Some algorithmic and combinatorial problems on permutation classes

⋆ ⋆ ⋆

The point of view of decomposition trees

Mathilde Bouvel

PhD Defense, 2009 December the 4th

Outline

Outline

Representation of permutations

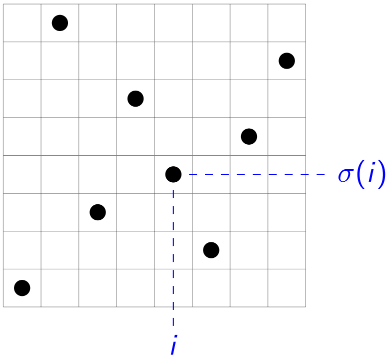
Permutation : Bijection from $[1..n]$ to itself. Set S_n.

- Linear representation :
  $\sigma = 1\ 8\ 3\ 6\ 4\ 2\ 5\ 7$

- Two lines representation :
  $$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 8 & 3 & 6 & 4 & 2 & 5 & 7 \end{pmatrix}$$

- Representation as a product of cycles :
  $\sigma = (1)\ (2\ 8\ 7\ 5\ 4\ 6)\ (3)$

- Graphical representation :



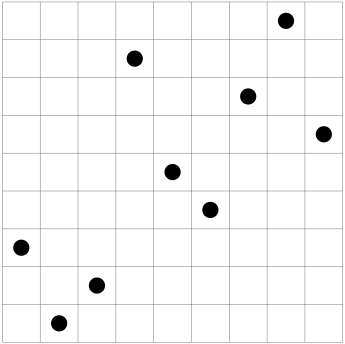
Patterns in permutations

Pattern (order) relation $\preccurlyeq$:
$\pi \in S_k$ is a pattern of $\sigma \in S_n$ if $\exists\ 1 \leq i_1 < \ldots < i_k \leq n$ such that $\sigma_{i_1} \ldots \sigma_{i_k}$ is order isomorphic ($\equiv$) to π.

Notation : $\pi \preccurlyeq \sigma$.

Equivalently :
The normalization of $\sigma_{i_1} \ldots \sigma_{i_k}$ on $[1..k]$ yields π.

Example : $2\,1\,3\,4 \preccurlyeq \mathbf{3}\,\mathbf{1}\,2\,8\,\mathbf{5}\,4\,\mathbf{7}\,9\,6$
since $3\,1\,5\,7 \equiv 2\,1\,3\,4$.



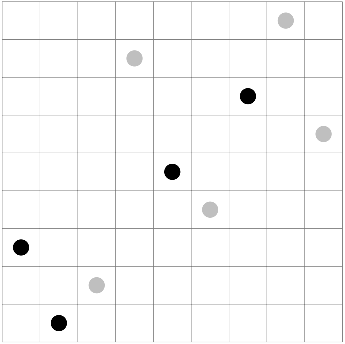
Patterns in permutations

Pattern (order) relation $\preccurlyeq$:
$\pi \in S_k$ is a pattern of $\sigma \in S_n$ if $\exists\ 1 \leq i_1 < \ldots < i_k \leq n$ such that $\sigma_{i_1} \ldots \sigma_{i_k}$ is order isomorphic ($\equiv$) to π.

Notation : $\pi \preccurlyeq \sigma$.

Equivalently :
The normalization of $\sigma_{i_1} \ldots \sigma_{i_k}$ on $[1..k]$ yields π.

Example : $2\,1\,3\,4 \preccurlyeq \mathbf{3}\,\mathbf{1}\,2\,8\,\mathbf{5}\,4\,\mathbf{7}\,9\,6$
since $3\,1\,5\,7 \equiv 2\,1\,3\,4$.

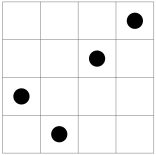


Patterns in permutations

Pattern (order) relation $\preccurlyeq$:
$\pi \in S_k$ is a pattern of $\sigma \in S_n$ if $\exists\, 1 \leq i_1 < \ldots < i_k \leq n$ such that $\sigma_{i_1} \ldots \sigma_{i_k}$ is order isomorphic $(\equiv)$ to π.

Notation : $\pi \preccurlyeq \sigma$.

Equivalently :
The normalization of $\sigma_{i_1} \ldots \sigma_{i_k}$ on $[1..k]$ yields π.



Example : $2\,1\,3\,4 \preccurlyeq \mathbf{3}\,\mathbf{1}\,2\,8\,\mathbf{5}\,4\,\mathbf{7}\,9\,6$
since $3\,1\,5\,7 \equiv 2\,1\,3\,4$.

Permutation classes

Permutation class : set of permutations downward-closed for $\preccurlyeq$.

$S(B)$: the class of permutations that avoid every pattern of B.
If B is an antichain then B is the basis of $S(B)$.

Conversly : Every class $\mathcal{C}$ can be characterized by its basis :

$$\mathcal{C} = S(B) \text{ for } B = \{\sigma \notin \mathcal{C} : \forall \pi \preccurlyeq \sigma \text{ such that } \pi \neq \sigma, \pi \in \mathcal{C}\}$$

A class has a unique basis.
A basis can be either finite or infinite.

