# Quelques problèmes combinatoires et algorithmiques sur les classes de permutations

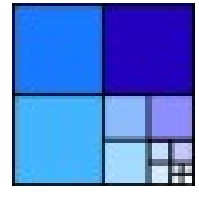
⋆ ⋆ ⋆

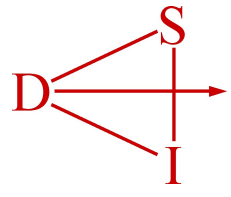
*Le point de vue des arbres de décomposition*

Mathilde Bouvel

Soutenance de thèse, le 4 décembre 2009







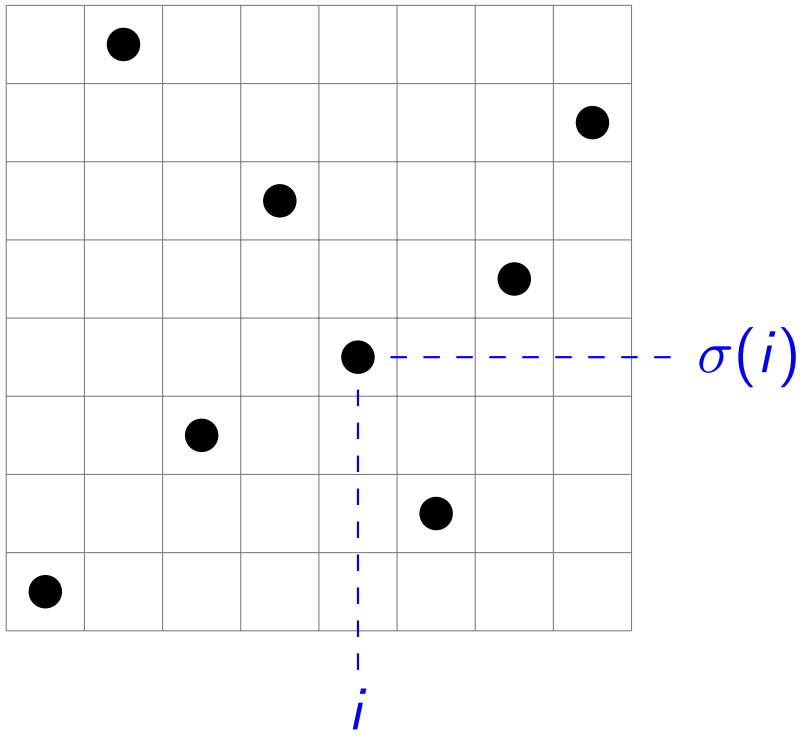
# Plan de l'exposé

# Plan de l'exposé

# Représentation des permutations

**Permutation** : Bijection de $[1..n]$ dans lui-même. Ensemble $S_n$.

- Représentation linéaire :
  $\sigma = 1\ 8\ 3\ 6\ 4\ 2\ 5\ 7$
- Représentation sur deux lignes :
  $\sigma = \begin{pmatrix} 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8 \\ 1\ 8\ 3\ 6\ 4\ 2\ 5\ 7 \end{pmatrix}$
- Représentation en produit de cycles :
  $\sigma = (1)\ (2\ 8\ 7\ 5\ 4\ 6)\ (3)$
- Représentation graphique :



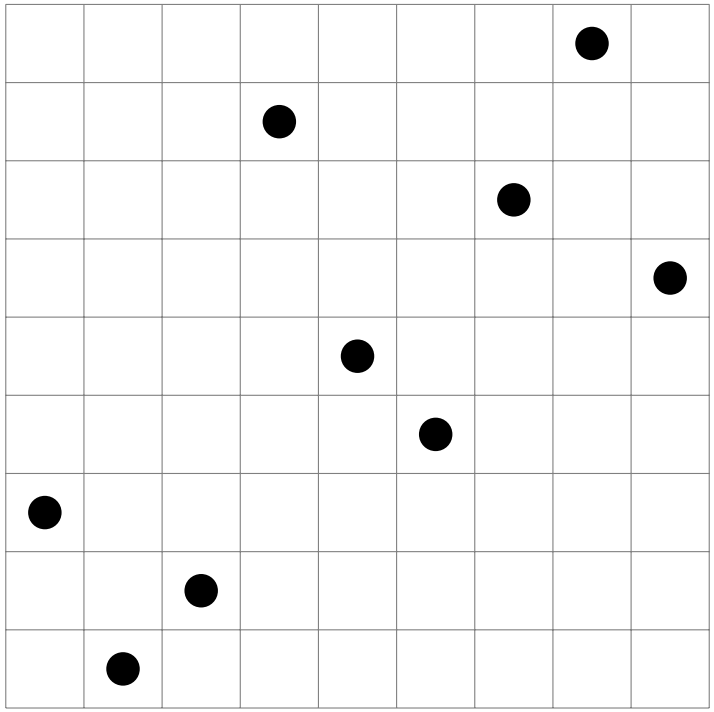
# Motif dans les permutations

**Relation (d'ordre) de motif $\preccurlyeq$ :**
$\pi \in S_k$ est un motif de $\sigma \in S_n$ si $\exists\ 1 \leq i_1 < \ldots < i_k \leq n$ tel que $\sigma_{i_1} \ldots \sigma_{i_k}$ est isomorphe en ordre ($\equiv$) à $\pi$.

Notation : $\pi \preccurlyeq \sigma$.

*De manière équivalente* :
La normalisation de $\sigma_{i_1} \ldots \sigma_{i_k}$ sur $[1..k]$ donne $\pi$.

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



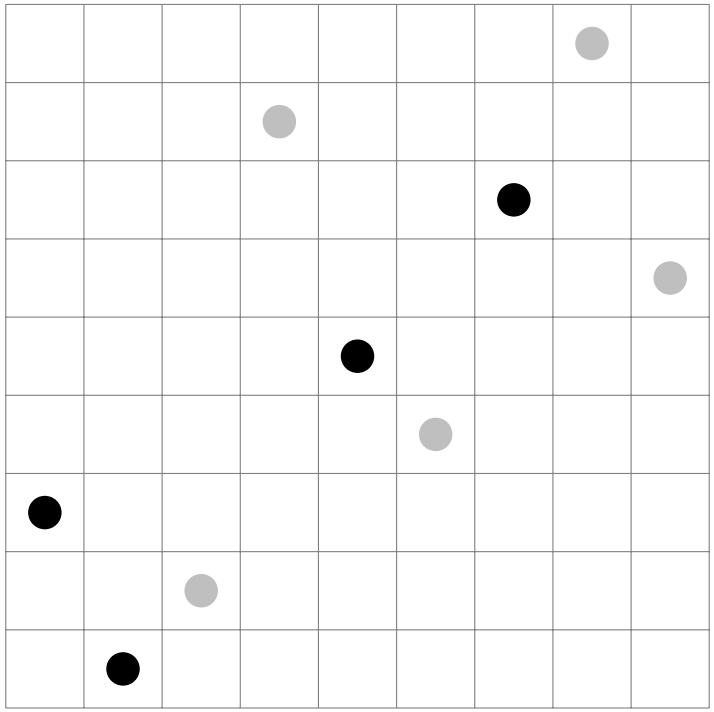
# Motif dans les permutations

**Relation (d'ordre) de motif $\preccurlyeq$ :**
$\pi \in S_k$ est un motif de $\sigma \in S_n$ si $\exists\ 1 \leq i_1 < \ldots < i_k \leq n$ tel que $\sigma_{i_1} \ldots \sigma_{i_k}$ est isomorphe en ordre ($\equiv$) à $\pi$.

Notation : $\pi \preccurlyeq \sigma$.

*De manière équivalente* :
La normalisation de $\sigma_{i_1} \ldots \sigma_{i_k}$ sur $[1..k]$ donne $\pi$.

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

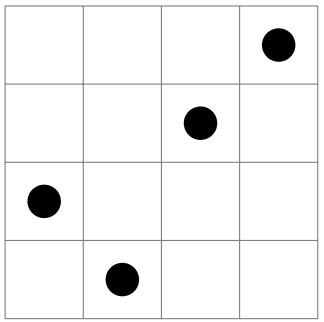


# Motif dans les permutations

**Relation (d'ordre) de motif $\preccurlyeq$ :**
$\pi \in S_k$ est un motif de $\sigma \in S_n$ si $\exists\ 1 \leq i_1 < \ldots < i_k \leq n$ tel que $\sigma_{i_1} \ldots \sigma_{i_k}$ est isomorphe en ordre ($\equiv$) à $\pi$.

Notation : $\pi \preccurlyeq \sigma$.

*De manière équivalente* :
La normalisation de $\sigma_{i_1} \ldots \sigma_{i_k}$ sur $[1..k]$ donne $\pi$.



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

# Classes de permutations

**Classe de permutations** : ensemble fermé par le bas pour $\preccurlyeq$.

$S(B)$ : la classe des permutations évitant tous les motifs de $B$.
Si $B$ est une antichaîne alors $B$ est la base de $S(B)$.

Réciproque : Toute classe $\mathcal{C}$ peut être caractérisée par sa base :

$$\mathcal{C} = S(B) \text{ pour } B = \{\sigma \notin \mathcal{C} : \forall \pi \preccurlyeq \sigma \text{ tel que } \pi \neq \sigma, \pi \in \mathcal{C}\}$$

La base d'une classe est unique.
Une base peut être finie ou infinie.

Origine : [Knuth 73] avec les triables par pile $= S(231)$

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

# Problématiques

- **Combinatoire** : étude de classes définies par leur base.

↪ Énumération.

↪ Génération exhaustive.

- **Algorithmique** : problématiques de l'algorithmique du texte.

↪ Recherche d'occurrence, de plus grand motif commun.

↪ En lien avec le test d'appartenance à une classe.

- **Combi. (et algo.)** : étude des classes dans leur ensemble.

↪ Une classe n'est pas toujours donnée par sa base.

↪ Automatiser la détection de structure dans une classe.

