Multiple GCD's: Probabilistic analysis of the plain algorithm

Valérie Berthé, Jean Creusefond, Loïck Lhote, Brigitte Vallée

GREYC, UMR CNRS 6072,
ENSICAEN & Université de Caen Basse-Normandie

Projet ANR Dyna3S

Plan

Plan

Average analysis of algorithms

Elements for an average analysis

- An algorithm $\mathcal{A}$ whose inputs belong to the set Ω
- A cost function $X : \Omega \rightarrow \mathbb{R}^+$ that "describes" the algorithm (bit complexity, size of the output, memory/space complexity, ...)
- A size function : $\Omega = \bigcup_n \Omega_n$
- Each set Ω_n is endowed with a probability distribution $\Pr_n$ $(\mathrm{E}_n, \mathrm{V}_n, \sigma_n)$ (in this work, it is the uniform distribution)

Average analysis of algorithms

Main aims of an average analysis

- [mean value] Compute the asymptotic mean value of X : $\mathrm{E}_n[X] \underset{n\to\infty}{\sim} ?$
  ex : what is the average bit complexity of the algorithm when the input size n is large ? Is it linear in n ? Quadratic in n ?...
- [variance] Compute the asymptotic of the variance : $\mathrm{V}_n[X] \underset{n\to\infty}{\sim} ?$
  ex : is the probability to be far from the mean value asymptotically close to 0 ?
- [limit law] what is the limit law of X : $\dfrac{X - \mathrm{E}_n[X]}{\sigma_n(X)} \underset{n\to\infty}{\to} ?$
  ex : what is asymptotically the probability that X is in the interval $[a, b]$?

Average analysis of algorithms

Main aims of an average analysis

- [mean value] Compute the asymptotic mean value of X : $\mathrm{E}_n[X] \underset{n\to\infty}{\sim}$?
  ex : what is the average bit complexity of the algorithm when the input size n is large ? Is it linear in n ? Quadratic in n ?...
- [variance] Compute the asymptotic of the variance : $\mathrm{V}_n[X] \underset{n\to\infty}{\sim}$?
  ex : is the probability to be far from the mean value asymptotically close to 0 ?
- [limit law] what is the limit law of X : $\dfrac{X - \mathrm{E}_n[X]}{\sigma_n(X)} \underset{n\to\infty}{\to}$?
  ex : what is asymptotically the probability that X is in the interval $[a, b]$?

Average analysis of algorithms

Main aims of an average analysis

- [mean value] Compute the asymptotic mean value of X : $\mathrm{E}_n[X] \underset{n\to\infty}{\sim} ?$
  ex : what is the average bit complexity of the algorithm when the input size n is large ? Is it linear in n ? Quadratic in n ?...

- [variance] Compute the asymptotic of the variance : $\mathrm{V}_n[X] \underset{n\to\infty}{\sim} ?$
  ex : is the probability to be far from the mean value asymptotically close to 0 ?

- [limit law] what is the limit law of X : $\dfrac{X - \mathrm{E}_n[X]}{\sigma_n(X)} \underset{n\to\infty}{\to} ?$
  ex : what is asymptotically the probability that X is in the interval $[a, b]$?

Average analysis of algorithms

Main aims of an average analysis

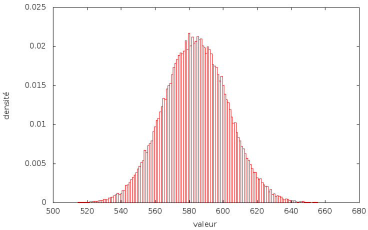
- [mean value] Compute the asymptotic mean value of X : $\mathrm{E}_n[X] \underset{n\to\infty}{\sim} ?$
  ex : what is the average bit complexity of the algorithm when the input size n is large ? Is it linear in n ? Quadratic in n ?...

- [variance] Compute the asymptotic of the variance : $\mathrm{V}_n[X] \underset{n\to\infty}{\sim} ?$
  ex : is the probability to be far from the mean value asymptotically close to 0 ?

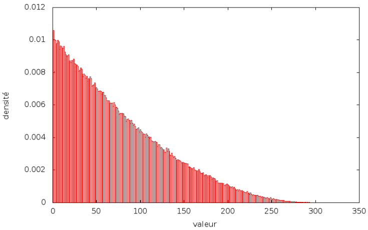
- [limit law] what is the limit law of X : $\dfrac{X - \mathrm{E}_n[X]}{\sigma_n(X)} \underset{n\to\infty}{\to} ?$
  ex : what is asymptotically the probability that X is in the interval $[a, b]$?

Examples of limit laws

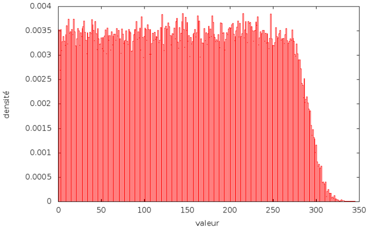
- x-axis : value of the studied parameter y-axis : probability density



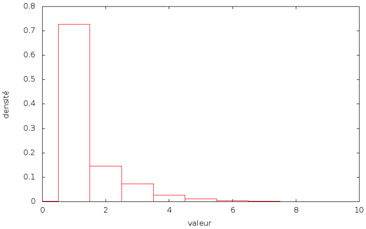
Gaussian law



Beta law



Uniform law



Geometric law

Previous analyses of euclidean algorithms

Euclid algorithm : number of euclidean divisions

- Lamé (1850) : *the worst case is linear w.r.t. the input binary size*
- Heilbron (69) and Dixon (70) : *the mean number of divisions is linear w.r.t. the input binary size*
- Hensley (1994) : *the number of divisions follows a gaussian limit law*
- A. Knopfmacher and J. Knopfmacher (86) : *(polynomial case) the number of divisions follows a binomial law over* Ω_n