Origine : [Knuth 73] with stack-sortable permutations $= S(231)$

Enumeration[Stanley & Wilf 92][Marcus & Tardos 04] : $|\mathcal{C} \cap S_n| \leq c^n$

Problematics

- **Combinatorics** : study of classes defined by their basis.

$\hookrightarrow$ Enumeration.

$\hookrightarrow$ Exhaustive generation.

- **Algorithmics** : problematics from text algorithmics.

$\hookrightarrow$ Pattern matching, longest common pattern.

$\hookrightarrow$ Linked with testing the membership of σ to a class.

- **Combinatorics (and algorithms)** : studying classes as a whole.

$\hookrightarrow$ A class is not always described by its basis.

$\hookrightarrow$ Detect automatically the structure of a class.

Outline

Substitution decomposition : main ideas

Analogous to the decomposition of integers as products of primes.

- [Möhring & Radermacher 84] : general framework.
- Specialization : Modular decomposition of graphs.

Relies on :

- a principle for building objects (permutations, graphs) from smaller objects : the substitution.
- some "basic objects" for this construction : **simple** permutations, **prime** graphs.

Required properties :

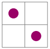
- every object can be decomposed using only "basic objects".
- this decomposition is unique.

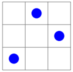
Substitution for permutations

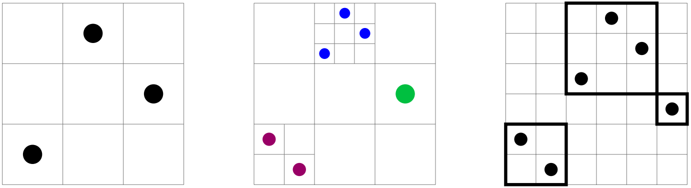
Substitution or inflation : $\sigma = \pi[\alpha^{(1)}, \alpha^{(2)}, \ldots, \alpha^{(k)}]$.

Example : Here, $\pi = 1\,3\,2$, and

$$\begin{cases} \alpha^{(1)} = 2\,1 = \\ \alpha^{(2)} = 1\,3\,2 = \\ \alpha^{(3)} = 1 = \end{cases}$$



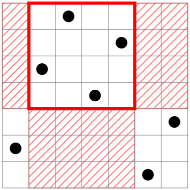




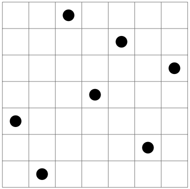
Hence $\sigma = 1\,3\,2[2\,1, 1\,3\,2, 1] = 2\,1\,4\,6\,5\,3$.

Simple permutations

Interval (or block) = set of elements of σ whose positions **and** values form intervals of integers
Example : 5 7 4 6 is an interval of 2 5 7 4 6 1 3



Simple permutation = permutation that has no interval, except the trivial intervals : $1, 2, \ldots, n$ and σ
Example : 3 1 7 4 6 2 5 is simple.



The smallest simple : 1 2, 2 1, 2 4 1 3, 3 1 4 2

Substitution decomposition of permutations

Theorem : Every σ ($\neq 1$) is uniquely decomposed as

- $12 \dots k[\alpha^{(1)}, \dots, \alpha^{(k)}]$, where the $\alpha^{(i)}$ are $\oplus$-indecomposable
- $k \dots 21[\alpha^{(1)}, \dots, \alpha^{(k)}]$, where the $\alpha^{(i)}$ are $\ominus$-indecomposable
- $\pi[\alpha^{(1)}, \dots, \alpha^{(k)}]$, where π is simple of size $k \geq 4$

Remarks :

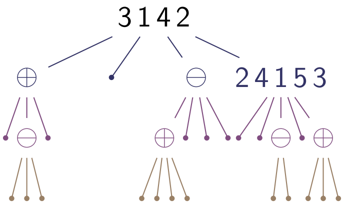
- $\oplus$-indecomposable : that cannot be written as $12[\alpha^{(1)}, \alpha^{(2)}]$
- Rephrasing a result of [Albert & Atkinson 05]
- The $\alpha^{(i)}$ are the maximal strong intervals of σ

Decomposing recursively inside the $\alpha^{(i)} \Rightarrow$ decomposition tree

Decomposition tree : witness of this decomposition

Example : Decomposition tree of $\sigma =$

10 13 12 11 14 1 18 19 20 21 17 16 15 4 8 3 2 9 5 6 7



Notations and properties :

- $\oplus = 12\ldots k$ and $\ominus = k\ldots 21$ = linear nodes.
- π simple of size ≥ 4 = prime node.
- No edge $\oplus - \oplus$ nor $\ominus - \ominus$.
- Ordered trees.

$\sigma = 3\,1\,4\,2[\oplus[1, \ominus[1, 1, 1], 1], 1, \ominus[\oplus[1, 1, 1, 1], 1, 1, 1], 2\,4\,1\,5\,3[1, 1, \ominus[1, 1], 1, \oplus[1, 1, 1]]]$

Bijection between permutations and their decomposition trees.

