Gcd computations & multidimensional continued fractions

V. Berthé, L. Lhote, B. Vallée

LIAFA-CNRS-Paris-France
berthe@liafa.univ-paris-diderot.fr
http ://www.liafa.jussieu.fr/~berthe

Discrete mathematics,
VMS-SMF Joint Conference, Hue, 2012

Gcd algorithms and beyond

We want to

- compute the gcd of n numbers

Gcd algorithms and beyond

We want to

- compute the gcd of n numbers
  - $n = 3$ or n large
  - small/big size
  - same size/different sizes

Gcd algorithms and beyond

We want to

- compute the gcd of n numbers
  - $n = 3$ or n large
  - small/big size
  - same size/different sizes
- find Bezout's coefficients : extended gcd
- find simultaneous rational approximations

Gcd algorithms and beyond

We want to

- compute the gcd of n numbers
  - $n = 3$ or n large
  - small/big size
  - same size/different sizes
- find Bezout's coefficients : extended gcd
- find simultaneous rational approximations

How to compare multidimensional gcd algorithms ?

Euclid algorithm

We start with two nonnegative integers u_0 and u_1

$$
\begin{aligned}
u_0 &= u_1 \left\lfloor \frac{u_0}{u_1} \right\rfloor + u_2 \\
u_1 &= u_2 \left\lfloor \frac{u_1}{u_2} \right\rfloor + u_3 \\
&\vdots \\
u_{m-1} &= u_m \left\lfloor \frac{u_{m-1}}{u_m} \right\rfloor + u_{m+1} \\
u_{m+1} &= \gcd(u_0, u_1) \\
u_{m+2} &= 0
\end{aligned}
$$

Euclid algorithm

We start with two nonnegative integers u_0 and u_1

$$u_0 = u_1 \left[\frac{u_0}{u_1}\right] + u_2$$

$$u_1 = u_2 \left[\frac{u_1}{u_2}\right] + u_3$$

$$\vdots$$

$$u_{m-1} = u_m \left[\frac{u_{m-1}}{u_m}\right] + u_{m+1}$$

$$u_{m+1} = \mathrm{gcd}(u_0, u_1)$$

$$u_{m+2} = 0$$

One subtracts the smallest number to the largest as much as we can

Euclid algorithm and continued fractions

We start with two coprime integers u_0 and u_1

$$u_0 = u_1 a_1 + u_2$$

$$\vdots$$

$$u_{m-1} = u_m a_m + u_{m+1}$$

$$u_m = u_{m+1} a_{m+1} + 0$$

$$u_{m+1} = 1 = \gcd(u_0, u_1)$$

Euclid algorithm and continued fractions

We start with two coprime integers u_0 and u_1

$$u_0 = u_1 a_1 + u_2$$

$$\vdots$$

$$u_{m-1} = u_m a_m + u_{m+1}$$

$$u_m = u_{m+1} a_{m+1} + 0$$

$$u_{m+1} = 1 = \gcd(u_0, u_1)$$

Euclid's algorithm yields the digits
for the continued fraction expansion of $\frac{u_1}{u_0}$

Euclid algorithm and continued fractions

We start with two coprime integers u_0 and u_1

$$u_0 = u_1 a_1 + u_2$$

$$\vdots$$

$$u_{m-1} = u_m a_m + u_{m+1}$$

$$u_m = u_{m+1} a_{m+1} + 0$$

$$u_{m+1} = 1 = \gcd(u_0, u_1)$$

$$\frac{u_1}{u_0} = \frac{1}{a_1 + \frac{u_2}{u_1}}$$

$$u_1/u_0 = \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cdots + \cfrac{1}{a_m + \frac{1}{a_{m+1}}}}}$$

Multidimensional case

- Euclid's algorithm

  ◁ Starting with two numbers, one subtracts the smallest to the largest

  ◁ Starting with three numbers, which subtraction/division has to be done ?

