Lacunaryx: *Computing bounded-degree factors of lacunary polynomials*

Bruno Grenet
LIRMM — CNRS
Université de Montpellier

ISSAC — July 7., 2015

Factorization of a polynomial f

Find $f_1, \dots, f_t$, irreducible, s.t. $f = f_1 \times \cdots \times f_t$.

Factorization of a polynomial f

Find $f_1, \dots, f_t$, irreducible, s.t. $f = f_1 \times \cdots \times f_t$.

- Many algorithms
  - over $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{Q}(\alpha)$, $\overline{\mathbb{Q}}$, $\mathbb{Q}_p$, $\mathbb{F}_q$, $\mathbb{R}$, $\mathbb{C}$, $\dots$;
  - in 1, 2, $\dots$, n variables.
- Complexity: **polynomial in** $\deg(f)$

Factorization of a polynomial f

Find $f_1, \dots, f_t$, irreducible, s.t. $f = f_1 \times \cdots \times f_t$.

- Many algorithms
  - over $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{Q}(\alpha)$, $\overline{\mathbb{Q}}$, $\mathbb{Q}_p$, $\mathbb{F}_q$, $\mathbb{R}$, $\mathbb{C}$, …;
  - in 1, 2, …, n variables.
- Complexity: **polynomial in** $\deg(f)$

$$X^{568742}Y^{568741} + X^{568741}Y^{568742} - X^{568741}Y^{568741} - X - Y + 1$$
$$= (X + Y - 1) \times (X^{568741}Y^{568741} - 1)$$

Factorization of a polynomial f

Find $f_1, \dots, f_t$, irreducible, s.t. $f = f_1 \times \cdots \times f_t$.

- Many algorithms
  - over $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{Q}(\alpha)$, $\overline{\mathbb{Q}}$, $\mathbb{Q}_p$, $\mathbb{F}_q$, $\mathbb{R}$, $\mathbb{C}$, $\dots$;
  - in $1, 2, \dots, n$ variables.
- Complexity: **polynomial in** $\deg(f)$

$$X^{568742}Y^{568741} + X^{568741}Y^{568742} - X^{568741}Y^{568741} - X - Y + 1$$
$$= (X + Y - 1) \times (X^{568741}Y^{568741} - 1)$$
$$= (X + Y - 1) \times (XY - 1) \times (1 + XY + \cdots + X^{568740}Y^{568740})$$

Definition

$$f(X_1, \dots, X_n) = \sum_{j=1}^{k} c_j X_1^{\alpha_{1j}} \cdots X_n^{\alpha_{nj}}$$

- $\mathrm{size}(f) \simeq k\left(\max_j(\mathrm{size}(c_j)) + n\log(\deg f)\right)$

Definition

$$f(X_1,\ldots,X_n) = \sum_{j=1}^{k} c_j X_1^{\alpha_{1j}} \cdots X_n^{\alpha_{nj}}$$

- $\text{size}(f) \simeq k\Big(\max_j(\text{size}(c_j)) + n\log(\deg f)\Big)$

Theorems

There exist deterministic polynomial-time algorithms computing

- **integer roots** of $f \in \mathbb{Z}[X]$; [Cucker-Koiran-Smale'98]
- **low-degree** factors of $f \in \mathbb{Q}(\alpha)[X]$; [H. Lenstra'99]
- **low-degree** factors of $f \in \mathbb{Q}(\alpha)[X_1,\ldots,X_n]$. [Kaltofen-Koiran'05-06]

[Chattopadhyay-G.-Koiran-Portier-Strozecki'13, G.'14]

Definition

$$f(X_1,\dots,X_n)=\sum_{j=1}^{k} c_j X_1^{\alpha_{1j}}\cdots X_n^{\alpha_{nj}}$$

- $\mathrm{size}(f)\simeq k\Big(\max_j(\mathrm{size}(c_j))+n\log(\deg f)\Big)$

Theorems

There exist deterministic polynomial-time algorithms computing

- **integer roots** of $f\in\mathbb{Z}[X]$; [Cucker-Koiran-Smale'98]
- **low-degree** factors of $f\in\mathbb{Q}(\alpha)[X]$; [H. Lenstra'99]
- **low-degree** factors of $f\in\mathbb{Q}(\alpha)[X_1,\dots,X_n]$. [Kaltofen-Koiran'05-06]