Computation and examples of application

Computation : in linear time. [Uno & Yagiura 00] [Bui Xuan, Habib & Paul 05] [Bergeron, Chauve, Montgolfier & Raffinot 08]

In algorithms :

- Pattern matching [Bose, Buss & Lubiw 98] [Ibarra 97]
- Algorithms for bio-informatics [Bérard, Bergeron, Chauve & Paul 07] [Bérard, Chateau, Chauve, Paul & Tannier 08]

In combinatorics :

- Simple permutations [Albert, Atkinson & Klazar 03]
- Classes closed by substitution product [Atkinson & Stitt 02] [Brignall 07] [Atkinson, Ruškuc & Smith 09]
- Exhibit the structure of classes [Albert & Atkinson 05] [Brignall, Huczynska & Vatter 08a,08b] [Brignall, Ruškuc & Vatter 08]

Outline

Pattern matching

Problem, which is *NP*-hard :

- Input : pattern σ (size k), permutation τ (size n).
- Output : an occurrence of σ in τ if it exists.

Restriction : σ is separable. Polynomial subproblem.

Separable permutations :

- Definition by excluded patterns : $S(2413, 3142)$
- Other definition : having a separating tree
- Characterization : decomposition tree with no prime node

Pattern matching of a separable pattern

Dynamic Programming [Bose, Buss & Lubiw 98] [Ibarra 97]

- following the guide = separating tree of σ
- from the leaves to the root
- for windows of positions and values

Complexity : [Bose, Buss & Lubiw 98]

- $\mathcal{O}(kn^6)$ in time
- $\mathcal{O}(kn^4)$ in space

[Ibarra 97]

- $\mathcal{O}(kn^4)$ in time
- $\mathcal{O}(kn^3)$ in space

$\Rightarrow$ Polynomial

Generalization with decomposition trees

Method :

- Dynamic programming.
- Consider further the prime nodes of decomposition trees.

Solutions obtained : [**B.** & Rossin 06] [**B.**, Rossin & Vialette 07]

- Pattern matching of any pattern in $\mathcal{O}(kn^{2d+2})$
- Finding a longest common pattern between two permutations, one of which is separable, in $\mathcal{O}(\min(n_1, n_2) n_1 n_2^6)$
- Finding a longest common pattern between two permutations in $\mathcal{O}(\min(n_1, n_2) n_1 n_2^{2d_1+2})$

with d = maximal arity of a prime node

Outline

Structure in permutation classes

Theorem [Albert & Atkinson 05] : If $\mathcal{C}$ contains a finite number of simple permutations, then

- $\mathcal{C}$ has a finite basis
- $\mathcal{C}$ has an algebraic generating function ($= \sum_n |\mathcal{C} \cap S_n| x^n$)

Proof : relies on the substitution decomposition.
Construction : compute the generating function from the simples in $\mathcal{C}$

Algorithmically :

- Semi-decision procedure

$\hookrightarrow$ Find simples of size $4, 5, 6, \ldots$ until k and $k+1$ for which there are 0 simples [Schmerl & Trotter 93]

- "Very exponential" ($\sim n!$) computation of the simples in $\mathcal{C}$

Finite number of simple permutations : decision

Theorem [Brignall, Ruškuc & Vatter 08] : It is decidable whether $\mathcal{C}$ given by its finite basis contains a finite number of simples.

Prop. $\mathcal{C} = S(B)$ contains infinitely many simples iff $\mathcal{C}$ contains :

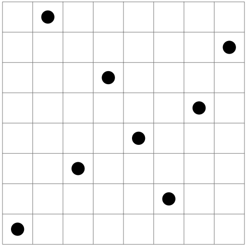
1. either infinitely many parallel permutations
2. or infinitely many simple wedge permutations
3. or infinitely many proper pin-permutations

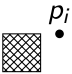
	Decision procedure	Complexity
1. and 2. :	pattern matching of patterns of size 3 or 4 in the $\beta \in B$.	Polynomial
3. :	Decidability with automata techniques	Decidable 2ExpTime

The class of pin-permutations

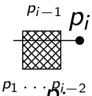
Pin-permutation = that admits a pin representation, *i.e.* a sequence $(p_1, \ldots, p_n)$ where each p_i satisfies :

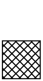
Example :



- the exteriority condition 

- and

• either the separation condition 

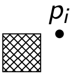
• or the independence condition 

 = *bounding box* of $\{p_1, \ldots, p_{i-1}\}$

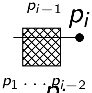
Encoding by pin words on $\{1, 2, 3, 4, L, R, U, D\}$ with $\begin{array}{c|c} 2 & 1 \\ \hline 3 & 4 \end{array}$

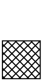
The class of pin-permutations

Pin-permutation = that admits a pin representation, *i.e.* a sequence $(p_1, \ldots, p_n)$ where each p_i satisfies :

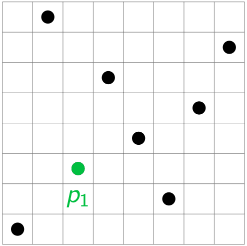
- the exterioty condition 

- and

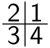
• either the separation condition 

• or the independence condition 

Example :