Previous analyses of euclidean algorithms

Euclid algorithm : number of euclidean divisions

- Lamé (1850) : *the worst case is linear w.r.t. the input binary size*
- Heilbron (69) and Dixon (70) : *the mean number of divisions is linear w.r.t. the input binary size*
- Hensley (1994) : *the number of divisions follows a gaussian limit law*
- A. Knopfmacher and J. Knopfmacher (86) : *(polynomial case) the number of divisions follows a binomial law over* Ω_n

Average dynamical analyses

$$\boxed{\text{Dynamical Analysis} = \text{Analysis of algorithms} + \text{Dynamical systems}}$$

- the method was developped in Caen around Brigitte Vallée
- Dynamical Analysis applies to various euclidean algorithms : *Euclid, centered, by default, by excess,* α*-euclideans, binary, ...*
- the method applies to various parameters : *number of steps, binary complexity, costs of moderate growth, size of the continuants, ...*

Previous analyses of euclidean algorithms

Distributional dynamical analyses

- Viviane Baladi and Brigitte Vallée in 2002
- main result : costs of moderate growth (including the number of divisions) follow **gaussian limit laws**,
- the result applies to 3 algorithms : Euclid, centered, odd
- their result was adapted to other parameters : binary complexity (extended algorithm), size of the continuants, ...
- average analysis of a divide and conquer algorithm (Knuth-Schönhage)

And now ?

Open problems

- there remain algorithms to analyze : plus-minus (quadratic), Stehlé-Zimmerman (quasi-linear), . . .
- Perform the distributional analyses of the α-euclidean algorithms, the binary algorithm, . . .
- some parameters are not completely analyzed (binary complexity, . . .)

And now ?

Open problems

- there remain algorithms to analyze : plus-minus (quadratic), Stehlé-Zimmerman (quasi-linear), ...
- Perform the distributional analyses of the α-euclidean algorithms, the binary algorithm, ...
- some parameters are not completely analyzed (binary complexity, ...)

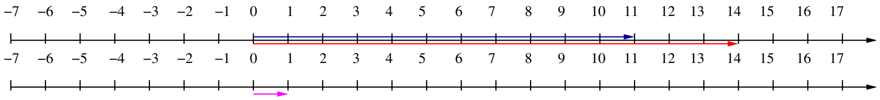
These are technical challenges
but
the main ideas are known

Generalizations of the Euclid algorithm

Lattice reduction

- Algorithms : LLL, HKZ, BKZ, . . .
- Models for the algorithms (sandpile, CFG, . . .)
- Models for the inputs (cryptography, factorization, . . .)

This afternoon. . .



Generalizations of the Euclid algorithm

Lattice reduction

- Algorithms : LLL, HKZ, BKZ, ...
- Models for the algorithms (sandpile, CFG, ...)
- Models for the inputs (cryptography, factorization, ...)

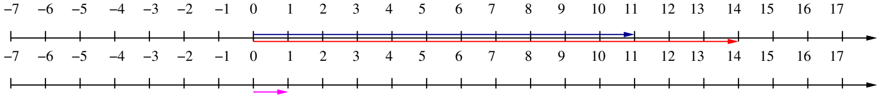
This afternoon...

GCD
Simultaneous Rational Approximation

Problem : Consider $\overrightarrow{y} \in \mathbb{R}^n$, find $q \in \mathbb{Z}$ with $q \leq M$ and $\overrightarrow{p} \in \mathbb{Z}^n$ such that $||q \cdot \overrightarrow{y} - \overrightarrow{p}||$ is small.

Continued Fraction Expansion

$$\frac{u}{v} = \cfrac{1}{m_1 + \cfrac{1}{m_2 + \cfrac{1}{\ddots \, \cfrac{}{m_{p-1} + \cfrac{1}{m_p + 0}}}}}$$



Plan

Strategies

Two input integers/polynomials

- one has to choose the division : euclidean, centered, odd, even, binary, . . .

Strategies

Two input integers/polynomials

- one has to choose the division : euclidean, centered, odd, even, binary, ...

3 or more input integers/polynomials

- One has to choose the division : euclidean, centered, odd, even, binary, ...
- what are the two elements used for each division ? The greatest ones ? The smallest ones ? Random choice ?
- After each division, what is the position of the remainder ?
- Is it better to sort the integers/polynomials ?
- Do we want to compute the gcd ? a good rational approximation ? perhaps both ?
- ...

Strategies

Two input integers/polynomials

- one has to choose the division : euclidean, centered, odd, even, binary, . . .

3 or more input integers/polynomials

- One has to choose the division : euclidean, centered, odd, even, binary, . . .
- what are the two elements used for each division ? The greatest ones ? The smallest ones ? Random choice ?
- After each division, what is the position of the remainder ?
- Is it better to sort the integers/polynomials ?
- Do we want to compute the gcd ? a good rational approximation ? perhaps both ?
- . . .

Strategies

Two input integers/polynomials

- one has to choose the division : euclidean, centered, odd, even, binary, . . .

3 or more input integers/polynomials

- One has to choose the division : euclidean, centered, odd, even, binary, . . .
- what are the two elements used for each division ? The greatest ones ? The smallest ones ? Random choice ?
- After each division, what is the position of the remainder ?
- Is it better to sort the integers/polynomials ?
- Do we want to compute the gcd ? a good rational approximation ? perhaps both ?
- . . .

Strategies

Two input integers/polynomials

- one has to choose the division : euclidean, centered, odd, even, binary, ...

