Permutations in comparative genomics

Anthony Labarre
Anthony.Labarre@cs.kuleuven.be

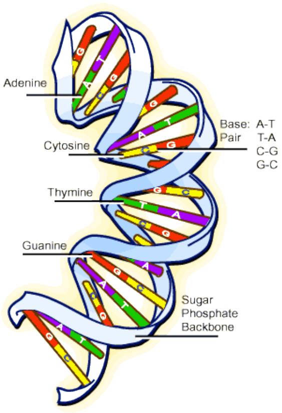
Katholieke Universiteit Leuven (K.U.L.)

February 21st, 2012

Algorithms & Permutations 2012

A few definitions

- **D**eoxyribo**n**ucleic **a**cide: double helix of *nucleotides* (A, C, G, T);
- Complementarity (A-T, C-G): one strand is enough;
- *Gene* = sequence of nucleotides (that codes for a specific protein);
- *Chromosome* = ordered set of genes;
- *Genome* = set of chromosomes;
- **Goal:** compare genomes;



Comparing genomes at the nucleotide level

- Most common comparisons: at the nucleotide level;

Example (sequence alignment)

$$\begin{array}{cccccccccccccc} S_1: & \cdots & T & C & C & G & C & C & A & - & - & C & T & A & \cdots \\ & & | & | & & | & & | & & & & | & & | & \\ S_2: & \cdots & T & C & G & G & A & C & T & G & G & C & - & A & \cdots \end{array}$$

- Matches, substitutions, insertions and deletions;
- Correspond to mutations;

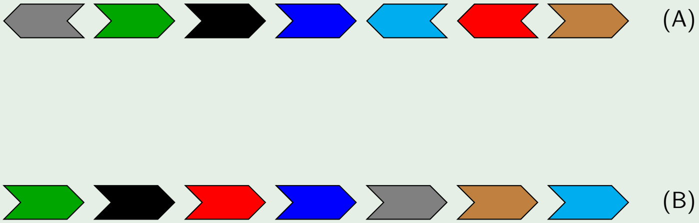
At the gene level

- Some mutations involve whole segments of nucleotides;
- Genomes = signed permutations if genomes:
  1. are totally ordered sets of genes, and
  2. only differ by gene order (no duplications, no deletions).

At the gene level

- Some mutations involve whole segments of nucleotides;
- Genomes = signed permutations if genomes:
  1. are totally ordered sets of genes, and
  2. only differ by gene order (no duplications, no deletions).

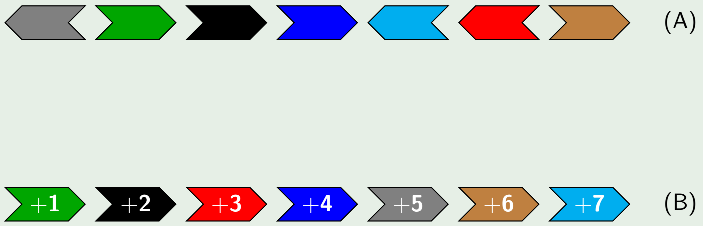
Example (genomes → signed permutations)



At the gene level

- Some mutations involve whole segments of nucleotides;
- Genomes = signed permutations if genomes:
  1. are totally ordered sets of genes, and
  2. only differ by gene order (no duplications, no deletions).

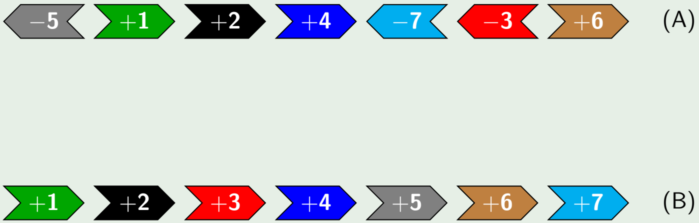
Example (genomes → signed permutations)



At the gene level

- Some mutations involve whole segments of nucleotides;
- Genomes = signed permutations if genomes:
  1. are totally ordered sets of genes, and
  2. only differ by gene order (no duplications, no deletions).

Example (genomes → signed permutations)



Comparing genomes

- Two main ways of comparing genomes:
  1. by identifying “common” or close content;
  2. by measuring their distance according to evolutionary events;
- Both approaches yield measures of (dis)similarity between permutations;
- (Many other measures are available, but they’re generally not biologically relevant (see e.g. [Estivill-Castro and Wood, 1992]));

What lies ahead

- I will give an overview of several representative problems:
  - comparing signed and unsigned permutations;
  - enumeration problems, with applications;
  - using comparisons to reconstruct evolution;
- Links with other interesting areas;
- Open problems and suggestions for future research;

Measuring similarity

- Search for segments that are “alike”;
- This is a form of approximate pattern matching:
  1. at the nucleotide level: look for subsequences that are “almost the same”;
     (this is how genomes are partitioned to yield permutations)
  2. at the gene level: look for subsequences that have exactly the same content BUT in a possibly different order;
- Point 2 motivates the next definition;

Common intervals

Definition

A **common interval** of permutations $\pi^1, \pi^2, \ldots, \pi^m$ in $S_n^{\pm}$ is a subset of $\{1, 2, \ldots, n\}$ whose elements form a substring of $\pi^1, \pi^2, \ldots, \pi^m$ (up to reordering and sign changes).

Biological motivation: genes that stuck together in an ancestor and in the present species are not likely to have been separated during evolution.

Common intervals

Definition

A **common interval** of permutations $\pi^1, \pi^2, \ldots, \pi^m$ in $S_n^{\pm}$ is a subset of $\{1, 2, \ldots, n\}$ whose elements form a substring of $\pi^1, \pi^2, \ldots, \pi^m$ (up to reordering and sign changes).

Biological motivation: genes that stuck together in an ancestor and in the present species are not likely to have been separated during evolution.

Example (<u>some</u> common intervals of two given permutations)

$$\begin{array}{llllllllll} \langle 9 & -8 & 4 & -5 & -6 & 7 & 1 & 2 & 3\rangle & \{4,5,6,7,8,9\}, \{1,2,3\} \\ \langle 1 & 2 & 3 & -8 & 7 & -4 & -5 & 6 & -9\rangle & \{4,5,8\}\text{: NO} \end{array}$$

Intervals and measures

- Other variants exist (e.g. conserved intervals);
- Measures of (dis)similarity can be built and computed efficiently [Bergeron and Stoye, 2006];
  - ✓ simple to compute;
  - ✓ biologically relevant;
  - × do not correspond to evolutionary events;
- ... which is why we'll now have a look at "event-based" measures;

Genome rearrangements

- Genomes evolve by point mutations, but also by means of mutations involving whole segments;
- **Parsimony principle** in biology: a shorter scenario of mutations is more likely;
- This motivates the study of the following problem;

Genome rearrangements

Problem (pairwise genome rearrangement problem)

Given: *permutations π, σ in $S_n^{\pm}$, and a generating set S of $S_n^{\pm}$.*
Find: *a sequence of t elements of S that:*

1. *transforms π into σ (or conversely), and*
2. *such that t is as small as possible.*

- This yields a *distance* $d_S(\cdot, \cdot)$ between permutations;

Genome rearrangements

Problem (pairwise genome rearrangement problem)

Given: *permutations π, σ in $S_n^{\pm}$, and a generating set S of $S_n^{\pm}$.*
Find: *a sequence of t elements of S that:*

1. *transforms π into σ (or conversely), and*
2. *such that t is as small as possible.*

- This yields a *distance* $d_S(\cdot, \cdot)$ between permutations;
- $d_S(\cdot, \cdot)$ is usually left-invariant, so we assume $\sigma = \iota$;