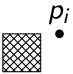


 = *bounding box* of $\{p_1, \ldots, p_{i-1}\}$

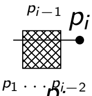
Encoding by pin words on $\{1, 2, 3, 4, L, R, U, D\}$ with 

The class of pin-permutations

Pin-permutation = that admits a pin representation, *i.e.* a sequence $(p_1, \ldots, p_n)$ where each p_i satisfies :

- the exteriority condition 

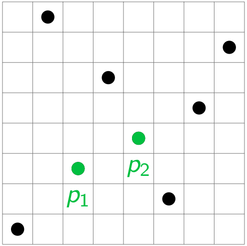
- and

• either the separation condition 

• or the independence condition 

 = *bounding box* of $\{p_1, \ldots, p_{i-1}\}$

Example :



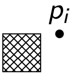
1

Encoding by pin words on $\{1, 2, 3, 4, L, R, U, D\}$ with $\frac{2}{3}\Big|\frac{1}{4}$

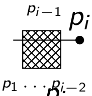
The class of pin-permutations

Pin-permutation = that admits a pin representation, *i.e.* a sequence $(p_1, \ldots, p_n)$ where each p_i satisfies :

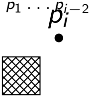
- the exteriority condition
- and



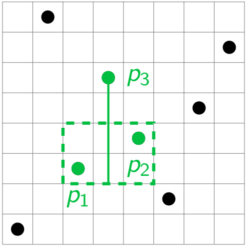
- either the separation condition



- or the independence condition



Example :



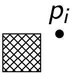
$1\ U$

 = *bounding box* of $\{p_1, \ldots, p_{i-1}\}$

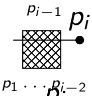
Encoding by pin words on $\{1, 2, 3, 4, L, R, U, D\}$ with $\frac{2}{3}\big|\frac{1}{4}$

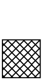
The class of pin-permutations

Pin-permutation = that admits a pin representation, *i.e.* a sequence $(p_1, \ldots, p_n)$ where each p_i satisfies :

- the exteriority condition 

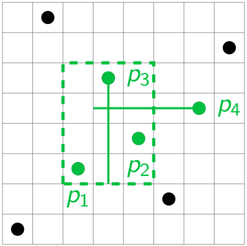
- and

• either the separation condition 

• or the independence condition 

 = *bounding box* of $\{p_1, \ldots, p_{i-1}\}$

Example :

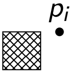


$1\,U\,R$

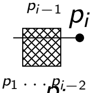
Encoding by pin words on $\{1, 2, 3, 4, L, R, U, D\}$ with $\begin{array}{c|c} 2 & 1 \\ \hline 3 & 4 \end{array}$

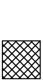
The class of pin-permutations

Pin-permutation = that admits a pin representation, *i.e.* a sequence $(p_1, \ldots, p_n)$ where each p_i satisfies :

- the exteriority condition 

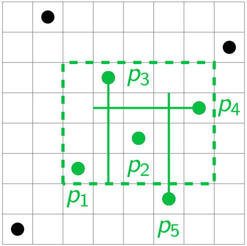
- and

- either the separation condition 

- or the independence condition 

 = *bounding box* of $\{p_1, \ldots, p_{i-1}\}$

Example :

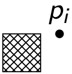


$1\,U\,R\,D$

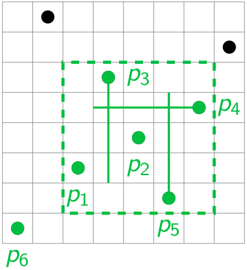
Encoding by pin words on $\{1, 2, 3, 4, L, R, U, D\}$ with $\begin{array}{c|c} 2 & 1 \\ \hline 3 & 4 \end{array}$

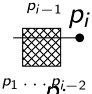
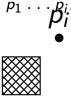
The class of pin-permutations

Pin-permutation = that admits a pin representation, *i.e.* a sequence $(p_1, \dots, p_n)$ where each p_i satisfies :

- the exteriority condition 
- and

Example :



- either the separation condition 
- or the independence condition 

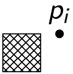
 = *bounding box* of $\{p_1, \dots, p_{i-1}\}$

$1\,U\,R\,D\,3$

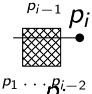
Encoding by pin words on $\{1, 2, 3, 4, L, R, U, D\}$ with $\frac{2}{3}\big|\frac{1}{4}$

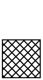
The class of pin-permutations

Pin-permutation = that admits a pin representation, *i.e.* a sequence $(p_1, \ldots, p_n)$ where each p_i satisfies :

- the exteriority condition 

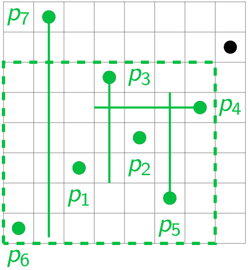
- and

- either the separation condition 

- or the independence condition 

 $=$ *bounding box* of $\{p_1, \ldots, p_{i-1}\}$

Example :

