1 < a_0\},$$
$$\Omega_n := \{(a_0, a_1) \in \Omega;\ 2^{n-1} \leqslant a_0 < 2^n\}, \qquad \widetilde{\Omega}_n := \{(a_0, a_1) \in \widetilde{\Omega};\ 2^{n-1} \leqslant a_0 < 2^n\}.$$

The following definition makes precise the probabilistic model.

Definition. Let f be a positive function defined on the unit interval I. We say that Ω (resp. $\widetilde{\Omega}$) is endowed with f if any element (a_0, a_1) of Ω is weighted with the quantity $f(\frac{a_1}{a_0})$.

This framework defines a sequence of probabilistic models on subsets $\Omega_n, \widetilde{\Omega}_n$. For each n, the corresponding probabilities and expectations on $\Omega_n, \widetilde{\Omega}_n$ are denoted by $\Pr_n, \mathrm{E}_n$; if we wish to insist on the dependence on function f, we put it as an exponent. Then, the symbols $\Pr_n^{(f)}[B]$ (for any subset $B \subset \widetilde{\Omega}$ or Ω) and $\mathrm{E}_n^{(f)}[X]$ (for any variable X defined on Ω or $\widetilde{\Omega}$) denote the following quantities:

$$\Pr_n^{(f)}[B] := \frac{\sum_{(a_0,a_1)\in\Omega_n\cap B} f\left(\frac{a_1}{a_0}\right)}{\sum_{(a_0,a_1)\in\Omega_n} f\left(\frac{a_1}{a_0}\right)}, \qquad \mathrm{E}_n^{(f)}[X] := \frac{\sum_{(a_0,a_1)\in\Omega_n} X(a_0,a_1) f\left(\frac{a_1}{a_0}\right)}{\sum_{(a_0,a_1)\in\Omega_n} f\left(\frac{a_1}{a_0}\right)}.$$

The exponent (f) will be omitted if not necessary.

https://doi.org/10.1017/S0963548304006261 Published online by Cambridge University Press

2.5. The Interrupted Algorithm: number of iterations

Our first main result relates the number P_α of iterations of the algorithm $\mathscr{E}_\alpha$ to the number P of iterations of the Euclid Algorithm $\mathscr{E}$. This theorem will be proved in Section 4.

Theorem 2.1. *Suppose that the valid sets $\Omega, \widetilde{\Omega}$ are endowed with some positive function f with bounded variation on the unit interval I. Let $\mathscr{E}$ denote the Euclid Algorithm, let $\mathscr{E}_\alpha$ be the Interrupted Euclid Algorithm of parameter α, let P_α be the number of iterations of $\mathscr{E}_\alpha$, and let P be the number of iterations of $\mathscr{E}$. Then:*

(i) *for any $\varepsilon > 0$, there exists some $K < 1$ such that*

$$\Pr_n\left[\left|\frac{P_\alpha}{P} - (1-\alpha)\right| > \varepsilon\right] = O(K^n), \quad \text{when } n \to \infty;$$

(ii) *the expectations of these costs on Ω_n or on $\widetilde{\Omega}_n$ satisfy, when $n \to \infty$,*

$$\mathrm{E}_n[P_\alpha] \sim (1-\alpha)\,\mathrm{E}_n[P] \sim (1-\alpha)\frac{12\log^2 2}{\pi^2}n.$$

Remark. The random variable P was first analysed around 1970 by Dixon [9] and Heilbronn [12], who proved independently that

$$\mathrm{E}_n[P] \sim \frac{12\log^2 2}{\pi^2}n,$$

in the case when $f \cong 1$. More recently, in the particular case $f \cong 1$, Hensley [13] proved that the random variable P asymptotically follows a normal law. He expressed the expectation and the variance with some function $\Lambda(s)$ which will play a fundamental rôle in this paper,

$$\mathrm{E}_n[P] \sim \frac{-\log 2}{\Lambda'(2)}n, \quad \mathrm{Var}_n[P] \sim -\log 2\frac{\Lambda''(2)}{\Lambda'(2)^3}n.$$

Remark. In (i), one can choose

$$K = 2^{-\eta\varepsilon^2} \quad \text{with} \quad \eta = |\Lambda'(2)|/\Lambda''(2).$$

2.6. The Interrupted Euclid Algorithm: bit complexity

The operations that are performed by the Euclid Algorithm $\mathscr{E}$ do not all have the same cost. The Extended Euclid Algorithm performs exchanges, multiplications and divisions. We let M denote the constant that arises in the multiplication and exchange costs, and let D be the constant that arises in the division cost. Then the cost of a multiplication between two integers u and v has a cost equal to $M \cdot \ell(u) \cdot \ell(v)$, while the cost of an exchange between u and v is $M(\ell(u) + \ell(v))$.

A Euclidean division $v = uq + r$ has a bit cost equal to $D \cdot \ell(u) \cdot \ell(q)$. It is followed by two exchanges, whose cost are $M \cdot \ell(u)$. When the algorithm $\mathscr{E}$ performs p iterations on input (a_0, a_1), the total bit cost B of the execution of the algorithm

$$B(a_0, a_1) = \sum_{i=1}^{p} \ell(a_i) \cdot b(q_i) \quad \text{with} \quad b(q) := D \cdot \ell(q) + 2M \tag{2.4}$$

involves both the quotient sequence (q_i) and the remainder sequence (a_i).

https://doi.org/10.1017/S0963548304006261 Published online by Cambridge University Press

For computing the Bezout coefficients, the Extended Euclid Algorithm performs multiplications and exchanges, so that the supplementary bit cost C for one Bezout coefficient v_i,

$$C(a_0, a_1) = \sum_{i=1}^{p} \ell(v_i) \cdot c(q_i), \quad \text{with} \quad c(q) := (\ell(q) + 2)M, \tag{2.5}$$

involves both the quotient sequence (q_i) and the co-sequence (v_i).

Theorem 2.1 proves that the algorithm $\mathscr{E}_\alpha$ stops (almost surely) at the $\lfloor (1-\alpha)P \rfloor$th iteration. We introduce another truncated algorithm, which we denote by $\overline{\mathscr{E}}_\alpha$, that stops exactly at the $\lfloor (1-\alpha)P \rfloor$th iteration: it is a regularized version of the $\mathscr{E}_\alpha$ algorithm. This algorithm – not very realistic, since it must 'guess' the value of P – is just a tool for the analysis; the following theorem shows that it behaves asymptotically in the same way as $\mathscr{E}_\alpha$ and it is easier to analyse. This theorem will be proved in Section 5.

Theorem 2.2. *Suppose that the valid sets $\Omega, \widetilde{\Omega}$ are endowed with some function f with bounded variation on the unit interval I. Let $\mathscr{E}$ denote the Euclid Algorithm, and let $\mathscr{E}_\alpha$ be the Interrupted Euclid Algorithm of parameter α. We consider three measures of cost for the Interrupted Algorithm $\mathscr{E}_\alpha$: the bit cost B_α, the supplementary bit cost C_α due to the computation of one Bezout coefficient, and the total bit cost E_α of the Extended Euclid Algorithm (i.e., $E_\alpha = B_\alpha + 2C_\alpha$). With a bar (i.e., $\overline{B}_\alpha, \overline{C}_\alpha, \overline{E}_\alpha$), the same quantities denote the corresponding costs for the $\overline{\mathscr{E}}_\alpha$ algorithm, and without index α (i.e., B, C, E) for the Euclid Algorithm $\mathscr{E}$.*

Then, for any positive function f with bounded variation on I, we have, for $n \to \infty$,

(i) $\mathrm{E}_n[B_\alpha] \sim \mathrm{E}_n[\overline{B}_\alpha] \sim (1-\alpha^2)\,\mathrm{E}_n[B]$,
(ii) $\mathrm{E}_n[C_\alpha] \sim \mathrm{E}_n[\overline{C}_\alpha] \sim (1-\alpha)^2\,\mathrm{E}_n[C]$,
(iii) $\mathrm{E}_n[E_\alpha] \sim \mathrm{E}_n[\overline{E}_\alpha] \sim \frac{1}{3}(1-\alpha)(3-\alpha)\,\mathrm{E}_n[E]$.

Remark. The random variables B, C, E were first analysed by Akhavi and Vallée [1, 27] in the case when $f \cong 1$. With the notation used here, their results can be translated in the following way:

$$\mathrm{E}_n[B] \sim (L_1 M + L_2 D)\, n^2, \qquad \mathrm{E}_n[C] \sim (L_1 + L_2) M\, n^2,$$
$$\text{with} \quad L_1 = \frac{12 \log^2 2}{\pi^2} \sim 0.58, \qquad L_2 = \frac{6 \log 2}{\pi^2} \log \prod_{k=0}^{\infty} \left(1 + \frac{1}{2^k}\right) \sim 0.66.$$

In the remainder of this section, we come back to the Lehmer–Euclid Algorithm $\mathscr{LE}_\mu$. We recall that we work with a truncation degree μ: from an input (A_0, A_1) of $\widetilde{\Omega}_n$ or Ω_n, we deal with pairs of length m with $m := \lfloor \mu n \rfloor$, and we always use the truncation map T_m, which creates a new input (a_0, a_1) belonging to $\widetilde{\Omega}_m$. Here, we wish to give an idea of what we can expect about the asymptotic behaviour of the Lehmer–Euclid Algorithm, if we 'insert' in it our previous results about the Interrupted Algorithm $\mathscr{E}_\alpha$. Later in this paper – namely Section 6 – we will prove these facts.

https://doi.org/10.1017/S0963548304006261 Published online by Cambridge University Press

2.7. Comparison between the Lehmer–Euclid Algorithm and the Euclid Algorithm: the first phase

If the algorithm starts with $f \cong 1$, the distribution is uniform for the inputs (A_0, A_1) in $\widetilde{\Omega}_n$, and the distribution of the truncated pairs (a_0, a_1) is also uniform on $\widetilde{\Omega}_m$.

Then Stage 2 is exactly the Interrupted Algorithm $\mathscr{E}_{1/2}$ on integers of length $m := \lfloor \mu n \rfloor$, and stops when the length of the current integers a_i, u_i, v_i is about $m/2$. In Stage 3, we compute four products, each between an integer of length n and an integer of length $m/2$ (see Figure 3). The output integers (A', B') now have a length of about $n(1 - \frac{\mu}{2})$. Consequently, the top horizontal line is the Interrupted Algorithm $\mathscr{E}_{1-(\mu/2)}$ on integers of length n.

Then we can easily compare the two costs, and the expressions involve the length n of the inputs, the degree μ of integer truncation, the constant M that arises in a multiplication cost, and the constant D that arises in a division cost. The constants L_1, L_2 are the constants that are important in the Euclid bit cost.

The cost of the top horizontal line corresponding to the first phase is equal to

$$\left(1 - \left(1 - \frac{\mu}{2}\right)^2\right)(L_1 M + L_2 D)n^2 = \left[\left(\mu - \frac{\mu^2}{4}\right)(L_1 M + L_2 D)\right]n^2.$$

This cost has to be compared with the total cost of the first phase (Stages 2 and 3), which is equal to

$$\left[\frac{3}{4}(L_1 M + L_2 D)\mu^2 + \frac{1}{2}(L_1 + L_2)M\mu^2 + 2\mu M\right]n^2.$$

2.8. Comparison between the Lehmer–Euclid Algorithm and the Euclid Algorithm: the other phases

During each phase, the length of the small pair decreases by a quantity equal to $m/2$. If this is also true for the large pair, then the average number of phases would be around $2/\mu$. At the end of the jth phase, one computes four products, each between an integer of length $m/2$ and an integer of length $(1 - (j-1)(\mu/2))n$ (that is, the length of the large integers at the beginning of the jth phase). If the argument given in the previous paragraph could be repeated for each phase, the average cost of the jth phase would be

$$\frac{3}{4}(L_1 M + L_2 D)\mu^2 n^2 + \frac{1}{2}(L_1 + L_2)M\mu^2 n^2 + 4\frac{\mu}{2}\left(1 - (j-1)\frac{\mu}{2}\right)Mn^2.$$

The average cost of the Lehmer–Euclid Algorithm would then be

$$\left[\frac{3}{2}(L_1 M + L_2 D)\mu + (L_1 + L_2)M\mu + (2 - \mu)M\right]n^2,$$

which can be compared to the bit cost of the classical Euclid Algorithm, that is, $(L_1 M + L_2 D)n^2$.

2.9. The final result

However, the previous argument cannot be repeated *a priori* for the other phases, because the distribution on Ω is modified by the execution of the Euclid Algorithm. At the beginning of the second phase, the distribution of the new inputs at Stage 1 is not the

https://doi.org/10.1017/S0963548304006261 Published online by Cambridge University Press

same as at the beginning of the algorithm. However, the evolution of the distribution of the integers during the execution of the Euclid Algorithm can be precisely described with tools of Dynamical Analysis. The main idea is then to simulate the cost of the Lehmer–Euclid Algorithm only on the Euclid Algorithm itself. Then all the previous remarks will be proved and we shall obtain our final result.