Multidimensional case

- Continued fractions and unimodularity

$$\frac{p_n}{q_n} := \cfrac{1}{a_1 + \cfrac{1}{\cdots + \frac{1}{a_n}}} \qquad \det \begin{bmatrix} p_{n+1} & q_{n+1} \\ p_n & q_n \end{bmatrix} = \pm 1$$

$\triangleleft$ $SL(2, \mathbb{N})$ is a finitely generated free monoid generated by

$$\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \text{ and } \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

$\triangleleft$ $SL(3, \mathbb{N})$ is not finitely generated. Consider the family of undecomposable matrices for $n \geq 3$ [Rivat]

$$\begin{pmatrix} 1 & 0 & n \\ 1 & n-1 & 0 \\ 1 & 1 & n-1 \end{pmatrix}$$

Multidimensional continued fractions

There is no canonical generalization of continued fractions to higher dimensions

There is no canonical generalization of Euclid algorithm to higher dimensions

Multidimensional continued fractions

There is no canonical generalization of continued fractions to higher dimensions

Several approaches are possible

- best simultaneous approximations but we then loose unimodularity, and the sequence of best approximations heavily depends on the chosen norm [Lagarias]

Multidimensional continued fractions

There is no canonical generalization of continued fractions to higher dimensions

Several approaches are possible

- best simultaneous approximations but we then loose unimodularity, and the sequence of best approximations heavily depends on the chosen norm [Lagarias]
- Klein polyhedra and sails [Arnold]

Multidimensional continued fractions

There is no canonical generalization of continued fractions to higher dimensions

Several approaches are possible

- best simultaneous approximations but we then loose unimodularity, and the sequence of best approximations heavily depends on the chosen norm [Lagarias]
- Klein polyhedra and sails [Arnold]
- unimodular algorithms
  - Lattice reduction (LLL) [Lagarias,Ferguson-Forcade,Just,Havas-Majewski-Matthews etc.]

Multidimensional continued fractions

There is no canonical generalization of continued fractions to higher dimensions

Several approaches are possible

- best simultaneous approximations but we then loose unimodularity, and the sequence of best approximations heavily depends on the chosen norm [Lagarias]
- Klein polyhedra and sails [Arnold]
- unimodular algorithms
  - Lattice reduction (LLL) [Lagarias,Ferguson-Forcade,Just,Havas-Majewski-Matthews etc.]
  - $\rightsquigarrow$ **continued fractions based on the iteration of piecewise fractional linear maps** Jacobi-Perron, Brun, Selmer, Poincaré etc. [Brentjes, Schweiger]

Multidimensional Euclid's algorithms

- Jacobi-Perron We subtract the first one to the two other ones with $0 \leq u_1, u_2 \leq u_3$

$$(u_1, u_2, u_3) \mapsto (u_2 - [\frac{u_2}{u_1}]u_1, u_3 - [\frac{u_3}{u_1}]u_1, u_1)$$

- Brun We subtract the second largest entry and we reorder. If $u_1 \leq u_2 \leq u_3$

$$(u_1, u_2, u_3) \mapsto (u_1, u_2, u_3 - u_2)$$

- Poincaré We subtract the previous entry and we reorder

$$(u_1, u_2, u_3) \mapsto (u_1, u_2 - u_1, u_3 - u_2)$$

- Selmer We subtract the smallest to the largest and we reorder

$$(u_1, u_2, u_3) \mapsto (u_1, u_2, u_3 - u_1)$$

- Fully subtractive We subtract the smallest one to the other ones and we reorder

$$(u_1, u_2, u_3) \mapsto (u_1, u_2 - u_1, u_3 - u_1)$$

Why chosing these algorithms ?

- They are not the best ones in terms of Diophantine approximation compared to algorithms based on lattice reduction
- They are not the fastest ones experimentally for gcd computation

There exist subquadractic gcd algorithms
[GMP= Möller'08]

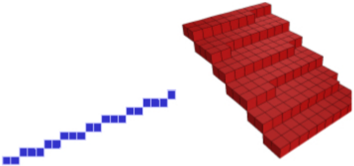
Why chosing these algorithms ?