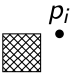


$1\,U\,R\,D\,3\,U$

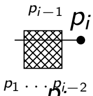
Encoding by pin words on $\{1, 2, 3, 4, L, R, U, D\}$ with $\frac{2}{3}\Big|\frac{1}{4}$

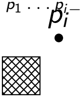
The class of pin-permutations

Pin-permutation = that admits a pin representation, *i.e.* a sequence $(p_1, \ldots, p_n)$ where each p_i satisfies :

- the exteriority condition 

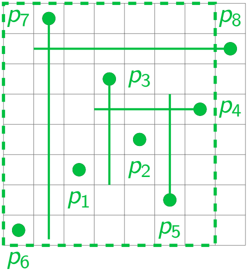
- and

- either the separation condition 

- or the independence condition 

 = *bounding box* of $\{p_1, \ldots, p_{i-1}\}$

Example :

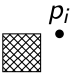


$1\,U\,R\,D\,3\,U\,R$

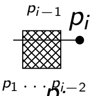
Encoding by pin words on $\{1, 2, 3, 4, L, R, U, D\}$ with $\frac{2}{3}\Big|\frac{1}{4}$

The class of pin-permutations

Pin-permutation = that admits a pin representation, *i.e.* a sequence $(p_1, \ldots, p_n)$ where each p_i satisfies :

- the exteriority condition 

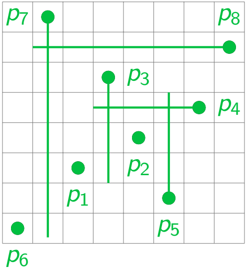
- and

• either the separation condition 

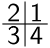
• or the independence condition 

 = *bounding box* of $\{p_1, \ldots, p_{i-1}\}$

Example :

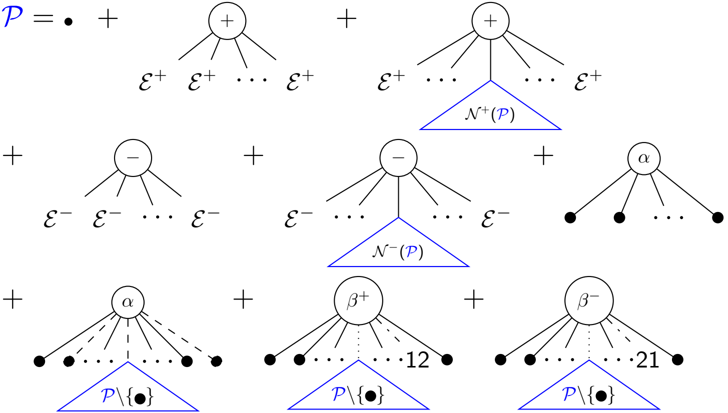


$1\,U\,R\,D\,3\,U\,R$

Encoding by pin words on $\{1, 2, 3, 4, L, R, U, D\}$ with 

Some results on pin-permutations (1/2)

- Characterization of their decomposition trees [Bassino, **B.** & Rossin 09]

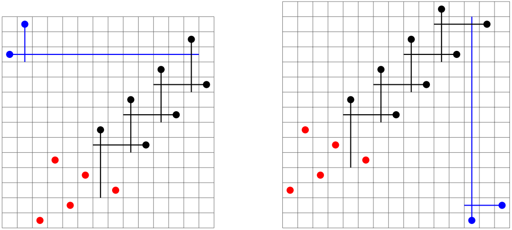


Some results on pin-permutations (2/2)

- Computation of the generating function : rational [BBR09]

$$P(z) = z\frac{8z^6-20z^5-4z^4+12z^3-9z^2+6z-1}{8z^8-20z^7+8z^6+12z^5-14z^4+26z^3-19z^2+8z-1}$$

- Infinite basis (still to be determined) [BBR09]



- Polynomial algorithm checking whether the number of simples in $S(B)$ is finite [Bassino, B., Pierrot & Rossin], instead of the decision procedure of [BRV08]

Polynomial algorithm for the finite number of simples

Points similar to [BRV08] :

- Encoding by pin words on $\{1, 2, 3, 4, L, R, U, D\}$
- Construction of automata

Study of pin-permutations $\Rightarrow$ better understanding of the relationship between pin words and patterns in permutations

Points specific to [B**B**PR] :

- Polynomial construction of a **(deterministic, complete)** automaton for the language $\mathcal{L}$ = pin words of proper pin-permutations containing some $\beta \in B$
- Is this language co-finite ? Polynomial.

$\hookrightarrow$ Yes iff the class contains finitely many simples.

Automatic computation of the generating function

What is done :

- Deciding the finite number of simples

$\hookrightarrow$ Polynomial

- Computing the simples in the class

$\hookrightarrow$ Exponential

- Computing the (algebraic) generating function from the simples

$\hookrightarrow$ Possible on any example

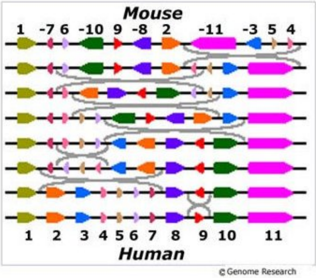
What remains to do :

- Automatically compute the generating function from the simples
- Polynomial computation of the set of simples in a class
- If $\mathcal{C}$ is not given by its finite basis ?