Genome rearrangements

Problem (pairwise genome rearrangement problem)

Given: *permutations π, σ in $S_n^{\pm}$, and a generating set S of $S_n^{\pm}$.*
Find: *a sequence of t elements of S that:*

1. *transforms π into σ (or conversely), and*
2. *such that t is as small as possible.*

- This yields a *distance* $d_S(\cdot,\cdot)$ between permutations;
- $d_S(\cdot,\cdot)$ is usually left-invariant, so we assume $\sigma = \iota$;
- S is closed under inverses ($x \in S \Leftrightarrow x^{-1} \in S$);

Genome rearrangements

Problem (pairwise genome rearrangement problem)

Given: *permutations* π, σ *in* $S_n^{\pm}$, *and a generating set* S *of* $S_n^{\pm}$.
Find: *a sequence of* t *elements of* S *that:*

1. *transforms* π *into* σ *(or conversely), and*
2. *such that* t *is as small as possible.*

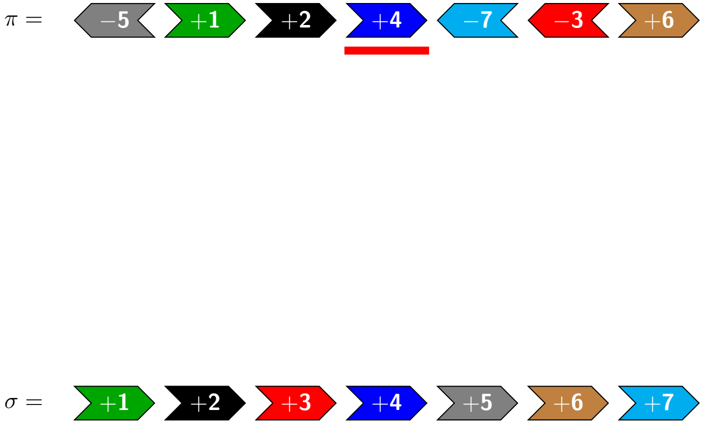
- This yields a *distance* $d_S(\cdot, \cdot)$ between permutations;
- $d_S(\cdot, \cdot)$ is usually left-invariant, so we assume $\sigma = \iota$;
- S is closed under inverses ($x \in S \Leftrightarrow x^{-1} \in S$);
- $\Rightarrow$ find a minimum-length factorisation of π into the product of elements of S;

Example: sorting by signed reversals

$\pi = \quad -5 \quad +1 \quad +2 \quad +4 \quad -7 \quad -3 \quad +6$

$\sigma = \quad +1 \quad +2 \quad +3 \quad +4 \quad +5 \quad +6 \quad +7$

Example: sorting by signed reversals



Example: sorting by signed reversals

$$\pi = \quad -5 \quad +1 \quad +2 \quad \underline{+4} \quad -7 \quad -3 \quad +6$$

$$-5 \quad +1 \quad +2 \quad -4 \quad -7 \quad -3 \quad +6$$

$$\sigma = \quad +1 \quad +2 \quad +3 \quad +4 \quad +5 \quad +6 \quad +7$$

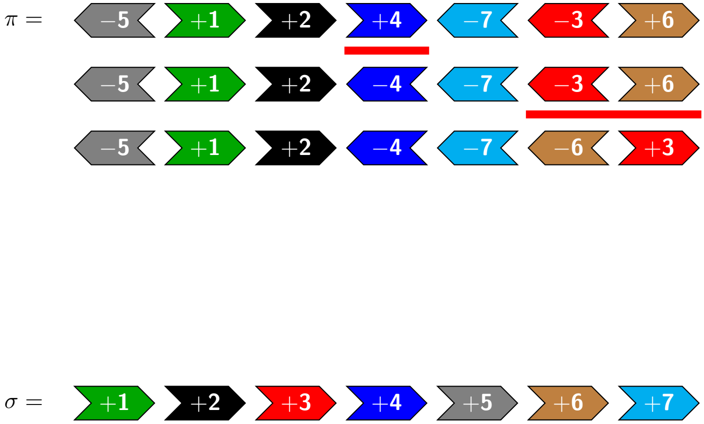
Example: sorting by signed reversals

$$\pi = \quad -5 \quad +1 \quad +2 \quad \underline{+4} \quad -7 \quad -3 \quad +6$$

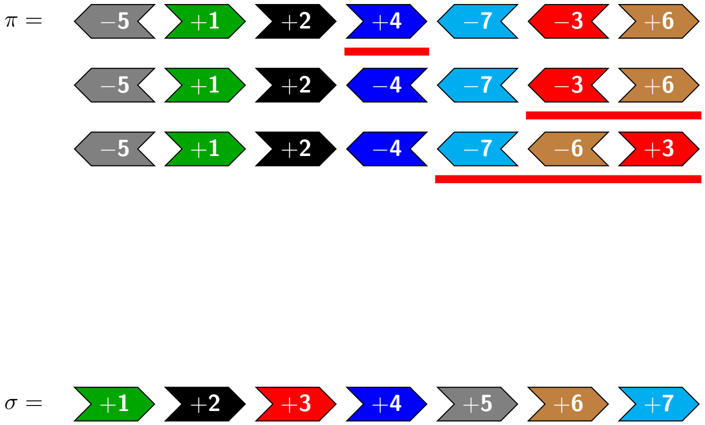
$$-5 \quad +1 \quad +2 \quad -4 \quad -7 \quad \underline{-3 \quad +6}$$