But

- They can be described by a simple dynamical system

$\rightsquigarrow$ dynamical analysis of multidimensional gcd algorithms

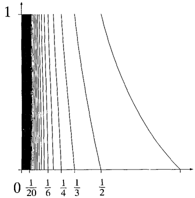
- They thus can be easily applied, for instance, in discrete geometry



Continued fractions and dynamical systems

Consider the Gauss map

$$T\colon [0,1] \to [0,1],\ x \mapsto \{1/x\}$$



Continued fractions and dynamical systems

Consider the Gauss map

$$T\colon [0,1] \to [0,1],\ x \mapsto \{1/x\}$$

$$x_1 = T(x) = \{1/x\} = \frac{1}{x} - \left[\frac{1}{x}\right] = \frac{1}{x} - a_1$$

$$x = \frac{1}{a_1 + x_1}$$

$$a_n = \left[\frac{1}{T^{n-1}x}\right]$$

$$x = \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{a_4 + \cdots}}}}$$

Rational vs. irrational parameters

Euclid algorithm $\rightsquigarrow$ gcd $\rightsquigarrow$ rational parameters

Continued fractions $\rightsquigarrow$ irrational parameters

Is it relevant to compare generic orbits
and orbits for integer parameters ?

Rational vs. irrational parameters

- When computing a gcd, we work with integer/rational parameters
- This set has zero measure
- Ergodic methods produce results that hold only almost everywhere

Average-case analysis vs. a.e. results

Fact Orbits of rational points tend to behave like generic orbits

And their probabilistic bevaviour can be captured thanks to the methods of dynamical analysis of algorithms

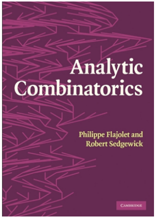
Dynamical analysis of algorithms [Vallée]

It belongs to the area of

- Analysis of algorithms [Knuth'63]

  probabilistic, combinatorial, and analytic methods

- Analytic combinatorics [Flajolet-Sedgewick]



generating functions and complex analysis,
analytic functions, analysis of the singularities

Dynamical analysis of algorithms [Vallée]

It mixes tools from

- dynamical systems (transfer operators, density transformers, Ruelle-Perron-Frobenius operators)
- analytic combinatorics (generating functions of Dirichlet type)

the singularities of (Dirichlet) generating functions
are expressed in terms of transfer operators

Euclidean dynamics [Vallée]

One starts with a discrete algorithm

- This algorithm is extended into a continuous one in terms of a dynamical system

  Orbits/trajectories = executions

- Main parameters of the algorithm are studied in the continuous framework

  rational trajectories $\leftrightarrow$ generic trajectories

- One comes back to the discrete algorithm

A transfer from continuous to discrete

"The probabilistic behaviour of gcd algorithms is quite similar to the behaviour of their continuous counterparts"

The floating-point Gauss map [Corless-Continued fractions and Chaos-(1992)]

Consider the Gauss map $T\colon [0,1] \to [0,1],\ x \mapsto \{1/x\}$

Theorem Orbits under the floating-point Gauss map are close to corresponding exact orbits cf. shadowing properties

The floating-point Gauss map [Corless-Continued fractions and Chaos-(1992)]

Consider the Gauss map $T\colon [0,1] \to [0,1],\ x \mapsto \{1/x\}$

Theorem Orbits under the floating-point Gauss map are close to corresponding exact orbits cf. shadowing properties

$$\lambda(x) = \lim_{n\to\infty} \frac{1}{n} \log \left(\prod_{i=0}^{n} |T'(T^i(x))| \right) = \frac{\pi^2}{6\log 2} \text{ for a.e. } x$$

This yields “a candidate for ‘the worlds’ worst’ algorithm for computing π. $[\cdots]$. This method is likely worse than nearly any other in existence, since it does **not** converge to the correct value in any particular fixed-precision system, since all orbits are eventually periodic, and the Lyapounov exponent of a periodic orbit is the logarithm of an algebraic number.$[\cdots]$. This method is clearly related to the Monte-Carlo methods, with the roundoff error associated with the floating-point arithmetic playing the part of the random number generator required”.