Theorem 2.3. *Suppose that the valid sets $\Omega, \widetilde{\Omega}$ are endowed with some function f with bounded variation on the unit interval I. When the Lehmer–Euclid Algorithm deals with a truncation degree μ, its average bit complexity on pairs of length n is asymptotically equal to*

$$\left[\frac{3}{2}(L_1 M + L_2 D)\mu + (L_1 + L_2)M\mu + (2-\mu)M\right] n^2.$$

Here, L_1 and L_2 are the two constants that appear in the average bit complexity of the Euclid Algorithm,

$$L_1 = \frac{12\log^2 2}{\pi^2} \sim 0.58, \qquad L_2 = \frac{6\log 2}{\pi^2}\log\prod_{k=0}^{\infty}\left(1+\frac{1}{2^k}\right) \sim 0.66,$$

and M, D are the constants resulting from the costs of a multiplication or a division.

In the rest of the paper, we shall prove the three theorems successively. We first recall the general methodology of what we call a dynamical analysis.

3. Principles of dynamical analysis

This method uses tools that come from dynamical systems theory, mainly transfer operators.

3.1. Dirichlet series and Tauberian theorem

We are interested in analysing some costs, and, from here on, we deal with the generating Dirichlet series of these costs. To any cost X, and to any weight defined on I, we associate Dirichlet series

$$F_X(s) = \sum_{(a_0,a_1)\in\Omega} \frac{X(a_0,a_1)}{a_0^s} f\left(\frac{a_1}{a_0}\right), \qquad \widetilde{F}_X(s) = \sum_{(a_0,a_1)\in\widetilde{\Omega}} \frac{X(a_0,a_1)}{a_0^s} f\left(\frac{a_1}{a_0}\right).$$

Then,

$$F_X(s) = \sum_{a\geqslant 1} \frac{x_a}{a^s}, \qquad \widetilde{F}_X(s) = \sum_{a\geqslant 1} \frac{\widetilde{x}_a}{a^s},$$

where x_a, $\widetilde{x}_a$ denote the cumulative costs of X on

$$\omega_a := \{(a_0,a_1)\in\Omega; a_0 = a\}, \qquad \widetilde{\omega}_a := \{(a_0,a_1)\in\widetilde{\Omega}; a_0 = a\}.$$

https://doi.org/10.1017/S0963548304006261 Published online by Cambridge University Press

For the trivial cost, t_a or $\widetilde{t}_a$ are just the weights of subsets $\omega_a, \widetilde{\omega}_a$. The mean values of the cost X on $\Omega_n, \widetilde{\Omega}_n$ are then given by the ratio of partial sums,

$$\mathrm{E}_n[X] = \frac{\sum_{\ell(a)=n} x_a}{\sum_{\ell(a)=n} t_a}, \qquad \widetilde{\mathrm{E}}_n[X] = \frac{\sum_{\ell(a)=n} \widetilde{x}_a}{\sum_{\ell(a)=n} \widetilde{t}_a}. \tag{3.1}$$

The asymptotics of such partial sums can be provided when applying the following Tauberian theorem, due to Delange [8].

Tauberian Theorem (Delange). *Let $F(s)$ be a Dirichlet series with nonnegative coefficients*

$$F(s) = \sum_{a \geqslant 1} \frac{x_a}{a^s}.$$

Assume that

(i) *$F(s)$ converges for $\Re(s) > \sigma > 0$ and is analytic on $\Re(s) = \sigma, s \neq \sigma$,*
(ii) *for some $\theta \geqslant 0$, we have, for s near σ,*

$$F(s) = \frac{A(s)}{(s-\sigma)^{\theta+1}} + C(s),$$

where A, C are analytic at σ, with $A(\sigma) \neq 0$.

Then, as $n \to \infty$,

$$\sum_{\ell(a)=n} x_a = \left[\left(1 - \frac{1}{2^\sigma}\right)(\log 2)^\theta \frac{A(\sigma)}{\sigma\Gamma(\theta+1)}\right] 2^{n\sigma} n^\theta \quad [1 + \varepsilon(n)], \quad \varepsilon(n) \to 0.$$

3.2. Costs of interest

We now describe the main costs that intervene in this paper. Theorem 2.1 deals with the parameter P_α, which denotes the number of iterations of the $\mathscr{E}_\alpha$ algorithm, and we are mainly interested in the events $[P_\alpha > \delta P]$ and $[P_\alpha \leqslant \delta P]$, with $\delta \in [0,1]$. Since the algorithm stops as soon as $a_i \leqslant a_0^\alpha$, these events satisfy

$$[P_\alpha > \delta P] = [P_\alpha > \lfloor \delta P \rfloor] = \left[\frac{a_{\lfloor \delta P \rfloor}}{a_0^\alpha} > 1\right],$$
$$[P_\alpha \leqslant \delta P] = [P_\alpha \leqslant \lfloor \delta P \rfloor] = \left[\frac{a_{\lfloor \delta P \rfloor}}{a_0^\alpha} \leqslant 1\right],$$

and Markov's inequality entails

$$\Pr_n[P_\alpha > \delta P] \leqslant \mathrm{E}_n\left[\left(\frac{a_{\lfloor \delta P \rfloor}}{a_0^\alpha}\right)^\gamma\right], \quad \text{for all } \gamma > 0, \tag{3.2}$$

$$\Pr_n[P_\alpha \leqslant \delta P] \leqslant \mathrm{E}_n\left[\left(\frac{a_{\lfloor \delta P \rfloor}}{a_0^\alpha}\right)^\gamma\right], \quad \text{for all } \gamma < 0. \tag{3.3}$$

We are then led to define the first cost as

$$M(a_0, a_1) = \left(\frac{a_{\lfloor \delta P \rfloor}}{a_0^\alpha}\right)^\gamma. \tag{3.4}$$

https://doi.org/10.1017/S0963548304006261 Published online by Cambridge University Press

This cost depends on three parameters, namely the interruption parameter α, the fraction δ of the depth of the continued fraction expansion, and finally γ, which appears in Markov's inequality.

Section 2.6 explained why the main costs studied in Theorem 2.2, namely $\overline{B}_\alpha$ and $\overline{C}_\alpha$, involve the main parameters of the Euclid Algorithm: the depth P, and the four main sequences that are computed, the quotients (q_i), the remainders (a_i), and the two co-sequences $(u_i), (v_i)$. We have

$$\overline{B}_\alpha(a_0, a_1) \sim \sum_{i=1}^{\lfloor (1-\alpha)P \rfloor} \ell(a_i) \cdot b(q_i), \qquad \overline{C}_\alpha(a_0, a_1) \sim \sum_{i=1}^{\lfloor (1-\alpha)P \rfloor} \ell(u_i) \cdot c(q_i), \tag{3.5}$$

with

$$b(q) := \ell(q)D + 2M, \qquad c(q) := (\ell(q) + 2)M. \tag{3.6}$$

We shall provide alternative expressions for the corresponding Dirichlet series of costs in Sections 4 and 5. These expressions deal with the transfer operator $\mathbf{H}_s$ relative to the Euclidean dynamical system. Then, the singularities of the Dirichlet series will become apparent and related to the dominant spectral objects of the transfer operator $\mathbf{H}_s$. We now describe this dynamical system and present the operator $\mathbf{H}_s$ together with its main spectral properties.

3.3. The Euclidean dynamical system

When computing the gcd of the integer-pair (a_0, a_1), Euclid's algorithm performs a sequence of p iterations of the form

$$a_0 = q_1 a_1 + a_2, \qquad a_1 = q_2 a_2 + a_3, \ldots, \qquad a_{p-1} = q_p a_p. \tag{3.7}$$

This sequence can be associated with the map S that satisfies

$$S\left(\frac{a_1}{a_0}\right) = \frac{a_2}{a_1}, \ldots, S\left(\frac{a_i}{a_{i-1}}\right) = \frac{a_{i+1}}{a_i}, \ldots, S\left(\frac{a_p}{a_{p-1}}\right) = 0.$$

When extended to the real interval $I = [0, 1]$, this map corresponds to the classical continued fraction expansion algorithm, and is defined by

$$S(x) = \frac{1}{x} - \left\lfloor \frac{1}{x} \right\rfloor, \quad \textit{i.e.}, \quad S(x) = \frac{1}{x} - q \quad \text{for} \quad x \in \left[\frac{1}{q+1}, \frac{1}{q}\right].$$

The pair (I, S) defines the dynamical system relative to the Euclid Algorithm. We let $\mathscr{H}$ denote the set of the inverse branches of S,

$$\mathscr{H} = \left\{ x \mapsto \frac{1}{q + x}; q \geqslant 1 \right\},$$

and let $\mathscr{H}^n$ be the set of inverse branches of depth n (*i.e.*, the set of inverse branches of S^n),

$$\mathscr{H}^n = \{ h = h_1 \circ \cdots \circ h_n; h_i \in \mathscr{H}, \forall i \}.$$

https://doi.org/10.1017/S0963548304006261 Published online by Cambridge University Press

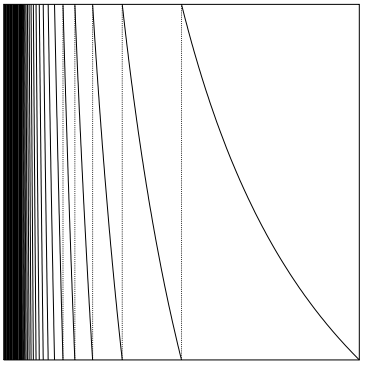


Figure 6. Euclidean dynamical system

Then the sequence (3.7) builds a continued fraction

$$\frac{a_1}{a_0} = \cfrac{1}{q_1 + \cfrac{1}{q_2 + \cfrac{1}{q_3 + \cfrac{1}{\ddots + \cfrac{1}{q_p}}}}}, \tag{3.8}$$

and can be written as

$$\frac{a_1}{a_0} = h(0) \quad \text{with} \quad h = h_1 \circ h_2 \circ \ldots \circ h_p \in \mathscr{H}^p. \tag{3.9}$$

We then associate to each execution of the algorithm a unique LFT (Linear Fractional Transformation) h whose depth is exactly the number p of divisions performed.

Note that all quantities of interest appear on the continued fraction (3.8) or in the decomposition (3.9). The depth P equals its height; note that the ith LFT h_i used by the algorithm is exactly the LFT relative to matrix $\mathscr{Q}_i$ of Section 2.1, so that the LFT $h_1 \circ h_2 \circ \ldots \circ h_i$ is relative to matrix $\mathscr{M}_i$ of Section 2.1. Then, when we 'split' the CF-expansion (3.8) of a_1/a_0 at depth i, we obtain two CF-expansions,

$$\cfrac{1}{q_1 + \cfrac{1}{q_2 + \cfrac{1}{q_3 + \cfrac{1}{\ddots + \cfrac{1}{q_{i-1}}}}}}, \qquad \cfrac{1}{q_i + \cfrac{1}{q_{i+1} + \cfrac{1}{q_{i+2} + \cfrac{1}{\ddots + \cfrac{1}{q_p}}}}}, \tag{3.10}$$

each defining a rational number. When we consider only the $i-1$ first LFTs, we get the 'beginning' rational, which can be expressed with the co-sequences $(u_i), (v_i)$ as

$$h_1 \circ h_2 \circ \cdots \circ h_{i-1}(0) = \frac{|u_i|}{|v_i|}.$$

https://doi.org/10.1017/S0963548304006261 Published online by Cambridge University Press

When we consider only the $p-i$ last LFTs, we get the 'ending' rational, which can be expressed with the sequence (a_i),

$$h_{i+1} \circ h_{i+2} \circ \cdots \circ h_p(0) = \frac{a_{i+1}}{a_i}.$$

3.4. Transfer operators

The main tool of dynamical analysis is the transfer operator, denoted by $\mathbf{H}_s$. It generalizes the density transformer $\mathbf{H}$ that describes the evolution of the density: if f_0 denotes the initial density on I, and f_1 the density on I after one iteration of S, then f_1 can be written as $f_1 = \mathbf{H}[f_0]$, where $\mathbf{H}$ is defined by

$$\mathbf{H}[f](x) = \sum_{h \in \mathscr{H}} |h'(x)|\, f \circ h(x). \tag{3.11}$$

We let $\mathbf{R}_h$ denote the 'component' operator relative to a single LFT $h \in \mathscr{H}$

$$\mathbf{R}_h[f](x) = |h'(x)|\, f \circ h(x). \tag{3.12}$$

The derivative $h'(x)$ can be expressed with the denominator function D defined by

$$D[g](x) = cx + d, \quad \text{for} \quad g(x) = \frac{ax+b}{cx+d} \quad \text{with} \quad \gcd(a,b,c,d) = 1,$$

as

$$h'(x) = \frac{\det h}{D[h](x)^2}.$$

Since all the LFTs have a determinant equal to ± 1, this implies an alternative expression for $\mathbf{R}_h$,

$$\mathbf{R}_h[f](x) = \frac{1}{D[h](x)^2}\, f \circ h(x).$$

It is convenient to use a more general operator that depends on a complex parameter s:

$$\mathbf{R}_{s,h}[f](x) := \frac{1}{D[h](x)^s}\, f \circ h(x).$$

The composition property of the denominator,

$$D[h \circ g](x) = D[h](g(x))D[g](x),$$

entails a composition property on operators, namely

$$\mathbf{R}_{s,h\circ g} = \mathbf{R}_{s,g} \circ \mathbf{R}_{s,h}.$$

In particular, the transfer operator $\mathbf{H}_s$, and its iterates are defined as

$$\mathbf{H}_s = \sum_{h \in \mathscr{H}} \mathbf{R}_{s,h}, \qquad \mathbf{H}_s^n = \sum_{h \in \mathscr{H}^n} \mathbf{R}_{s,h}.$$

Since $\mathbf{H}_2 = \mathbf{H}$, the transfer operator can be viewed as a generalization of the density transformer. Finally, the operator that is the quasi-inverse of $\mathbf{H}_s$,

$$(\mathbf{I} - \mathbf{H}_s)^{-1} := \sum_{n \geqslant 0} \mathbf{H}_s^n = \sum_{h \in \mathscr{H}^\star} \mathbf{R}_{s,h}$$

https://doi.org/10.1017/S0963548304006261 Published online by Cambridge University Press

will play a fundamental rôle hereafter, and it can be viewed as a generating operator of the set $\mathscr{H}^\star$ formed by all possible transformations used by the Euclid Algorithm.