# Plan de l'exposé

# Décomposition par substitution : les grands principes

Analogue de la décomposition des entiers en facteurs premiers.

- [Möhring et Radermacher 84] : contexte général.
- Spécialisation : décomposition modulaire des graphes.

Repose sur :

- un principe de construction d'objets (permutations, graphes) à partir d'objets plus petits : la substitution.
- des "briques de base" pour cette construction : permutations **simples**, graphes **premiers**.

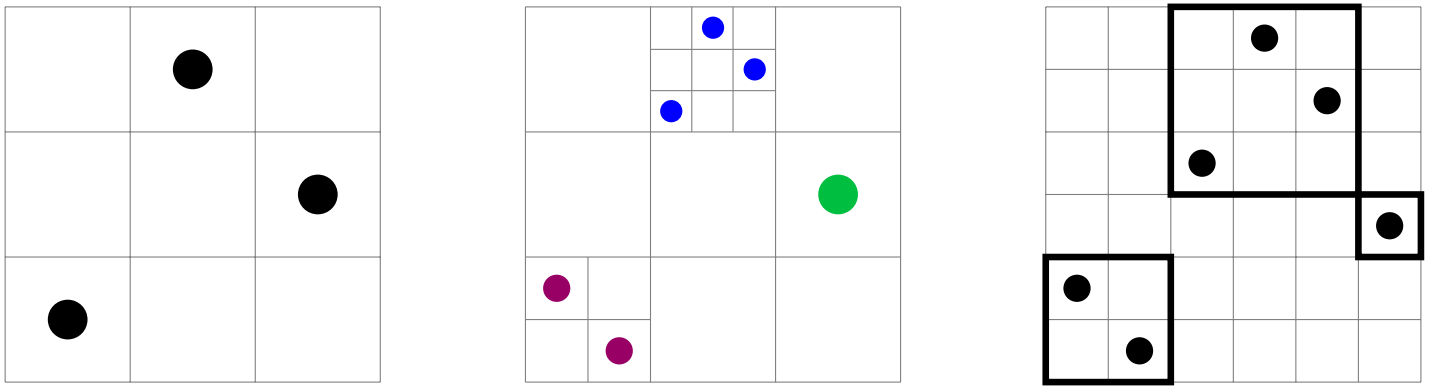
Propriétés essentielles :

- tout objet peut être décomposé en ces "briques de base".
- cette décomposition est unique.

# Principe de substitution sur les permutations

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

Exemple : Ici, $\pi = 1\,3\,2$, et $\begin{cases} \alpha^{(1)} = 2\,1 = \\ \alpha^{(2)} = 1\,3\,2 = \\ \alpha^{(3)} = 1 = \end{cases}$ .

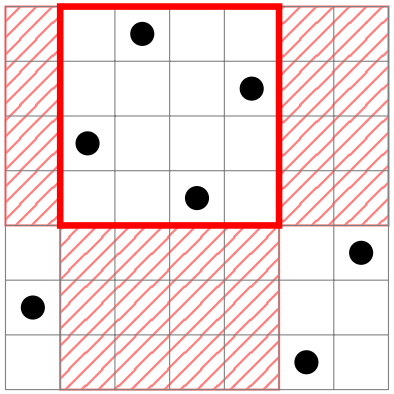


Donc $\sigma = 1\,3\,2[2\,1, 1\,3\,2, 1] = 2\,1\,4\,6\,5\,3$.

# Permutations simples

Intervalle (ou bloc) = ensemble d'éléments de $\sigma$ dont les positions **et** les valeurs sont des intervalles

Exemple : 5 7 4 6 est un intervalle de 2 5 7 4 6 1 3

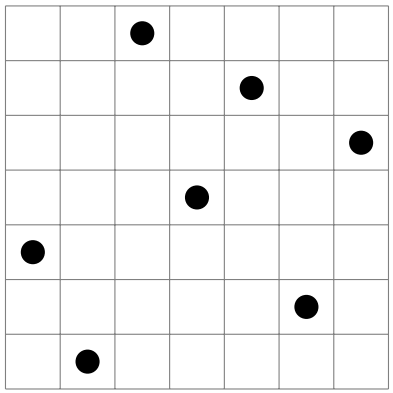


Permutation simple = ne possède pas d'intervalle, sauf les intervalles triviaux : $1, 2, \ldots, n$ et $\sigma$

Exemple : 3 1 7 4 6 2 5 est simple.

*Les plus petites simples* :

1 2, 2 1, 2 4 1 3, 3 1 4 2



# Décomposition par substitution des permutations

Théorème : Chaque $\sigma$ ($\neq 1$) se décompose de manière unique en

- $12\dots k[\alpha^{(1)},\dots,\alpha^{(k)}]$, où les $\alpha^{(i)}$ sont $\oplus$-indécomposables
- $k\dots 21[\alpha^{(1)},\dots,\alpha^{(k)}]$, où les $\alpha^{(i)}$ sont $\ominus$-indécomposables
- $\pi[\alpha^{(1)},\dots,\alpha^{(k)}]$, où $\pi$ est simple de taille $k \geq 4$

Remarques :

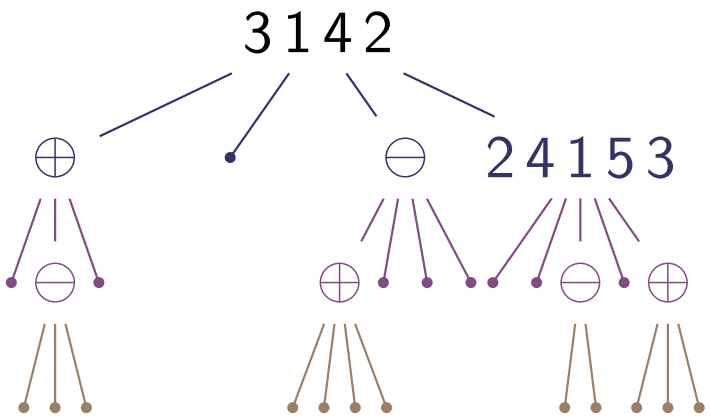
- $\oplus$-indécomposable : ne s'écrivant pas comme $12[\alpha^{(1)},\alpha^{(2)}]$
- Reformulation d'un résultat de [Albert et Atkinson 05]
- Les $\alpha^{(i)}$ sont les intervalles forts maximaux de $\sigma$

Décomposition récursive dans les $\alpha^{(i)} \Rightarrow$ arbre de décomposition

# Arbre de décomposition : témoin de cette décomposition

Exemple : Arbre de décomposition de $\sigma =$

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




Notations et propriétés :

- $\oplus$ et $\ominus$ = nœuds linéaires.
- $\pi$ simple de taille $\geq 4$ = nœud premier.
- Pas d'arête $\oplus - \oplus$ ou $\ominus - \ominus$.
- Arbres ordonnés.

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

Bijection entre permutations et leurs arbres de décomposition.

# Calcul et exemples d'utilisation

Calcul : en temps linéaire. [Uno et Yagiura 00] [Bui Xuan, Habib et Paul 05] [Bergeron, Chauve, Montgolfier et Raffinot 08]

En algorithmique :

- Recherche de motifs [Bose, Buss et Lubiw 98] [Ibarra 97]
- Algorithmes pour la bio-informatique [Bérard, Bergeron, Chauve et Paul 07] [Bérard, Chateau, Chauve, Paul et Tannier 08]

En combinatoire :

- Permutations simples [Albert, Atkinson et Klazar 03]
- Classes fermées par produit de substitution [Atkinson et Stitt 02] [Brignall 07] [Atkinson, Ruškuc et Smith 09]
- Exhiber la structure des classes [Albert et Atkinson 05] [Brignall, Huczynska et Vatter 08a,08b] [Brignall, Ruškuc et Vatter 08]

# Plan de l'exposé

# Recherche d'occurrence d'un motif

Problème *NP*-difficile :

- Entrée : $\sigma$ motif (taille $k$), $\tau$ permutation (taille $n$).
- Sortie : une occurrence de $\sigma$ dans $\tau$ si elle existe.

Restriction à $\sigma$ séparable : sous-problème polynomial.

Permutations séparables :

- Définition par motifs exclus : $S(2413, 3142)$
- Définition alternative : possédant un arbre de séparation
- Caractérisation : arbre de décomposition sans nœud premier

# Recherche d'occurrence d'un motif séparable

Programmation dynamique [Bose, Buss et Lubiw 98] [Ibarra 97]

- en suivant le guide = arbre de séparation de $\sigma$
- depuis les feuilles vers la racine
- pour des fenêtres d'indices et de valeurs

Complexité : [Bose, Buss et Lubiw 98]

- $\mathcal{O}(kn^6)$ en temps
- $\mathcal{O}(kn^4)$ en espace

[Ibarra 97]

- $\mathcal{O}(kn^4)$ en temps
- $\mathcal{O}(kn^3)$ en espace

$\Rightarrow$ Polynomial

# Généralisation avec les arbres de décomposition

Méthode :

- Programmation dynamique.
- Traitement des nœuds premiers à ajouter.

Solutions apportées : [**B.** et Rossin 06] [**B.**, Rossin et Vialette 07]

- Recherche d'occurrence d'un motif quelconque en $\mathcal{O}(kn^{2d+2})$
- Recherche de plus grand motif commun à deux permutations dont une séparable en $\mathcal{O}(\min(n_1, n_2)n_1 n_2^6)$
- Recherche de plus grand motif commun à deux permutations en $\mathcal{O}(\min(n_1, n_2)n_1 n_2^{2d_1+2})$

avec $d$ = arité maximale d'un nœud premier

# Plan de l'exposé

# Structure dans les classes de permutations

Théorème [Albert et Atkinson 05] : Si $\mathcal{C}$ contient un nombre fini de permutations simples, alors

- $\mathcal{C}$ a une base finie
- $\mathcal{C}$ a une série génératrice $(= \sum_n |\mathcal{C} \cap S_n| x^n)$ algébrique

Preuve : repose sur la décomposition par substitution.
Constructive : série génératrice calculable à partir des simples de $\mathcal{C}$

Algorithmiquement :

- procédure de semi-décision

$\hookrightarrow$ Chercher les simples de taille $4, 5, 6, \ldots$ jusqu'à $k$ et $k+1$ donnant 0 simples [Schmerl et Trotter 93]

- calcul "très exponentiel" $(\sim n!)$ des simples dans $\mathcal{C}$

# Nombre fini de permutations simples : décision

Théorème [Brignall, Ruškuc et Vatter 08] : On peut décider si $\mathcal{C}$ donnée par sa base finie contient un nombre fini de simples.