$$\sigma = \quad +1 \quad +2 \quad +3 \quad +4 \quad +5 \quad +6 \quad +7$$

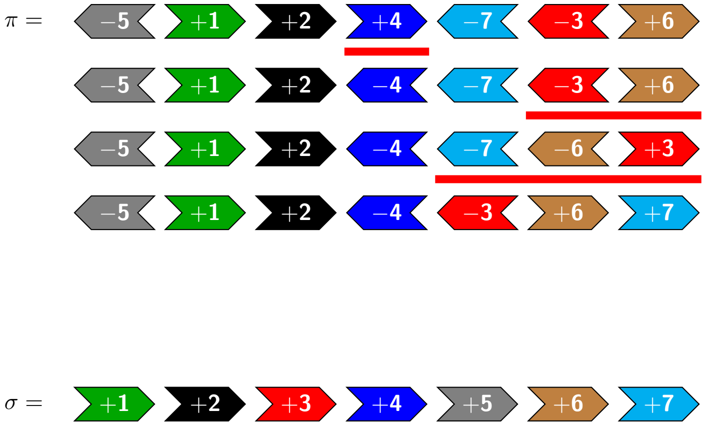
Example: sorting by signed reversals



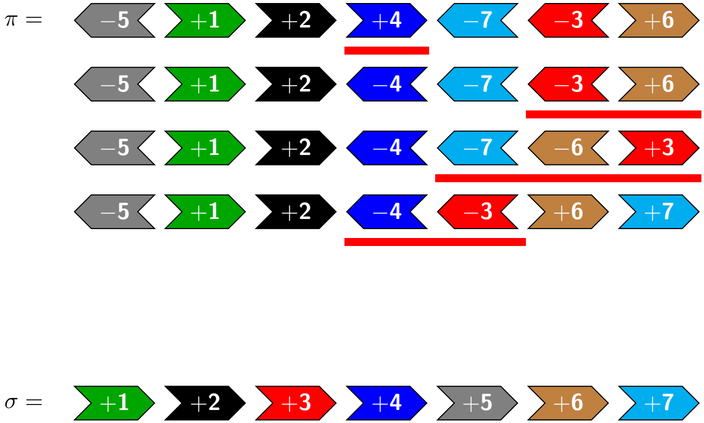
Example: sorting by signed reversals



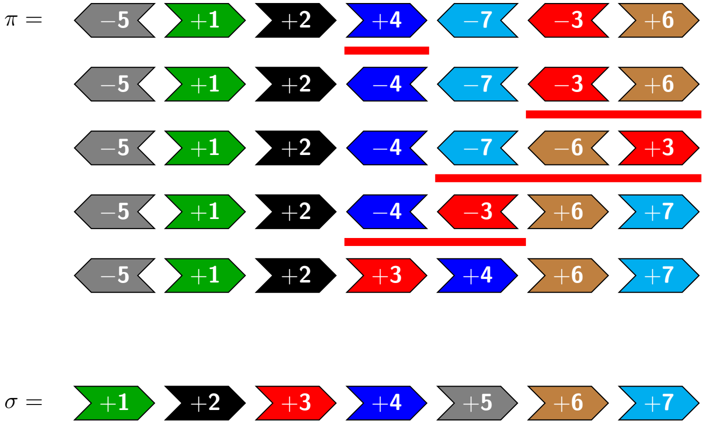
Example: sorting by signed reversals



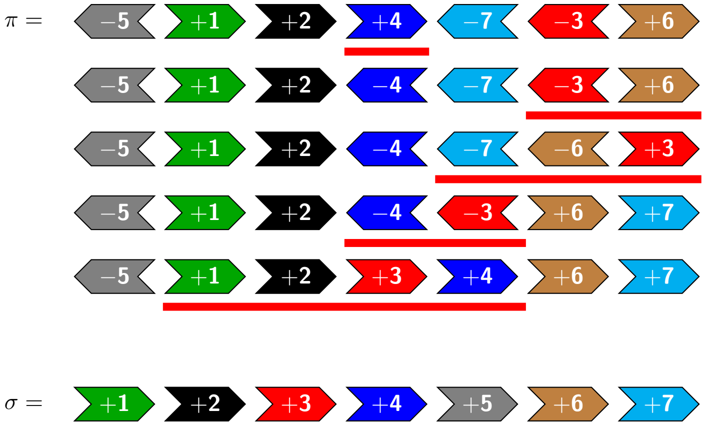
Example: sorting by signed reversals



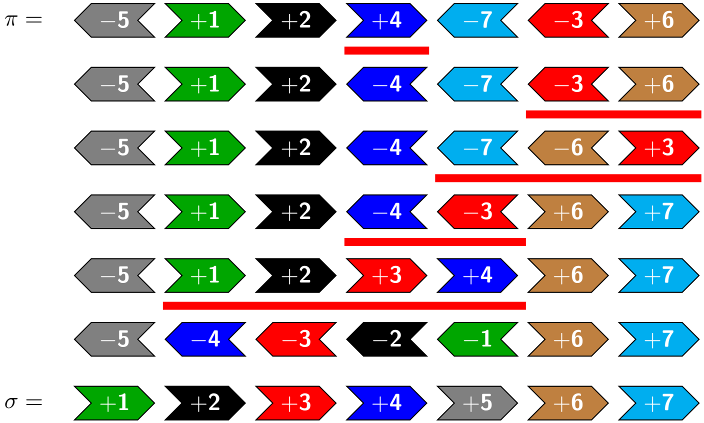
Example: sorting by signed reversals



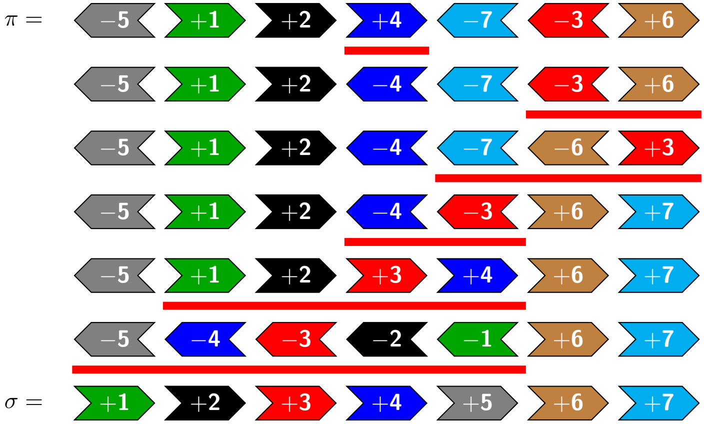
Example: sorting by signed reversals



Example: sorting by signed reversals



Example: sorting by signed reversals



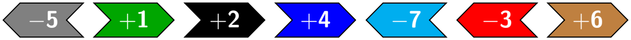
$$srd(\pi,\sigma) \leq 6$$

Computing genome rearrangement distances

- Proving upper bounds is “easy” (find a sequence that works);
- But how do we guarantee that a given sequence is optimal?
- We usually rely on variants of the *breakpoint graph*, first introduced by [Bafna and Pevzner, 1996];
- That structure proved very useful in:
  1. obtaining bounds and approximations;
  2. computing distances exactly in polynomial time;
  3. proving complexity results;

The *breakpoint graph*

- Going back to our example:

$\pi =$ 

The *breakpoint graph*

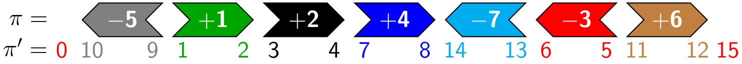
- Going back to our example:

$$\pi = \quad -5 \quad +1 \quad +2 \quad +4 \quad -7 \quad -3 \quad +6$$
$$\pi' = 0 \quad 10 \quad 9 \quad 1 \quad 2 \quad 3 \quad 4 \quad 7 \quad 8 \quad 14 \quad 13 \quad 6 \quad 5 \quad 11 \quad 12 \quad 15$$

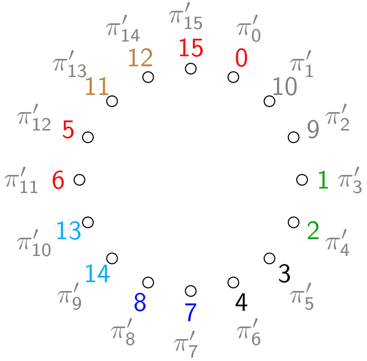
1. double π's elements $(i \mapsto \{2|i|-1, 2|i|\})$ and add $\mathbf{0}$ and $\mathbf{2n+1}$

The *breakpoint graph*

- Going back to our example:

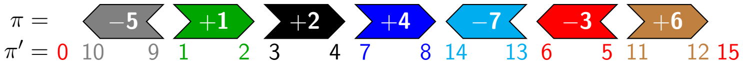


1. double π's elements ($i \mapsto \{2|i|-1, 2|i|\}$) and add **0** and $\mathbf{2n+1}$
2. elements of π' = vertices

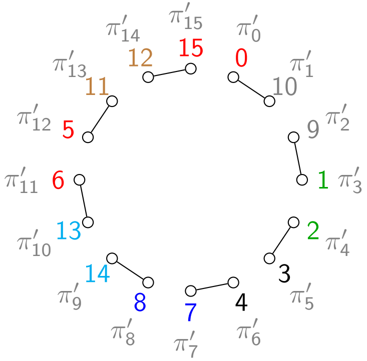

π'_{14} π'_{15} π'_0
π'_{13} 12 15 0 π'_1
11 10
π'_{12} 5 9 π'_2
π'_{11} 6 1 π'_3
π'_{10} 13 2 π'_4
14 3
π'_9 8 7 4 π'_5
π'_8 π'_7 π'_6

The *breakpoint graph*

- Going back to our example:

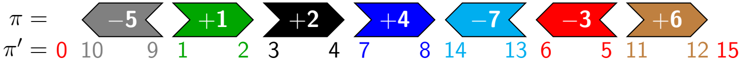


1. double π's elements ($i \mapsto \{2|i|-1, 2|i|\}$) and add **0** and **2n + 1**
2. elements of π' = vertices
3. **black edges** connect distinct adjacent genes

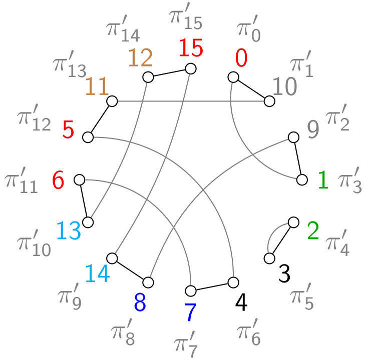


The *breakpoint graph*

- Going back to our example:

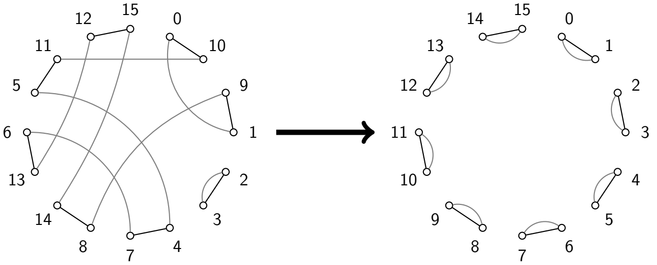


1. double π's elements $(i \mapsto \{2|i|-1, 2|i|\})$ and add **0** and $\mathbf{2n+1}$
2. elements of π' = vertices
3. **black edges** connect distinct adjacent genes
4. **grey edges** connect distinct consecutive genes



Using the breakpoint graph

- The breakpoint graph is 2-regular and decomposes as such into **alternating cycles** in a unique way;
- The breakpoint graph of $\langle 1\ 2\ \cdots\ n \rangle$ contains the largest number of cycles:



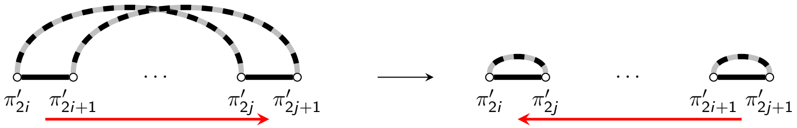
- ⇒ goal: create new cycles in as few moves as possible;

A lower bound on the signed reversal distance

- A signed reversal involves black edges belonging to at most two cycles;

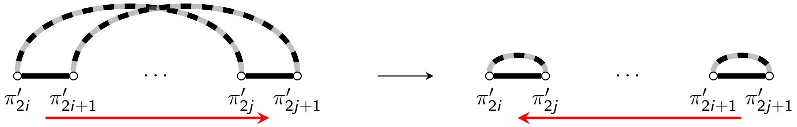
A lower bound on the signed reversal distance

- A signed reversal involves black edges belonging to at most two cycles;
- The only way to increase $c(BG(\pi))$ is to split cycles:



A lower bound on the signed reversal distance

- A signed reversal involves black edges belonging to at most two cycles;
- The only way to increase $c(BG(\pi))$ is to split cycles:



- Therefore, for all π in $S_n^{\pm}$:

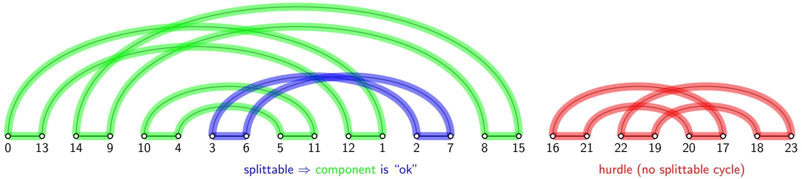
$$srd(\pi) \geq n + 1 - c(BG(\pi)).$$

An exact formula for computing the signed reversal distance

- It is not always possible to split cycles;

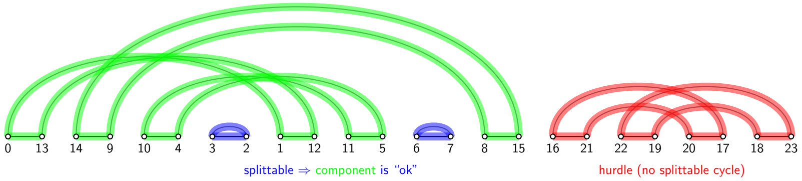
An exact formula for computing the signed reversal distance

- It is not always possible to split cycles;
- Worse: some collections of cycles in the breakpoint graph, called hurdles, each require an additional move;



An exact formula for computing the signed reversal distance

- It is not always possible to split cycles;
- Worse: some collections of cycles in the breakpoint graph, called hurdles, each require an additional move;



An exact formula for computing the signed reversal distance

- It is not always possible to split cycles;
- Worse: some collections of cycles in the breakpoint graph, called hurdles, each require an additional move;

Theorem ([Hannenhalli and Pevzner, 1999])

For all π in $S_n^{\pm}$:

$$srd(\pi) = n + 1 - c(BG(\pi)) + \underbrace{h(BG(\pi))}_{\text{number of hurdles}} + \underbrace{f(BG(\pi))}_{\text{special ``fortress'' case}} .$$

An exact formula for computing the signed reversal distance

- It is not always possible to split cycles;
- Worse: some collections of cycles in the breakpoint graph, called hurdles, each require an additional move;

Theorem ([Hannenhalli and Pevzner, 1999])

For all π in $S_n^{\pm}$:

$$srd(\pi) = n + 1 - c(BG(\pi)) + \underbrace{h(BG(\pi))}_{\text{number of hurdles}} + \underbrace{f(BG(\pi))}_{\text{special ``fortress'' case}} .$$

- Computing $srd(\cdot)$ / sorting can be done in polynomial time;

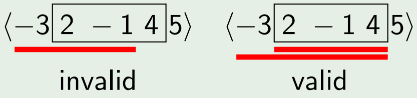
Integrating other constraints: “perfection”

- (recall Mathilde Bouvel’s talk)
- As seen before, some genes “stick together”;

Problem (perfect sorting by signed reversals)

Given: *a permutation π in $S_n^{\pm}$, and a set S of intervals of π.*
Find: *a minimum-length sequence of signed reversals that sorts π and whose elements do not overlap with intervals from S.*

Example (<u>not</u> sorting sequences)


$\langle -3\ \boxed{2\ -1\ 4}\ 5\rangle$
invalid
$\langle -3\ \boxed{2\ -1\ 4}\ 5\rangle$
valid

A more general view: double cut-and-join

- The “double-cut-and-join” (DCJ) operation is defined directly on the breakpoint graph:
- Idea: **cut** two black edges and **join** their endpoints;
    - Simulates signed reversals and block-interchanges (2 DCJs for the latter);
    - $\Rightarrow$ sorting by DCJs $\equiv$ sorting by signed reversals with weight 1 and block-interchanges with weight 2;

Theorem ([Yancopoulos et al., 2005])

For any π in $S_n^{\pm}$: $dcj(\pi) = n + 1 - c(BG(\pi))$.