3.5. Spectral properties of the transfer operator

In dynamical analysis context, singularity analysis of generating functions is closely related to spectral properties of the transfer operator. These properties depend on the functional space where the operator acts. Previous analyses of Euclidean algorithms dealt with spaces of analytic functions defined on a complex neighbourhood of the unit interval I. Here we have to work with characteristic functions of some intervals, and we are led to work with a larger space, the space of functions with bounded variation on the unit interval I. This functional space was used previously in dynamical analysis [4], and the main properties of the transfer operator, when acting on this functional space, can be found there.

For $\Re(s) > 1$, the operator $\mathbf{H}_s$ acts on $BV(I)$ and the map $s \mapsto \mathbf{H}_s$ is analytic. We denote by $R(s)$ the spectral radius of $\mathbf{H}_s$. The function $s \mapsto R(s)$ is strictly decreasing along the real axis, and satisfies $R(s) \leqslant R(r)$ for $\Re(s) = r$. For $s = 2$, the operator is quasi-compact: there exists a spectral gap between the unique dominant eigenvalue (which equals 1, since the operator is a density transformer) and the remainder of the spectrum. By perturbation theory, these facts – existence of a dominant eigenvalue $\lambda(s)$ and of a spectral gap – remain true in a neighbourhood of $s = 2$. There, the operator splits into two parts: the part relative to the dominant eigenvalue, denoted $\mathbf{P}_s$, and the part relative to the remainder of the spectrum, denoted $\mathbf{N}_s$, whose spectral radius is strictly less than $|\lambda(s)|$. This leads to the spectral decomposition

$$\mathbf{H}_s[f](x) = \lambda(s)\mathbf{P}_s[f](x) + \mathbf{N}_s[f](x). \tag{3.13}$$

The projector $\mathbf{P}_s$ can also be written as $\mathbf{P}_s[f](x) = \psi_s(x)E_s[f]$, where ψ_s is the dominant eigenfunction and E_s is the dominant eigenfunction of the dual operator $\mathbf{H}_s^\star$ with normalization condition $E_s[\psi_s] = 1$.

The decomposition (3.13) extends to the powers of the operator

$$\mathbf{H}_s^n[f](x) = \lambda^n(s)\mathbf{P}_s[f](x) + \mathbf{N}_s^n[f](x),$$

and finally to the quasi-inverse $(\mathbf{I} - \mathbf{H}_s)^{-1}$:

$$(\mathbf{I} - \mathbf{H}_s)^{-1}[f](x) = \frac{\lambda(s)}{1 - \lambda(s)}\mathbf{P}_s[f](x) + (\mathbf{I} - \mathbf{N}_s)^{-1}[f](x). \tag{3.14}$$

The first term on the right is singular at $s = 2$, while the second term is analytic on the half-plane $\{\Re(s) > 2\}$.

3.6. Decomposition of the quasi-inverse: properties of the dominant eigenvalue

We summarize here the main properties that will be extensively used hereafter, in particular, when one applies the Tauberian Theorem.

https://doi.org/10.1017/S0963548304006261 Published online by Cambridge University Press

When the operator $\mathbf{H}_s$ acts on $BV(I)$, the following is true.

(a) At $s = 2$, dominant spectral objects are all explicit,

$$\lambda(2) = 1, \qquad \psi_2(x) = \frac{1}{\log 2}\frac{1}{1+x},$$
$$E_2[f] = \int_0^1 f(t)\,dt, \qquad \lambda'(2) = \frac{-\pi^2}{12\log 2}.$$

(b) The dominant eigenvalue $s \to \lambda(s)$ is well defined on a neighbourhood $\mathscr{V}$ of $s = 2$. On the interval $\mathscr{V} \cap \mathbf{R}$, the function $s \mapsto \lambda(s)$ is positive, analytic, strictly decreasing and strictly log-convex.

(c) For any $\sigma > 1$ and any s with $\Re(s) = \sigma$, the spectral radius $R(s)$ satisfies $R(s) \leqslant R(\sigma)$. For any $\sigma \in \mathscr{V} \cap \mathbf{R}$ and any s with $\Re(s) = \sigma, s \neq \sigma$, the spectral radius $R(s)$ satisfies $R(s) < R(\sigma) = \lambda(\sigma)$.

(d) The quasi-inverse $(I - \mathbf{H}_s)^{-1}$ of $\mathbf{H}_s$ is analytic on the half-plane $\{\Re(s) \geqslant 2, s \neq 2\}$, and has a simple pole at $s = 2$. Near $s = 2$, one has, for any positive function f of $BV(I)$, and any $x \in I$,

$$(I - \mathbf{H}_s)^{-1}[f](x) \sim \frac{1}{s-2}\,\frac{-1}{\lambda'(2)}\,\frac{1}{\log 2}\,\frac{1}{1+x}\left(\int_I f(t)\,dt\right). \tag{3.15}$$

3.7. The basic Dirichlet series $F_1(s)$

We explain now, as a kind of test, how the previous results can be used in the study of the Dirichlet series $F_1(s)$ relative to the trivial cost $X = 1$. First, this series admits another expression that involves the quasi-inverse $(I - \mathbf{H}_s)^{-1}$. Consider an input (a_0, a_1) of Ω. There exists a unique LFT h of $\mathscr{H}^\star$ for which $a_1/a_0 = h(0)$. Then

$$F_1(s) = \sum_{(a_0,a_1)\in\Omega} \frac{1}{a_0^s} f\left(\frac{a_1}{a_0}\right) = \sum_{h\in\mathscr{H}^\star} \frac{1}{D[h](0)^s} f\circ h(0) = (I - \mathbf{H}_s)^{-1}[f](0).$$

Moreover, the Riemann Zeta function $\zeta(s)$ relates $F_1(s)$ and $\widetilde{F}_1(s)$ via the equality $\widetilde{F}_1(s) = \zeta(s)F_1(s)$. Then the Tauberian Theorem applies to $F_1(s), \widetilde{F}_1(s)$ with $\sigma = 2, \theta = 0$, and, in the case when f is a density, we obtain

$$F_1(s) \sim \frac{1}{(s-2)}\,\frac{-1}{\lambda'(2)}\,\frac{1}{\log 2} = \frac{1}{(s-2)}\,\frac{6}{\pi^2}$$

so that

$$\sum_{\ell(a)=n} t_a \sim \frac{9}{2\pi^2}4^n, \qquad \sum_{\ell(a)=n} \widetilde{t}_a \sim \frac{3}{4}4^n.$$

Of course, these results can be obtained directly! But, the previous lines are in a sense 'generic' in the dynamical analysis methods.

3.8. Evolution of the density during the execution of the Euclid Algorithm

When all the real inputs of the Continued Fraction Algorithm are considered, the evolution of the density during the algorithm is well known. In this case, the algorithm does not terminate (almost surely), and the asymptotic density on the unit interval is the Gauss

https://doi.org/10.1017/S0963548304006261 Published online by Cambridge University Press

density $\psi = \psi_2$ defined in Section 3.6,

$$\psi(t) := \frac{1}{\log 2}\frac{1}{1+t}.$$

But, when we only consider rationals of Ω, the situation is not so clear, since the algorithm always terminates; moreover, at the end of the algorithm, all the rationals are now at the point 0, so the limit measure on Ω is the Dirac measure concentrated at $x = 0$.

The following two lemmas describe the evolution of density on rational inputs; the first one at the beginning of the algorithm, and the second one at any fraction of the depth.

Lemma 3.1. *Suppose that Ω is endowed with a density f with bounded variation. Then, for any interval B,*

$$\lim_{n\to\infty} \Pr_n[B] = \int_B f(t)\,dt.$$

Proof. Since Ω is endowed with f, we have

$$\Pr_n[B] = \frac{\sum_{x\in\Omega_n\cap B} f(x)}{\sum_{x\in\Omega_n} f(x)} = \frac{\sum_{x\in\Omega_n} \mathbf{1}_B(x) f(x)}{\sum_{x\in\Omega_n} f(x)}.$$

The Dirichlet series of costs for numerator and denominator are

$$G(s) = \sum_{(a_0,a_1)\in\Omega} \frac{1}{a_0^s} (\mathbf{1}_B f)\left(\frac{a_1}{a_0}\right), \qquad F_1(s) = \sum_{(a_0,a_1)\in\Omega} \frac{1}{a_0^s} f\left(\frac{a_1}{a_0}\right),$$

which have alternative expressions involving the quasi-inverse $(I - \mathbf{H}_s)^{-1}$,

$$F_1(s) = (I - \mathbf{H}_s)^{-1}[f](0), \qquad G(s) = (I - \mathbf{H}_s)^{-1}[\mathbf{1}_B f](0).$$

Then the Tauberian Theorem applies to $F(s)$ and $G(s)$ with $\sigma = 2$ and $\theta = 0$ and gives the result. □

The next lemma describes the situation 'at a fraction of the depth'.

Lemma 3.2. *Suppose that Ω is endowed with a density f with bounded variation. For $\alpha \in]0,1[$ and any $x \in \Omega$ with depth p, let x_α denote the pair that arises at the $\lfloor \alpha p \rfloor$th iteration of the Euclid Algorithm when applied to x. Then, for any interval B,*

$$\lim_{n\to\infty} \Pr_n[x_\alpha \in B] = \int_B \psi(t)\,dt, \quad \textit{where} \quad \psi(x) = \frac{1}{\log 2}\frac{1}{1+x}$$

is the stationary density of the Euclid Algorithm.

Proof. Consider a rational $x := a_1/a_0$ of depth p and its transform $g(x)$ after $i := i(p)$ steps of the Euclid Algorithm. Then $g(x) = a_{i+1}/a_i$. There exists some LFTs h of depth i, r of depth $p - i$ for which $x := h \circ r(0), g(x) := r(0)$. Then

$$\Pr_n[x; g(x) \in B] = \frac{\sum_{x\in\Omega_n} \mathbf{1}_B \circ g(x) f(x)}{\sum_{x\in\Omega_n} f(x)}.$$

https://doi.org/10.1017/S0963548304006261 Published online by Cambridge University Press

Since $\mathbf{1}_B \circ g(x) f(x) = \mathbf{1}_B \circ r(0) f \circ h \circ r(0)$, the Dirichlet series relative to the numerator is

$$G(s) = \sum_p \mathbf{H}_s^{p-i(p)} \big[\mathbf{1}_B \mathbf{H}_s^{i(p)}[f]\big](0).$$

It converges on the punctured half-plane $\{\Re(s) > 2, s \neq 2\}$. Moreover, in the case when $i(p)$ and $p - i(p)$ both tend to ∞ for $p \to \infty$, the 'dominant' part of this Dirichlet series at $s = 2$ is

$$G^+(s) = \sum_p \lambda(s)^p \mathbf{P}_s \big[\mathbf{1}_B \mathbf{P}_s[f]\big](0);$$

it has a pole at $s = 2$, whose residue is

$$\frac{-1}{\lambda'(2)} \mathbf{P}_2 \big[\mathbf{1}_B \mathbf{P}_2[f]\big](0) = \frac{-1}{\lambda'(2)} \psi(0) \int_I \mathbf{1}_B(t) \mathbf{P}_2[f](t)\, dt.$$

Since $\mathbf{P}_2[f](t) = \psi(t)\, dt$, we finally obtain the result. □

Remark. We have in fact proved a more general result. Consider a function $Q : \mathbb{N} \to \mathbb{N}$ such that $0 < Q(n) < n$, and both $Q(n)$ and $n - Q(n)$ tend to infinity. For any $x \in \Omega$ with depth p, we denote by x_Q the integer pair that arises at the $Q(p)$th iteration of the Euclid Algorithm when applied to x. Then the following holds:

$$\lim_{n\to\infty} \Pr_n[x_Q \in B] = \int_B \psi(t)\, dt.$$

The previous lemma is the special case $Q(n) = \lfloor \alpha n \rfloor$.