Outline

Motivations and the model

- Genomes = sequences of genes ≈ Signed permutations
- Only one type of mutation is possible ≈ Reversal = reversing a window while changing the signs
- Goal : evolution scenario ≈ Sequence of reversals
- Group of common genes ≈ Interval of permutations



$$1\ \bar{7}\ 6\ \overline{10}\ 9\ \bar{8}\ 2\ \overline{11}\ \bar{3}\ 5\ 4$$

$\Downarrow$ Reversal $\Downarrow$

$$1\ \bar{7}\ 6\ \overline{10}\ 9\ \bar{8}\ 2\ \bar{4}\ \bar{5}\ 3\ 11$$

Additional constraint for perfect sorting :
do not break any interval

Perfect sorting by reversals

- Input : Two signed permutations σ_1 and σ_2
- Output : A parcimonious **perfect** scenario from σ_1 to σ_2 or $\overline{\sigma_2}$

We can always assume that $\sigma_2 = Id = 1\ 2 \dots n$

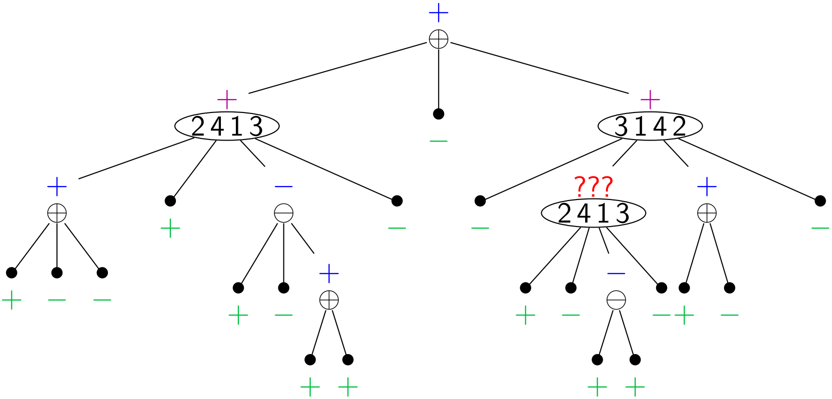
Sorting by reversals : polynomial [Hannenhalli & Pevzner 99]

Perfect sorting by reversals :

- *NP*-hard problem [Figeac & Varré 04]
- FPT algorithm [Bérard, Bergeron, Chauve & Paul 07] : uses the decomposition tree, in time $\mathcal{O}\left(2^{p} \cdot n^{\mathcal{O}(1)}\right)$
- Complexity parametrized by
  p = number of prime nodes (with a prime parent)

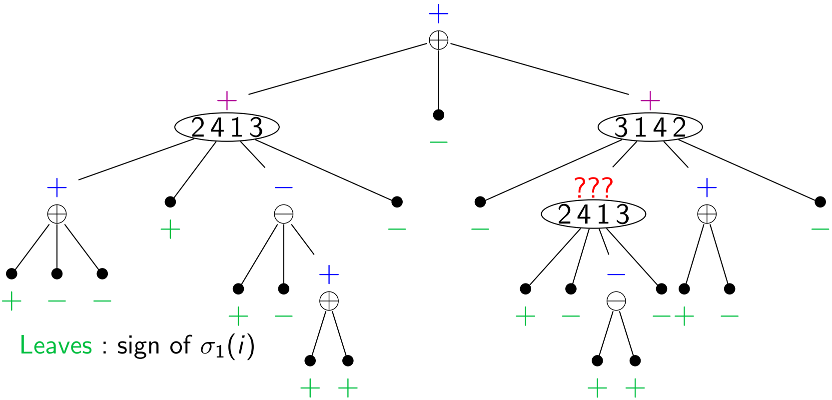
Idea of the algorithm on an example

$$\sigma_1 = 5\ \overline{6}\ \overline{7}\ 9\ 4\ \overline{3}\ 1\ 2\ \overline{8}\ \overline{10}\ \overline{17}\ 13\ \overline{15}\ 12\ 11\ \overline{14}\ 18\ \overline{19}\ \overline{16}$$