3 or more input integers/polynomials

- One has to choose the division : euclidean, centered, odd, even, binary, ...
- what are the two elements used for each division ? The greatest ones ? The smallest ones ? Random choice ?
- After each division, what is the position of the remainder ?
- Is it better to sort the integers/polynomials ?
- Do we want to compute the gcd ? a good rational approximation ? perhaps both ?
- ...

Strategies

Two input integers/polynomials

- one has to choose the division : euclidean, centered, odd, even, binary, . . .

3 or more input integers/polynomials

- One has to choose the division : euclidean, centered, odd, even, binary, . . .
- what are the two elements used for each division ? The greatest ones ? The smallest ones ? Random choice ?
- After each division, what is the position of the remainder ?
- Is it better to sort the integers/polynomials ?
- Do we want to compute the gcd ? a good rational approximation ? perhaps both ?
- . . .

Strategies

Two input integers/polynomials

- one has to choose the division : euclidean, centered, odd, even, binary, ...

3 or more input integers/polynomials

- One has to choose the division : euclidean, centered, odd, even, binary, ...
- what are the two elements used for each division ? The greatest ones ? The smallest ones ? Random choice ?
- After each division, what is the position of the remainder ?
- Is it better to sort the integers/polynomials ?
- Do we want to compute the gcd ? a good rational approximation ? perhaps both ?
- ...

Examples of strategies

Plain algorithm

Sequentially call the Euclid algorithm

$$\mathrm{GCD}(x_1, \ldots, x_d) = \mathrm{GCD}(\mathrm{Euclid}(x_1, x_2), x_3, \ldots, x_d), \quad \mathrm{GCD}(x_1) = x_1.$$

Examples of strategies

Plain algorithm

Sequentially call the Euclid algorithm

$$\mathrm{GCD}(x_1, \ldots, x_d) = \mathrm{GCD}(\mathrm{Euclid}(x_1, x_2), x_3, \ldots, x_d), \quad \mathrm{GCD}(x_1) = x_1.$$

Brun algorithm

Suppose that $0 \leq x_1 \leq x_2 \leq \ldots \leq x_d$. Each step is of the form

$$(x_1, \ldots, x_d) \to \mathrm{ord}(x_d \mod x_{d-1}, [x_1, \ldots, x_{d-1}]).$$

Examples of strategies

Plain algorithm

Sequentially call the Euclid algorithm

$$\mathrm{GCD}(x_1, \ldots, x_d) = \mathrm{GCD}(\mathrm{Euclid}(x_1, x_2), x_3, \ldots, x_d), \quad \mathrm{GCD}(x_1) = x_1.$$

Brun algorithm

Suppose that $0 \leq x_1 \leq x_2 \leq \ldots \leq x_d$. Each step is of the form

$$(x_1, \ldots, x_d) \to \mathrm{ord}(x_d \mod x_{d-1}, [x_1, \ldots, x_{d-1}]).$$

Jacobi-Perron algorithm

Suppose that $0 \leq x_1, \ldots, x_{d-1} \leq x_d$. Each step is of the form

$$(x_1, \ldots, x_d) \to (x_2 \mod x_1, \ldots, x_d \mod x_1, x_1)$$

Examples of strategies

Plain algorithm

Sequentially call the Euclid algorithm

$$\mathrm{GCD}(x_1, \ldots, x_d) = \mathrm{GCD}(\mathrm{Euclid}(x_1, x_2), x_3, \ldots, x_d), \quad \mathrm{GCD}(x_1) = x_1.$$

Brun algorithm

Suppose that $0 \leq x_1 \leq x_2 \leq \ldots \leq x_d$. Each step is of the form

$$(x_1, \ldots, x_d) \to \mathrm{ord}(x_d \mod x_{d-1}, [x_1, \ldots, x_{d-1}]).$$

Jacobi-Perron algorithm

Suppose that $0 \leq x_1, \ldots, x_{d-1} \leq x_d$. Each step is of the form

$$(x_1, \ldots, x_d) \to (x_2 \mod x_1, \ldots, x_d \mod x_1, x_1)$$

and also...

Selmer, Poincaré, fully substractive, ...

Plain algorithm

Input: $(x_1, \ldots, x_d)$ d polynômes
Output: le PGCD de $(x_1, \ldots, x_d)$

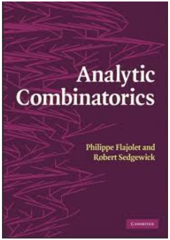
$u_1 = x_1$
for $i = 2$ **until** d **do**
  if $\deg u_{i-1} > \deg x_i$ **then**
    exchange u_{i-1} and x_i
  end if
  $u_i = \textbf{Euclid}(u_{i-1}, x_i)$
end for
return u_d

- I will detail the analysis for polynomial inputs
- Brigitte will detail the analysis for integer inputs
- method : Analytic Combinatorics ("simpler")
- philosophy : "*the analysis over the polynomials can be adapted to the integer case*"

Plan

Analytic Combinatorics

The bible :
Philippe Flajolet, Robert Sedgewick
Analytic Combinatorics,
Cambridge University Press, 2009



Main steps

algorithme modèle → séries génératrices —bijection / dictionnaire→ expr. altern. des séries —singularités→ asymptotiques des coefficients → espérance, variance loi limite

paramètre

"définies formellement"

les coefficients sont liés
à l'espérance, la variance,
la fonction caractéristique du
paramètre

règles du type
$\Omega \approx \mathcal{A} \oplus \mathcal{B} \Rightarrow \Omega(z) = A(z) + B(z)$