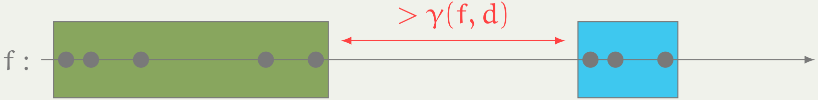
[Chattopadhyay-G.-Koiran-Portier-Strozecki'13, G.'14]

Lenstra's algorithm

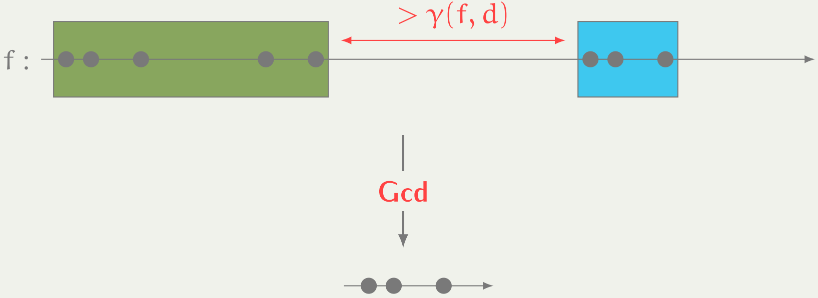
Input: f (list of monomial) and d
Output: Degree-d *non-cyclotomic* factors of f

Input: f (list of monomial) and d

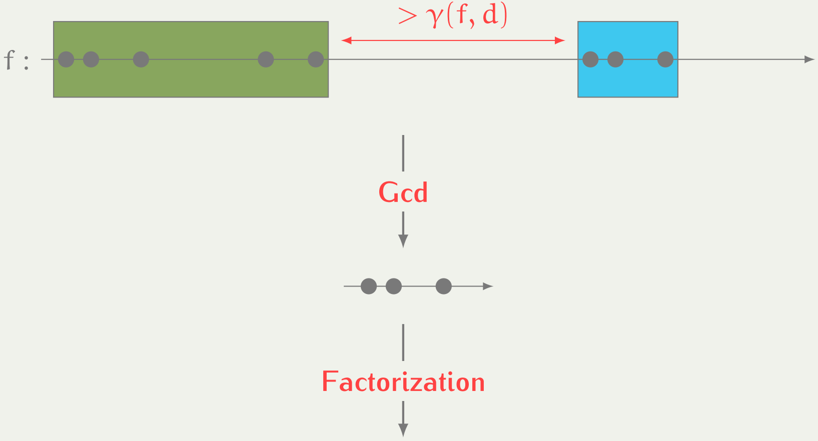
Output: Degree-d *non-cyclotomic* factors of f



Input: f (list of monomial) and d
Output: Degree-d *non-cyclotomic* factors of f



Input: f (list of monomial) and d
Output: Degree-d *non-cyclotomic* factors of f



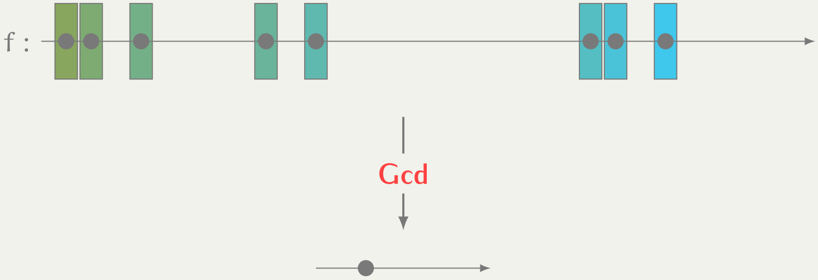
Degree-d factors of f

Modified Lenstra's algorithm

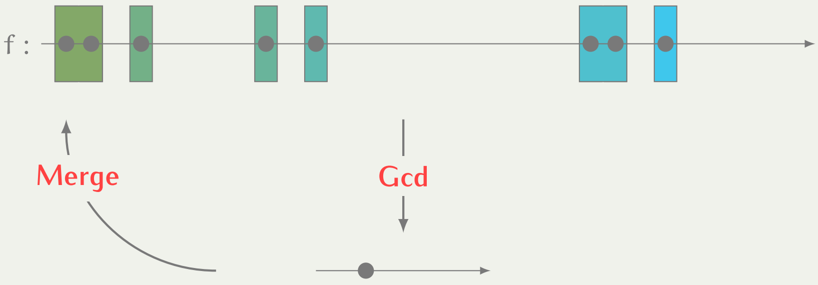
f :