Transfer operators

Perron-Frobenius operator

associated with the Gauss map $T \colon x \mapsto \{1/x\}$

$$P[f](x) = \sum_{y\,:\,T(y)=x} \frac{1}{|T'(y)|} f(y) = \sum_{k \geq 1} \left(\frac{1}{k+x}\right)^2 f\left(\frac{1}{k+x}\right)$$

Transfer operators

Perron-Frobenius operator

associated with the Gauss map $T: x \mapsto \{1/x\}$

$$P[f](x) = \sum_{y\,:\,T(y)=x} \frac{1}{|T'(y)|} f(y) = \sum_{k\geq 1} \left(\frac{1}{k+x}\right)^2 f\left(\frac{1}{k+x}\right)$$

Take $f = \frac{1}{1+x}$, one has $P[f] = f$

The Gauss measure is defined on $[0, 1]$ as

$$\mu(B) = \frac{1}{\log 2} \int_B \frac{1}{1+x} \mathrm{dx}$$

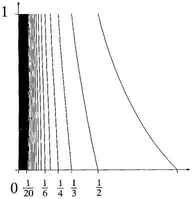
It is T-invariant : $\mu(B) = \mu(T^{-1}B), \forall B \in \mathcal{B}$

Transfer operators

Perron-Frobenius operator

associated with the Gauss map $T\colon x \mapsto \{1/x\}$

$$P[f](x) = \sum_{y\,:\,T(y)=x} \frac{1}{|T'(y)|} f(y) = \sum_{k\geq 1} \left(\frac{1}{k+x}\right)^2 f\left(\frac{1}{k+x}\right)$$



Let $\mathcal{H}$ stand for the set of inverse branches of the Gauss map

$$P[f](x) = \sum_{h\in\mathcal{H}} h'(x)\, f \circ h(x)$$

Transfer operators

Perron-Frobenius operator

associated with the Gauss map $T\colon x \mapsto \{1/x\}$

$$P[f](x) = \sum_{y\,:\,T(y)=x} \frac{1}{|T'(y)|} f(y) = \sum_{k\geq 1} \left(\frac{1}{k+x}\right)^2 f\left(\frac{1}{k+x}\right)$$

$$P[f](x) = \sum_{h\in\mathcal{H}} h'(x)\, f\circ h(x)$$

Ruelle operator

$$P_s[f](x) = \sum_{h\in\mathcal{H}} h'(x)^s\, f\circ h(x) \qquad s\in\mathbb{C}$$

Transfer operators and Brun algorithm

Each step of the algorithm is a linear fractional transformation
Let h_a be an inverse branch and J_a its Jacobian

$$P_{[a],s}[f](x) = J[h_a]^s(x)\ f \circ h_a(x)$$

Transfer operators and Brun algorithm

Defined on $\{(x_1, \ldots, x_d) \in [0,1]^d \; ; \; x_1 \geq x_2 \geq \ldots x_d\}$

- Brun transformation

$$T_B(x_1, x_2, \ldots, x_d) = \mathtt{ord}\left(\frac{x_2}{x_1}, \ldots, \frac{x_d}{x_1}, \left\{\frac{1}{x_1}\right\}\right)$$

$$m(x) = \left[\frac{1}{x_1}\right], \qquad j(x) = \mathtt{Pos}\left[\left\{\frac{1}{x_1}\right\}, \left(\frac{x_2}{x_1}, \ldots, \frac{x_{d-1}}{x_1}\right)\right]$$

- Inverse branch

$$h_{(m,j)}(y_1, y_2, \ldots, y_d) = \left(\frac{1}{m+y_j}, \frac{y_1}{m+y_j}, \ldots, \frac{y_{j-1}}{m+y_j}, \frac{y_{j+1}}{m+y_j}, \ldots, \frac{y_d}{m+y_j}\right)$$