Some results and some questions (non exhaustive)

- What has been done:

Operation	Sorting	Distance	Best approximation
signed reversal	$O(n^{3/2})$ [Han, 2006]	$O(n)$ [Bader et al., 2001]	1
perfect signed reversal	NP-hard [Figeac and Varré, 2004]		?
prefix signed reversal	?	?	2 [Cohen and Blum, 1995]
double cut and join	$O(n)$ [Yancopoulos et al., 2005]		1

- What could be done:
  - Prefix signed reversals:
    1. complexity of sorting / computing the distance?
    2. largest value the distance can reach?
    3. "better-than-2"-approximation?
  - Characterise "hard instances" using pattern matching/avoidance;

Unsigned permutations

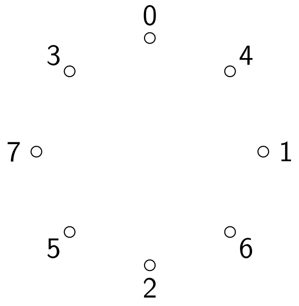
- In some situations, gene orientation may be unknown or can be disregarded;
- Genomes are then modelled by the more traditional unsigned permutations;
- Of course, rearrangements in that setting do not affect orientation;

The breakpoint graph in the unsigned case

- More direct construction than in the signed case;
- Let's build the breakpoint graph of $\pi = \langle 4\ 1\ 6\ 2\ 5\ 7\ 3 \rangle$:

The breakpoint graph in the unsigned case

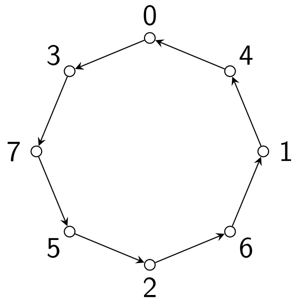
- More direct construction than in the signed case;
- Let's build the breakpoint graph of $\pi = \langle 4\ 1\ 6\ 2\ 5\ 7\ 3 \rangle$:



1. build the ordered vertex set $(\pi_0 = 0, \pi_1, \pi_2, \ldots, \pi_n)$;

The breakpoint graph in the unsigned case

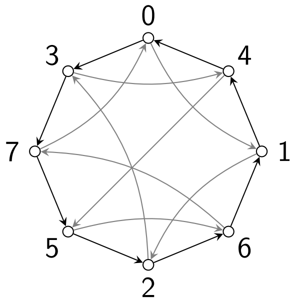
- More direct construction than in the signed case;
- Let's build the breakpoint graph of $\pi = \langle 4\ 1\ 6\ 2\ 5\ 7\ 3 \rangle$:



1. build the ordered vertex set $(\pi_0 = 0, \pi_1, \pi_2, \ldots, \pi_n)$;
2. add **black arcs** for every ordered pair $(\pi_i, \pi_{i-1 \pmod{n+1}})$;

The breakpoint graph in the unsigned case

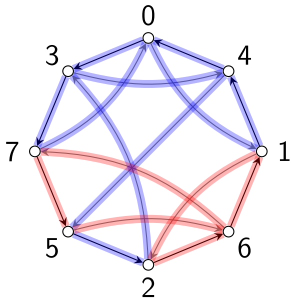
- More direct construction than in the signed case;
- Let's build the breakpoint graph of $\pi = \langle 4\ 1\ 6\ 2\ 5\ 7\ 3 \rangle$:



1. build the ordered vertex set $(\pi_0 = 0, \pi_1, \pi_2, \ldots, \pi_n)$;
2. add **black arcs** for every ordered pair $(\pi_i, \pi_{i-1 \pmod{n+1}})$;
3. add **grey arcs** for every ordered pair $(i, i+1 \pmod{n+1})$;

The breakpoint graph in the unsigned case

- More direct construction than in the signed case;
- Let's build the breakpoint graph of $\pi = \langle 4\ 1\ 6\ 2\ 5\ 7\ 3 \rangle$:



1. build the ordered vertex set ($\pi_0 = 0, \pi_1, \pi_2, \ldots, \pi_n$);
2. add **black arcs** for every ordered pair ($\pi_i, \pi_{i-1 \pmod{n+1}}$);
3. add **grey arcs** for every ordered pair $(i, i+1 \pmod{n+1})$;

$BG(\pi)$ decomposes in a unique way into alternating cycles

Breakpoint graphs as permutations

- Nice property of $BG(\pi)$: its alternating cycle decomposition matches the traditional disjoint cycle decomposition of

$$\overline{\pi} = (0, 1, 2, \ldots, n) \circ (0, \pi_n, \pi_{n-1}, \ldots, \pi_1);$$

- As a consequence, we can express the action of *any* rearrangement σ on π using $\overline{\pi}$:

Lemma ([Labarre, 2012])

For all π, σ in S_n, we have $\overline{\pi \circ \sigma} = \overline{\pi} \circ \overline{\sigma}^{\pi}$.

- ... and we can recycle known results in this context;

Obtaining lower bounds

- Lower bounds can be automatically obtained as follows:

Initial problem:
computing $d_S(\pi)$:

$$\pi \longrightarrow \pi \circ x_1 \longrightarrow \pi \circ x_1 \circ x_2 \longrightarrow \cdots \longrightarrow \iota$$

Obtaining lower bounds

- Lower bounds can be automatically obtained as follows:

Initial problem:
computing $d_S(\pi)$:

New problem:
computing $d_C(\overline{\pi})$:

$$\begin{tikzcd}
\pi \arrow[r] \arrow[d, bend right] & \pi \circ x_1 \arrow[r] & \pi \circ x_1 \circ x_2 \arrow[r, dash] & \cdots \arrow[r] & \iota \arrow[d, bend left] \\
\overline{\pi} & & & & \overline{\iota} = \iota
\end{tikzcd}$$

Obtaining lower bounds

- Lower bounds can be automatically obtained as follows:

Initial problem:
computing $d_S(\pi)$:

$\boxed{d_C(\overline{\pi}) \leq d_S(\pi)}$

New problem:
computing $d_C(\overline{\pi})$:

$$\begin{tikzcd}
\pi \arrow[r] \arrow[d, bend right] & \pi \circ x_1 \arrow[r] & \pi \circ x_1 \circ x_2 \arrow[r, dash] & \cdots \arrow[r] & \iota \arrow[d, bend left] \\
\overline{\pi} \arrow[r] & \overline{\pi \circ y_1} \arrow[r] & \overline{\pi \circ y_1 \circ y_2} \arrow[r, dash] & \cdots \arrow[r] & \overline{\iota} = \iota
\end{tikzcd}$$

Obtaining lower bounds

- Lower bounds can be automatically obtained as follows:

Initial problem:
computing $d_S(\pi)$:

$\boxed{d_{\mathcal{C}}(\overline{\pi}) \leq d_S(\pi)}$

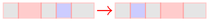
New problem:
computing $d_{\mathcal{C}}(\overline{\pi})$:

$$\begin{tikzcd}
\pi \arrow[r] \arrow[d, bend right] & \pi \circ x_1 \arrow[r] & \pi \circ x_1 \circ x_2 \arrow[r, dash] & \cdots \arrow[r] & \iota \arrow[d, bend left] \\
\overline{\pi} \arrow[r] & \overline{\pi \circ y_1} \arrow[r] & \overline{\pi \circ y_1 \circ y_2} \arrow[r, dash] & \cdots \arrow[r] & \overline{\iota} = \iota
\end{tikzcd}$$

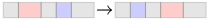
- This yields tight lower bounds on:
  1. the block-interchange distance [Christie, 1996];
  2. the transposition distance [Bafna and Pevzner, 1998];
  3. the prefix transposition distance [Labarre, 2012];

Three examples

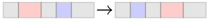
- How one can obtain bounds on the following distances:
  1. block-interchange distance ($bid(\cdot)$):



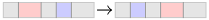
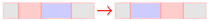
Three examples

- How one can obtain bounds on the following distances:
  1. block-interchange distance ($bid(\cdot)$):

     1. $\overline{\text{block-interchange}}$ = pair of 2-cycles;