Modified Lenstra's algorithm

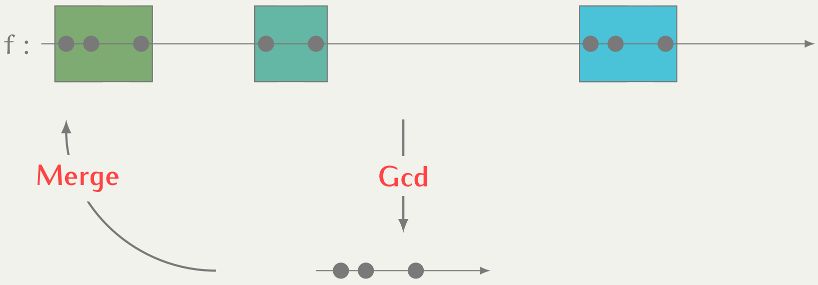
Modified Lenstra's algorithm



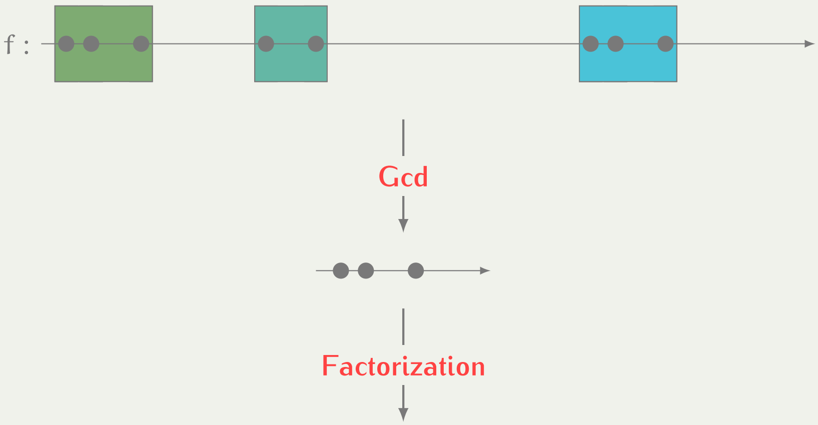
Modified Lenstra's algorithm



Modified Lenstra's algorithm

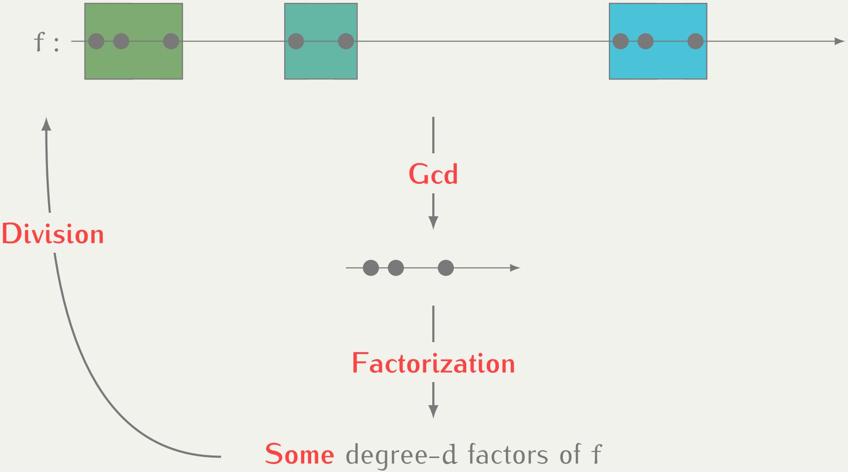


Modified Lenstra's algorithm



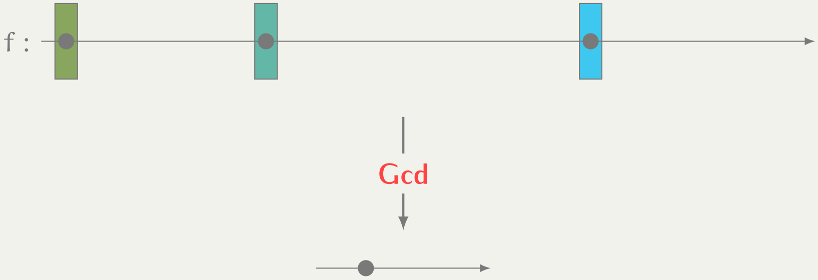
Some degree-d factors of f

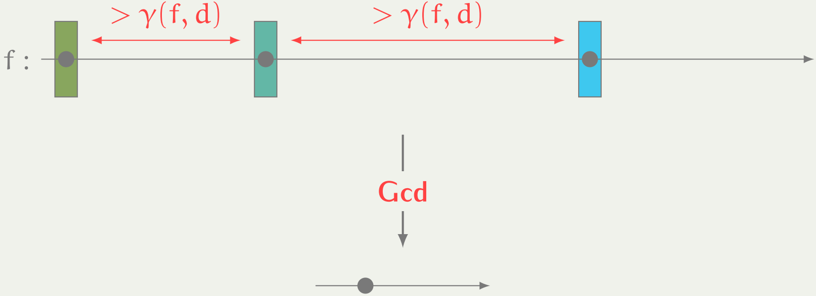
Modified Lenstra's algorithm



Modified Lenstra's algorithm

Modified Lenstra's algorithm




$> \gamma(f, d)$
$> \gamma(f, d)$
f :
Gcd

- Cyclotomic factors: test if each ϕ_r of degree $\leqslant d$ divides f:

$$\phi_r \text{ divides } f \iff \phi_r \text{ divides } f^{\bmod r} = \sum_j c_j X^{\alpha_j \bmod r}$$

- Multivariate polynomials: Same modifications

Implementation

- Cyclotomic factors: test if each ϕ_r of degree $\leqslant d$ divides f:

$$\phi_r \text{ divides } f \iff \phi_r \text{ divides } f^{\bmod r} = \sum_j c_j X^{\alpha_j \bmod r}$$

- Multivariate polynomials: Same modifications

- Implementation in MATHEMAGIX as a package named LACUNARYX
  - `http://mathemagix.org > Packages > Lacunaryx`
  - Polynomials over $\mathbb{Z}$ and $\mathbb{Q}$
  - Partially glued inside MATHEMAGIX

$$f = l \times c \times s$$

- l: product of low-degree polynomials
- c: product of $X^r - 1$
- s: *perturbated* sparse polynomial: $s = \sum_{j=1}^{n} X^{\alpha_j} p_j(X)$

Example

$$f = l \times c \times s$$