Idea of the algorithm on an example

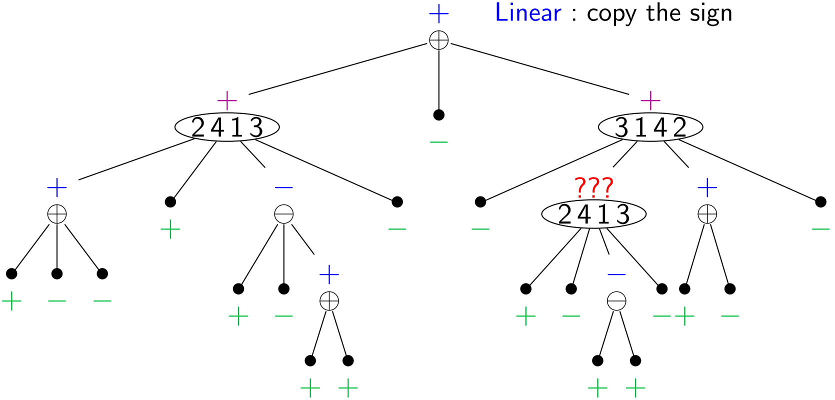
$$\sigma_1 = 5\ \overline{6}\ \overline{7}\ 9\ 4\ \overline{3}\ 1\ 2\ \overline{8}\ \overline{10}\ \overline{17}\ 13\ \overline{15}\ 12\ 11\ \overline{14}\ 18\ \overline{19}\ \overline{16}$$



Leaves : sign of $\sigma_1(i)$

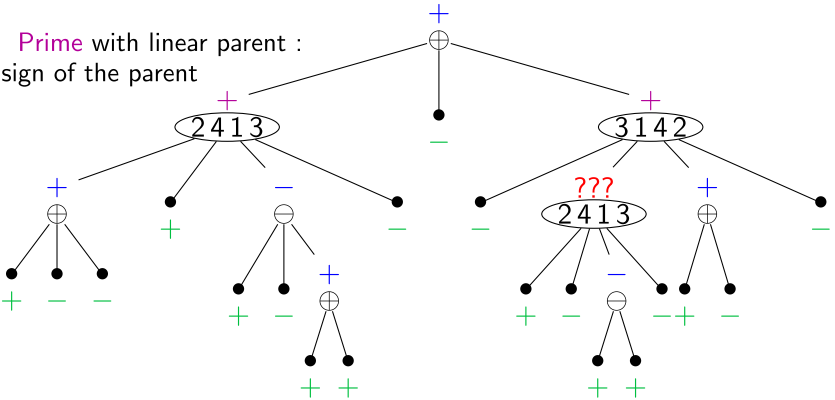
Idea of the algorithm on an example

$$\sigma_1 = 5\ \overline{6}\ \overline{7}\ 9\ 4\ \overline{3}\ 1\ 2\ \overline{8}\ \overline{10}\ \overline{17}\ 13\ \overline{15}\ 12\ 11\ \overline{14}\ 18\ \overline{19}\ \overline{16}$$



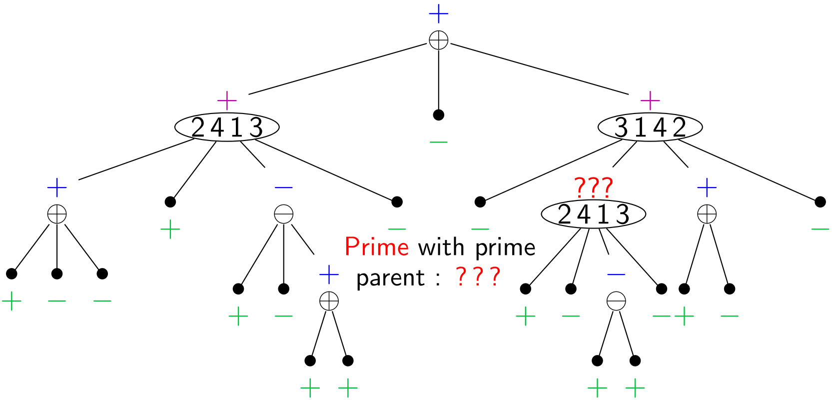
Idea of the algorithm on an example

$$\sigma_1 = 5\ \overline{6}\ \overline{7}\ 9\ 4\ \overline{3}\ 1\ 2\ \overline{8}\ \overline{10}\ \overline{17}\ 13\ \overline{15}\ 12\ 11\ \overline{14}\ 18\ \overline{19}\ \overline{16}$$



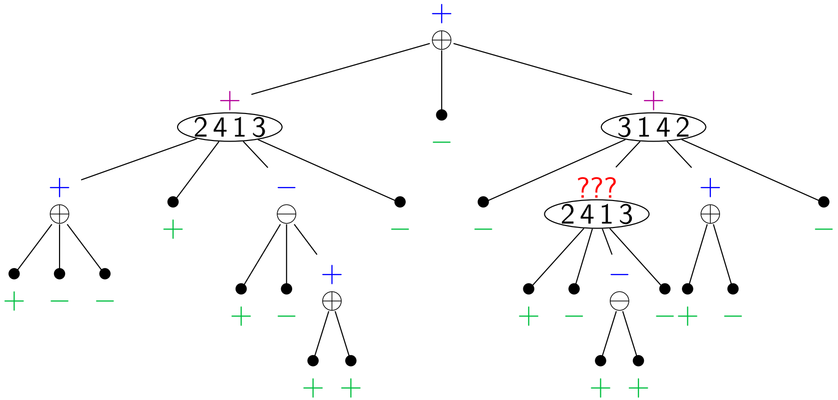
Idea of the algorithm on an example

$$\sigma_1 = 5\ \overline{6}\ \overline{7}\ 9\ 4\ \overline{3}\ 1\ 2\ \overline{8}\ \overline{10}\ \overline{17}\ 13\ \overline{15}\ 12\ 11\ \overline{14}\ 18\ \overline{19}\ \overline{16}$$