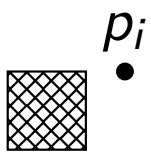
Prop. $\mathcal{C} = S(B)$ contient une infinité de simples ssi $\mathcal{C}$ contient :

1. soit une infinité de permutations parallèles
2. soit une infinité de permutations en chevrons simples
3. soit une infinité de permutations en épingles propres

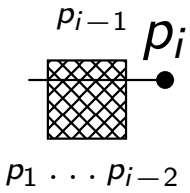
| | Procédure de décision | Complexité |
|---|---|---|
| 1. et 2. : | test d'occurrence de motifs de taille 3 ou 4 dans les $\beta \in B$. | Polynomiale |
| 3. : | Décidabilité avec des techniques de théorie des automates | Décidable 2ExpTime |

# Classes des permutations en épingles

Permutation en épingles = qui admet une représentation en épingles, *i.e.* une séquence $(p_1, \ldots, p_n)$ où chaque $p_i$ vérifie :

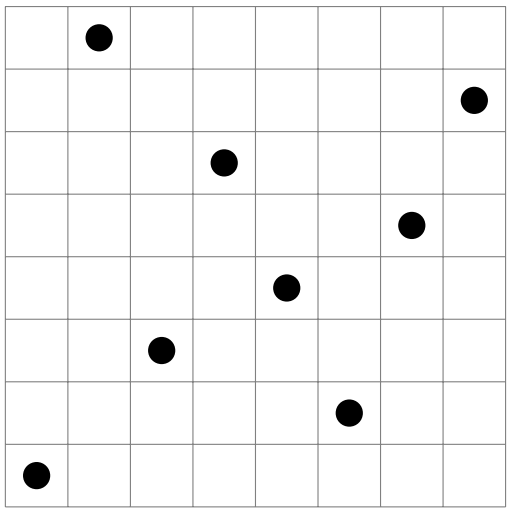
- la condition d'extériorité 

- et

- la condition de séparation 


- ou la condition d'indépendance 

Exemple :

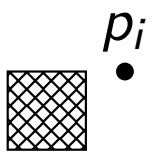


 = *boîte englobante* de $\{p_1, \ldots, p_{i-1}\}$

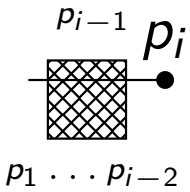
Codage par des mots d'épingles sur $\{1, 2, 3, 4, L, R, U, D\}$ avec $\begin{array}{c|c} 2 & 1 \\ \hline 3 & 4 \end{array}$

# Classes des permutations en épingles

Permutation en épingles = qui admet une représentation en épingles, *i.e.* une séquence $(p_1, \ldots, p_n)$ où chaque $p_i$ vérifie :

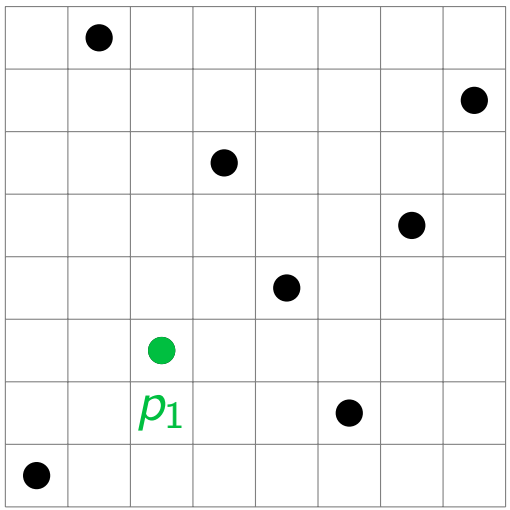
- la condition d'extériorité 

- et

- la condition de séparation 

- ou la condition d'indépendance 


Exemple :

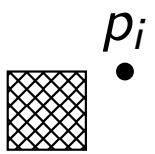



 = *boîte englobante* de $\{p_1, \ldots, p_{i-1}\}$

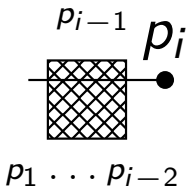
Codage par des mots d'épingles sur $\{1, 2, 3, 4, L, R, U, D\}$ avec $\begin{array}{|c|c|}\hline 2 & 1 \\ \hline 3 & 4 \\ \hline \end{array}$

# Classes des permutations en épingles

Permutation en épingles = qui admet une représentation en épingles, *i.e.* une séquence $(p_1, \ldots, p_n)$ où chaque $p_i$ vérifie :

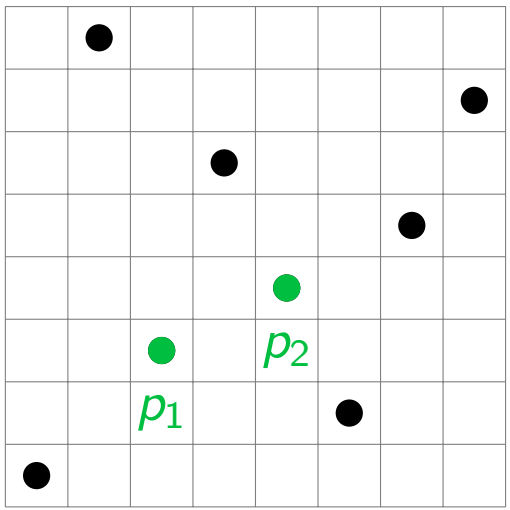
- la condition d'extériorité 

- et

- la condition de séparation 


- ou la condition d'indépendance 


Exemple :



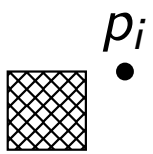

 = *boîte englobante* de $\{p_1, \ldots, p_{i-1}\}$

1

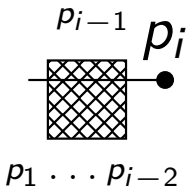
Codage par des mots d'épingles sur $\{1, 2, 3, 4, L, R, U, D\}$ avec $\begin{array}{c|c} 2 & 1 \\ \hline 3 & 4 \end{array}$

# Classes des permutations en épingles

Permutation en épingles = qui admet une représentation en épingles, *i.e.* une séquence $(p_1, \ldots, p_n)$ où chaque $p_i$ vérifie :

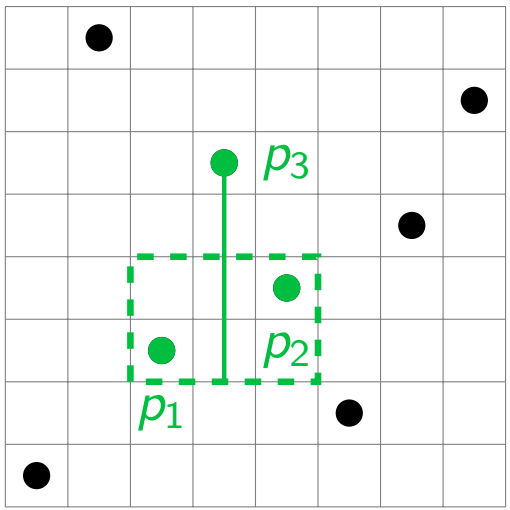
- la condition d'extériorité 

- et

- la condition de séparation 

- ou la condition d'indépendance 


Exemple :



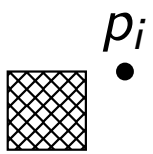

 = *boîte englobante* de $\{p_1, \ldots, p_{i-1}\}$

1 $U$

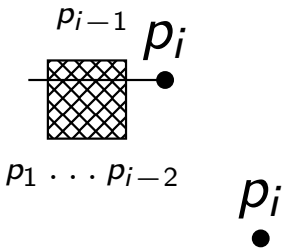
Codage par des mots d'épingles sur $\{1, 2, 3, 4, L, R, U, D\}$ avec $\begin{array}{c|c} 2 & 1 \\ \hline 3 & 4 \end{array}$

# Classes des permutations en épingles

Permutation en épingles = qui admet une représentation en épingles, *i.e.* une séquence $(p_1, \ldots, p_n)$ où chaque $p_i$ vérifie :

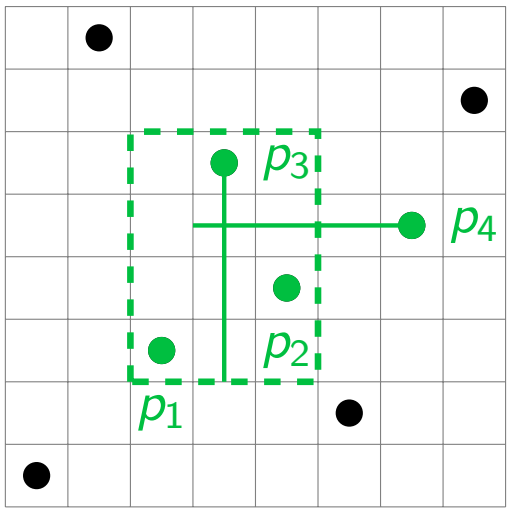
- la condition d'extériorité 

- et

- la condition de séparation 


- ou la condition d'indépendance 

Exemple :