We thus can describe (in an approximate way) the distribution at the beginning of each phase of the Lehmer–Euclid Algorithm. This provides a first argument towards our heuristic reasoning in Section 2.8.

Proposition 3.3. *Suppose that the set of the initial inputs of the Euclid Algorithm is endowed with some density f with bounded variation on the unit interval I. Then, at the beginning of each phase of the Lehmer–Euclid Algorithm – except the first one – the large inputs and the small inputs have a distribution that is 'close' to the distribution of Ω weighted by the stationary density of the Euclid Algorithm.*

4. Analysis of the Interrupted Euclid Algorithm: number of iterations

In this section we prove Theorem 2.1. We deal here with the cost M defined in (3.4) and we denote by $F_M, \widetilde{F}_M$ the Dirichlet series relative to this cost. We use in this section the notation gathered in Table 1. Note that in Sections 4.1, 4.2 and 4.3 we only consider strictly positive values for γ. We come back to the general case in Section 4.4.

4.1. The Dirichlet series $F_M(s)$

We first provide an expression for $F_M(s)$,

$$F_M(s) := \sum_{(a_0,a_1)\in\Omega} \frac{a_{\lfloor \delta P \rfloor}^{\gamma}}{a_0^{\alpha\gamma+s}} f\left(\frac{a_1}{a_0}\right),$$

as a function of operator $\mathbf{H}_t$ that involves two values of t, namely $t = s^+$ and $t = s^-$.

https://doi.org/10.1017/S0963548304006261 Published online by Cambridge University Press

Table 1. The notation used in the Proof of Theorem 2.1

α is the 'interruption parameter' in the Euclid Algorithm; $\beta := 1 - \alpha$.

γ is strictly positive; it appears when using Markov's inequality.

s denotes the 'Dirichlet series parameter'. We let $s^-(\alpha,\gamma) = s - \beta\gamma$ and $s^+(\alpha,\gamma) = s + \alpha\gamma$. For a fixed value of the pair (α,γ), we just write s^+ and s^-.

P denotes the number of iterations of the Euclid Algorithm (the depth).

δ (with $\delta \in [0,1]$) is used to denote the index of a generic step in the Euclid Algorithm, $\lfloor \delta P \rfloor$.

For easier computations, δ is a rational $\delta = c/(c+d)$. Thus, $P = (c+d)k + j$, with $j < c + d$, and we set $j' := \lfloor \delta j \rfloor$.

Lemma 4.1. *The Dirichlet series $F_M(s), \widetilde{F}_M(s)$ can be expressed in terms of the transfer operators,*

$$F_M(s) = \sum_{p \geqslant 0} \mathbf{H}_{s^-}^{p-\lfloor \delta p \rfloor} \circ \mathbf{H}_{s^+}^{\lfloor \delta p \rfloor}[f](0), \qquad \widetilde{F}_M(s) = \zeta(s^+) F_M(s). \tag{4.1}$$

Proof. Consider an input (a_0, a_1) of Ω on which the algorithm performs p iterations. There exists a unique LFT h of depth p such that $a_1/a_0 = h(0)$. We can decompose h in two LFTs, g and r, of depth $\lfloor \delta p \rfloor$ and $p - \lfloor \delta p \rfloor$ such that $h = g \circ r$. In this case, we have $D[g \circ r](0) = a_0$ and $D[r](0) = a_{\lfloor \delta p \rfloor}$. Now, the general term of the series $F_M(s)$ decomposes as

$$\frac{a_{\lfloor \delta p \rfloor}^{\gamma}}{a_0^{\alpha\gamma + s}} = \frac{D[r]^{\gamma}(0)}{D[g \circ r]^{s^+}(0)} = \frac{1}{D[r]^{s^-}(0)} \frac{1}{D[g]^{s^+} \circ r(0)}.$$

Furthermore, when a_1/a_0 varies in the set of all inputs of Ω with a given depth p, we obtain

$$\sum_{\substack{(a_0,a_1)\in\Omega \\ P(a_0,a_1)=p}} \frac{a_{\lfloor \delta p \rfloor}^{\gamma}}{a_0^{\alpha\gamma + s}} f\left(\frac{a_1}{a_0}\right) = \mathbf{H}_{s^-}^{p-\lfloor \delta p \rfloor} \circ \mathbf{H}_{s^+}^{\lfloor \delta p \rfloor}[f](0).$$

Finally, a summation over p gives the result. Moreover, the relation $\widetilde{F}_M(s) = \zeta(s^+) F_M(s)$ proves that it is sufficient to work with inputs of Ω. □

4.2. Dominant singularity of $F_M(s)$

With the notation of Table 1, the expression of the previous lemma takes the form

$$F_M(s) = \sum_{j=0}^{c+d-1} \mathbf{H}_{s^-}^{j-j'} \circ \left(\sum_{k \geqslant 0} \mathbf{H}_{s^-}^{dk} \circ \mathbf{H}_{s^+}^{ck} \right) \circ \mathbf{H}_{s^+}^{j'}[f](0), \tag{4.2}$$

and the possible singularities of the series $F_M(s)$ become apparent.

https://doi.org/10.1017/S0963548304006261 Published online by Cambridge University Press

Lemma 4.2. *Let s be a real $s > 1$, and let $R(s)$ be the spectral radius of the operator $\mathbf{H}_s$. Let $\phi(s)$ be the function*

$$\phi(s) := R^d(s^-)\, R^c(s^+). \tag{4.3}$$

For $0 < \gamma < 1$, the equation $\phi(s) = 1$ has a unique real solution ρ. This solution belongs to the interval $[2 - \alpha\gamma, 2 + \beta\gamma]$.

The function $F_M(s)$ is analytic on the half-plane $\Re(s) > \rho$.

Proof. The function ϕ is defined for $s > 1 + \beta\gamma$. Notice that if the solution ρ exists, then $\rho^- < 2 < \rho^+$. It is thus sufficient to study $\phi(s)$ on the interval

$$[2 - \alpha\gamma, 2 + \beta\gamma], \tag{4.4}$$

on which ϕ is defined: the inequality $0 < \gamma < 1$ entails $2 - \alpha\gamma > 1 + \beta\gamma$. Since $s \mapsto R(s)$ is strictly decreasing on the real axis, the same is true for $s \mapsto \phi(s)$ (on the real axis), and the sequence of inequalities

$$\phi(2 - \alpha\gamma) = R^d(2 - \gamma) > \lambda(2) = 1 > R^c(2 + \gamma) = \phi(2 + \beta\gamma).$$

proves that the equation $\phi(s) = 1$ has a unique solution ρ.

From (4.2), we obtain, when taking norms on space $BV(I)$,

$$|F_M(s)| \leqslant \|f\| \left(\sum_{j=0}^{c+d-1} \|\mathbf{H}_{s^-}^{j-j'}\| \, \|\mathbf{H}_{s^+}^{j'}\| \right) \left(\sum_{k \geqslant 0} \|\mathbf{H}_{s^-}^{dk}\| \, \|\mathbf{H}_{s^+}^{ck}\| \right).$$

The right part defines a series whose general term is equivalent to $R(s^-)^{dk} R(s^+)^{ck}$. This series is convergent when $\phi(s) = R(s^-)^d R(s^+)^c$ is less than one. □

Lemma 4.3. *Suppose that γ is positive and sufficiently small. Then, the series $F_M(s)$ has a pole of order 1 at $s = \rho$ where ρ is defined as the unique solution of the equation $\phi(s) = 1$. Moreover, $F_M(s)$ is analytic on the half-plane $\{\Re(s) \geqslant \rho, s \neq \rho\}$.*

Proof. If γ is sufficiently small, the interval $(2 - \gamma, 2 + \gamma)$ is contained in the neighbourhood $\mathscr{V}$ of Section 3.6. Then, when s belongs to the interval defined by (4.4), the two values s^+ and s^- belong to $\mathscr{V}$, and the two operators $\mathbf{H}_{s^+}, \mathbf{H}_{s^-}$ are quasi-compact. Thus the dominant eigenvalue $\lambda(s^+), \lambda(s^-)$ of the operators $\mathbf{H}_{s^+}, \mathbf{H}_{s^-}$ are well defined. The spectral decomposition (3.13) of $\mathbf{H}_t$ described in Section 3 applies to $t = s^+, t = s^-$ and extends to the series $F_M(s)$ via the equality (4.2) so that, on the interval (4.4), the series $F_M(s)$ decomposes into a sum of two terms, a 'dominant' term and a 'remainder' term. The dominant term is obtained when replacing each occurrence of $\mathbf{H}_t$ by the term $\lambda(t)\mathbf{P}_t$,

https://doi.org/10.1017/S0963548304006261 Published online by Cambridge University Press

and is of the form $F_M^+(s)\,\mathbf{P}_{s^-} \circ \mathbf{P}_{s^+}[f](0)$, with

$$F_M^+(s) = \left(\sum_{j=0}^{c+d-1} \lambda^{j-j'}(s^-)\lambda^{j'}(s^+) \right) \left(\sum_{k\geqslant 0} (\lambda^d(s^-)\lambda^c(s^+))^k \right),$$
$$= \frac{1}{1-\phi(s)} \left(\sum_{j=0}^{c+d-1} \lambda^{j-j'}(s^-)\lambda^{j'}(s^+) \right). \tag{4.5}$$

Spectral properties of the operator $\mathbf{H}_t$, for $t = s^+, t = s^-$, prove that the dominant poles of the series $F_M(s)$ are only brought by the dominant part $F_M^+(s)$. Thus, the dominant pole of $F_M(s)$ arises at $s = \rho$ if ρ is solution of the equation $\phi(s) = 1$. Near $s = \rho$, we have

$$F_M(s) \sim \frac{1}{s-\rho}\ \frac{-1}{\phi'(\rho)} \sum_{j=0}^{c+d-1} \lambda^{j-j'}(\rho^-)\lambda^{j'}(\rho^+)\mathbf{P}_{\rho^-} \circ \mathbf{P}_{\rho^+}[f](0). \tag{4.6}$$

The maximum properties of the function $s \mapsto \lambda(s)$ along vertical and horizontal lines are inherited by $s \mapsto \phi(s)$, so that $\phi(s) < 1$ for $\Re(s) = \rho, s \neq \rho$. □

According to the last lemma, the series $F_M(s)$ fulfils all the conditions of the Tauberian theorem with $\sigma = \rho$ and $\theta = 0$ when γ is positive. Since the Dirichlet series $F_1(s)$ fulfils all the conditions of Tauberian theorem with $\sigma = 2$ and $\theta = 0$ (as proved in Section 3.7), this leads to the main result of this section.

Proposition 4.4. *For all strictly positive γ sufficiently close to 0, for all $\alpha, \delta \in [0,1]$, there exists ρ (depending on α, γ and δ) such that the expectation of the cost M satisfies*

$$\mathrm{E}_n\left[\left(\frac{a_{\lfloor \delta p \rfloor}}{a_0^\alpha} \right)^\gamma \right] = O(2^{n(\rho-2)}).$$

4.3. Particular case when δ is near $1-\alpha$

We now prove that ρ is less than 2 when δ equals $(1-\alpha)+\varepsilon$ with $\varepsilon > 0$.