Algorithm, size, parameter and random model

- **Algorithm :** plain algorithm that computes the GCD of d polynomials in $\mathbb{F}_q[X]$ using successive calls to the Euclid Algorithm
- **Inputs :** $\Omega = \{(x_1, \ldots, x_d) \in \mathbb{F}_q[X]^d \mid \forall i = 1..d, x_i \neq 0, \ x_i \text{ monic } \}$
- **Studied parameter :** for $j = 1..d$,

  $X_j =$ number of euclidean divisions during the j-th call to the Euclid Algorithm

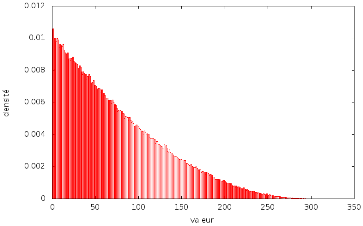
- **Size :** $||(x_1, \ldots, x_d)|| = \deg x_1 + \ldots + \deg x_d$
- **Inputs of size** n **:** $\Omega_n = \{(x_1, \ldots, x_d) \in \Omega \mid ||(x_1, \ldots, x_d)|| = n\}$
- **Probability distributions :** Pr_n is the uniform distribution over Ω_n

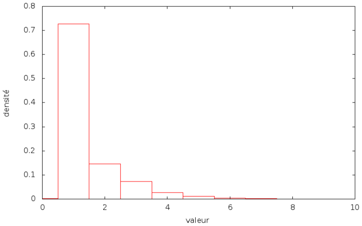
Algorithm, size, parameter and random model

- **Algorithm :** plain algorithm that computes the GCD of d polynomials in $\mathbb{F}_q[X]$ using successive calls to the Euclid Algorithm
- **Inputs :** $\Omega = \{(x_1, \ldots, x_d) \in \mathbb{F}_q[X]^d \mid \forall i = 1..d, x_i \neq 0,\ x_i \text{ monic } \}$
- **Studied parameter :** for $j = 1..d$,

  $X_j =$ number of euclidean divisions during the j-th call to the Euclid Algorithm

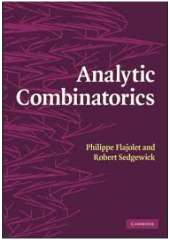
- **Size :** $||(x_1, \ldots, x_d)|| = \deg x_1 + \ldots + \deg x_d$
- **Inputs of size n :** $\Omega_n = \{(x_1, \ldots, x_d) \in \Omega \mid ||(x_1, \ldots, x_d)|| = n\}$
- **Probability distributions :** Pr_n is the uniform distribution over Ω_n





Analytic Combinatorics

The bible :
Philippe Flajolet, Robert Sedgewick
Analytic Combinatorics,
Cambridge University Press, 2009



Main steps

algorithme modèle → séries génératrices → (bijection / dictionnaire) → expr. altern. des séries → (singularités) → asymptotiques des coefficients → espérance, variance loi limite

paramètre

"définies formellement"

les coefficients sont liés
à l'espérance, la variance,
la fonction caractéristique du
paramètre

règles du type
$\Omega \approx \mathcal{A} \oplus \mathcal{B} \Rightarrow \Omega(z) = A(z) + B(z)$

“Formal” generating functions

Multivariate generating function

$$S(z_1, \ldots, z_d, u) = \sum_{(x_1,\ldots,x_d)\in\Omega} z_1^{\deg x_1} z_2^{\deg x_2} \ldots z_d^{\deg x_d} u^{X_j(x_1,\ldots,x_d)}$$

"Formal" generating functions

Multivariate generating function

$$S(z_1, \ldots, z_d, u) \quad = \sum_{(x_1,\ldots,x_d)\in\Omega} z_1^{\deg x_1} z_2^{\deg x_2} \ldots z_d^{\deg x_d} u^{X_j(x_1,\ldots,x_d)}$$

Bivariate generating function

If $z_1 = z_2 = \cdots = z_d = z$, we write

$$S(z, u) := S(z, \ldots, z, u) = \sum_{(x_1,\ldots,x_d)\in\Omega} z^{||(x_1,\ldots,x_d)||} u^{X_j(x_1,\ldots,x_d)}$$

“Formal” generating functions

Multivariate generating function

$$S(z_1, \ldots, z_d, u) \quad = \sum_{(x_1,\ldots,x_d)\in\Omega} z_1^{\deg x_1} z_2^{\deg x_2} \ldots z_d^{\deg x_d} u^{X_j(x_1,\ldots,x_d)}$$

Bivariate generating function

If $z_1 = z_2 = \cdots = z_d = z$, we write

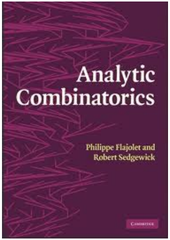
$$S(z, u) := S(z, \ldots, z, u) = \sum_{(x_1,\ldots,x_d)\in\Omega} z^{||(x_1,\ldots,x_d)||} u^{X_j(x_1,\ldots,x_d)}$$

$$\mathrm{Pr}_n[X_j = k] = \frac{[z^n u^k]S(z, u)}{[z^n]S(z, 1)}$$

Analytic Combinatorics

The bible :

Philippe Flajolet, Robert Sedgewick
Analytic Combinatorics,
Cambridge University Press, 2009



Main steps

algorithme modèle $\longrightarrow$ séries génératrices $\xrightarrow[\text{dictionnaire}]{\text{bijection}}$ expr. altern. des séries $\xrightarrow{\text{singularités}}$ asymptotiques des coefficients $\longrightarrow$ espérance, variance loi limite

paramètre

"définies formellement"

les coefficients sont liés
à l'espérance, la variance,
la fonction caractéristique du
paramètre

règles du type
$\Omega \approx \mathcal{A} \oplus \mathcal{B} \Rightarrow \Omega(z) = A(z) + B(z)$

Dictionary

Consider two combinatorial structures $(\mathcal{A}, |\cdot|_{\mathcal{A}})$ and $(\mathcal{B}, |\cdot|_{\mathcal{B}})$ and two parameters $X_{\mathcal{A}}$ and $X_{\mathcal{B}}$ over $\mathcal{A}$ and $\mathcal{B}$.