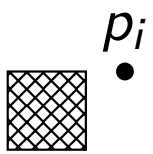

 = *boîte englobante* de $\{p_1, \ldots, p_{i-1}\}$

1 *U* *R*

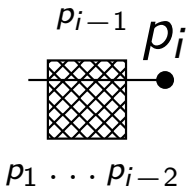
Codage par des mots d'épingles sur $\{1, 2, 3, 4, L, R, U, D\}$ avec $\begin{array}{c|c} 2 & 1 \\ \hline 3 & 4 \end{array}$

# Classes des permutations en épingles

Permutation en épingles = qui admet une représentation en épingles, *i.e.* une séquence $(p_1, \ldots, p_n)$ où chaque $p_i$ vérifie :

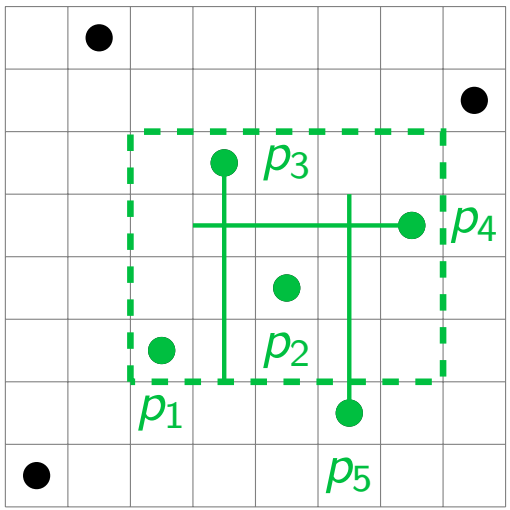
- la condition d'extériorité 

- et

- la condition de séparation 


- ou la condition d'indépendance 


Exemple :



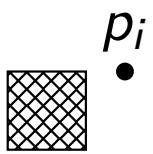

1 *U* *R* *D*

 = *boîte englobante* de $\{p_1, \ldots, p_{i-1}\}$

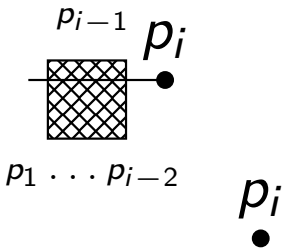
Codage par des mots d'épingles sur $\{1, 2, 3, 4, L, R, U, D\}$ avec $\begin{array}{c|c} 2 & 1 \\ \hline 3 & 4 \end{array}$

# Classes des permutations en épingles

Permutation en épingles = qui admet une représentation en épingles, *i.e.* une séquence $(p_1, \ldots, p_n)$ où chaque $p_i$ vérifie :

- la condition d'extériorité 

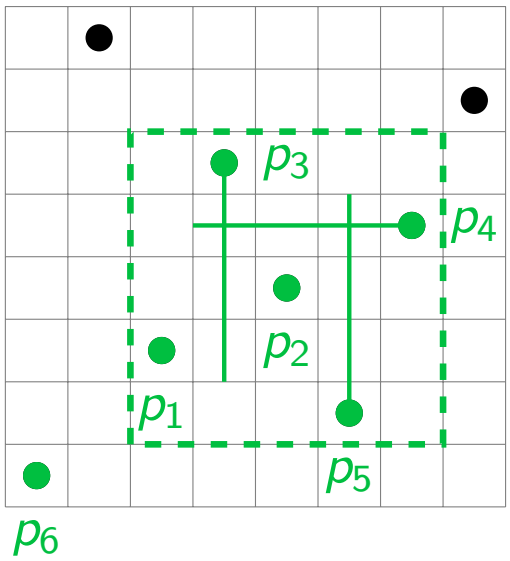
- et

- la condition de séparation 


- ou la condition d'indépendance 

 = *boîte englobante* de $\{p_1, \ldots, p_{i-1}\}$

Exemple :

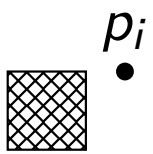



1 *U R D* 3

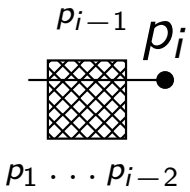
Codage par des mots d'épingles sur $\{1, 2, 3, 4, L, R, U, D\}$ avec $\begin{array}{c|c} 2 & 1 \\ \hline 3 & 4 \end{array}$

# Classes des permutations en épingles

Permutation en épingles = qui admet une représentation en épingles, *i.e.* une séquence $(p_1, \ldots, p_n)$ où chaque $p_i$ vérifie :

- la condition d'extériorité 

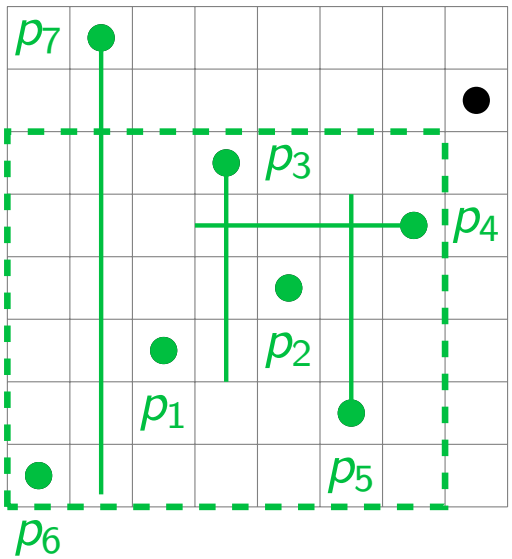
- et

- la condition de séparation 

- ou la condition d'indépendance 


 = *boîte englobante* de $\{p_1, \ldots, p_{i-1}\}$

Exemple :

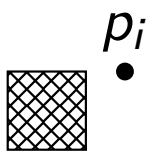



$1\ U\ R\ D\ 3\ U$

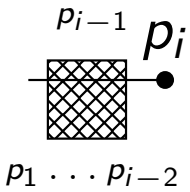
Codage par des mots d'épingles sur $\{1, 2, 3, 4, L, R, U, D\}$ avec $\begin{array}{c|c} 2 & 1 \\ \hline 3 & 4 \end{array}$

# Classes des permutations en épingles

Permutation en épingles = qui admet une représentation en épingles, *i.e.* une séquence $(p_1, \ldots, p_n)$ où chaque $p_i$ vérifie :

- la condition d'extériorité 

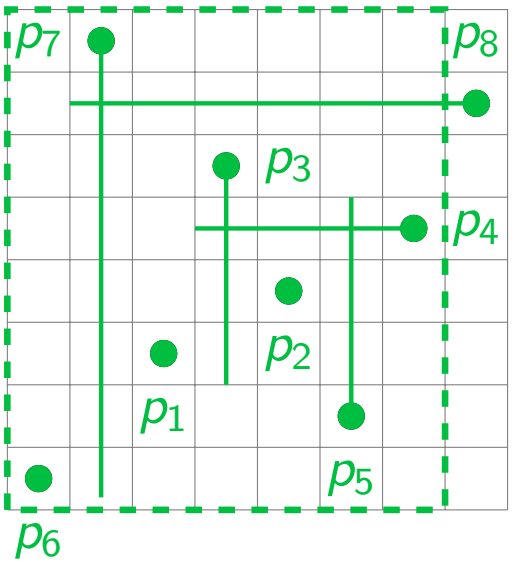
- et

- la condition de séparation 

- ou la condition d'indépendance 


 = *boîte englobante* de $\{p_1, \ldots, p_{i-1}\}$

Exemple :

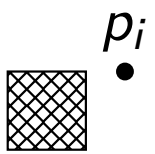



$1\ U\ R\ D\ 3\ U\ R$

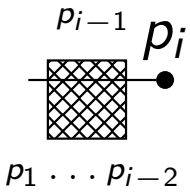
Codage par des mots d'épingles sur $\{1, 2, 3, 4, L, R, U, D\}$ avec $\begin{array}{c|c} 2 & 1 \\ \hline 3 & 4 \end{array}$

# Classes des permutations en épingles

Permutation en épingles = qui admet une représentation en épingles, *i.e.* une séquence $(p_1, \ldots, p_n)$ où chaque $p_i$ vérifie :

- la condition d'extériorité 

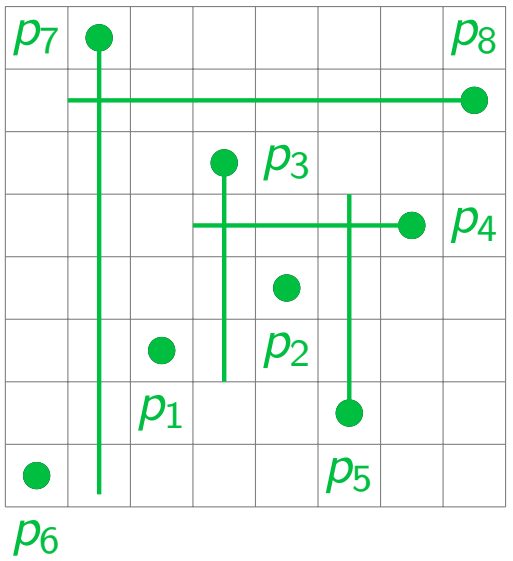
- et

- la condition de séparation 


- ou la condition d'indépendance 


 = *boîte englobante* de $\{p_1, \ldots, p_{i-1}\}$

Exemple :

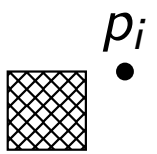



$1\ U\ R\ D\ 3\ U\ R$

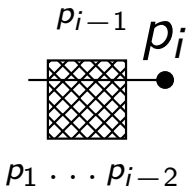
Codage par des mots d'épingles sur $\{1, 2, 3, 4, L, R, U, D\}$ avec $\begin{array}{c|c} 2 & 1 \\ \hline 3 & 4 \end{array}$

# Classes des permutations en épingles

Permutation en épingles = qui admet une représentation en épingles, *i.e.* une séquence $(p_1, \ldots, p_n)$ où chaque $p_i$ vérifie :