Three examples

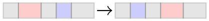
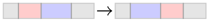
- How one can obtain bounds on the following distances:
  1. block-interchange distance ($bid(\cdot)$):
     
     1. $\overline{\text{block-interchange}}$ = pair of 2-cycles;
     2. $bid(\pi) \geq d_{\mathcal{C}}(\overline{\pi}) = \frac{n+1-c(BG(\pi))}{2}$;

Three examples

- How one can obtain bounds on the following distances:
  1. block-interchange distance ($bid(\cdot)$): 
     1. $\overline{\text{block-interchange}}$ = pair of 2-cycles;
     2. $bid(\pi) \geq d_{\mathcal{C}}(\overline{\pi}) = \frac{n+1-c(BG(\pi))}{2}$;
  2. transposition distance ($td(\cdot)$): 

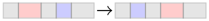
Three examples

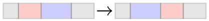
- How one can obtain bounds on the following distances:
  1. block-interchange distance $(bid(\cdot))$:
     1. $\overline{\text{block-interchange}}$ = pair of 2-cycles;
     2. $bid(\pi) \geq d_{\mathcal{C}}(\overline{\pi}) = \frac{n+1-c(BG(\pi))}{2}$;
  2. transposition distance $(td(\cdot))$:
     1. $\overline{\text{transposition}}$ = 3-cycle;

Three examples

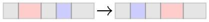
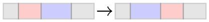
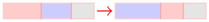
- How one can obtain bounds on the following distances:
  1. block-interchange distance ($bid(\cdot)$):
     1. $\overline{\text{block-interchange}}$ = pair of 2-cycles;
     2. $bid(\pi) \geq d_{\mathcal{C}}(\overline{\pi}) = \frac{n+1-c(BG(\pi))}{2}$;
  2. transposition distance ($td(\cdot)$):
     1. $\overline{\text{transposition}}$ = 3-cycle;
     2. $td(\pi) \geq d_{\mathcal{C}}(\overline{\pi}) = \frac{n+1-c_{odd}(BG(\pi))}{2}$;



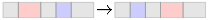
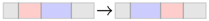
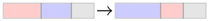


Three examples

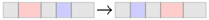
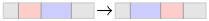
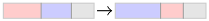
- How one can obtain bounds on the following distances:
  1. block-interchange distance ($bid(\cdot)$):
     1. $\overline{\text{block-interchange}}$ = pair of 2-cycles;
     2. $bid(\pi) \geq d_{\mathcal{C}}(\overline{\pi}) = \frac{n+1-c(BG(\pi))}{2}$;
  2. transposition distance ($td(\cdot)$):
     1. $\overline{\text{transposition}}$ = 3-cycle;
     2. $td(\pi) \geq d_{\mathcal{C}}(\overline{\pi}) = \frac{n+1-c_{odd}(BG(\pi))}{2}$;
  3. prefix transposition distance ($ptd(\cdot)$):

Three examples

- How one can obtain bounds on the following distances:
  1. block-interchange distance ($bid(\cdot)$):
     
     1. $\overline{\text{block-interchange}}$ = pair of 2-cycles;
     2. $bid(\pi) \geq d_{\mathcal{C}}(\overline{\pi}) = \frac{n+1-c(BG(\pi))}{2}$;
  2. transposition distance ($td(\cdot)$):
     
     1. $\overline{\text{transposition}}$ = 3-cycle;
     2. $td(\pi) \geq d_{\mathcal{C}}(\overline{\pi}) = \frac{n+1-c_{odd}(BG(\pi))}{2}$;
  3. prefix transposition distance ($ptd(\cdot)$):
     
     1. $\overline{\text{prefix transposition}}$ = 3-cycle containing 0;

Three examples

- How one can obtain bounds on the following distances:
  1. block-interchange distance ($bid(\cdot)$): 
     1. $\overline{\text{block-interchange}}$ = pair of 2-cycles;
     2. $bid(\pi) \geq d_{\mathcal{C}}(\overline{\pi}) = \frac{n+1-c(BG(\pi))}{2}$;
  2. transposition distance ($td(\cdot)$): 
     1. $\overline{\text{transposition}}$ = 3-cycle;
     2. $td(\pi) \geq d_{\mathcal{C}}(\overline{\pi}) = \frac{n+1-c_{odd}(BG(\pi))}{2}$;
  3. prefix transposition distance ($ptd(\cdot)$): 
     1. $\overline{\text{prefix transposition}}$ = 3-cycle containing 0;
     2. $ptd(\pi) \geq d_{\mathcal{C}}(\overline{\pi}) =$
        $\frac{n+1+c(BG(\pi))}{2} - 2c_1(BG(\pi)) - \begin{cases} 0 \text{ if } \pi_1 = 1, \\ 1 \text{ otherwise.} \end{cases}$;

A word of caution

- Don’t think that sorting unsigned permutations is trivial;
- $bid(\cdot)$ and $exc(\cdot)$ are indeed easy to compute, but:
  1. sorting by (prefix or arbitrary) reversals becomes NP-hard;
  2. sorting by transpositions is NP-hard;
  3. sorting by double cut-and-joins is NP-hard;
  4. sorting by prefix transpositions is open;

Results on sorting unsigned permutations

- What has been done:

	Operation	Sorting	Distance	Best approximation
	exchange	$O(n)$ [Knuth, 1995]		1
	block-interchange	$O(n)$ [Christie, 1996]		1
	double cut-and-joins	NP-hard [Chen, 2010]		?
	reversal	NP-hard [Caprara, 1999b]		11/8 [Berman et al., 2002]
	transposition	NP-hard [Bulteau et al., 2011b]		11/8 [Elias and Hartman, 2006]
prefix	exchange	$O(n)$ [Akers et al., 1987]		1
prefix	reversal	NP-hard [Bulteau et al., 2011a]		2 [Fischer and Ginzinger, 2005]
prefix	transposition	?	?	2 [Dias and Meidanis, 2002]

- What could be done:
  - better approximations;
  - complexity of prefix transposition problems?

Counting problems

- What is the distribution of a given rearrangement distance?
- Most tight bounds on rearrangement distances are obtained using the breakpoint graph and its cycles;
- Use the distribution of cycles in the breakpoint graph to approximately answer the above;

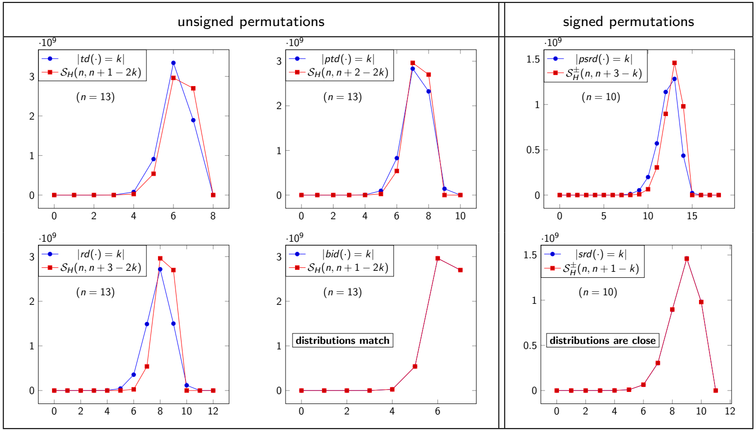
Hultman numbers

- Hultman numbers count permutations whose breakpoint graph contains k cycles:

$$\mathcal{S}_H(n,k) = |\{\pi \in S_n \mid c(BG(\pi)) = k\}| \qquad \text{unsigned case}$$
$$\mathcal{S}_H^{\pm}(n,k) = |\{\pi \in S_n^{\pm} \mid c(BG(\pi)) = k\}| \qquad \text{signed case}$$

- Similar in spirit to Stirling numbers of the first kind;
- Explicit formulas are available for computing $\mathcal{S}_H(n,k)$ and $\mathcal{S}_H^{\pm}(n,k)$ (see e.g. [Grusea and Labarre, 2011]);
- Also available: generating functions, expected value and variance;

Approximating distance distributions using Hultman numbers



(distance distributions from [Galvão and Dias, 2011])

Experiments wrap-up and perspectives

- Some distance distributions are extremely well approximated using (some function of) the Hultman numbers;
- Can bounds be tightened by trying to minimise the difference between the distributions?
- Can the proximity of distributions be used to argue that exact computation of hard distances is overrated?