Relation	taille	paramètre	Série génératrice
$\mathcal{A} \oplus \mathcal{B}$	$\|x\| = \begin{cases} \|x\|_{\mathcal{A}} \text{ si } x \in \mathcal{A} \\ \|x\|_{\mathcal{B}} \text{ si } x \in \mathcal{B} \end{cases}$	$X(x) = \begin{cases} X_{\mathcal{A}}(x) \text{ si } x \in \mathcal{A} \\ X_{\mathcal{B}}(x) \text{ si } x \in \mathcal{B} \end{cases}$	$S(z,u) = A(z,u) + B(z,u)$
$\mathcal{A} \times \mathcal{B}$	$\|(x,y)\| = \|x\|_{\mathcal{A}} + \|y\|_{\mathcal{B}}$	$X(x,y) = X_{\mathcal{A}}(x) + X_{\mathcal{B}}(y)$	$S(z,u) = A(z,u) \times B(z,u)$
$\mathrm{Seq}(\mathcal{A})$	$\|(x_1,\dots,x_k)\| =$ $\|x_1\|_{\mathcal{A}} + \cdots + \|x_k\|_{\mathcal{A}}$	$X(x_1,\dots,x_k) =$ $X_{\mathcal{A}}(x_1) + \dots + X_{\mathcal{A}}(x_k)$	$S(z,u) = \dfrac{1}{1 - A(z,u)}$

Dictionary

Consider two combinatorial structures $(\mathcal{A}, |\cdot|_{\mathcal{A}})$ and $(\mathcal{B}, |\cdot|_{\mathcal{B}})$ and two parameters $X_{\mathcal{A}}$ and $X_{\mathcal{B}}$ over $\mathcal{A}$ and $\mathcal{B}$.

Relation	taille	paramètre	Série génératrice
$\mathcal{A} \oplus \mathcal{B}$	$\lvert x\rvert = \begin{cases} \lvert x\rvert_{\mathcal{A}} \text{ si } x \in \mathcal{A} \\ \lvert x\rvert_{\mathcal{B}} \text{ si } x \in \mathcal{B} \end{cases}$	$X(x) = \begin{cases} X_{\mathcal{A}}(x) \text{ si } x \in \mathcal{A} \\ X_{\mathcal{B}}(x) \text{ si } x \in \mathcal{B} \end{cases}$	$S(z,u) = A(z,u) + B(z,u)$
$\mathcal{A} \times \mathcal{B}$	$\lvert(x,y)\rvert = \lvert x\rvert_{\mathcal{A}} + \lvert y\rvert_{\mathcal{B}}$	$X(x,y) = X_{\mathcal{A}}(x) + X_{\mathcal{B}}(y)$	$S(z,u) = A(z,u) \times B(z,u)$
$\mathrm{Seq}(\mathcal{A})$	$\lvert(x_1,\ldots,x_k)\rvert = \lvert x_1\rvert_{\mathcal{A}} + \cdots + \lvert x_k\rvert_{\mathcal{A}}$	$X(x_1,\ldots,x_k) = X_{\mathcal{A}}(x_1) + \ldots + X_{\mathcal{A}}(x_k)$	$S(z,u) = \dfrac{1}{1 - A(z,u)}$

Two polynomials case (Euclid Algorithm) :

$$\Omega = \mathcal{U} \times \mathcal{U} \qquad \text{with} \quad \mathcal{U} = \{\text{monic polynomials}\}$$

Generating functions

$$U(z) = \sum_{x \in \mathcal{U}} z^{\deg x} = \frac{1}{1 - qz}, \qquad S(z_1, z_2, 1) = U(z_1)U(z_2)$$

Combinatorial bijection

Bijection given by the Euclid Algorithm

An input (x_1, x_2) is entirely determined by the sequence of quotients $(m_1, \ldots, m_p)$ and the gcd y the Euclid Algorithm computes.

$$\begin{array}{ccccccc} \Omega & \approx & \{\text{first quotient}\} & \times & \{\text{sequence of quotients}\} & \times & \{\text{GCD}\} \\ (x_1, x_2) & \approx & m_1 & \times & (m_2, \ldots, m_p) & \times & y \end{array}$$

Constraints

- the first quotient is monic (x_1 and x_2 are monic)
- for $i \geq 2$, $\deg m_i \geq 1$ and m_i is not necessarily monic

$$\mathcal{G} = \{x \in \mathbb{F}_q[X] | \deg x \geq 1\}, \qquad G(z) = \sum_{x \in \mathcal{G}} z^{\deg x} = \frac{(q-1)qz}{1-qz}$$

- the degree of the first quotient has a particular role

Combinatorial bijection

Bijection given by the Euclid Algorithm

An input (x_1, x_2) is entirely determined by the sequence of quotients $(m_1, \ldots, m_p)$ and the gcd y the Euclid Algorithm computes.

$$\begin{array}{ccccccc} \Omega & \approx & \{\text{first quotient}\} & \times & \{\text{sequence of quotients}\} & \times & \{\text{GCD}\} \\ \Omega & \approx & \mathcal{U} & \times & & \times & \\ (x_1, x_2) & \approx & m_1 & \times & (m_2, \ldots, m_p) & \times & y \end{array}$$

