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

Bruno Grenet
LIRMM
Université de Montpellier

Mathemagix Days — September 21., 2015

Factorization of a polynomial f

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

Factorization of a polynomial f

Find $f_1, \ldots, f_t$, 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** $\deg(f)$

Factorization of a polynomial f

Find $f_1, \ldots, f_t$, 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 $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{Q}(\alpha)$, $\overline{\mathbb{Q}}$, $\mathbb{Q}_p$, $\mathbb{F}_q$, $\mathbb{R}$, $\mathbb{C}$, …;
  - in 1, 2, …, n variables.
- Complexity: **polynomial in** $\deg(f)$

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

Definition

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

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

Definition

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

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

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 $f \in \mathbb{Q}(\alpha)[X_1, \dots, X_n]$. **[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

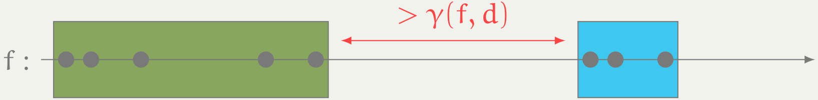
There exist deterministic polynomial-time algorithms computing

- **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]

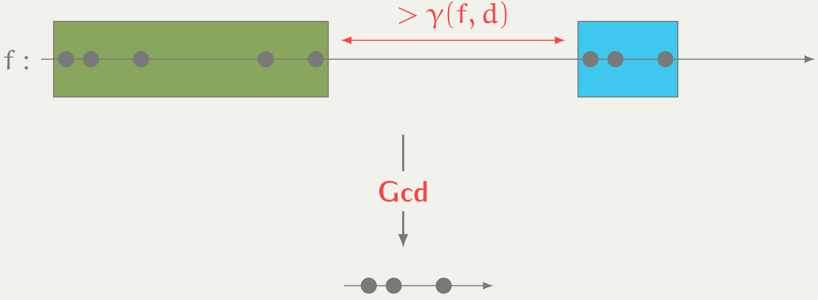
[Chattopadhyay-G.-Koiran-Portier-Strozecki'13, G.'14]

Lenstra's algorithm (non-cyclotomic factors)

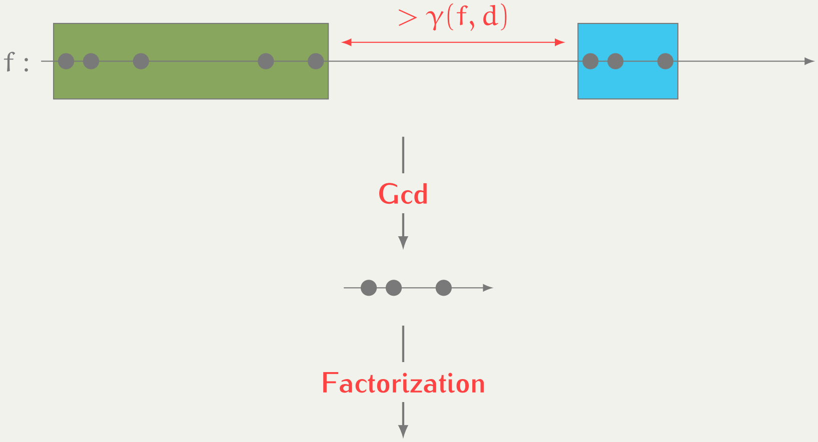
Lenstra's algorithm (non-cyclotomic factors)



Lenstra's algorithm (non-cyclotomic factors)



Lenstra's algorithm (non-cyclotomic factors)



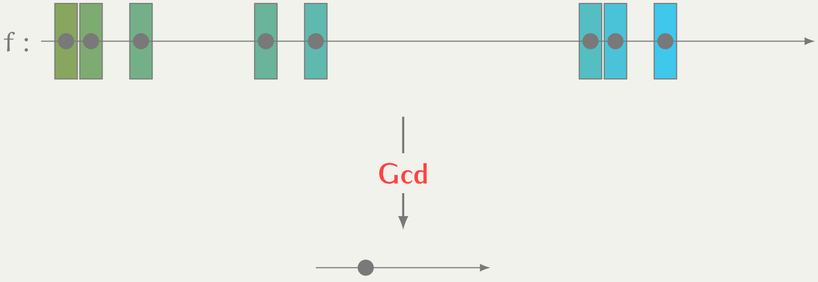
Degree-d non-cyclotomic factors of f

Implemented algorithm (non-cyclotomic factors)

