Algorithms in Genome Research

Pedro Feijao

Winter 2013/14

pfeijao@cebitec.uni-bielefeld.de

Lecture 3 - **The Double Cut and Join Operation**

Genome Rearrangements - Some History

Since the beginning of the genome rearrangement field, many models were studied. First, with only one operation.

- **Inversions (reversals)** Watterson et al. 1982; Sankoff 1992; Bafna & Pevzner 1993; **Hannenhalli & Pevzner 1995**; Kaplan, Shamir & Tarjan 1999; Bader, Moret & Yan 2001; Bergeron 2001; Bergeron, Heber & S 2002; Bergeron, Mixtacki & S 2004
- **Transpositions** Meidanis, Walter & Dias, 1997; Elias & Hartman 2006; Bulteau, Fertin, Rusu 2011
- **Block interchanges** Christie 1996
- **Translocations** Hannenhalli 1996; Bergeron, Mixtacki & S 2005

Genome Rearrangements - Some History

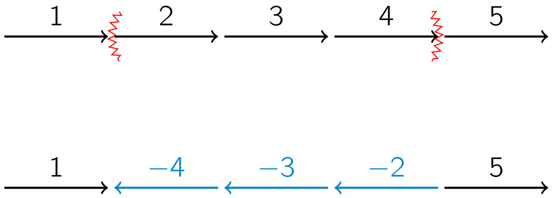
Then, models combining more than one operation:

- **Translocations and Reversals("general HP model")** Hannenhalli & Pevzner 1995; Tesler 2002; Ozery-Flato & Shamir 2003; Jean & Nikolski 2007; Bergeron, Mixtacki & S 2008; Erdõs, Sokoup & S 2011
- **Inversions + Transpositions**: Walter, Dias & Meidanis 1998; Christie & Irving 2001
- **Fusion/Fission + Transpositions**: Meidanis & Dias 2001
- **Double Cut and Join (DCJ)** Yancopoulos, Attie & Friedberg 2005; Bergeron, Mixtacki & S 2006.

DCJ Operation

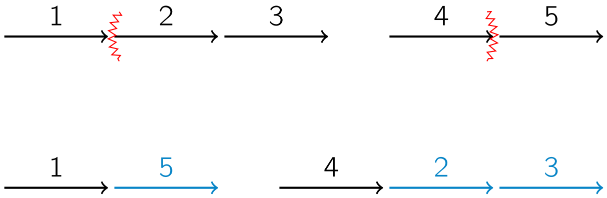
- The **DCJ operation** was proposed by Yancopoulos et al. in 2005.
- It is based on the fact that lots of rearrangement operations can be modeled by applying **two cuts** followed by **two joins** in a genome.

Genome Rearrangements



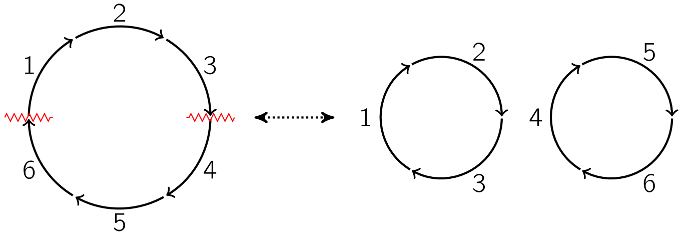
Signed Reversal/Inversion

Genome Rearrangements



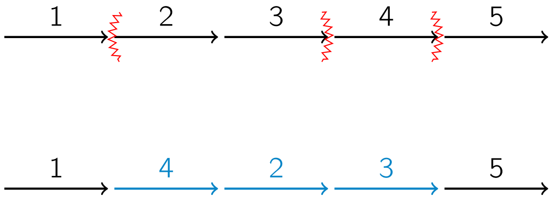
Translocation (*multichromosomal* operation)

Genome Rearrangements



Circular Fussion / Fission

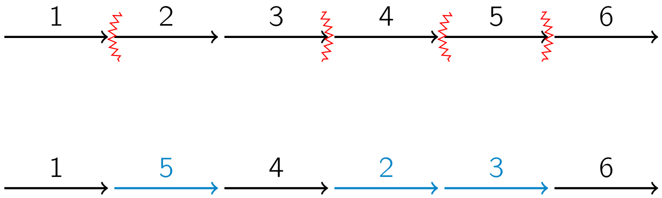
Genome Rearrangements



Transposition

More than two cuts!

Genome Rearrangements

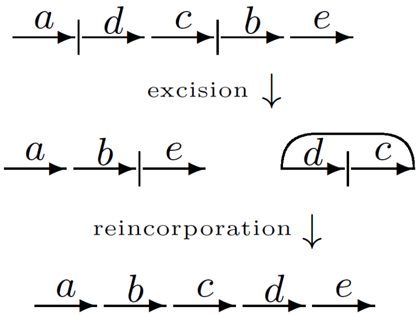


Block Interchange

More than two cuts! But...

Operations modelled with 2 DCJs

- Transpositions and Block-Interchanges can be achieved with 2 DCJs: an **excision** followed by a **reincorporation**.