Lemma 4.5. *When $\delta = (1-\alpha)+\varepsilon$, with $\varepsilon > 0$, there exists $\gamma > 0$ such that the unique solution ρ of equation $\phi(s) = 1$ is strictly less than 2. Moreover, one can choose $2-\rho = \Omega(\varepsilon^2)$.*

Proof. Suppose that $\delta = \frac{c}{c+d}$ is of the form $\delta = (1-\alpha)+\varepsilon$. We have

$$\frac{c}{d} = \frac{1-\alpha+\varepsilon}{\alpha-\varepsilon} = \frac{\beta+\varepsilon}{\alpha-\varepsilon}.$$

Now, equation $\phi(s) = 1$ can be written with the function $\Lambda(s) := \log \lambda(s)$ as

$$\Phi(s) := \frac{|\Lambda(s^-)|}{|\Lambda(s^+)|} = \frac{c}{d} = \frac{\beta+\varepsilon}{\alpha-\varepsilon}.$$

https://doi.org/10.1017/S0963548304006261 Published online by Cambridge University Press

On a neighbourhood of $s = 2$, the left term defines a strictly decreasing function Φ of s. It is thus sufficient to show that there exists some $\gamma > 0$ (depending on ε) for which

$$\Phi(2) < \Phi(\rho) = \frac{\beta + \varepsilon}{\alpha - \varepsilon}.$$

The function $s \mapsto \Lambda(s)$ satisfies

$$\Lambda(2) = 0, \qquad \Lambda'(2) < 0, \qquad \Lambda''(2) > 0,$$

so that, for a sufficiently small γ,

$$\Phi(2) = \frac{|\Lambda(2 - \beta\gamma)|}{|\Lambda(2 + \alpha\gamma)|} < \frac{\beta}{\alpha}\,\frac{1 + \beta\varepsilon_1}{1 - \alpha\varepsilon_1} < \frac{\beta + \varepsilon_1}{\alpha - \varepsilon_1}, \quad \text{with} \quad \varepsilon_1 := \frac{3\gamma}{4}\frac{\Lambda''(2)}{|\Lambda'(2)|}.$$

One can choose

$$\gamma = \varepsilon\frac{|\Lambda'(2)|}{\Lambda''(2)} \quad \text{so that} \quad \varepsilon_1 = \frac{3\varepsilon}{4} < \varepsilon \quad \text{and} \quad \Phi(2) < \Phi(\rho).$$

In this case, we have $\rho < 2$, and this proves the first part of the lemma. We now wish to evaluate $2 - \rho$ as a function of ε. First

$$2 - \rho \sim \frac{|\Phi(2) - \Phi(\rho)|}{\Phi'(2)}.$$

Then,

$$|\Phi(2) - \Phi(\rho)| \geqslant \frac{\beta + \varepsilon}{\alpha - \varepsilon} - \frac{\beta + \varepsilon_1}{\alpha - \varepsilon_1} \geqslant \frac{\varepsilon - \varepsilon_1}{\alpha^2} = \frac{\varepsilon}{4\alpha^2}.$$

On the other hand, when taking logarithmic derivatives, and using the fact that $|(\Lambda'/\Lambda)(2 + x)| \sim (1/x)$ (near $x = 2$), we have

$$\frac{|\Phi'(2)|}{\Phi(2)} = \left|\frac{\Lambda'}{\Lambda}(2 - \beta\gamma)\right| + \left|\frac{\Lambda'}{\Lambda}(2 + \alpha\gamma)\right| \quad \text{so that} \quad |\Phi'(2)| \sim \frac{1}{\alpha^2\gamma},$$

and finally

$$2 - \rho \geqslant \frac{\varepsilon\gamma}{4} = \frac{\varepsilon^2}{4}\frac{|\Lambda'(2)|}{\Lambda''(2)}. \qquad \square$$

4.4. The case $\gamma < 0$

We now consider the case when the cost M is relative to parameters $(\alpha, \delta, -\gamma)$ with $\gamma > 0$. We remark that $s^-(\alpha, \gamma) = s^+(1 - \alpha, -\gamma)$. Moreover, even if the two costs, the cost M relative to $(\alpha, \delta, -\gamma)$ and the cost $\widetilde{M}$ relative to $(1 - \alpha, 1 - \delta, \gamma)$ are not *a priori* the same, the two Dirichlet series $F_M(s)$ and $F_{\widetilde{M}}(s)$ have the same dominant part

$$\sum_{k \geqslant 0} \mathbf{H}_{s^-}^{dk} \circ \mathbf{H}_{s^+}^{ck},$$

and the function ϕ is the same in both cases. Let $\rho(\alpha, \delta, \gamma)$ denote the unique solution of the equation $\phi(s) = 1$, where ϕ is relative to parameters (α, δ, γ); then the equality $\rho(\alpha, \delta, -\gamma) = \rho(1 - \alpha, 1 - \delta, \gamma)$ and Lemma 4.5 imply that for any $\varepsilon > 0$, there exists $\gamma > 0$ such that for any α

$$\rho(\alpha, 1 - \alpha - \varepsilon, -\gamma) = \rho(1 - \alpha, \alpha + \varepsilon, \gamma) < 2.$$

https://doi.org/10.1017/S0963548304006261 Published online by Cambridge University Press

This entails that, for any $\varepsilon > 0$, there exists $\gamma > 0$ such that $\rho^+ = \rho(\alpha, 1-\alpha+\varepsilon, \gamma)$ and $\rho^- = \rho(\alpha, 1-\alpha-\varepsilon, -\gamma)$ are both less than 2. Then, with (3.2) and (3.3),

$$\Pr_n\left[\left|\frac{P_\alpha}{P} - (1-\alpha)\right| > \varepsilon\right] = O(K^n),$$

with $K = 2^{\max(\rho^+,\rho^-)-2}$. This proves Theorem 2.1(i).

4.5. Proof of Theorem 2.1(ii)

Let Q_α denote the random variable $Q_\alpha := |P_\alpha - (1-\alpha)P|$, and consider, for some $\varepsilon > 0$, the exceptional event $A(\varepsilon) := [Q_\alpha \geqslant \varepsilon P]$. The worst case of the Euclid Algorithm entails that, on Ω_n, we always have $P = O(n)$. This yields

$$\mathrm{E}_n[Q_\alpha] \leqslant K n \left(\Pr_n[A(\varepsilon)] + \varepsilon\right),$$

and, together with Theorem 2.1(i), we obtain

$$\mathrm{E}_n[P_\alpha] \sim (1-\alpha)\mathrm{E}_n[P].$$

This ends the proof of Theorem 2.1. □

5. Analysis of the Interrupted Algorithm: bit complexity

In this section we prove Theorem 2.2. We are interested in studying the average bit complexity of the Interrupted Euclid Algorithm $\mathscr{E}_\alpha$, which terminates as soon as the current integer a_i is less than a_0^α. As claimed in Section 3.2, we replace this algorithm by a 'regularized' algorithm $\overline{\mathscr{E}}_\alpha$, which always terminates at a fraction of the depth. Thanks to Theorem 2.1, the two algorithms are quite close, and their bit complexities will be asymptotically the same.

5.1. Expression of Dirichlet series with transfer operators

As in the previous section, the first step relates the Dirichlet series of costs $\overline{B}_\alpha, \overline{C}_\alpha$,

$$\overline{B}_\alpha(a_0, a_1) = \sum_{i=1}^{\lfloor (1-\alpha)p \rfloor} \ell(a_i) \cdot b(q_i) \quad \text{with} \quad b(q) := D\ell(q) + 2M, \tag{5.1}$$

$$\overline{C}_\alpha(a_0, a_1) = \sum_{i=1}^{\lfloor (1-\alpha)p \rfloor} \ell(v_i) \cdot c(q_i) \quad \text{with} \quad c(q) := (\ell(q) + 2)M, \tag{5.2}$$

to some transfer operators. These costs involve different parameters, namely the length $\ell(q)$ of the quotients q relative to the LFT $h : x \mapsto 1/(q+x)$, the length $\ell(a_i)$ of the remainders, and the lengths of the Bezout terms u_i, v_i.

As in (3.9), (3.10), we consider an input (a_0, a_1) of the algorithm such that $a_1/a_0 = h(0) = h_1 \circ h_2 \circ \cdots \circ h_p(0)$ and we split the LFT h in three parts:

(a) the beginning part $b_i(h) = h_1 \circ \cdots \circ h_{i-1}$,
(b) the ending part $e_i(h) = h_{i+1} \circ \cdots \circ h_p$,
(c) the ith component h_i.

https://doi.org/10.1017/S0963548304006261 Published online by Cambridge University Press

Then, the operator relative to the LFT h decomposes as

$$\mathbf{R}_{s,h} = \mathbf{R}_{s,e_i(h)} \circ \mathbf{R}_{s,h_i} \circ \mathbf{R}_{s,b_i(h)}.$$

First, we weight the operator $\mathbf{R}_{s,h_i}$ with some cost $d(q_i)$ relative to the quotient q_i of the LFT h_i and introduce, for any LFT g of depth 1, the operator

$$\mathbf{R}^{[d]}_{s,g}[f](x) := \frac{d(q)}{D[g](x)^s} f \circ g(x),$$

and the total operator $\mathbf{H}^{[d]}_s$ is thus defined by

$$\mathbf{H}^{[d]}_s[f](x) := \sum_{g \in \mathscr{H}} \mathbf{R}^{[d]}_{s,g}[f](x).$$

Second, we take derivatives with respect to s, and use the derivative functional Δ defined for an operator $\mathbf{L}_s$ which depends on a parameter s by

$$\Delta \mathbf{L}_s := -\frac{1}{\log 2} \frac{d}{ds} \mathbf{L}_s.$$

When applied to $\mathbf{R}_{s,g}$, it produces at the numerator the logarithm $\log_2 D[g]$. Since $D[e_i(h)](0) = a_i, D[b_i(h)](x) = v_i + v_{i-1}x$, we obtain, with costs b, c defined in (5.1), (5.2),

$$\Delta \mathbf{R}_{s,e_i(h)} \circ \mathbf{R}^{[b]}_{s,h_i} \circ \mathbf{R}_{s,b_i(h)}[f](0) = \frac{1}{a_0^s} \log_2 a_i \, b(q_i) f\left(\frac{a_1}{a_0}\right),$$
$$\mathbf{R}_{s,e_i(h)} \circ \mathbf{R}^{[c]}_{s,h_i} \circ \Delta \mathbf{R}_{s,b_i(h)}[f](0) = \frac{1}{a_0^s} \log_2 \left(v_i + v_{i-1}\frac{a_i}{a_{i-1}}\right) c(q_i) \, f\left(\frac{a_1}{a_0}\right).$$

The inequalities

$$|\log_2 a_i - \ell(a_i)| \leqslant 1, \qquad \left|\log_2 \left(v_i + v_{i-1}\frac{a_i}{a_{i-1}}\right) - \ell(v_i)\right| \leqslant 1 \tag{5.3}$$

prove that the costs $\widehat{B}_\alpha, \widehat{C}_\alpha$ defined as

$$\widehat{B}_\alpha(a_0, a_1) = \sum_{i=1}^{\lfloor (1-\alpha)p \rfloor} \log_2(a_i) \cdot b(q_i), \tag{5.4}$$

$$\widehat{C}_\alpha(a_0, a_1) = \sum_{i=1}^{\lfloor (1-\alpha)p \rfloor} \log_2 \left(v_i + v_{i-1}\frac{a_i}{a_{i-1}}\right) \cdot c(q_i), \tag{5.5}$$

can be viewed as 'approximations' for costs $\overline{B}_\alpha, \overline{C}_\alpha$ (we shall prove this statement in a precise way in Section 5.3). Now, the operators

$$\mathbf{B}^{(\alpha)}_{s,h} = \sum_{i=1}^{\lfloor (1-\alpha)p \rfloor} \Delta \mathbf{R}_{s,e_i(h)} \circ \mathbf{R}^{[b]}_{s,h_i} \circ \mathbf{R}_{s,b_i(h)},$$
$$\mathbf{C}^{(\alpha)}_{s,h} = \sum_{i=1}^{\lfloor (1-\alpha)p \rfloor} \mathbf{R}_{s,e_i(h)} \circ \mathbf{R}^{[c]}_{s,h_i} \circ \Delta \mathbf{R}_{s,b_i(h)},$$

https://doi.org/10.1017/S0963548304006261 Published online by Cambridge University Press

are generating operators for the costs $\widehat{B_\alpha}$ and $\widehat{C_\alpha}$ on the input (a_0, a_1), since

$$\mathbf{B}^{(\alpha)}_{s,h}[f](0) = \frac{1}{a_0^s}\widehat{B}_\alpha(a_0,a_1) f\left(\frac{a_1}{a_0}\right), \qquad \mathbf{C}^{(\alpha)}_{s,h}[f](0) = \frac{1}{a_0^s}\widehat{C}_\alpha(a_0,a_1) f\left(\frac{a_1}{a_0}\right).$$

When (a_0, a_1) is a generic element of Ω with depth p, the LFT h is a generic element of $\mathscr{H}^p$. When summing over p, we obtain the following lemma.