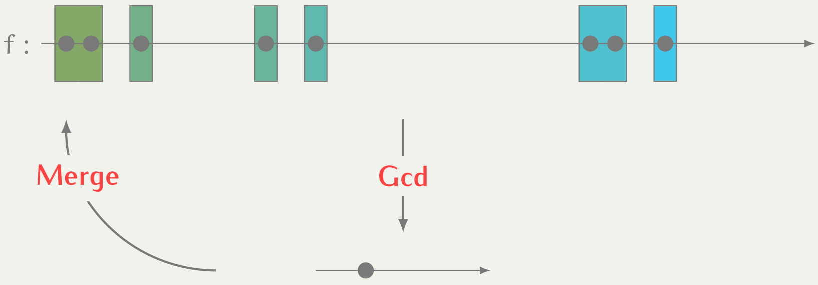
f :

Implemented algorithm (non-cyclotomic factors)

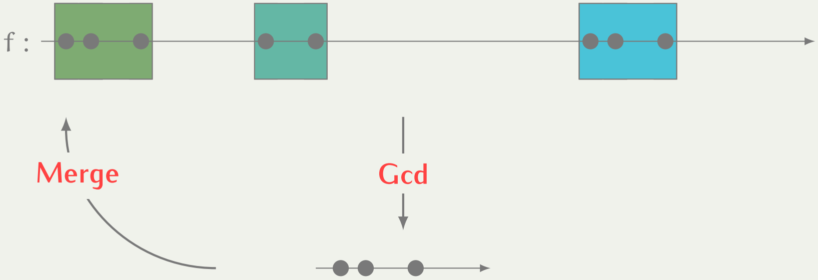
Implemented algorithm (non-cyclotomic factors)



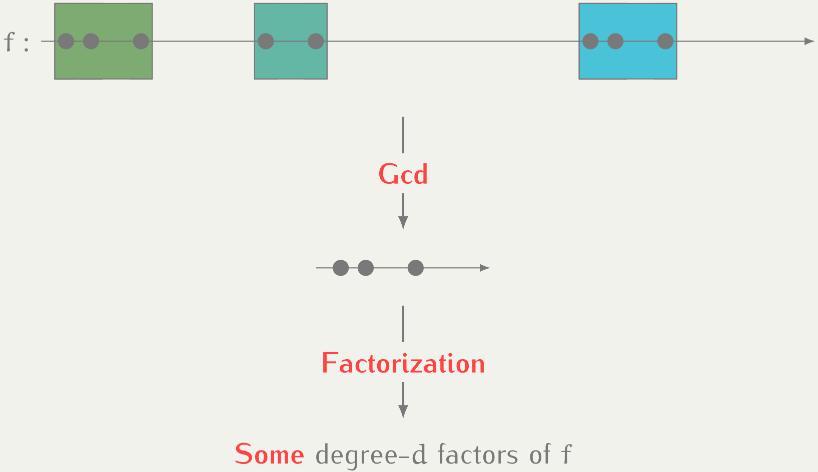
Implemented algorithm (non-cyclotomic factors)



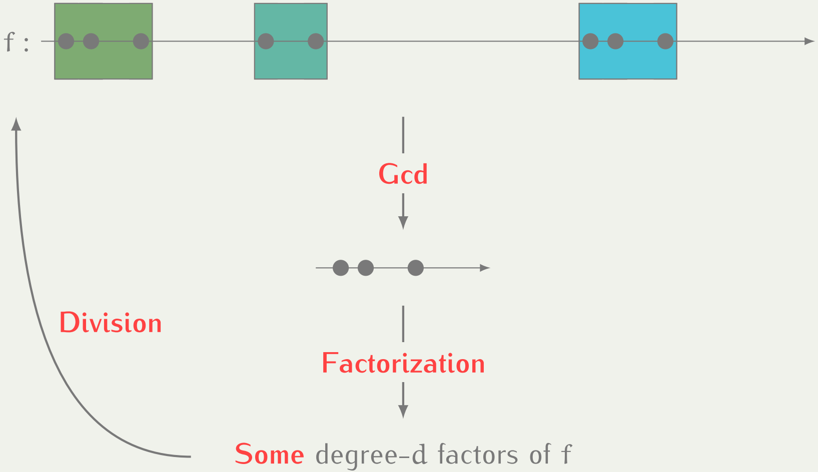
Implemented algorithm (non-cyclotomic factors)



Implemented algorithm (non-cyclotomic factors)