Idea of the algorithm on an example

$$\sigma_1 = 5\ \overline{6}\ \overline{7}\ 9\ 4\ \overline{3}\ 1\ 2\ \overline{8}\ \overline{10}\ \overline{17}\ 13\ \overline{15}\ 12\ 11\ \overline{14}\ 18\ \overline{19}\ \overline{16}$$



Complexity results

Previous results [BBCP07] :

- $\mathcal{O}(2^p n\sqrt{n \log n})$, where $p =$ number of prime nodes
- polynomial on separable permutations ($p = 0$)

Complexity analysis [**B.**, Chauve, Mishna & Rossin 09] :

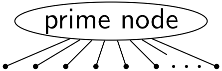
- polynomial with probability 1 asymptotically
- polynomial on average
- in a parsimonious scenario for separable permutations
  - average number of reversals $\sim 1.2n$
  - average size of a reversal $\sim 1.02\sqrt{n}$

Probability distribution : always uniforme

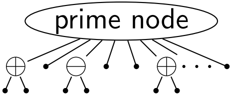
"Average" shape of decomposition trees

Enumeration of simple permutations : asymptotically $\frac{n!}{e^2}$

$\Rightarrow$ Asymptotically, a proportion $\frac{1}{e^2}$ of decom- -position trees are reduced to one prime node.



Thm : Asymptotically, the proportion of decomposition trees made of a prime root with children that are leaves or twins is 1



twin = linear node with only two children, that are leaves

Consequence : Asymptotically, with probability 1, the algorithm runs in polynomial time.

Average complexity

Average complexity on permutations of size n :

$$\frac{\sum_{p=0}^{n} \sharp\{\sigma \text{ with } p \text{ prime nodes}\}\ C\ 2^p n\sqrt{n \log n}}{n!}$$

Thm : When $p \geq 2$,
number of permutations of size n with p prime nodes $\leq \frac{48(n-1)!}{2^p}$

Consequence : Average complexity on permutations of size n is $\leq 50Cn\sqrt{n \log n}$.
In particular, **polynomial on average.**

Parameters for separable permutations

Schröder trees $\approx$ decomposition trees of separable permutations :

- Average number of internal nodes : $\sim \frac{n}{\sqrt{2}}$
- Average value of the sum of the sizes of all subtrees : $\sim 2^{3/4}\sqrt{3-2\sqrt{2}}\,\sqrt{\pi n^3}$

Signed separable permutations :

- Average number of reversals : $\sim \frac{1+\sqrt{2}}{2} n$
- Average value of the sum of the sizes of all reversals : $\sim 2^{3/4}\sqrt{3-2\sqrt{2}}\,\sqrt{\pi n^3}$
- Average size of a reversal : $\sim \frac{2^{7/4}\sqrt{3-2\sqrt{2}}}{1+\sqrt{2}}\sqrt{\pi n} \sim 1.02\sqrt{n}$

Outline

Conclusions

With decomposition trees :

- Parametrized algorithms for finding patterns
  - pattern matching
  - longest common pattern [**B**R06, **B**RV07]
- Combinatorial study of pin-permutations
  - example of a permutation class [B**B**R09]
  - application for detecting structure [B**B**PR]
- Complexity analysis of algorithms
  - perfect sorting by reversals [**B**CMR09]

But also :

- Limits in the problem of finding longest common patterns, with patterns restricted to a class [**B.**, Rossin & Vialette 07]
- Combinatorial study of the model of tandem duplication - random loss [**B.** et Rossin 09] [**B.** & Pergola 08]

Perspectives

- Pattern matching : NP-hard. Does there exist an algorithm polynomial in n with a preprocessing of the pattern ?
- Computation of generating functions of $S(B)$ when containing a finite number of simples : some steps still missing
- Application to random generation
- Precise analysis of other algorithms involving decomposition trees (*Double-Cut and Join*)
- Extend concepts and results from graph theory to permutations, and vice-versa

Perspectives

- Pattern matching : *NP*-hard. Does there exist an algorithm polynomial in n with a preprocessing of the pattern ?
- Computation of generating functions of $S(B)$ when containing a finite number of simples : some steps still missing
- Application to random generation
- Precise analysis of other algorithms involving decomposition trees (*Double-Cut and Join*)
- Extend concepts and results from graph theory to permutations, and vice-versa

Thank you !

Perspectives

- Pattern matching : *NP*-hard. Does there exist an algorithm polynomial in n with a preprocessing of the pattern ?
- Computation of generating functions of $S(B)$ when containing a finite number of simples : some steps still missing
- Application to random generation
- Precise analysis of other algorithms involving decomposition trees (*Double-Cut and Join*)
- Extend concepts and results from graph theory to permutations, and vice-versa

Thank you !