Lemma 5.1. *The Dirichlet series $F_X(s)$ relative to costs $X = \widehat{B_\alpha}$ or $X = \widehat{C_\alpha}$ defined in* (5.4), (5.5) *are expressed in terms of the transfer operator $\mathbf{H}_s$, the weighted operator $\mathbf{H}_s^{[d]}$ relative to cost $d = b, c$ defined in* (5.1), (5.2), *and the functional derivative Δ,*

$$F_{\widehat{B}_\alpha}(s) = \sum_{p\geqslant 0} \sum_{i=1}^{\lfloor (1-\alpha)p\rfloor} \Delta\mathbf{H}_s^{p-i} \circ \mathbf{H}_s^{[b]} \circ \mathbf{H}_s^{i-1}[f](0),$$

$$F_{\widehat{C}_\alpha}(s) = \sum_{p\geqslant 0} \sum_{i=1}^{\lfloor (1-\alpha)p\rfloor} \mathbf{H}_s^{p-i} \circ \mathbf{H}_s^{[c]} \circ \Delta\mathbf{H}_s^{i-1}[f](0).$$

5.2. Spectral decomposition and Tauberian theorem

We now work with s near to 2 and use the spectral decomposition (3.13), which splits the operator $\mathbf{H}_s^n$ in two parts,

$$\mathbf{H}_s^n = \lambda^n(s)\mathbf{P}_s + \mathbf{N}_s^n,$$

where $\lambda(s)$ is the dominant eigenvalue of the operator. This decomposition extends to the series. There appears a 'dominant' term that is obtained when replacing all the powers $\mathbf{H}_s^n$ by $\lambda(s)^n\mathbf{P}_s$. Note also that

$$\Delta\mathbf{H}_s^n = \sum_{j=1}^{n} \mathbf{H}_s^{j-1} \circ \Delta\mathbf{H}_s \circ \mathbf{H}_s^{n-j}$$

brings a dominant term equal to $n\lambda^{n-1}(s)\mathbf{P}_s \circ \Delta\mathbf{H}_s \circ \mathbf{P}_s$. Finally, the dominant terms $F^+_{\widehat{B}_\alpha}(s), F^+_{\widehat{C}_\alpha}(s)$ of $F_{\widehat{B}_\alpha}(s), F_{\widehat{C}_\alpha}(s)$ involve the operators

$$\mathbf{Q}_s = \mathbf{P}_s \circ \Delta\mathbf{H}_s \circ \mathbf{P}_s \circ \mathbf{H}_s^{[b]} \circ \mathbf{P}_s, \qquad \mathbf{T}_s = \mathbf{P}_s \circ \mathbf{H}_s^{[c]} \circ \mathbf{P}_s \circ \Delta\mathbf{H}_s \circ \mathbf{P}_s,$$

under the form

$$F^+_{\widehat{B}_\alpha}(s) = \widehat{B}^{(\alpha)}(s)\ \mathbf{Q}_s[f](0), \qquad F^+_{\widehat{C}_\alpha}(s) = \widehat{C}^{(\alpha)}(s)\ \mathbf{T}_s[f](0),$$

with (near $s = 2$)

$$\widehat{B}^{(\alpha)}(s) = \sum_{p\geqslant 0}\left(\sum_{i=0}^{\lfloor (1-\alpha)p\rfloor}(p-i)\right)\lambda^{p-1}(s) \sim (1-\alpha^2)\left(\frac{1}{1-\lambda(s)}\right)^3,$$

$$\widehat{C}^{(\alpha)}(s) = \sum_{p\geqslant 0}\left(\sum_{i=0}^{\lfloor (1-\alpha)p\rfloor} i\right)\lambda^{p-1}(s) \sim (1-\alpha)^2\left(\frac{1}{1-\lambda(s)}\right)^3.$$

https://doi.org/10.1017/S0963548304006261 Published online by Cambridge University Press

Near $s = 2$, the relations

$$\frac{\widehat{B}^{(\alpha)}(s)}{\widehat{B}^{(0)}(s)} \sim (1-\alpha^2), \qquad \frac{\widehat{C}^{(\alpha)}(s)}{\widehat{C}^{(0)}(s)} \sim (1-\alpha)^2$$

prove an analogue of Theorem 2.2 when the bar is replaced by a hat, namely,

$$\mathrm{E}_n[\widehat{B}_\alpha] \sim (1-\alpha^2)\,\mathrm{E}_n[\widehat{B}_0], \qquad \mathrm{E}_n[\widehat{C}_\alpha] \sim (1-\alpha)^2\,\mathrm{E}_n[\widehat{C}_0], \tag{5.6}$$

for $n \to \infty$.

Furthermore, the asymptotic behaviour of the expectations $\mathrm{E}_n[\widehat{B}_0], \mathrm{E}_n[\widehat{C}_0]$ can be obtained when applying Tauberian Theorem as in [1, 27]. Near $s = 2$, the two quantities $\mathbf{Q}_s[f](0), \mathbf{T}_s[f](0)$ define analytic functions of s whose values at $s = 2$ involve two integrals

$$\int_0^1 \mathbf{H}^{[d]}[\psi](t)\,dt, \qquad \int_0^1 \Delta\mathbf{H}[\psi](t)\,dt, \tag{5.7}$$

which deal with the stationary density ψ, and the possible costs $d = b, c$. As in [1, 27] (we recall that ℓ is the binary length),

$$\int_0^1 \mathbf{H}^{[\ell]}[\psi](t)\,dt = \log_2 \prod_{k=0}^{\infty}\left(1+\frac{1}{2^k}\right), \qquad \int_0^1 \Delta\mathbf{H}[\psi](t)\,dt = -\lambda'(2)$$

so that

$$\mathrm{E}_n[\widehat{B}_0] \sim (L_1 M + L_2 D)\,n^2, \qquad \mathrm{E}_n[\widehat{C}_0] \sim (L_1+L_2)M\,n^2,$$
$$\text{with} \qquad L_1 = \frac{12\log^2 2}{\pi^2}, \qquad L_2 = \frac{6\log 2}{\pi^2}\log\prod_{k=0}^{\infty}\left(1+\frac{1}{2^k}\right).$$

We now come back to the original costs, first with a bar, second without a bar.

5.3. Relation between costs with a hat and costs with a bar

The Dirichlet series of the cost $Y(a_0, a_1) := \sum_{i=1}^{p} d(q_i)$ corresponding to $d = b, c$ is exactly

$$F_Y(s) = (I - \mathbf{H}_s)^{-1} \circ \mathbf{H}_s^{[d]} \circ (I - \mathbf{H}_s)^{-1}[f](0).$$

Since this series has a pole of order 2 at $s = 2$, the expectation $\mathrm{E}_n[Y]$ is $O(n)$. Finally, the inequalities (5.3) entail that

$$\mathrm{E}_n[\widehat{B}_\alpha] - E_n[\overline{B}_\alpha] = O(n), \qquad \mathrm{E}_n[\widehat{C}_\alpha] - E_n[\overline{C}_\alpha] = O(n).$$

5.4. Relation between the costs with a bar and without a bar

In the same vein as in Section 4.5, let Q_α denote the random variable $Q_\alpha := |P_\alpha - (1-\alpha)P|$ and let R_α be one of the two random variables $R_\alpha := |B_\alpha - \overline{B_\alpha}|$ or $R_\alpha := |C_\alpha - \overline{C_\alpha}|$. The worst case of the Euclid Algorithm implies that, on Ω_n, we always have $R_\alpha = O(n^2)$. Furthermore, when dealing with the exceptional event $A(\varepsilon) := [Q_\alpha \geqslant \varepsilon P]$, we obtain the relation

$$\mathrm{E}_n[R_\alpha] \leqslant K'\,n^2\,\Pr_n[A(\varepsilon)] + n\,\mathrm{E}_n[X_\alpha] \quad \text{with} \quad X_\alpha := \sum_{i=\lfloor(1-\alpha-\varepsilon)P\rfloor}^{\lfloor(1-\alpha+\varepsilon)P\rfloor} d(q_i), \tag{5.8}$$

https://doi.org/10.1017/S0963548304006261 Published online by Cambridge University Press

where d is one of the two costs b or c. Theorem 2.1 proves that the first term is $o(n^2)$. The Dirichlet series relative to cost X_α,

$$\sum_{p\geqslant 0}\sum_{i=\lfloor(1-\alpha-\varepsilon)p\rfloor}^{\lfloor(1-\alpha+\varepsilon)p\rfloor}\mathbf{H}_s^{p-i}\circ\mathbf{H}_s^{[d]}\circ\mathbf{H}_s^{i-1}[f](0),$$

has a dominant term equal to

$$\varepsilon\left(\sum_p p\lambda(s)^{p-1}\right)\mathbf{P}_s\circ\mathbf{H}_s^{[d]}\circ\mathbf{P}_s[f](0)=\varepsilon\left(\frac{1}{1-\lambda(s)}\right)^2\mathbf{P}_s\circ\mathbf{H}_s^{[d]}\circ\mathbf{P}_s[f](0),$$

so that the second term of (5.8) is of the form $o(n^2)$. Furthermore, for both costs R_α, we have $\mathrm{E}_n[R_\alpha]=o(n^2)$, and

$$\mathrm{E}_n[B_\alpha]\sim\mathrm{E}_n[\overline{B_\alpha}],\qquad \mathrm{E}_n[C_\alpha]\sim\mathrm{E}_n[\overline{C_\alpha}].$$

Finally, Theorem 2.2 is obtained with (5.6), Sections 5.3 and 5.4.

6. Analysis of the Lehmer–Euclid Algorithm

We now prove Theorem 2.3. There are two main ideas, summarized by the next two lemmas. The first lemma compares, during each phase, the quantities that would appear if one used the usual Euclid Algorithm (the top horizontal line of Figure 7) with the quantities that actually appear in the bottom horizontal line of Figure 7.

Since these quantities are closely related, we can 'simulate' the bit complexity of the Lehmer–Euclid Algorithm only on the plain Euclid Algorithm itself. The second lemma shows that the algorithm is almost surely 'regular' in the sense that the duration of each phase is (almost surely) equal to a fraction of the depth. Then, as in the previous section, we first study, in the third lemma, a regularized version of the Lehmer–Euclid Algorithm $\mathscr{LE}_\mu$, which we denote by $\overline{\mathscr{LE}}_\mu$, where the duration of each phase is exactly equal to a fraction of the depth, *i.e.*, $\lfloor(\mu/2)P\rfloor$. This section ends by comparing the bit complexity of the two algorithms – the $\mathscr{LE}_\mu$ algorithm and the $\overline{\mathscr{LE}}_\mu$ algorithm.

In the top horizontal line, the jth phase begins with the pair $(A_0^{(j)},A_1^{(j)})$, while the bottom horizontal line starts with

$$\left(a_0^{(j)},a_1^{(j)}\right):=T_m\left(A_0^{(j)},A_1^{(j)}\right).$$

At the ith step of the jth phase, the Lehmer–Euclid Algorithm deals with the pair $(a_{i-1}^{(j)},a_i^{(j)})$, while the Euclid Algorithm (if performed) would deal with the pair $(A_{i-1}^{(j)},A_i^{(j)})$. Except at the first step of the jth phase, it is not true that the small pair $(a_{i-1}^{(j)},a_i^{(j)})$ is the truncation of the large pair $(A_{i-1}^{(j)},A_i^{(j)})$: usually this is even wrong. However, the next lemma shows that these pairs have almost the same length.

Lemma 6.1. *Let $(a_{i-1}^{(j)},a_i^{(j)})$ denote the pair used by the Lehmer–Euclid Algorithm at the ith step of the jth phase, and let $(A_{i-1}^{(j)},A_i^{(j)})$ be the pair that would be used by the Euclid Algorithm at the corresponding step. Then,*

$$\left|\ell\left(a_i^{(j)}\right)-\ell\left(A_i^{(j)}\right)+\ell\left(A_0^{(j)}\right)-m\right|\leqslant 2.$$

https://doi.org/10.1017/S0963548304006261 Published online by Cambridge University Press

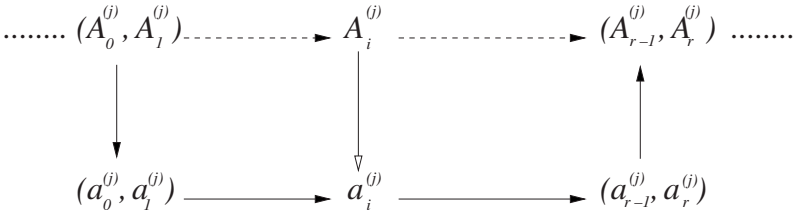


Figure 7. The jth phase of the Lehmer–Euclid Algorithm

Proof. We fix a phase, and we omit the index j. We use the notation of Section 2.1. Since the two matrices $\mathscr{M}_i$ (the 'large' one and the 'small' one) are equal, we have

$$\begin{pmatrix} A_{i+1} \\ A_i \end{pmatrix} = \mathscr{M}_i^{-1} \begin{pmatrix} A_1 \\ A_0 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} a_{i+1} \\ a_i \end{pmatrix} = \mathscr{M}_i^{-1} \begin{pmatrix} a_1 \\ a_0 \end{pmatrix}. \tag{6.1}$$

On the other hand, since the pair (a_0, a_1) is the m-truncation of the pair (A_0, A_1),

$$\begin{pmatrix} A_1 \\ A_0 \end{pmatrix} = 2^{\ell(A_0)-m} \left[\begin{pmatrix} a_1 \\ a_0 \end{pmatrix} + \begin{pmatrix} \delta_1 \\ \delta_0 \end{pmatrix} \right],$$

where δ_0, δ_1 satisfy $0 \leqslant \delta_0, \delta_1 < 1$. Then,

$$\begin{pmatrix} A_{i+1} \\ A_i \end{pmatrix} = 2^{\ell(A_0)-m} \left[\begin{pmatrix} a_{i+1} \\ a_i \end{pmatrix} + \mathscr{M}_i^{-1} \begin{pmatrix} \delta_1 \\ \delta_0 \end{pmatrix} \right].$$

Moreover, during each step, the relation $a_0 = |v_{i-1}|\, a_i + |v_i|\, a_{i-1}$ together with the inequalities $a_i, a_{i-1} > \sqrt{a_0}$ prove that the absolute values of all the coefficients of the matrices $\mathscr{M}_{i-1}, \mathscr{M}_{i-1}^{-1}$ are less than $(1/2)\sqrt{a_0} \leqslant (1/2)a_i$. Finally, the relation

$$|A_i - 2^{\ell(A_0)-m} a_i| \leqslant 2^{\ell(A_0)-m} \frac{a_i}{2}$$

proves the lemma. □

The bit cost of the jth phase decomposes into three bit costs. The first one (type 1) is the bit cost of an Interrupted Euclid Algorithm, the second one (type 2) is the extra cost due to the computation of the two co-sequences, and the third one (type 3) is due to the four multiplications of Stage 3. The second and the third costs (types 2 and 3) involve quantities that could have been computed during the Euclid Algorithm of the top horizontal line, so the Dirichlet series of these two costs admit expressions that involve the transfer operators relative to the Euclidean dynamical system, with a possible intervention of the cost $d = c$ or $d = b$, and of the functional Δ. The first cost (type 1) involves the quantities previously denoted by $\ell(a_i^{(j)})$ that are not computed by the 'top' Euclid Algorithm. However, Lemma 6.1 proves that these quantities can be approximated by other quantities $\ell(A_0^{(j)}), \ell(A_i^{(j)})$, computed by the 'top' Euclid Algorithm.