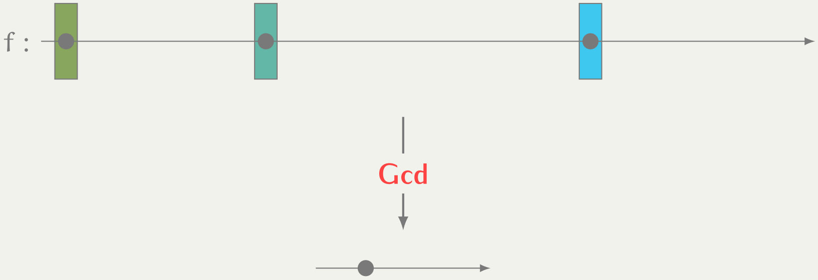


Implemented algorithm (non-cyclotomic factors)

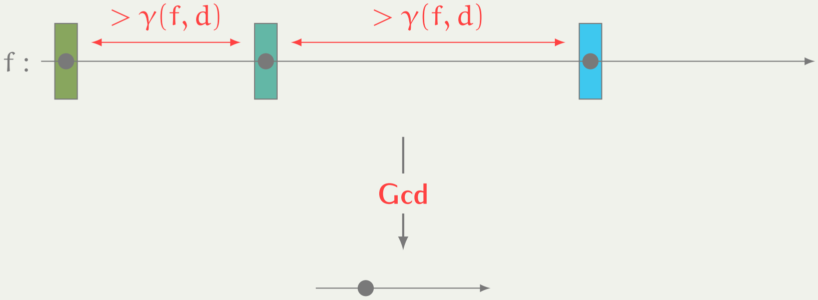


Implemented algorithm (non-cyclotomic factors)

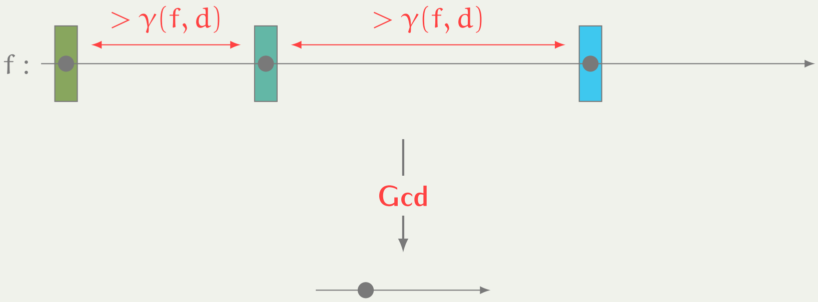
Implemented algorithm (non-cyclotomic factors)



Implemented algorithm (non-cyclotomic factors)



Implemented algorithm (non-cyclotomic factors)



Partial factorization:

$$f = \underbrace{\prod_{g \in G} g}_{\text{low-degree factors}} \times \underbrace{\prod_{h \in H} h}_{\text{sparse polynomials}}$$

- Lenstra's algorithm:
  - ϕ_r divides $f \iff \phi_r$ divides $\gcd(f, X^r - 1)$
  - $\gcd(f, X^r - 1) = \gcd(f^{\bmod r}, X^r - 1)$ $\quad (f^{\bmod r} = \sum_j c_j X^{\alpha_j \bmod r})$

- Lenstra's algorithm:
  - ϕ_r divides $f \iff \phi_r$ divides $\gcd(f, X^r - 1)$
  - $\gcd(f, X^r - 1) = \gcd(f^{\bmod r}, X^r - 1)$ $\left(f^{\bmod r} = \sum_j c_j X^{\alpha_j \bmod r}\right)$

1. For $r = 1$ to $2d^2$, **factor** $\gcd(f, X^r - 1)$; (deg $\phi_r = \varphi(r) \geqslant \sqrt{r/2}$)
2. Return the union of these factors.

Cyclotomic factors

- Lenstra's algorithm:
  - ϕ_r divides $f \iff \phi_r$ divides $\gcd(f, X^r - 1)$
  - $\gcd(f, X^r - 1) = \gcd(f^{\bmod r}, X^r - 1)$ $(f^{\bmod r} = \sum_j c_j X^{\alpha_j \bmod r})$

  1. For $r = 1$ to $2d^2$, **factor** $\gcd(f, X^r - 1)$; $(\deg \phi_r = \varphi(r) \geqslant \sqrt{r/2})$
  2. Return the union of these factors.