- la condition d'extériorité 

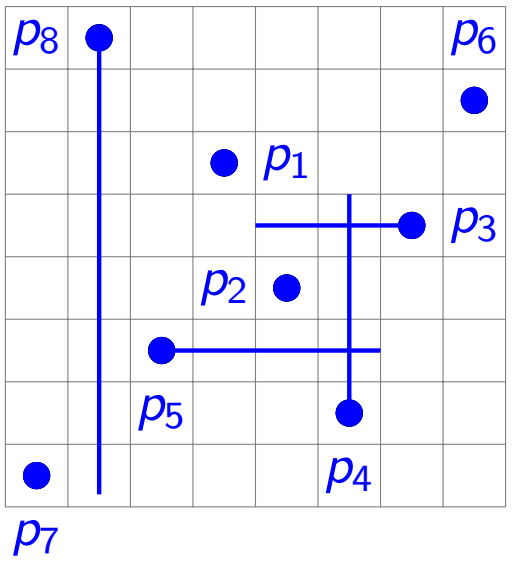
- et

- la condition de séparation 

- ou la condition d'indépendance 


 = *boîte englobante* de $\{p_1, \ldots, p_{i-1}\}$

Exemple :

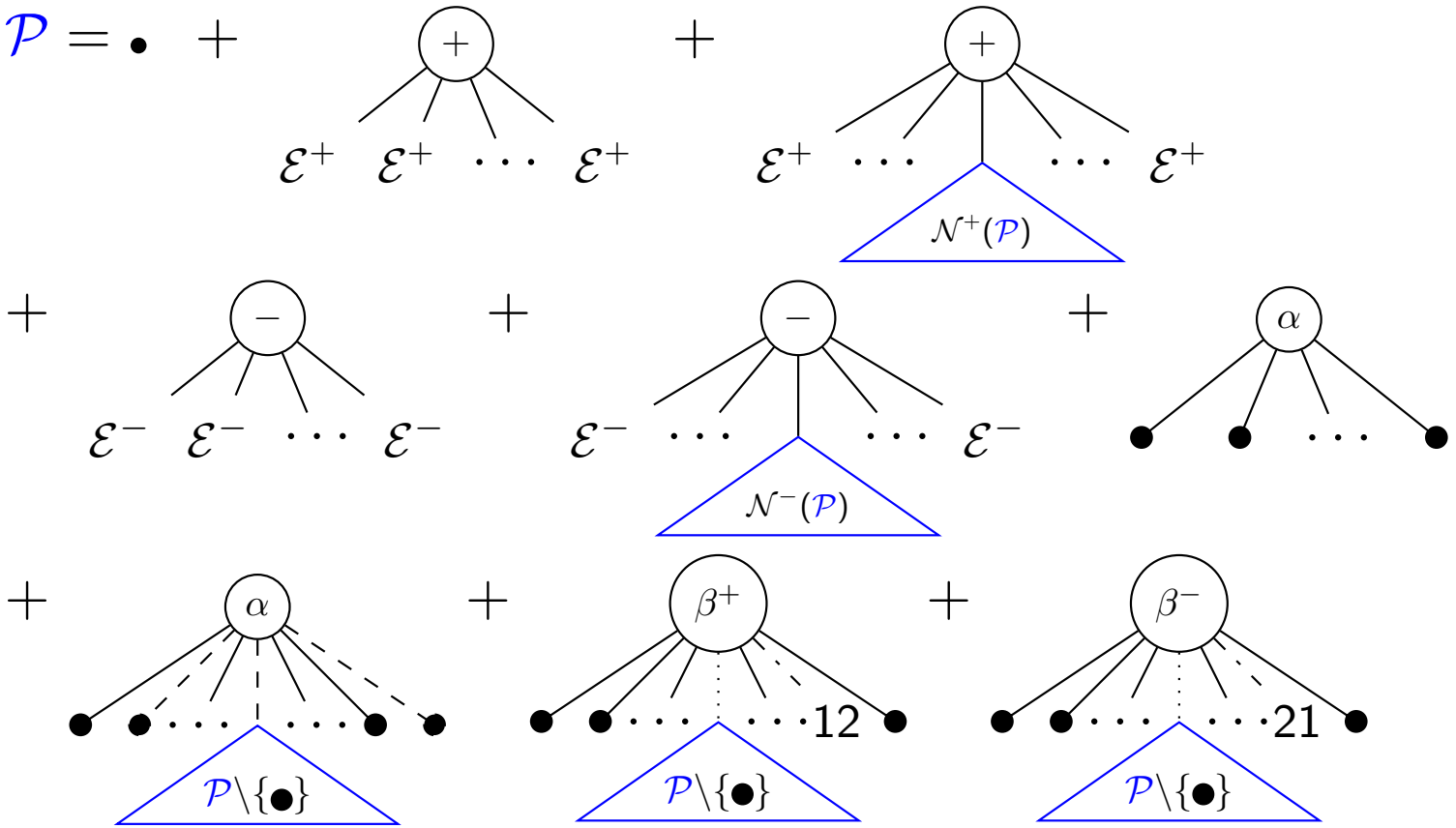



$4\,R\,D\,L\,1\,3\,U$

Codage par des mots d'épingles sur $\{1, 2, 3, 4, L, R, U, D\}$ avec $\begin{array}{c|c} 2 & 1 \\ \hline 3 & 4 \end{array}$

# Résultats sur les permutations en épingles (1/2)

- Caractérisation des arbres de décomposition [Bassino, **B.** et Rossin 09]

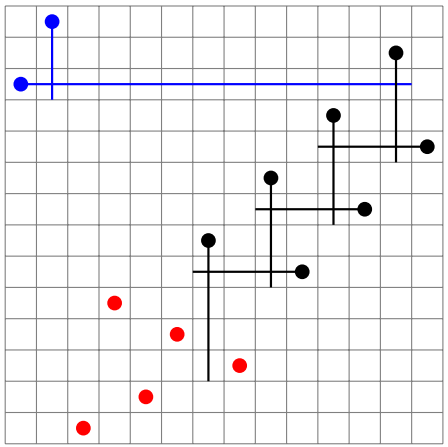


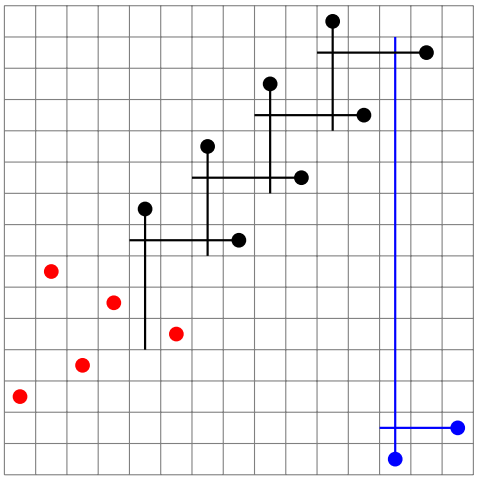
# Résultats sur les permutations en épingles (2/2)

- Calcul de la série génératrice : rationnelle [BBR09]

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

- Base infinie (qui reste à déterminer) [BBR09]





- Algorithme polynomial calculant si le nombre de simples de $S(B)$ est fini [Bassino, **B.**, Pierrot et Rossin], au lieu de la procédure de décision de [BRV08]

# Algorithme polynomial pour le nombre fini de simples

Points communs avec [BRV08] :

- Codage par des mots d'épingles sur $\{1, 2, 3, 4, L, R, U, D\}$
- Construction d'automates

Étude des permutations en épingles $\Rightarrow$ meilleure compréhension des relations entre mots d'épingles et motifs dans les permutations

Spécificités de [B**B**PR] :

- Construction polynomiale d'un automate **(déterministe, complet)** reconnaissant les mots d'épingles des permutations en épingles propres contenant un $\beta \in B$
- Langage co-fini $\Leftrightarrow$ la classe contient un nombre fini de simples
- Test de co-finitude polynomial

# Calcul automatique de la série génératrice d'une classe

Chemin parcouru :

- Décider du nombre fini de simples

↪ Polynomial

- Calculer les simples dans la classe

↪ Exponentiel

- Calculer la série génératrice (algébrique) à partir des simples

↪ Possible sur tout exemple

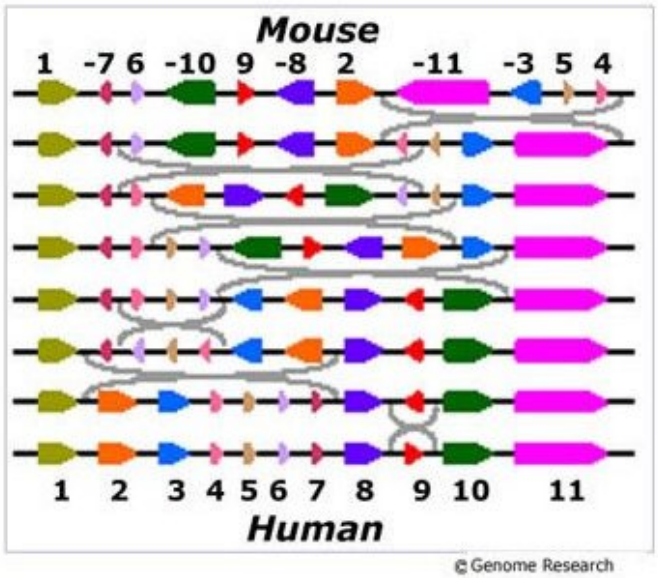
Étapes restantes :

- Calculer automatiquement la série à partir des simples
- Calcul polynomial de l'ensemble des simples dans une classe
- Si $\mathcal{C}$ n'est pas donné par sa base finie ?