We let J denote the number of phases, let $p(j)$ be the beginning index of the jth phase, and let $v[r, t]$ be the coefficient v that is computed between the two indices $i = r$ and $i = t$.

https://doi.org/10.1017/S0963548304006261 Published online by Cambridge University Press

More precisely, $v[r,t]$ is the Bezout coefficient that would be computed if one were to start the Extended Euclid Algorithm with input pair (A_r, A_{r+1}) and stop with integer A_t. Here are the different costs involved during the algorithm.

Array 1. Costs:

$$\text{cost of type 1} \quad \sum_{j=1}^{J} \sum_{i=p(j)}^{p(j+1)-1} [\ell(A_i) - \ell(A_{p(j)}) + m] \cdot b(q_i),$$

$$\text{cost of type 2} \quad 2\sum_{j=1}^{J} \sum_{i=p(j)}^{p(j+1)-1} \ell(v[p(j), i]) \cdot c(q_i),$$

$$\text{cost of type 3} \quad 4\sum_{j=1}^{J} \ell(A_{p(j)}) \cdot \ell(v[p(j), p(j+1)]).$$

In fact, the cost of type 1 decomposes into three different costs. The first one gives rise to the bit complexity of the plain Euclid Algorithm during the phase and the third one was analysed in Section 5.3. There remains the second part, which we call the cost of type 1′:

$$\text{cost of type 1}' \quad \sum_{j=1}^{J} \ell(A_{p(j)}) \cdot \sum_{i=p(j)}^{p(j+1)-1} b(q_i).$$

All the previous expressions heavily depend on the sequence of indices $p(j)$ that denote the beginning of the jth phase. The next lemma proves that the length of each phase is almost surely the expected one, namely equal to $\bar{p} := \lfloor (\mu/2)p \rfloor$, so that the sequence $p(j)$ is not too far from an arithmetic progression. This lemma is an (easy) extension of Theorem 2.1.

Lemma 6.2. *Let $\delta(j)$ denote the number of iterations performed during the jth phase. For any ε, there exists $K < 1$ for which, as $n \to \infty$,*

$$\Pr_n \left[\left| \frac{\delta(j)}{P} - \frac{\mu}{2} \right| > \varepsilon \right] = O(K^n).$$

Proof. If a phase begins at the rth iteration of the Euclid Algorithm, it ends at the $(r+t)$th iteration, as soon as the (small) sequence a_i satisfies some condition. However, the previous lemma proves that there is a precise relation between the length of small a_i and the length of large A_i. In fact, the phase ends as soon as

$$A_{r+t} \sim \frac{A_r}{A_0^{\mu/2}}.$$

As in the proof of Theorem 2.1, we use Markov's inequality and we are led to study the Dirichlet series relative to the cost

$$N(A_0, A_1) := \left(\frac{A_{r+t}}{A_r A_0^{-\mu/2}} \right)^{\gamma}$$

for some $\gamma > 0$. We use the same notation as in Table 1 with $\alpha = 1 - (\mu/2)$ and $\beta = \mu/2$. This Dirichlet series admits an alternative expression that involves the transfer operator $\mathbf{H}_s$, as

$$\mathbf{H}_{s^-}^{p-r-t} \circ \mathbf{H}_{s^+}^{t} \circ \mathbf{H}_{s^-}^{r}.$$

https://doi.org/10.1017/S0963548304006261 Published online by Cambridge University Press

It is important to note that the dominant part of the function does not depend on the index r that denotes the moment when the phase begins. If we let $p := (c+d)k, t := ck$, then the function ϕ that is relative to the dominant singularity is exactly the same as in Lemma 4.2, *i.e.*, $\phi(s) := \lambda^d(s^-)\lambda^c(s^+)$. The end of the proof is exactly the same as in Section 4. □

Since the length of each phase is almost surely equal to $\bar{p} := \lfloor(\mu/2)p\rfloor$, we now consider the regularized version of the Lehmer–Euclid Algorithm, where the length of each phase exactly equals $\bar{p} := \lfloor(\mu/2)p\rfloor$. We denote this algorithm by $\overline{\mathscr{LE}}_\mu$. We shall prove two facts. First, the expectation of the bit cost of the $\overline{\mathscr{LE}}_\mu$ algorithm on $\Omega_n, \widetilde{\Omega}_n$ is given by the expression of Theorem 2.3. Second, the difference between the two average bit costs, the average bit cost of the $\overline{\mathscr{LE}}_\mu$ algorithm on $\Omega_n, \widetilde{\Omega}_n$ and the average bit cost of the $\mathscr{LE}_\mu$ algorithm on $\Omega_n, \widetilde{\Omega}_n$ is $o(n^2)$. We first consider the regularized version of the algorithm.

Lemma 6.3. *The expectation of the bit cost during the jth phase of the $\overline{\mathscr{LE}}_\mu$ algorithm on $\Omega_n, \widetilde{\Omega}_n$ is asymptotically equal to*

$$\frac{3}{4}(L_1M + L_2D)\mu^2n^2 + \frac{1}{2}(L_1+L_2)M\mu^2n^2 + 4\frac{\mu}{2}\left(1-(j-1)\frac{\mu}{2}\right)Mn^2.$$

Proof. We first obtain an expression for the costs of the $\overline{\mathscr{LE}}_\mu$ algorithm during the jth phase when the depth of the algorithm equals p.

Array 2. Costs of the regularized algorithm during the jth phase:

$$\begin{aligned}
&\text{cost of type } 1' \quad \sum_{i=(j-1)\bar{p}}^{j\bar{p}-1} \ell(A_{(j-1)\bar{p}}) \cdot b(q_i),\\
&\text{cost of type } 2 \quad 2\sum_{i=(j-1)\bar{p}}^{j\bar{p}-1} \ell(v[(j-1)\bar{p}, i]) \cdot c(q_i),\\
&\text{cost of type } 3 \quad 4\ell\left(A_{(j-1)\bar{p}}\right) \cdot \ell(v[(j-1)\bar{p}, j\bar{p}]).
\end{aligned}$$

We easily deduce an expression for the Dirichlet series of 'regularized' costs when the depth equals p. In fact, we work with some approximated costs that are the analogues of the costs denoted by hats in the previous section.

Array 3. Dirichlet series for costs of the regularized algorithm during the jth phase:

$$\begin{aligned}
&\text{cost of type } 1' \quad \sum_{i=1}^{\bar{p}} \Delta\left[\mathbf{H}_s^{p-j\bar{p}} \circ \mathbf{H}_s^{\bar{p}-i} \circ \mathbf{H}_s^{[b]} \circ \mathbf{H}_s^{i-1}\right] \circ \mathbf{H}_s^{(j-1)\bar{p}}[f](0),\\
&\text{cost of type } 2 \quad 2\sum_{i=1}^{\bar{p}} \mathbf{H}_s^{p-j\bar{p}} \circ \mathbf{H}_s^{\bar{p}-i} \circ \mathbf{H}_s^{[c]} \circ \Delta\left[\mathbf{H}_s^{i-1}\right] \circ \mathbf{H}_s^{(j-1)\bar{p}}[f](0),\\
&\text{cost of type } 3 \quad 4\Delta\left[\mathbf{H}_s^{p-j\bar{p}} \circ \Delta\left(\mathbf{H}_s^{\bar{p}}\right)\right] \circ \mathbf{H}_s^{(j-1)\bar{p}}[f](0).
\end{aligned}$$

When summing over all possible values of depth p, the dominant terms of the Dirichlet series of Array 3 all involve a product of two factors. The first factor is the same for all

https://doi.org/10.1017/S0963548304006261 Published online by Cambridge University Press

the three series and is equal to

$$\sum_{p\geqslant 0} p^2\lambda(s)^p \sim 2\left(\frac{1}{1-\lambda(s)}\right)^3. \tag{6.2}$$

Array 4. The other factors are respectively equal to:

$$\text{cost of type } 1' \quad \frac{\mu}{2}\left[1-\frac{\mu}{2}(j-1)\right]\ \mathbf{P}_s\circ\Delta\mathbf{H}_s\circ\mathbf{P}_s\circ\mathbf{H}_s^{[b]}\circ\mathbf{P}_s[f](0),$$

$$\text{cost of type } 2 \quad 2\frac{1}{2}\left(\frac{\mu}{2}\right)^2\ \mathbf{P}_s\circ\mathbf{H}_s^{[c]}\circ\mathbf{P}_s\circ\Delta\mathbf{H}_s\circ\mathbf{P}_s[f](0),$$

$$\text{cost of type } 3 \quad 4\frac{\mu}{2}\left[1-\frac{\mu}{2}(j-1)\right]\ \mathbf{P}_s\circ\Delta\mathbf{H}_s\circ\mathbf{P}_s\circ\Delta\mathbf{H}_s\circ\mathbf{P}_s[f](0).$$

Near $s=2$, all these functions are analytic and their values at $s=2$ involve the same integrals as in (5.7). As in Section 5.4, it is easy to compare costs with a hat and costs with a bar. This ends the proof of the lemma. □

When summing over the index j, which varies between 1 and $\bar{J} := \lceil 2/\mu\rceil$, we get the expression given in Theorem 2.3. It then remains to compare the bit complexity of the two algorithms, the Lehmer–Euclid Algorithm $\mathscr{LE}_\mu$ and its regularized version $\overline{\mathscr{LE}}_\mu$. This is the purpose of the following lemma.

Lemma 6.4. *When $n\to\infty$, the average bit complexity of the Lehmer–Euclid Algorithm $\mathscr{LE}_\mu$ and the average bit complexity of its regularized version $\overline{\mathscr{LE}}_\mu$ are asymptotically the same.*

Proof. Lemma 6.2 proves that the length $\delta(j)$ of the jth phase is almost surely close to $\bar{p} := \lfloor(\mu/2)p\rfloor$. We then split the set of inputs Ω into two subsets: a subset of inputs that have an exceptional behaviour, and an ordinary subset. Consider, for some $\varepsilon>0$, the event

$$D(\varepsilon) := [\exists j\leqslant J,\quad |\delta(j)-\bar{p}|>\varepsilon p]. \tag{6.3}$$

Lemma 6.2 proves that there exists a $K<1$ for which $\Pr_n[D(\varepsilon)] = O(K^n)$, so that the subset $D(\varepsilon)$ is exceptional. It is sufficient to study the bit complexity of the $\mathscr{LE}_\mu$ algorithm on the complementary subset of $D(\varepsilon)$. Here, there are two 'extremal ordinary values',

$$\bar{p}_- := \left\lfloor\left(\frac{\mu}{2}-\varepsilon\right)p\right\rfloor, \qquad \bar{p}_+ := \left\lceil\left(\frac{\mu}{2}+\varepsilon\right)p\right\rceil,$$

and the beginning indices $p(j), p(j+1)$ satisfy

$$j\bar{p}_-\leqslant p(j)\leqslant j\bar{p}_+,$$

$$[j\bar{p}_+,(j+1)\bar{p}_-]\subset[p(j),p(j+1)]\subset[j\bar{p}_-,(j+1)\bar{p}_+].$$

https://doi.org/10.1017/S0963548304006261 Published online by Cambridge University Press

Note that the number J of phases satisfies

$$J^- := \frac{2}{\mu + 2\varepsilon} \leqslant J \leqslant J^+ := \frac{2}{\mu - 2\varepsilon}, \tag{6.4}$$

and note that the length of the large interval $[j\bar{p}_-, (j+1)\bar{p}_+]$ is less than $\bar{p}_{++} := \bar{p} + J^+\varepsilon p$, while the length of the small interval is greater than $\bar{p}_{--} := \bar{p} - J^-\varepsilon p$. The function $i \mapsto \ell(A_i)$ is a decreasing function, and the function $[r,t] \mapsto v[r,t]$ is an increasing function. Finally, for each cost that has been previously defined, we provide an upper bound and a lower bound on the ordinary subset.