- Implemented algorithm:
  - ϕ_r divides $f \iff \phi_r$ divides $f^{\bmod r} = \sum_j c_j X^{\alpha_j \bmod r}$
  - Applied to each $h \in H$

- Lenstra's algorithm:
  - ϕ_r divides $f \iff \phi_r$ divides $\gcd(f, X^r - 1)$
  - $\gcd(f, X^r - 1) = \gcd(f^{\bmod r}, X^r - 1)$ $\quad (f^{\bmod r} = \sum_j c_j X^{\alpha_j \bmod r})$

1. For $r = 1$ to $2d^2$, **factor** $\gcd(f, X^r - 1)$; $(\deg \phi_r = \varphi(r) \geqslant \sqrt{r/2})$
2. Return the union of these factors.

- Implemented algorithm:
  - ϕ_r divides $f \iff \phi_r$ divides $f^{\bmod r} = \sum_j c_j X^{\alpha_j \bmod r}$
  - Applied to each $h \in H$

1. Compute $R = \{r : \varphi(r) \leqslant d\}$ (*naive*)
2. For $r \in R$, compute $\phi_r = (X^r - 1)/\gcd(X^r - 1, \prod_{\substack{s \in R \\ s < r}} \phi_s)$
3. Test if ϕ_r **divides** $h^{\bmod r}$

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

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

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

- l: product of low-degree polynomials

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

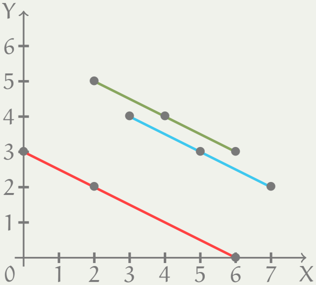
- c: product of $X^r - 1$

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

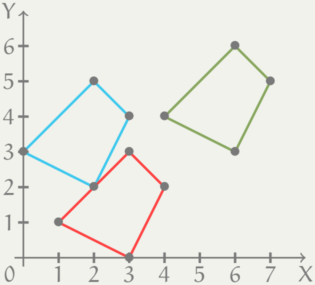
- s: *perturbated* sparse polynomial: $s = \sum_{j=1}^{n} X^{\alpha_j} p_j(X)$

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

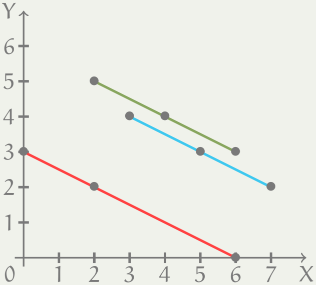
$\rightsquigarrow$ degree $\geqslant 1\,000\,000$, $\geqslant 10\,000$ terms, coefficients $\geqslant 5 \times 10^9$



Unidimensional factors



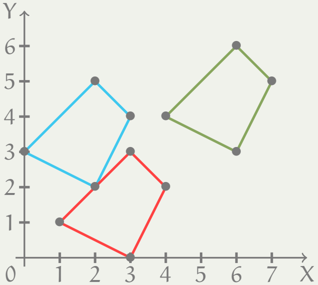
Multidimensional factors



Unidimensional factors

⇓

Univariate lacunary factorization

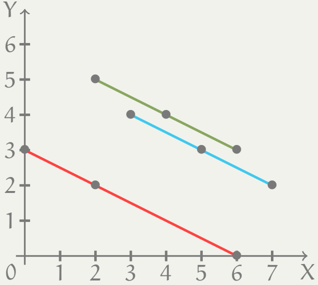


Multidimensional factors

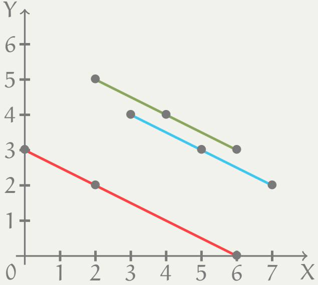
⇓

Multivariate low-degree factorization

Unidimensional factors



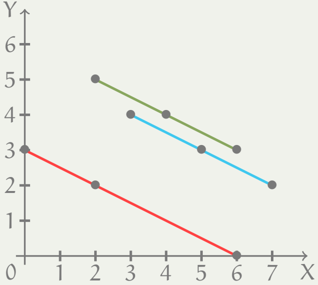
- Directions given by the support
- Each direction δ independently



- Directions given by the support
- Each direction δ independently

1. Write f as a sum of **δ-components**;
2. **Project** each component to a univariate lacunary polynomial;
3. Compute the common **bounded-degree factors** of the projections;
   Compute the factors of the first, then test divisibility
4. **Lift** the univariate factors to unidimensional factors.