# Plan de l'exposé

# Motivations et modèle

- Génomes = suites de gènes ≈ Permutations signées
- Un seul type de mutation envisagé ≈ Renversement = renverser une fenêtre en changeant les signes
- But : scénario d'évolution ≈ Séquence de renversements
- Groupe de gènes communs ≈ Intervalle des permutations




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

$$\Downarrow \text{ Renversement } \Downarrow$$

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

Contrainte supplémentaire pour le tri parfait : **ne pas casser d'intervalle**

# Tri **parfait** par renversements

- Entrée : Deux permutations signées $\sigma_1$ et $\sigma_2$
- Sortie : Un scénario parcimonieux **parfait** de $\sigma_1$ à $\sigma_2$ ou $\overline{\sigma_2}$

On peut toujours supposer que $\sigma_2 = Id = 1\ 2 \dots n$

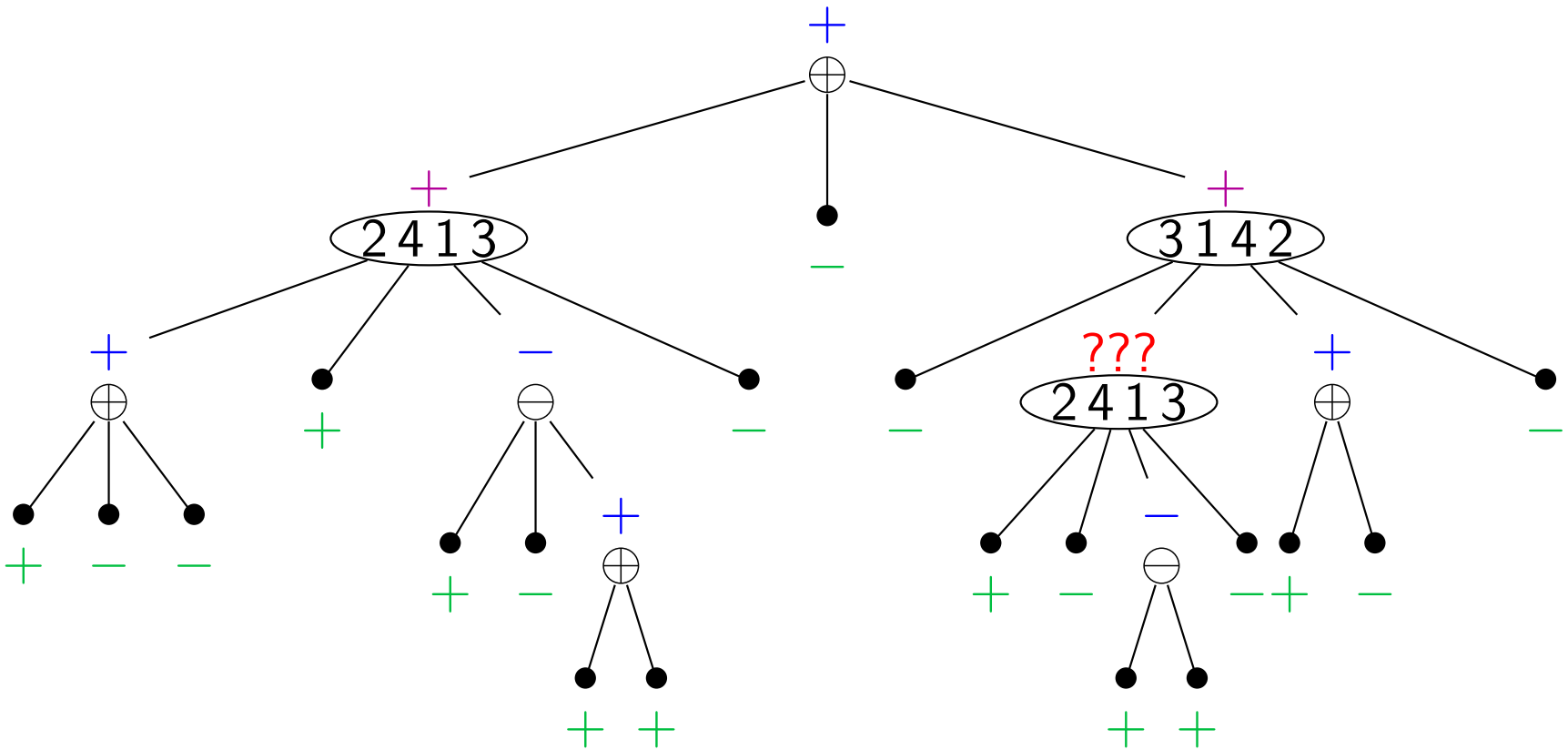
Tri par renversements : polynomial [Hannenhalli et Pevzner 99]

Tri **parfait** par renversements :

- Problème *NP*-difficile [Figeac et Varré 04]
- Algorithme FPT [Bérard, Bergeron, Chauve et Paul 07] : utilise l'arbre de décomposition, en $\mathcal{O}\left(2^p \cdot n^{\mathcal{O}(1)}\right)$
- Complexité paramétrée par
  $p$ = nombre de nœuds premiers de père premier