Constraints

- the first quotient is monic (x_1 and x_2 are monic)
- for $i \geq 2$, $\deg m_i \geq 1$ and m_i is not necessarily monic

$$\mathcal{G} = \{x \in \mathbb{F}_q[X] \,|\, \deg x \geq 1\}, \qquad G(z) = \sum_{x \in \mathcal{G}} z^{\deg x} = \frac{(q-1)qz}{1-qz}$$

- the degree of the first quotient has a particular role

Combinatorial bijection

Bijection given by the Euclid Algorithm

An input (x_1, x_2) is entirely determined by the sequence of quotients $(m_1, \ldots, m_p)$ and the gcd y the Euclid Algorithm computes.

$$
\begin{array}{ccccccc}
\Omega & \approx & \{\text{first quotient}\} & \times & \{\text{sequence of quotients}\} & \times & \{\text{GCD}\} \\
\Omega & \approx & \mathcal{U} & \times & \mathrm{Seq}(\mathcal{G}) & \times & \\
(x_1, x_2) & \approx & m_1 & \times & (m_2, \ldots, m_p) & \times & y
\end{array}
$$

Constraints

- the first quotient is monic (x_1 and x_2 are monic)
- for $i \geq 2$, $\deg m_i \geq 1$ and m_i is not necessarily monic

$$\mathcal{G} = \{x \in \mathbb{F}_q[X] \mid \deg x \geq 1\}, \qquad G(z) = \sum_{x \in \mathcal{G}} z^{\deg x} = \frac{(q-1)qz}{1-qz}$$

- the degree of the first quotient has a particular role

Combinatorial bijection

Bijection given by the Euclid Algorithm

An input (x_1, x_2) is entirely determined by the sequence of quotients $(m_1, \ldots, m_p)$ and the gcd y the Euclid Algorithm computes.

$$\begin{array}{ccccccc}
\Omega & \approx & \{\text{first quotient}\} & \times & \{\text{sequence of quotients}\} & \times & \{\text{GCD}\} \\
\Omega & \approx & \mathcal{U} & \times & \mathrm{Seq}(\mathcal{G}) & \times & \mathcal{U} \\
(x_1, x_2) & \approx & m_1 & \times & (m_2, \ldots, m_p) & \times & y
\end{array}$$

Constraints

- the first quotient is monic (x_1 and x_2 are monic)
- for $i \geq 2$, $\deg m_i \geq 1$ and m_i is not necessarily monic

$$\mathcal{G} = \{x \in \mathbb{F}_q[X] | \deg x \geq 1\}, \qquad G(z) = \sum_{x \in \mathcal{G}} z^{\deg x} = \frac{(q-1)qz}{1-qz}$$

- the degree of the first quotient has a particular role

Combinatorial bijection

Bijection given by the Euclid Algorithm

An input (x_1, x_2) is entirely determined by the sequence of quotients $(m_1, \ldots, m_p)$ and the gcd y the Euclid Algorithm computes.

$$\begin{array}{ccccccc}
\Omega & \approx & \{\text{first quotient}\} & \times & \{\text{sequence of quotients}\} & \times & \{\text{GCD}\} \\
\Omega & \approx & \mathcal{U} & \times & \mathrm{Seq}(\mathcal{G}) & \times & \mathcal{U} \\
(x_1, x_2) & \approx & m_1 & \times & (m_2, \ldots, m_p) & \times & y
\end{array}$$

Constraints

- the first quotient is monic (x_1 and x_2 are monic)
- for $i \geq 2$, $\deg m_i \geq 1$ and m_i is not necessarily monic

$$\mathcal{G} = \{x \in \mathbb{F}_q[X] | \deg x \geq 1\}, \qquad G(z) = \sum_{x \in \mathcal{G}} z^{\deg x} = \frac{(q-1)qz}{1-qz}$$

- the degree of the first quotient has a particular role

Combinatorial bijection and generating functions

Case $\deg x_1 \geq \deg x_2$:

$$\begin{aligned} \deg x_1 &= \deg m_1 + \deg m_2 + \ldots + \deg m_p + \deg y \\ \deg x_2 &= \deg m_2 + \ldots + \deg m_p + \deg y \end{aligned}$$

$$z_1^{\deg x_1} z_2^{\deg x_2} = z_1^{\deg m_1} (z_1 z_2)^{\deg m_2 + \ldots + \deg m_p} (z_1 z_2)^{\deg y}$$

Case $\deg x_1 \leq \deg x_2$:

$$\begin{aligned} \deg x_1 &= \deg m_2 + \ldots + \deg m_p + \deg y \\ \deg x_2 &= \deg m_1 + \deg m_2 + \ldots + \deg m_p + \deg y \end{aligned}$$

$$z_1^{\deg x_1} z_2^{\deg x_2} = z_2^{\deg m_1} (z_1 z_2)^{\deg m_2 + \ldots + \deg m_p} (z_1 z_2)^{\deg y}$$

Combinatorial bijection and generating functions

Bijection given by the Euclid Algorithm

An input (x_1, x_2) is entirely determined by the sequence of quotients $(m_1, \ldots, m_p)$ and the gcd y the Euclid Algorithm computes.

$$\begin{array}{cccccccc} \Omega & \approx & \{\text{first quotient}\} & \times & \{\text{sequence of quotients}\} & \times & \{\text{GCD}\} \\ \Omega & \approx & \mathcal{U} & \times & \mathrm{Seq}(\mathcal{G}) & \times & \mathcal{U} \\ (x_1, x_2) & \approx & m_1 & \times & (m_2, \ldots, m_p) & \times & y \end{array}$$

$$S(z_1, z_2, 1) = U(z_1)U(z_2) = \frac{T(z_1, z_2)}{1 - G(z_1 z_2)} \cdot U(z_1 z_2)$$

with

$$T(z_1, z_2) = U(z_1) + U(z_2) - 1$$

Combinatorial bijection and generating functions

And we apply this idea recursively...