Listing optimal sequences

- Distances are informative;

Listing optimal sequences

- Distances are informative;
- ... but an actual sorting sequence is more informative;

Listing optimal sequences

- Distances are informative;
- ... but an actual sorting sequence is more informative;
- What if the given sequence we found makes no biological sense?

Listing optimal sequences

- Distances are informative;
- ... but an actual sorting sequence is more informative;
- What if the given sequence we found makes no biological sense?
- $\Rightarrow$ can we list all optimal sequences for a given instance?

Listing optimal sequences

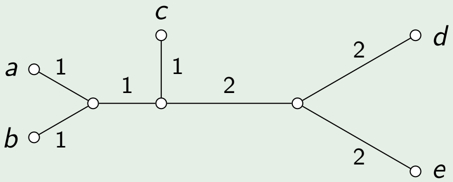
- Distances are informative;
- ... but an actual sorting sequence is more informative;
- What if the given sequence we found makes no biological sense?
- $\Rightarrow$ can we list all optimal sequences for a given instance?
- Efficient algorithms exist for listing:
    - all optimal signed reversals [Swenson et al., 2011];
    - all optimal sequences [Badr et al., 2011];

Median problems

- Measures of similarities between genomes are useful in reconstructing phylogenies;

Example (phylogeny from distance matrix)

	a	b	c	d	e
a	0	2	3	6	6
b	2	0	3	6	6
c	3	3	0	5	5
d	6	6	5	0	4
e	6	6	5	4	0



- (The matrix must satisfy some conditions [Buneman, 1971]);

Median problems

- Parsimony again: search for a tree that minimises the total number of evolutionary events (i.e. the sum of all edge weights);
- In its simplest form, the problem we want to solve is:

Problem (median of three)

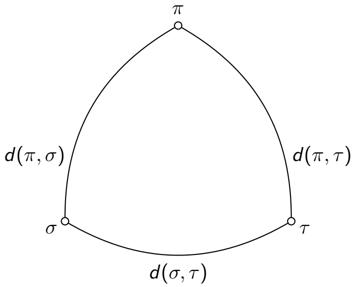
Given: π, σ, τ *in* $S_n^{\pm}$*; a distance* $d : S_n^{\pm} \times S_n^{\pm} \rightarrow \mathbb{N}$.
Find: *a permutation* μ *in* $S_n^{\pm}$ *that minimises*

$$w(\mu) = d(\pi, \mu) + d(\sigma, \mu) + d(\tau, \mu).$$

- Can be generalised to more than three input permutations;

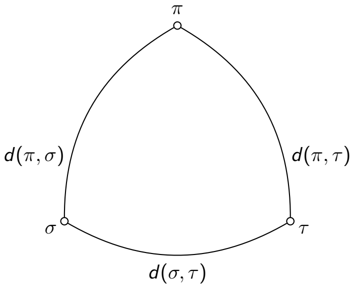
Generic bounds [Siepel and Moret, 2001]

- Generic lower and upper bounds for any distance:



Generic bounds [Siepel and Moret, 2001]

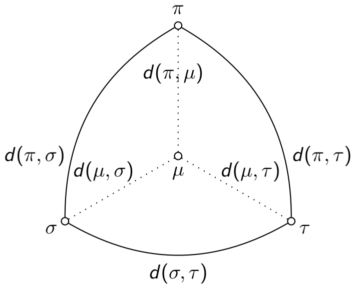
- Generic lower and upper bounds for any distance:



- $w(\mu) \leq \min\{\overbrace{d(\pi,\sigma)+d(\pi,\tau)}^{\text{if } \mu=\pi}, \overbrace{d(\pi,\sigma)+d(\sigma,\tau)}^{\text{if } \mu=\sigma}, \overbrace{d(\pi,\tau)+d(\sigma,\tau)}^{\text{if } \mu=\tau}\}.$

Generic bounds [Siepel and Moret, 2001]

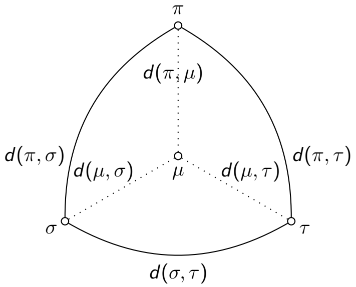
- Generic lower and upper bounds for any distance:



- $w(\mu) \leq \min\{\overbrace{d(\pi,\sigma)+d(\pi,\tau)}^{\text{if } \mu=\pi}, \overbrace{d(\pi,\sigma)+d(\sigma,\tau)}^{\text{if } \mu=\sigma}, \overbrace{d(\pi,\tau)+d(\sigma,\tau)}^{\text{if } \mu=\tau}\}.$

Generic bounds [Siepel and Moret, 2001]

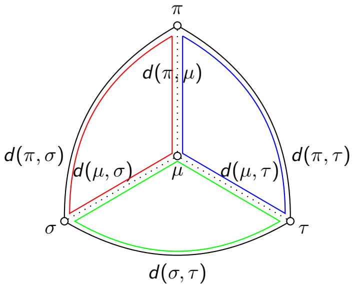
- Generic lower and upper bounds for any distance:



- $w(\mu) \leq \min\{\overbrace{d(\pi,\sigma)+d(\pi,\tau)}^{\text{if } \mu=\pi}, \overbrace{d(\pi,\sigma)+d(\sigma,\tau)}^{\text{if } \mu=\sigma}, \overbrace{d(\pi,\tau)+d(\sigma,\tau)}^{\text{if } \mu=\tau}\}.$
- $2w(\mu) = d(\pi,\mu)+d(\pi,\mu)+d(\sigma,\mu)+d(\sigma,\mu)+d(\tau,\mu)+d(\tau,\mu)$

Generic bounds [Siepel and Moret, 2001]

- Generic lower and upper bounds for any distance:



- $w(\mu) \le \min\{\overbrace{d(\pi,\sigma)+d(\pi,\tau)}^{\text{if } \mu=\pi}, \overbrace{d(\pi,\sigma)+d(\sigma,\tau)}^{\text{if } \mu=\sigma}, \overbrace{d(\pi,\tau)+d(\sigma,\tau)}^{\text{if } \mu=\tau}\}$.
- $2w(\mu) = d(\pi,\mu)+d(\pi,\mu)+d(\sigma,\mu)+d(\sigma,\mu)+d(\tau,\mu)+d(\tau,\mu)$
  $\ge d(\pi,\sigma)+d(\pi,\tau)+d(\sigma,\tau)$ (triangle inequalities)

Results on median problems

- What has been done:

Operation or measure	Median of three	Best approximation
breakpoint	NP-hard [Bryant, 1998]	5/3 [Caprara, 2002]
signed breakpoint	NP-hard [Bryant, 1998]	7/6 [Pe'er and Shamir, 2000]
exchange	?	?
signed reversal	NP-hard [Caprara, 2003]	4/3 [Caprara, 1999a]
signed double-cut-and-join	NP-hard [Caprara, 2003]	4/3 [Caprara, 1999a]
transposition	NP-hard [Bader, 2011]	?