Unidimensional factors

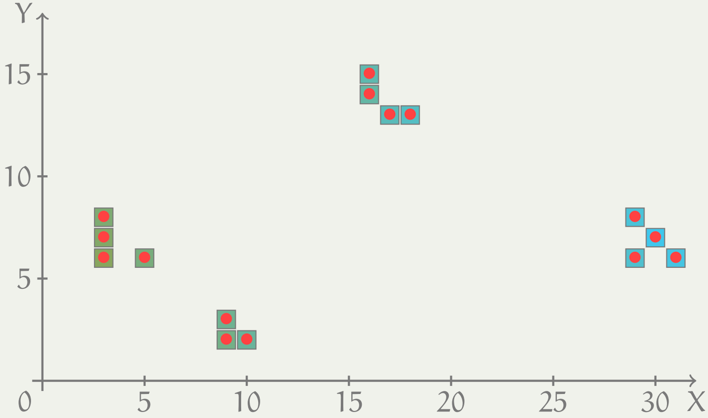


- Directions given by the support
- Each direction δ independently

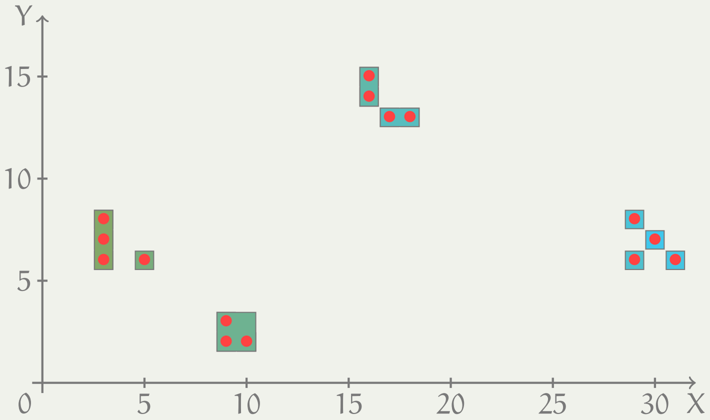
- Early termination:
  - Degenerate δ-component
  - No common factors

1. Write f as a sum of **δ-components**;
2. **Project** each component to a univariate lacunary polynomial;
3. Compute the common **bounded-degree factors** of the projections;
   Compute the factors of the first, then test divisibility
4. **Lift** the univariate factors to unidimensional factors.

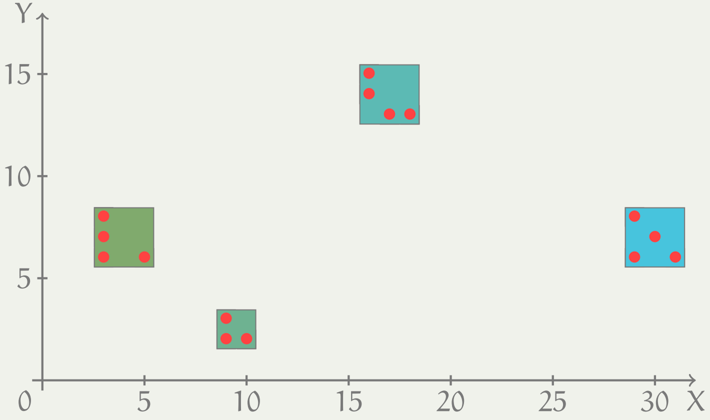
Multidimensional factors

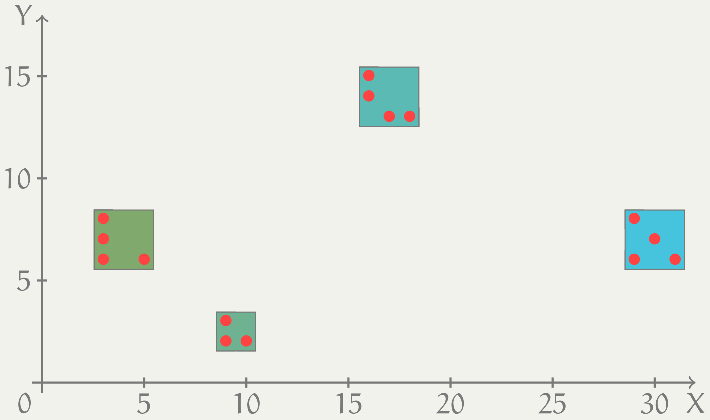


Multidimensional factors



Multidimensional factors





Single-linkage clustering

- 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
  - 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
  - Bound the *size* of the computation rather than the size of the factors
- Techniques useful for *classical* factorization?

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

Thank you!


Joyeux anniversaire Romain
aujourd'hui
c'est toi
la star !
joliecarte.com