Proposition

$$S(z_1, \ldots, z_d, 1) = U(z_1) \cdot \ldots \cdot U(z_d) = U(t_d) \prod_{k=1}^{d-1} \frac{T(z_{k+1}, t_k)}{1 - G(z_{k+1} t_k)}$$

with $t_k = z_1 \cdots z_k$.

Combinatorial bijection and generating functions

And we apply this idea recursively...

Proposition

$$S(z_1, \ldots, z_d, 1) = U(z_1) \cdot \ldots \cdot U(z_d) = U(t_d) \prod_{k=1}^{d-1} \frac{T(z_{k+1}, t_k)}{1 - G(z_{k+1} t_k)}$$

with $t_k = z_1 \cdots z_k$.

And now, with $z = z_1 = \ldots = z_d$

$$S(z, 1) = U(z)^d = U(z^d) \prod_{k=1}^{d-1} \frac{T(z, z^k)}{1 - G(z^{k+1})}$$

Moment generating functions of X_j

$$S(z,1) = U(z)^d = U(z^d) \prod_{k=1}^{d-1} \frac{T(z,z^k)}{1 - G(z^{k+1})}$$

We mark each step (division) of the j-th call to the Euclid Algorithm by the variable u

$$S(z,u) \quad = \quad U(z^d) \left[\prod_{k=1,\ k\neq j}^{d-1} \frac{T(z,z^k)}{1 - G(z^{k+1})} \right] \cdot \frac{uT(z,z^j)}{1 - uG(z^{j+1})}$$

Moment generating functions of X_j

$$S(z,1) = U(z)^d = U(z^d) \prod_{k=1}^{d-1} \frac{T(z,z^k)}{1-G(z^{k+1})}$$

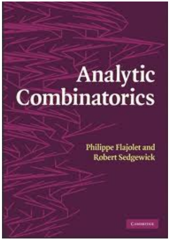
We mark each step (division) of the j-th call to the Euclid Algorithm by the variable u

$$\begin{aligned} S(z,u) &= U(z^d) \left[\prod_{k=1,\ k\neq j}^{d-1} \frac{T(z,z^k)}{1-G(z^{k+1})} \right] \cdot \frac{uT(z,z^j)}{1-uG(z^{j+1})} \\ &= U(z)^d \cdot u \cdot \frac{1-G(z^{j+1})}{1-uG(z^{j+1})} \end{aligned}$$

Analytic Combinatorics

The bible :

Philippe Flajolet, Robert Sedgewick
Analytic Combinatorics,
Cambridge University Press, 2009



Main steps

algorithme modèle $\longrightarrow$ séries génératrices $\xrightarrow{\text{bijection}}$ expr. altern. des séries $\xrightarrow{\text{singularités}}$ asymptotiques des coefficients $\longrightarrow$ espérance, variance loi limite

paramètre

dictionnaire

"définies formellement"

les coefficients sont liés
à l'espérance, la variance,
la fonction caractéristique du
paramètre

règles du type
$\Omega \approx \mathcal{A} \oplus \mathcal{B} \Rightarrow \Omega(z) = A(z) + B(z)$

Extraction w.r.t u

Bivariate generating function :

$$S(z,u) = U(z)^d \cdot 1 - G(z^{j+1}) \cdot \frac{u}{1 - uG(z^{j+1})}$$

Extraction w.r.t u

Bivariate generating function :

$$S(z,u) = U(z)^d \;\cdot\; 1 - G(z^{j+1}) \;\cdot\; \frac{u}{1 - uG(z^{j+1})}$$

Extraction w.r.t u : since $[u^k]\frac{u}{1-au} = a^{k-1}$

$$[u^k]S(z,u) = \frac{1}{(1-qz)^d} \cdot (1 - G(z^{j+1}))G(z^{j+1})^{k-1}$$

Extraction w.r.t u

Bivariate generating function :

$$S(z,u) = U(z)^d \cdot 1 - G(z^{j+1}) \cdot \frac{u}{1 - uG(z^{j+1})}$$

Extraction w.r.t u : since $[u^k]\frac{u}{1-au} = a^{k-1}$

$$[u^k]S(z,u) = \frac{1}{(1-qz)^d} \cdot (1 - G(z^{j+1}))G(z^{j+1})^{k-1}$$

Simplification (without loss of generality) :

$$\sum_{i>k}[u^i]S(z,u) = [u^{>k}]S(z,u) = \frac{1}{(1-qz)^d} \cdot G(z^{j+1})^k.$$

Extraction w.r.t u

Bivariate generating function :

$$S(z,u) = U(z)^d \;\cdot\; 1 - G(z^{j+1}) \;\cdot\; \frac{u}{1 - uG(z^{j+1})}$$

Extraction w.r.t u : since $[u^k]\frac{u}{1-au} = a^{k-1}$

$$[u^k]S(z,u) = \frac{1}{(1-qz)^d} \cdot (1 - G(z^{j+1}))G(z^{j+1})^{k-1}$$

Simplification (without loss of generality) :

$$\sum_{i>k}[u^i]S(z,u) = [u^{>k}]S(z,u) = \frac{1}{(1-qz)^d} \cdot G(z^{j+1})^k.$$

Recall : $\mathbb{P}_n[X_j > k] = \frac{1}{\mathrm{card}(\Omega_n)}[z^n][u^{>k}]S(z,u)$

Generating function :

$$[u^{>k}]S_j(z,u) = \frac{1}{(1-qz)^d} \cdot G(z^{j+1})^k.$$

- the GF admits a pole of order d in $z = 1/q$
- $G(z^{j+1})$ is analytic in a ball whose radius is $> 1/q$

and "clearly", the asymptotic will depend on :

- the power k (large or small power ?)
- whether $G(1/q^{j+1}) = 1$ or $G(1/q^{j+1}) < 1$ or $\cancel{G(1/q^{j+1}) > 1}$.