- l: product of low-degree polynomials

  5 polynomials, degree $\leqslant 10$, coefficients $\leqslant 50$

- c: product of $X^r - 1$

  3 polynomials, $r \leqslant 100\,000$

- s: *perturbated* sparse polynomial: $s = \sum_{j=1}^{n} X^{\alpha_j} p_j(X)$

  $n = 40$, $\alpha_j \leqslant 1\,000\,000$, $\deg(p_j) \leqslant 20$, coefficients $\leqslant 50$

$\rightsquigarrow$ degree $\geqslant 1\,000\,000$, $\geqslant 10\,000$ terms, coefficients $\geqslant 5 \times 10^9$

Conclusion

- First implementation for testing purposes
- Comparison with other softwares irrelevant

Conclusion

- First implementation for testing purposes
- Comparison with other softwares irrelevant
- Very versatile
  - *A priori* estimations
  - Other strategy: bound the *size* of the computation rather than the size of the factors

Conclusion

- First implementation for testing purposes
- Comparison with other softwares irrelevant
- Very versatile
  - *A priori* estimations
  - Other strategy: bound the *size* of the computation rather than the size of the factors
- Techniques useful for *classical* factorization?

Conclusion

- First implementation for testing purposes
- Comparison with other softwares irrelevant
- Very versatile
  - *A priori* estimations
  - Other strategy: bound the *size* of the computation rather than the size of the factors
- Techniques useful for *classical* factorization?

Thank you!