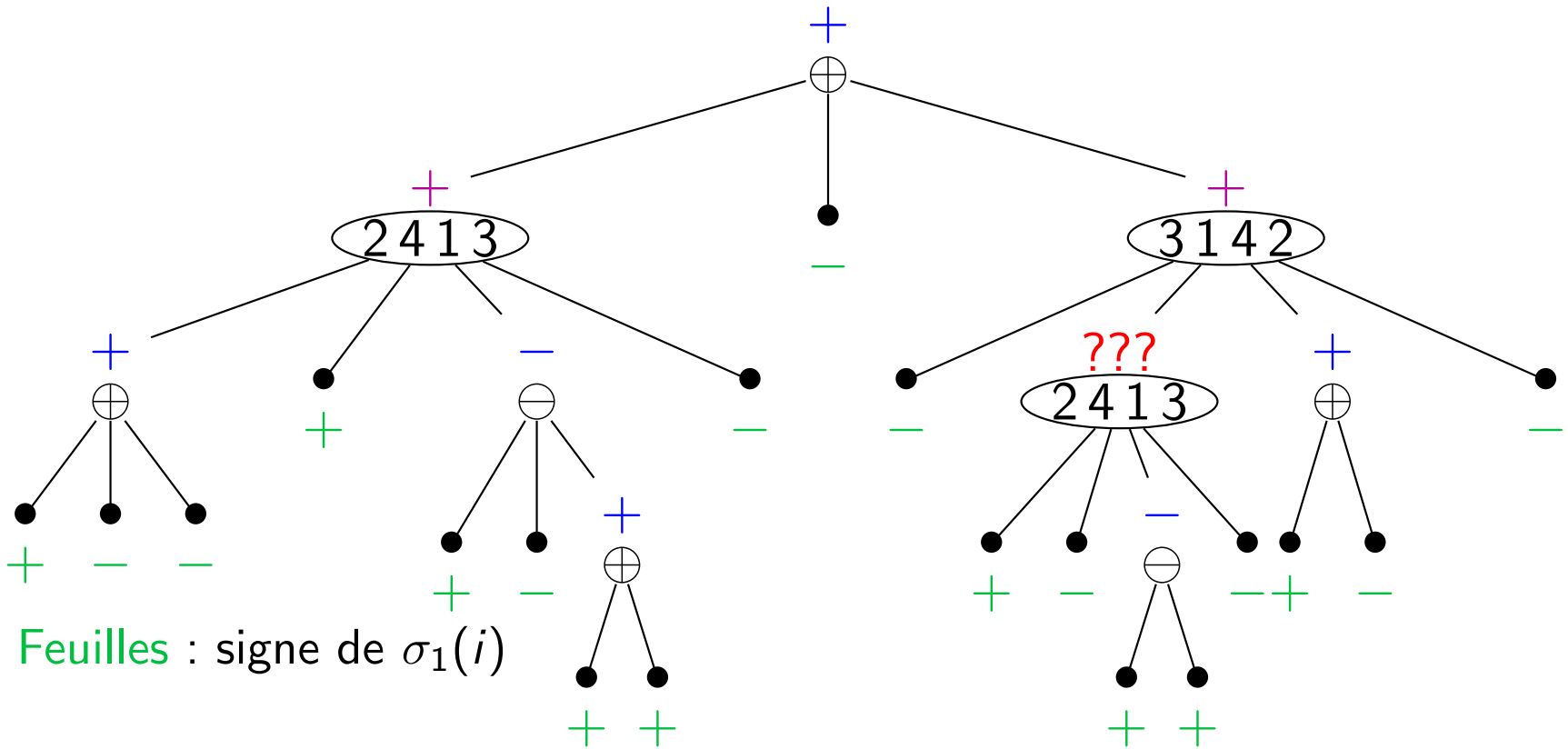
# Idée de l'algorithme sur un exemple

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



# Idée de l'algorithme sur un exemple

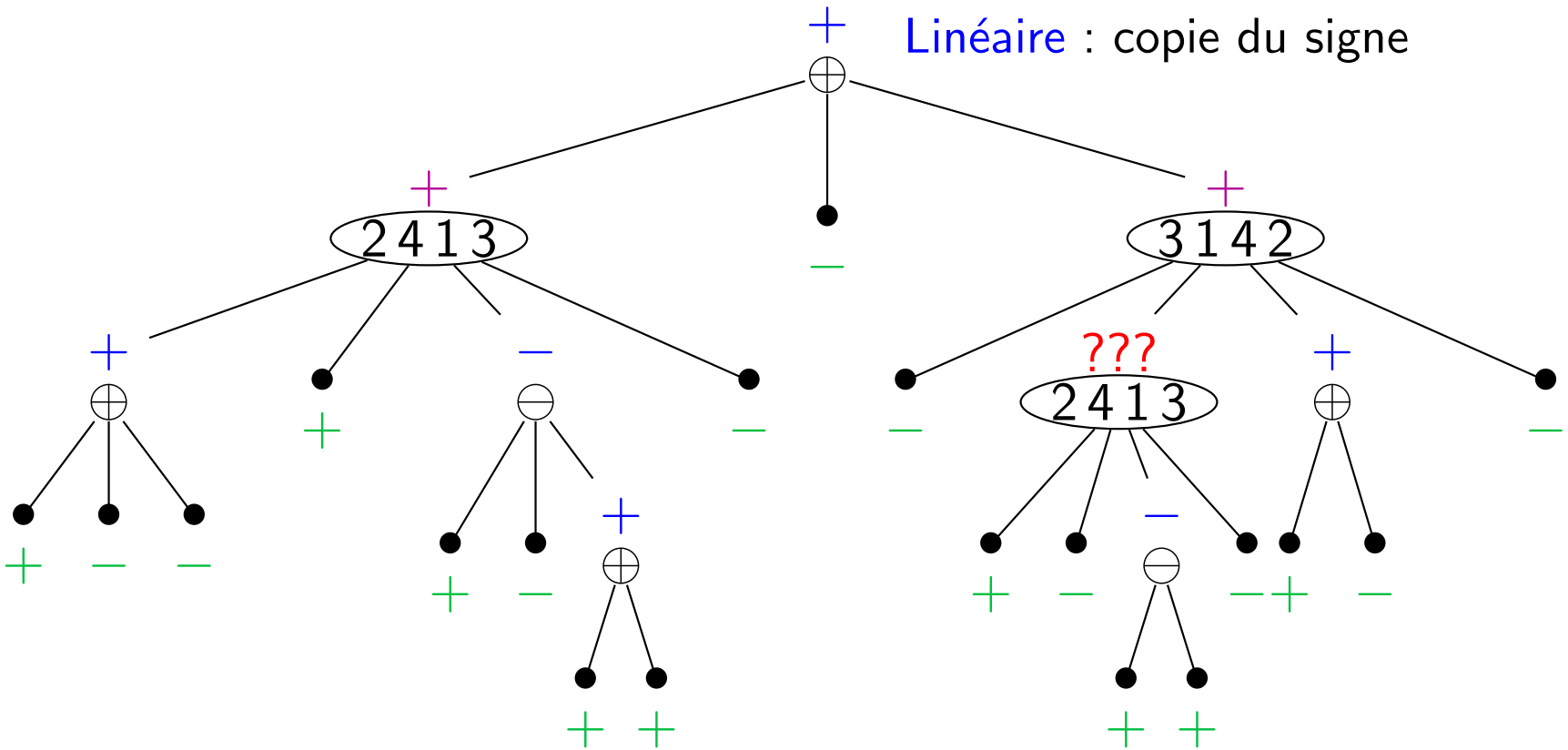
$$\sigma_1 = 5\ \overline{6}\ \overline{7}\ 9\ 4\ \overline{3}\ 1\ 2\ \overline{8}\ \overline{10}\ \overline{17}\ 13\ \overline{15}\ 12\ 11\ \overline{14}\ 18\ \overline{19}\ \overline{16}$$