Semi-classical theorem

The following result is a mixing between “large power theorems” and “meromorphic theorems”

Theorem

Consider the function

$$F^{[k]}(z) = \frac{1}{(1-z)^d} \cdot G(z)^k,$$

where :

- $G(z)$ is analytic on the disk $|z| \leq \rho$ with $\rho > 1$,
- $a := G(1) \neq 0, \quad b := G'(1) > 0,$
- for $|z|$ close enough to 1, $|G(z)| \leq G(|z|)$.

Then, when $k/n \to c$ for some $c \in [0, a/b[$ then

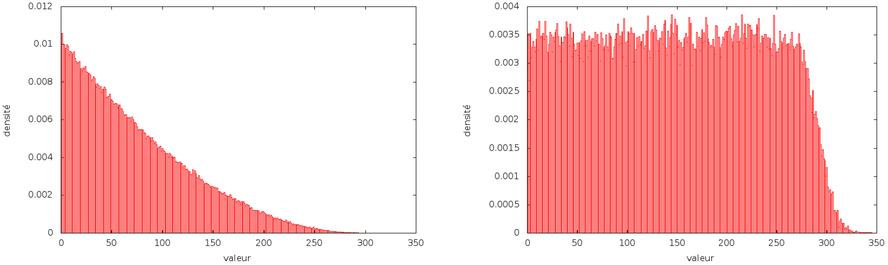
$$[z^n]F^{[k]}(z) = \frac{n^{d-1}}{(d-1)!} a^k \left(1 - \frac{b}{a}c\right)^{d-1} \left[1 + O\left(\frac{1}{n}\right)\right].$$

First call ($j = 1$)

Result 1 :

The number of euclidean divisions performed by the plain algorithm during the first call to the Euclid Algorithm follows an asymptotic beta law with parameters $(1, d-1)$. Precisely, for all $x \in [0, 1[$,

$$\Pr_n \left[X_1 > \frac{q-1}{2q} \cdot n \cdot x \right] \sim \frac{1}{K(d)} \int_x^1 (1-t)^{d-2} dt.$$



Other calls (case $j \geq 2$)

Result 2 :

For $j \geq 2$, the number of euclidean divisions performed by the plain algorithm during the j-th call to the Euclid Algorithm follows an asymptotic geometric law with parameter $p_j = G(1/q^{j+1}) = \dfrac{q-1}{q^j - 1}$. Precisely,

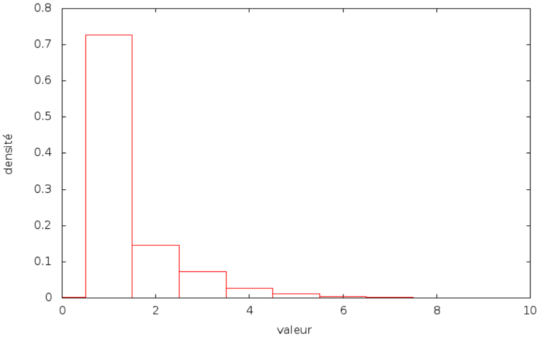
$$\Pr_n[X_j = k] \sim (1 - p_j)p_j^{k-1}, \qquad \text{when } k = o(n).$$

Other calls (case $j \geq 2$)

Result 2 :

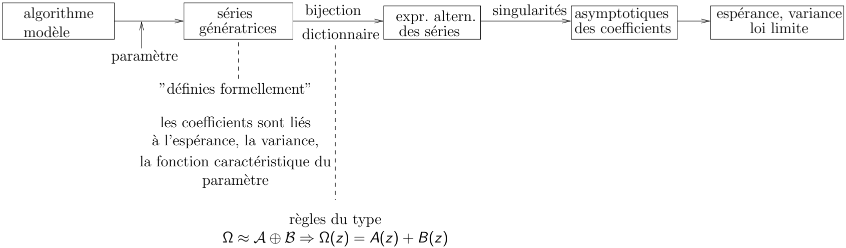
For $j \geq 2$, the number of euclidean divisions performed by the plain algorithm during the j-th call to the Euclid Algorithm follows an asymptotic geometric law with parameter $p_j = G(1/q^{j+1}) = \dfrac{q-1}{q^j - 1}$. Precisely,

$$\Pr_n[X_j = k] \sim (1 - p_j) p_j^{k-1}, \qquad \text{when } k = o(n).$$

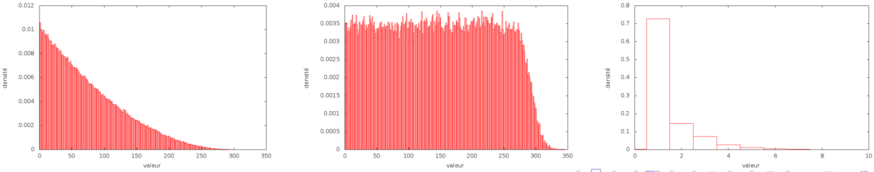


Conclusion for the polynomial case

- Methodology : classical and less classical tools in Analytic Combinatorics



- Main results :
  - as far as we know, it is the first distributional analysis of the plain algorithm
  - some results are *unexpected* (uniform distribution versus gaussian law in dimension 2)



Plan

1. Introduction
2. GCD algorithms
3. Analysis for polynomial inputs
4. Analysis for integer inputs

General approach

Brigitte . . .