Array 5. Upper ordinary bounds:

$$\begin{aligned}
&\text{cost of type } 1' \quad \sum\nolimits_{j=1}^{J^+} \ell\left(A_{(j-1)\bar{p}_-}\right) \cdot \sum\nolimits_{i=(j-1)\bar{p}_-}^{j\bar{p}_+-1} b(q_i),\\
&\text{cost of type } 2 \quad 2\sum\nolimits_{j=1}^{J^+} \sum\nolimits_{i=(j-1)\bar{p}_-}^{j\bar{p}_+-1} \ell(v[(j-1)\bar{p}_-, i]) \cdot c(q_i),\\
&\text{cost of type } 3 \quad 4\sum\nolimits_{j=1}^{J^+} \ell\left(A_{(j-1)\bar{p}_-}\right) \cdot \ell(v[(j-1)\bar{p}_-, j\bar{p}_+]).
\end{aligned}$$

Array 6. Lower ordinary bounds:

$$\begin{aligned}
&\text{cost of type } 1' \quad \sum\nolimits_{j=1}^{J^-} \ell\left(A_{(j-1)\bar{p}_+}\right) \cdot \sum\nolimits_{i=(j-1)\bar{p}_+}^{j\bar{p}_--1} b(q_i),\\
&\text{cost of type } 2 \quad 2\sum\nolimits_{j=1}^{J^-} \sum\nolimits_{i=(j-1)\bar{p}_+}^{j\bar{p}_--1} \ell(v[(j-1)\bar{p}_+, i]) \cdot c(q_i),\\
&\text{cost of type } 3 \quad 4\sum\nolimits_{j=1}^{J^-} \ell\left(A_{(j-1)\bar{p}_+}\right) \cdot \ell(v[(j-1)\bar{p}_+, j\bar{p}_-]).
\end{aligned}$$

Here are the Dirichlet series of various costs relative to jth phase, when the total depth of the algorithm equals p. In fact, we work with some approximate costs that are the analogues of the costs denoted by hats in the previous section.

Array 7. Dirichlet series for upper ordinary bounds of costs during the jth phase:

$$\begin{aligned}
&\text{cost of type } 1' \quad \sum\nolimits_{i=1}^{\bar{p}_{++}} \Delta\left[\mathbf{H}_s^{p-(j-1)\bar{p}_- - i} \circ \mathbf{H}_s^{[b]} \circ \mathbf{H}_s^{i-1}\right] \circ \mathbf{H}_s^{(j-1)\bar{p}_-}[f](0),\\
&\text{cost of type } 2 \quad 2\sum\nolimits_{i=1}^{\bar{p}_{++}} \mathbf{H}_s^{p-(j-1)\bar{p}_- - i} \circ \mathbf{H}_s^{[c]} \circ \Delta\left[\mathbf{H}_s^{i-1}\right] \circ \mathbf{H}_s^{(j-1)\bar{p}_-}[f](0),\\
&\text{cost of type } 3 \quad 4\Delta\left[\mathbf{H}_s^{p-j\bar{p}_+} \circ \Delta\left(\mathbf{H}_s^{\bar{p}_{++}}\right)\right] \circ \mathbf{H}_s^{(j-1)\bar{p}_-}[f](0).
\end{aligned}$$

Array 8. Dirichlet series for lower ordinary bounds of costs during the jth phase:

$$\begin{aligned}
&\text{cost of type } 1' \quad \sum\nolimits_{i=1}^{\bar{p}_{--}} \Delta\left[\mathbf{H}_s^{p-(j-1)\bar{p}_+ - i} \circ \mathbf{H}_s^{[b]} \circ \mathbf{H}_s^{i-1}\right] \circ \mathbf{H}_s^{(j-1)\bar{p}_+}[f](0),\\
&\text{cost of type } 2 \quad 2\sum\nolimits_{i=1}^{\bar{p}_{--}} \mathbf{H}_s^{p-(j-1)\bar{p}_+ - i} \circ \mathbf{H}_s^{[c]} \circ \Delta\left[\mathbf{H}_s^{i-1}\right] \circ \mathbf{H}_s^{(j-1)\bar{p}_+}[f](0),\\
&\text{cost of type } 3 \quad 4\Delta\left[\mathbf{H}_s^{p-j\bar{p}_-} \circ \Delta\left(\mathbf{H}_s^{\bar{p}_{--}}\right)\right] \circ \mathbf{H}_s^{(j-1)\bar{p}_+}[f](0).
\end{aligned}$$

https://doi.org/10.1017/S0963548304006261 Published online by Cambridge University Press

When summing over all possible values of depth p, the dominant terms of the Dirichlet series of Array 7 or Array 8 all involve a product of two factors. The first factor is the same for all the three series of the two arrays and is the same as in (6.2), that is,

$$\sum_{p\geqslant 0} p^2 \lambda(s)^p \sim 2\left(\frac{1}{1-\lambda(s)}\right)^3.$$

It remains to study the second factors and compare them to the expressions given in Array 4. The operators involved are the same as in Array 4, and the constants of Array 4,

$$A(\mu) := \frac{\mu}{2}\left[1-\frac{\mu}{2}(j-1)\right], \qquad B(\mu) := \frac{1}{2}\left(\frac{\mu}{2}\right)^2,$$

are respectively replaced by $A(\mu)+O(\varepsilon), B(\mu)+O(\varepsilon)$. We now sum over all possible values of index j. This index varies (according to the cases) between 1 and J^- or between 1 and J^+ (these values are defined in (6.4)). We finally obtain the result. □

This ends the proof of Theorem 2.3, which is the main result of the paper.

The previous results prove that all transpires as if we could directly apply the heuristic reasoning of Section 2. We recall that Section 3 provides a first explanation of this fact, based on the description of the evolution of the density on Ω during the execution of the Euclid Algorithm.

7. Conclusion

This paper has provided for the first time an average-case analysis of the Lehmer–Euclid Algorithm when the truncation degree μ is constant. This is not the algorithm that is used in most applications, though. Usually, truncations are designed so as to transform multi-precision integers into fixed-precision integers. In this case, the truncation degree is no longer a constant but is of the form c/n for some constant c. In order to deal with this algorithm, we would have to change our analysis, as it currently applies only to constant values of μ. This would require in an essential way uniform bounds and error terms, which are not obtainable by means of Tauberian Theorems. Hopefully new results [2] concerning the transfer operator associated to the Euclidean algorithm can perhaps serve for this purpose. An alternative approach might be to analyse directly the Interrupted Euclid Algorithm, which stops as soon as the current integer has lost a constant number of bits.

This first study may be of interest from several other points of view. Here, we have analysed a standard version of the parametrized Lehmer–Euclid Algorithm. Collins [7], Jebelean [14], and many others have proposed improvements on the Lehmer–Euclid Algorithm. For instance, it is sufficient to compute only one co-sequence and recover the second one at the end of each phase, at the expense of an additional product. On the other hand, Jebelean remarks that, when truncated numbers with m bits are used, almost all the operations performed involve numbers of $m/2$ bits, very few operations actually requiring numbers with m bits. Accordingly, he proposes to work with double-precision numbers. So far, such variants have only been assessed via experiments and empirically determined

https://doi.org/10.1017/S0963548304006261 Published online by Cambridge University Press

computation times. We provide in this paper a new theoretical tool that can provide an alternative way to quantify precisely the effect of such algorithmic optimizations.

Finally, we should mention the recursive version of the Lehmer–Euclid Algorithm, $\mathscr{RLE}$ in short. It is based on a divide-and-conquer principle, and replaces computations on n-bit integers by operations on μn-bit integers (with μ near $1/2$). The design of this algorithm, initially proposed by Schönhage [21], is now more precise, after recent progress due to the works of Cesari [6], Zimmermann [30] and Stehlé [24]. It is well described in the book by Yap [29]. It is our intention to analyse the $\mathscr{RLE}$ Algorithm and we believe that many mathematical tools introduced here can be used for this task.

Acknowledgements

We wish to thank Philippe Flajolet and Damien Stehlé for many interesting discussions related to this paper.

References

[1] Akhavi, A. and Vallée, B. (2000) Average bit-complexity of Euclidean algorithms. In *Proc. ICALP'2000*, Vol. 1853 of *Lecture Notes in Computer Science*, Springer, pp. 373–387.
[2] Baladi, V. and Vallée, B. (2003) Euclidean algorithms are Gaussian. *Les Cahiers du GREYC*. Also available from the ArXiv. Short version: Distributional analyses of Euclidean algorithms, `http://users.info.unicaen.fr/%7Ebrigitte/Publications/Siam-analco.ps`, *Proc. ANALCO'04*, to appear.
[3] Bedford, T., Keane, M. and Series, C., eds (1991) *Ergodic Theory, Symbolic Dynamics and Hyperbolic Spaces*, Oxford University Press.
[4] Bourdon, J., Daireaux, B. and Vallée, B. (2002) Dynamical analysis of α-Euclidean algorithms. *J. Algorithms* **44** 246–285.
[5] Brent, R. P. (1976) Analysis of the Binary Euclidean algorithm. In *Algorithms and Complexity: New Directions and Recent Results* (J. F. Traub, ed.), Academic Press, pp. 321–355.
[6] Cesari, G. (1998) Parallel Implementation of Schönhage's Integer GCD Algorithm. In *Proc. ANTS-III*, Vol. 1423 of *Lecture Notes in Computer Science*, Springer, pp. 64–76.
[7] Collins, G. E. Lecture notes on arithmetic algorithms, University of Wisconsin.
[8] Delange, H. (1954) Généralisation du Théorème d'Ikehara. *Ann. Sc. ENS* **71** 213–242.
[9] Dixon, J. D. (1970) The number of steps in the Euclidean algorithm. *J. Number Theory* **2** 414–422.
[10] Flajolet, P. (1992) Analytic analysis of algorithms. In *Proc. 19th International Colloquium 'Automata, Languages and Programming', Vienna* (W. Kuich, ed.), Vol. 623 of *Lecture Notes in Computer Science*, Springer pp. 186–210.
[11] Flajolet, P. and Sedgewick, R. (1999) *Analytic Combinatorics*, book in preparation. See also INRIA Research Reports 1888, 2026, 2376, 2956.
[12] Heilbronn, H. (1969) On the average length of a class of continued fractions. In *Number Theory and Analysis* (P. Turan, ed.), Plenum, New York, pp. 87–96.
[13] Hensley, D. (1994) The number of steps in the Euclidean algorithm. *J. Number Theory* **49** 142–182.
[14] Jebelean, T. (1995) A double-digit Lehmer–Euclid algorithm for finding the GCD of long integers. *J. Symbolic Comput*. **19** 145–157.
[15] Jebelean, T. Practical integer division with Karatsuba complexity. In *Proc. ISSAC'97*, pp. 339–341.
[16] Knuth, D. E. (1998) *The Art of Computer Programming*, Vol. 2, 3rd edn, Addison-Wesley, Reading, MA.

https://doi.org/10.1017/S0963548304006261 Published online by Cambridge University Press

[17] Krandick, W. and Johnson, J. R. (1993) Efficient multi-precision floating point multiplication with optimal directional rounding. In *Proc. 11th IEEE Symposium in Computer Arithmetic*, pp. 228–233.
[18] Lehmer, D. H. (1938) Euclid's algorithm for large numbers. *Amer. Math. Mon.* **45** 227–233.
[19] Lercier, R. (1997) Algorithmique des courbes elliptiques dans les corps finis, PhD Thesis, Ecole Polytechnique.
[20] Ruelle, D. (1978) *Thermodynamic Formalism*, Addison-Wesley.
[21] Schönhage, A. (1971) Schnelle Berechnung von Kettenbruchentwicklungen. *Acta Informatica*, pp. 139–144.
[22] Schönhage, A., Grotefeld, A. F. W. and Vetter, E. (1994) *Fast Algorithms: A Multitape Turing Machine Implementation*, BI-Wissenschaftsverlag, Mannheim.
[23] Sorenson, J. (1995) An analysis of Lehmer's Euclidean GCD Algorithm. In *Proc. ISSAC'95*, pp. 254–258.
[24] Stehlé, D. (2001) Les algorithmes hGCD sur les entiers et les polynômes, Master's Thesis, University of Caen.
[25] Tenenbaum, G. (1990) *Introduction à la Théorie Analytique des Nombres*, Vol. 13, Institut Élie Cartan, Nancy, France.
[26] Vallée, B. (1998) Dynamics of the Binary Euclidean Algorithm: Functional analysis and operators. *Algorithmica* **22** 660–685.
[27] Vallée, B. (2000) Digits and continuants in Euclidean algorithms: Ergodic versus Tauberian theorems. *J. Théorie des Nombres de Bordeaux* **12** 531–570.
[28] Vallée, B. (2003) Dynamical analysis of a class of Euclidean algorithms. *Theoret. Comput. Sci.* **297** 447–486.
[29] Yap, C. K. (1996) *Fundamental Problems in Algorithmic Algebra*, Princeton University Press.
[30] Zimmermann, P. (2001) A proof of subquadratic gcd. Personal communication.

https://doi.org/10.1017/S0963548304006261 Published online by Cambridge University Press