Feuilles : signe de $\sigma_1(i)$

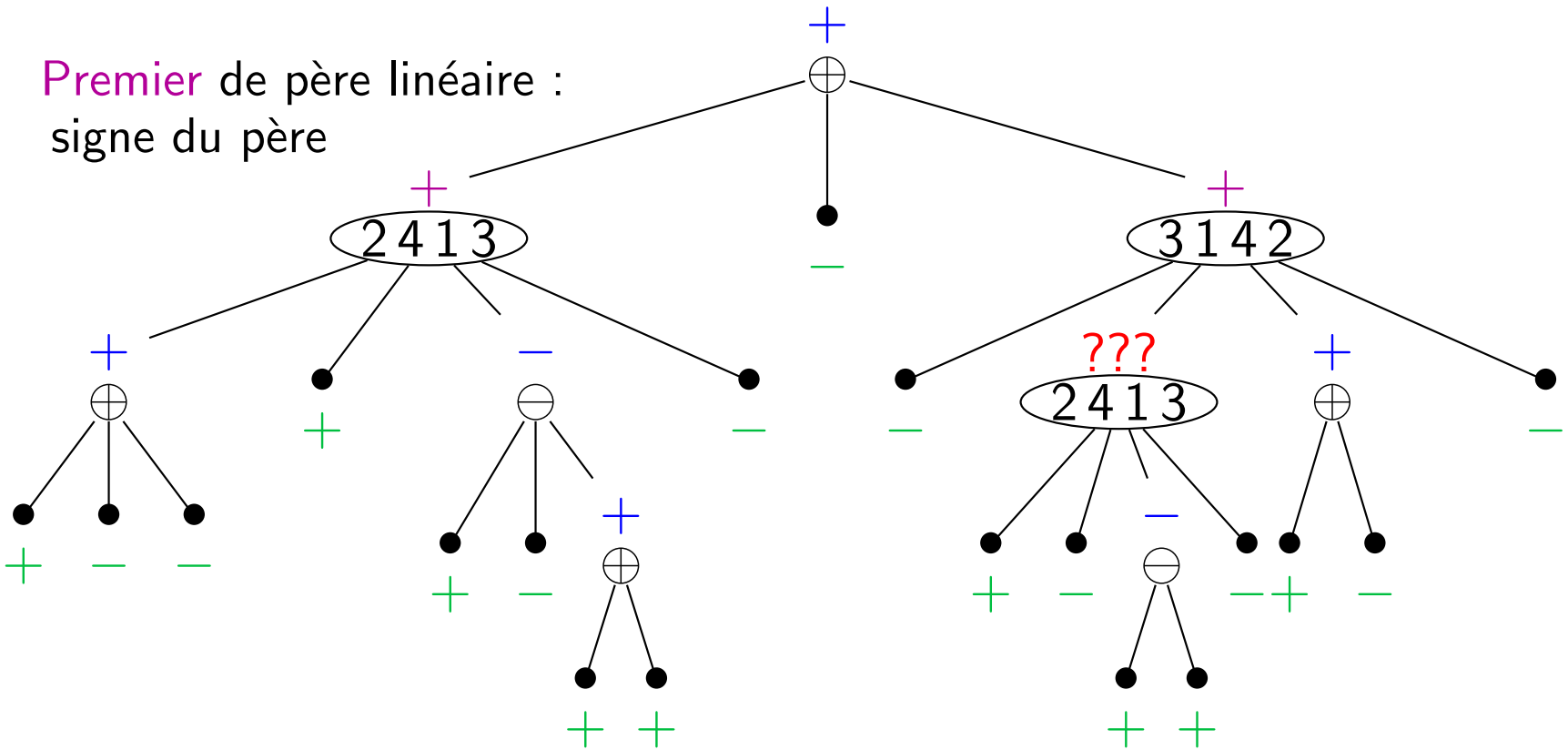
# Idée de l'algorithme sur un exemple

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



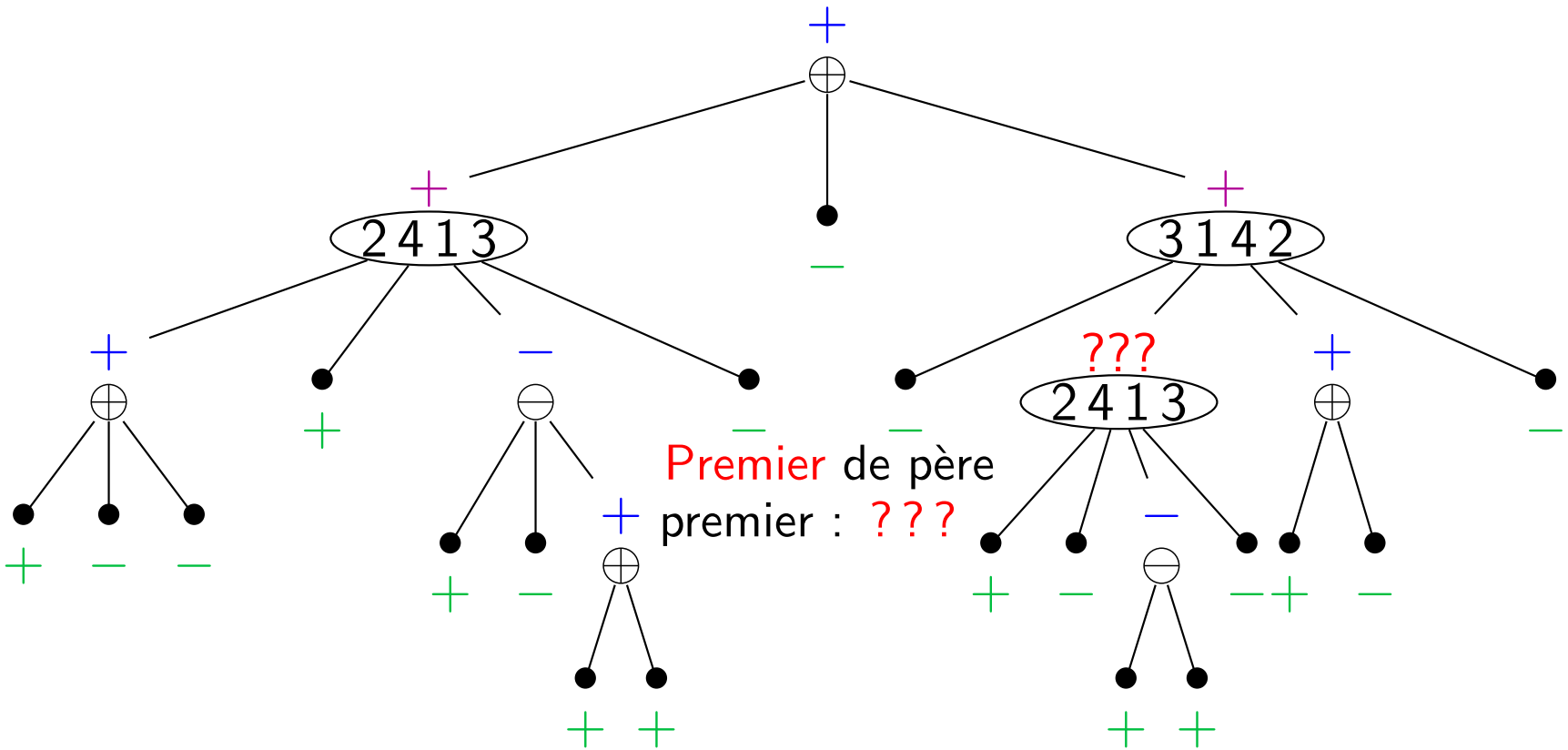
# Idée de l'algorithme sur un exemple

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



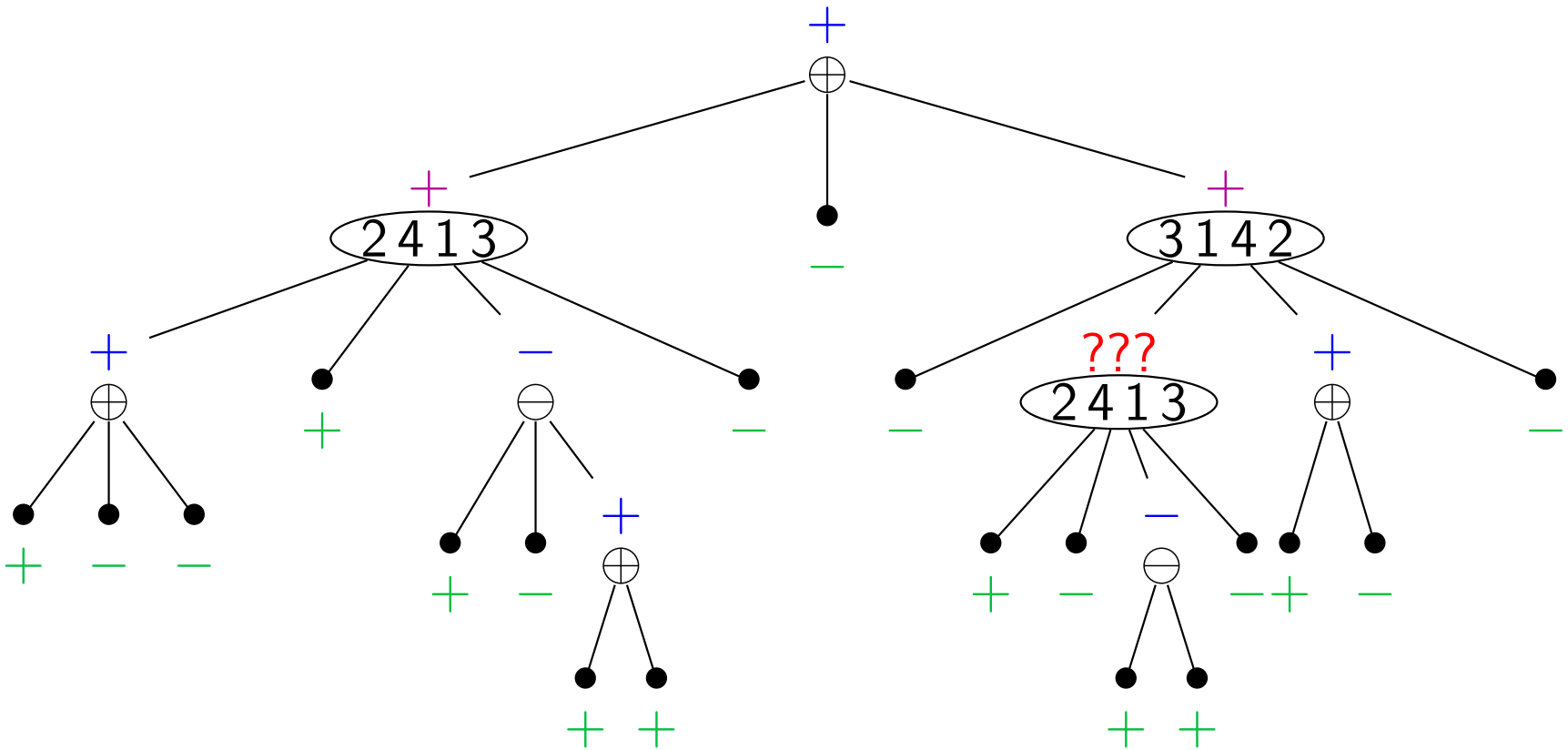
# Idée de l'algorithme sur un exemple

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



# Idée de l'algorithme sur un exemple

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



# Résultats de complexité

Résultats antérieurs [BBCP07] :

- $\mathcal{O}(2^p n\sqrt{n \log n})$, où $p =$ nombre de nœuds premiers
- polynomial sur les permutations séparables ($p = 0$)

Analyse de complexité [**B.**, Chauve, Mishna et Rossin 09] :

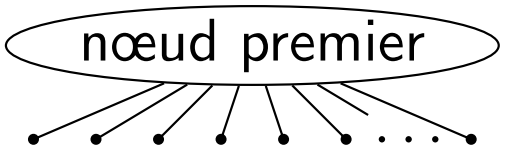
- polynomial avec probabilité 1 asymptotiquement
- polynomial en moyenne
- dans un scénario parcimonieux pour les permutations séparables
  - nombre moyen de renversements $\sim 1.2n$
  - taille moyenne d'un renversement $\sim 1.02\sqrt{n}$

Distribution de probabilité : toujours uniforme

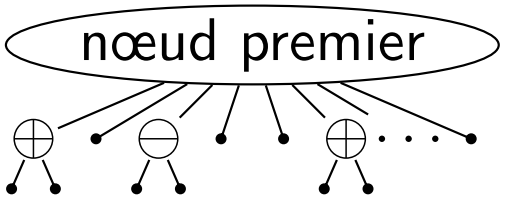
# Forme "moyenne" des arbres de décomposition

Énumération des permutations simples : asymptotiquement $\frac{n!}{e^2}$

$\Rightarrow$ Asymptotiquement, une proportion $\frac{1}{e^2}$ d'arbres de décomposition sont réduits à un nœud premier.




Théorème : Asymptotiquement, la proportion d'arbres de décomposition avec une racine première dont les fils sont des feuilles ou des cerises est 1.




cerise = nœud linéaire à deux fils, qui sont des feuilles

Conséquence : Asymptotiquement, avec probabilité 1, l'algorithme tourne en temps polynomial.

# Complexité moyenne

Complexité moyenne sur les permutations de taille $n$ :

$$\frac{\sum_{p=0}^{n} |\{\sigma \text{ à } p \text{ nœuds premiers}\}| \; C \; 2^p n\sqrt{n \log n}}{n!}$$

Théorème : Lorsque $p \geq 2$, le nombre de permutations de taille $n$ à $p$ nœuds premiers est $\leq \frac{48(n-1)!}{2^p}$

Conséquence : La complexité moyenne sur les permutations de taille $n$ est $\leq 50Cn\sqrt{n \log n}$.
En particulier, algorithme **polynomial en moyenne.**

# Paramètres pour les permutations séparables

Arbres de Schröder $\approx$ arbres de décomposition des séparables :

- Nombre moyen de nœuds internes : $\sim \frac{n}{\sqrt{2}}$
- Valeur moyenne de la somme des tailles des sous-arbres : $\sim 2^{3/4}\sqrt{3-2\sqrt{2}}\,\sqrt{\pi n^3}$

Permutations séparables **signées** :

- Nombre moyen de renversements : $\sim \frac{1+\sqrt{2}}{2} n$
- Valeur moyenne de la somme des tailles des renversements : $\sim 2^{3/4}\sqrt{3-2\sqrt{2}}\,\sqrt{\pi n^3}$
- Taille moyenne d'un renversement : $\sim \frac{2^{7/4}\sqrt{3-2\sqrt{2}}}{1+\sqrt{2}}\sqrt{\pi n} \sim 1.02\sqrt{n}$

# Plan de l'exposé

# Conclusions

Grâce aux arbres de décomposition :

- Algorithmes paramétrés pour la recherche de motifs
  - occurrence de $\sigma$ dans $\tau$
  - plus grand motif commun à $\tau_1$ et $\tau_2$ [**B**R06, **B**RV07]
- Étude combinatoire des permutations en épingles
  - exemple de classe de permutations [B**B**R09]
  - application à la détection de structure [B**B**PR]
- Analyse de complexité d'algorithmes
  - tri parfait par renversement [**B**CMR09]

Autres travaux :

- Limites de la recherche de plus grands motifs communs restreints à une classe [**B.**, Rossin et Vialette 07]
- Étude combinatoire du modèle de duplication en tandem avec perte aléatoire [**B.** et Rossin 09] [**B.** et Pergola 08]

# Perspectives

- Recherche de motif : *NP*-difficile. A-t-on existence d'un algorithme polynomial en $n$ avec prétraitement du motif ?
- Calcul des séries génératrices des $S(B)$ contenant un nombre fini de simples : quelques étapes manquantes
- Utilisation pour la génération aléatoire
- Analyse fine d'autres algorithmes utilisant des arbres de décomposition (*Double-Cut and Join*)
- Extension des concepts et résultats de théorie des graphes vers les permutations, et vice-versa

# Perspectives

- Recherche de motif : *NP*-difficile. A-t-on existence d'un algorithme polynomial en $n$ avec prétraitement du motif ?
- Calcul des séries génératrices des $S(B)$ contenant un nombre fini de simples : quelques étapes manquantes
- Utilisation pour la génération aléatoire
- Analyse fine d'autres algorithmes utilisant des arbres de décomposition (*Double-Cut and Join*)
- Extension des concepts et résultats de théorie des graphes vers les permutations, et vice-versa


Merci !

# Perspectives

- Recherche de motif : *NP*-difficile. A-t-on existence d'un algorithme polynomial en $n$ avec prétraitement du motif ?
- Calcul des séries génératrices des $S(B)$ contenant un nombre fini de simples : quelques étapes manquantes
- Utilisation pour la génération aléatoire
- Analyse fine d'autres algorithmes utilisant des arbres de décomposition (*Double-Cut and Join*)
- Extension des concepts et résultats de théorie des graphes vers les permutations, et vice-versa


Merci !