- Jacobian $J[h_{(m,j)}](y) = \frac{1}{(m+y_j)^{d+1}}$

Generating functions and transfer operators

We are given a generalized Euclid algorithm

Ω : coprime entries $u = (u_0, \cdots, u_d)$ with $u_0 = \max(u_i)$

Generating cost function

$$S_C(s) := \sum_{u \in \Omega} \frac{C(u)}{u_0^s}$$

where C is a cost function

For instance, $C(u)$ is the number of steps performed by the generalized Euclid algorithm on $u = (u_0, \cdots, u_d)$

Generating functions and transfer operators

We are given a generalized Euclid algorithm

Ω : coprime entries $u = (u_0, \cdots, u_d)$ with $u_0 = \max(u_i)$

Generating cost function

$$S_C(s) := \sum_{u \in \Omega} \frac{C(u)}{u_0^s}$$

where C is a cost function

Fact

For the cost $C \equiv 1$

$$S_C(d+1) \sim (I - P_s)^{-1}[1](0)$$

since

$$\frac{1}{u_0^{d+1}} = J[h](0)$$

Generating functions and transfer operators

We are given a generalized Euclid algorithm

Ω : coprime entries $u = (u_0, \cdots, u_d)$ with $u_0 = \max(u_i)$

Generating cost function

$$S_C(s) := \sum_{u \in \Omega} \frac{C(u)}{u_0^s}$$

where C is a cost function

For a general cost C, we introduce a further parameter w

$$T_C(s, w) := \sum_{u \in \Omega} \frac{1}{u_0^s} \exp[wC(u)]$$

$$P_{[a],s,w}[f](x) := J[h_a]^s(x) \exp[wc(h_a)]\, f \circ h_a(x)$$

Mean behaviour of the number of steps Euclid algorithm

Consider parameters $(u_1, \cdots, u_d)$ with $0 \leq u_1, \cdots, u_d \leq N$

Thm Expectation of the number of steps $= \dfrac{\text{dimension}}{\text{Entropy}} \times \log N$

Dimension

- d= Number of parameters

Entropy

- Growth rate of convergents
- Speed of convergence
- Chaotic dynamical systems

Mean behaviour of the number of steps
Euclid algorithm

Consider parameters $(u_1, \cdots, u_d)$ with $0 \le u_1, \cdots, u_d \le N$

Thm Expectation of the number of steps $= \dfrac{\text{dimension}}{\text{Entropy}} \times \log N$

- Euclid algorithm

$$\frac{2}{\pi^2/(6\log 2)} \log N$$

[Heilbronn'69,Dixon'70,Hensley'94,Baladi-Vallée'03...]

Mean behaviour of the number of steps
Euclid algorithm

Consider parameters $(u_1, \cdots, u_d)$ with $0 \leq u_1, \cdots, u_d \leq N$

Thm Expectation of the number of steps $= \dfrac{\text{dimension}}{\text{Entropy}} \times \log N$

- Jacobi-Perron

  [Fischer-Schweiger'75]

- Brun

  [B.-Lhote-Vallée]

Mean behaviour of the number of steps
Euclid algorithm

Consider parameters $(u_1, \cdots, u_d)$ with $0 \le u_1, \cdots, u_d \le N$

Thm Expectation of the number of steps $= \dfrac{\text{dimension}}{\text{Entropy}} \times \log N$

- Formal power series with coefficients in a finite field and ploynomials with degree less than m

$$\frac{2}{2\frac{q}{q-1}} m = \frac{q-1}{q} m$$

[Knopfmacher-Knopfmacher'88, Friesen-Hensley'96, B.-Nakada-Natsui-Vallée'11]

Comparing Euclid and cf algorithms

- **Number of steps** and costs functions for algorithms defined on rational entries

  worst-case, mean behavior, average-case analysis

- Convergence properties
- Ergodic properties

  ergodic invariant measure, natural extension

- Arithmetic properties

  cubic numbers and periodic expansions,
  Diophantine approximation

And now...

- Comparison with Knuth algorithm
- Formal power series case
- Analysis in distribution
- How to understand algorithms based on lattice reduction in dynamical terms ? [LLL and sandpile models, LAREDA]