- What could be done:
  1. complexity of the exchange median problem?
     (trivial for 2 permutations, NP-hard for ≥ 4; what about 3?)
  2. better approximations;
  3. "median clouds" [Eriksen, 2009];

Possible future directions

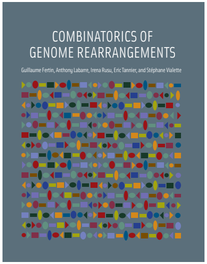
- More realistic distances:
  - several kinds of operations, weighted differently;
  - learning *ad hoc* distances?
- Generalising the bijection $\pi \mapsto \overline{\pi}$:
  - obtain *upper* bounds;
  - extend to *signed* permutations;
- Can we build bridges to other fields (e.g. pattern matching)?
- Complexity and approximability issues;

Thanks!!!

... and all apologies for (to) everything (everyone) I had to leave out.

Thanks!!!

... and all apologies for (to) everything (everyone) I had to leave out.


COMBINATORICS OF GENOME REARRANGEMENTS
Guillaume Fertin, Anthony Labarre, Irena Rusu, Eric Tannier, and Stéphane Vialette

with Guillaume Fertin,
Irena Rusu, Eric Tannier
and Stéphane Vialette.
The MIT Press, 2009.

References I

Akers, S. B., Krishnamurthy, B., and Harel, D. (1987).
The star graph: An attractive alternative to the n-cube.
In *ICPP'87*, pages 393–400. Pennsylvania State University Press.

Bader, D. A., Moret, B. M. E., and Yan, M. (2001).
A linear-time algorithm for computing inversion distance between signed permutations with an experimental study.
Journal of Computational Biology, 8(5):483–491.

Bader, M. (2011).
The transposition median problem is NP-complete.
Theoretical Computer Science, 412(12-14):1099–1110.

Badr, G., Swenson, K. M., and Sankoff, D. (2011).
Listing all parsimonious reversal sequences: New algorithms and perspectives.
Journal of Computational Biology, 18(9):1201–1210.

Bafna, V. and Pevzner, P. A. (1996).
Genome rearrangements and sorting by reversals.
SIAM Journal on Computing, 25(2):272–289.

Bafna, V. and Pevzner, P. A. (1998).
Sorting by transpositions.
SIAM Journal on Discrete Mathematics, 11(2):224–240.

References II

Bergeron, A. and Stoye, J. (2006).
On the similarity of sets of permutations and its applications to genome comparison.
Journal of Computational Biology, 13(7):1340–1354.

Berman, P., Hannenhalli, S., and Karpinski, M. (2002).
1.375-approximation algorithm for sorting by reversals.
In *ESA'02*, volume 2461 of *LNCS*, pages 200–210. Springer-Verlag.

Bryant, D. (1998).
The complexity of the breakpoint median problem.
Technical report, Université De Montréal.

Bulteau, L., Fertin, G., and Rusu, I. (2011a).
Pancake flipping is hard.
CoRR, abs/1111.0434.

Bulteau, L., Fertin, G., and Rusu, I. (2011b).
Sorting by transpositions is difficult.
In *ICALP'11*, volume 6755 of *LNCS*, pages 654–665. Springer-Verlag.

Buneman, P. (1971).
The recovery of trees from measures of dissimilarity.
Mathematics in the Archaeological and Historical Sciences, pages 387–395.

References III

Caprara, A. (1999a).
Formulations and hardness of multiple sorting by reversals.
In *RECOMB'99*, pages 84–93, New York, NY, USA. ACM.

Caprara, A. (1999b).
Sorting permutations by reversals and eulerian cycle decompositions.
SIAM Journal on Discrete Mathematics, 12(1):91–110 (electronic).

Caprara, A. (2002).
Additive bounding, worst-case analysis, and the breakpoint median problem.
SIAM Journal on Optimization, 13:508–519.

Caprara, A. (2003).
The reversal median problem.
INFORMS Journal on Computing, 15:93–113.

Chen, X. (2010).
On sorting permutations by double-cut-and-joins.
In *COCOON'10*, volume 6196 of *LNCS*, pages 439–448. Springer-Verlag.

Christie, D. A. (1996).
Sorting permutations by block-interchanges.
Information Processing Letters, 60(4):165–169.

References IV

Cohen, D. S. and Blum, M. (1995).
On the problem of sorting burnt pancakes.
Discrete Applied Mathematics, 61(2):105–120.

Dias, Z. and Meidanis, J. (2002).
Sorting by prefix transpositions.
In *SPIRE'02*, volume 2476 of *LNCS*, pages 65–76. Springer-Verlag.

Elias, I. and Hartman, T. (2006).
A 1.375-approximation algorithm for sorting by transpositions.
IEEE/ACM Transactions on Computational Biology and Bioinformatics, 3(4):369–379.

Eriksen, N. (2009).
Median clouds and a fast transposition median solver.
In *FPSAC'09*, Discrete Math. Theor. Comput. Sci. Proc., AK, pages 373–384. Assoc. Discrete Math. Theor. Comput. Sci., Nancy.

Estivill-Castro, V. and Wood, D. (1992).
A survey of adaptive sorting algorithms.
ACM Computing Surveys, 24(4):441–476.

Figeac, M. and Varré, J.-S. (2004).
Sorting by reversals with common intervals.
In *WABI'04*, volume 3240 of *LNCS*, pages 26–37. Springer-Verlag.

References V

Fischer, J. and Ginzinger, S. W. (2005).
A 2-approximation algorithm for sorting by prefix reversals.
In *ESA'05*, volume 3669 of *LNCS*, pages 415–425. Springer-Verlag.

Galvão, G. R. and Dias, Z. (2011).
Rearrangement distance database.
Published electronically at `http://mirza.ic.unicamp.br:8080/bioinfo/index.jsf`.

Grusea, S. and Labarre, A. (2011).
The distribution of cycles in breakpoint graphs of signed permutations.
CoRR, abs/1104.3353.

Han, Y. (2006).
Improving the efficiency of sorting by reversals.
In *Proceedings of The 2006 International Conference on Bioinformatics and Computational Biology*, page 4, Las Vegas, Nevada, USA. CSREA Press.

Hannenhalli, S. and Pevzner, P. A. (1999).
Transforming cabbage into turnip: Polynomial algorithm for sorting signed permutations by reversals.
Journal of the ACM, 46(1):1–27.

Knuth, D. E. (1995).
Sorting and Searching, volume 3 of *The art of Computer Programming*.
Addison-Wesley.

References VI

Labarre, A. (2012).
Lower bounding edit distances between permutations.
CoRR, abs/1201.0365.

Pe'er, I. and Shamir, R. (2000).
Approximation algorithms for the median problem in the breakpoint model.
D. Sankoff, J.H. Nadeau (Eds.), Comparative Genomics, Kluwer, Dordrecht, 2000:225–241.

Siepel, A. C. and Moret, B. M. E. (2001).
Finding an optimal inversion median: Experimental results.
In *WABI'01*, volume 2149 of *LNCS*, pages 189–203. Springer-Verlag.

Swenson, K., Badr, G., and Sankoff, D. (2011).
Listing all sorting reversals in quadratic time.
Algorithms for Molecular Biology, 6(11).

Yancopoulos, S., Attie, O., and Friedberg, R. (2005).
Efficient sorting of genomic permutations by translocation, inversion and block interchange.
Bioinformatics, 21(16):3340–3346.