Adapted from Braga and Stoye, BSB 2013

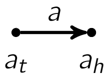
DCJ rearrangement problem

As usual, we are interested in the following questions:

- What is the minimum number of DCJ operations we need to transform one genome into another? **(distance)**
- Finding DCJ operations that actually transform one genome into another in minimal number of steps. **(sorting scenario)**

Genes, extremities and adjacencies

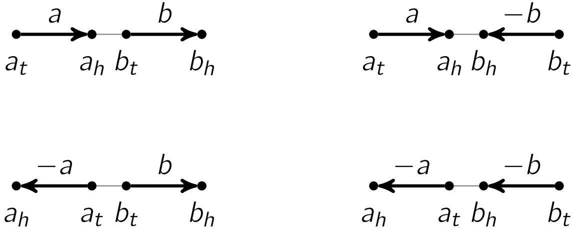
- A **block** (marker, gene) a is an oriented sequence of DNA that starts with a **tail** a_t and ends with a **head** a_h.
- Head and tail are called the **extremities** of a block.



- In the **graph representation**, each extremity is a vertex and there is a black directed edge from the tail to the head.

Genes, extremities and adjacencies

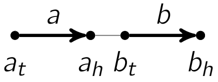
- An **adjacency** is a pair of extremities, representing the linkage between two consecutive blocks a and b.
- Depending on their respective orientation, can be of four different types: $a_h b_t, a_h b_h, a_t b_t, a_t b_h$



- In the **graph representation**, adjacencies are represented by grey edges between the extremities.

Genes, extremities and adjacencies

- An extremity that is not adjacent to any other block is called a **telomere**.



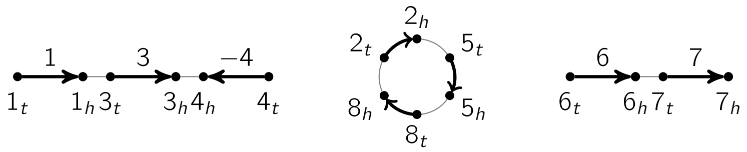
- In this example, a_t and b_h are telomeres.

Genomes

- A **genome** is set of adjacencies and telomeres such that each extremity appears in exactly **one** adjacency or telomere.

$$A = \{1_t, 1_h3_t, 3_h4_h, 4_t, 2_h5_t, 5_h8_t, 8_h2_t, 6_t, 6_h7_t, 7_h\}$$

- Graph representation:



- **Linear chromosomes** are paths, **circular chromosomes** are cyles.

DCJ Operation

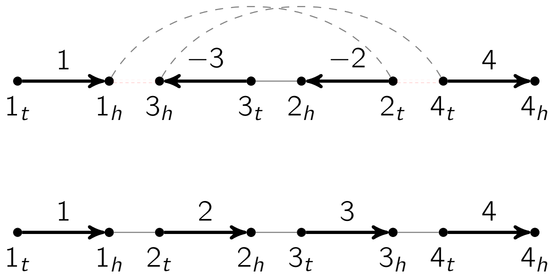
The **double cut and join** (DCJ) operation acts in the adjacencies and telomeres of a genome one of the following three ways:

- **(a)** Adjacencies $\{pq, rs\}$ are replaced by $\begin{cases} \{pr, sq\} \\ \textbf{or} \\ \{ps, qr\} \end{cases}$

- **(b)** Adjacency $\{pq\}$ and telomere $\{r\}$ are replaced by $\begin{cases} \{pr, q\} \\ \textbf{or} \\ \{qr, p\} \end{cases}$

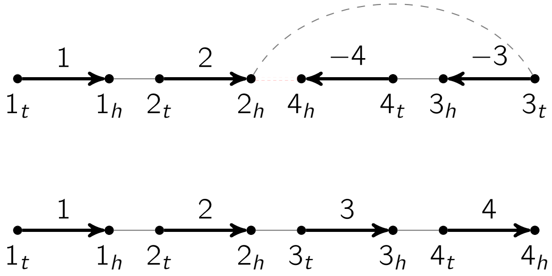
- **(c)** Telomeres $\{q, r\}$ are replaced by adjacency $\{qr\}$, or the inverse operation.

DCJ Operation - Type (a) example



- Cuts: 1_h3_h, 2_t4_t
- Joins: 1_h2_t, 3_h4_t
- **DCJ operation**: $\{1_h3_h, 2_t4_t\} \rightarrow \{1_h2_t, 3_h4_t\}$

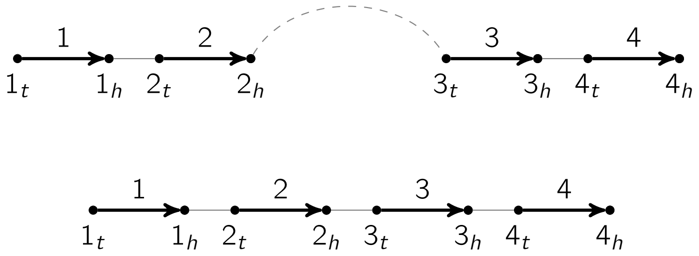
DCJ Operation - Type (b) example



1
2
−4
−3
1_t 1_h 2_t 2_h 4_h 4_t 3_h 3_t
1
2
3
4
1_t 1_h 2_t 2_h 3_t 3_h 4_t 4_h

- Cut: $2_h 4_h$ (telomere 3_t does not need a cut)
- Join: $2_h 3_t$ (new telomere 4_h does not need a join)
- **DCJ operation**: $\{2_h 4_h, 3_t\} \to \{2_h 3_t, 4_h\}$

DCJ Operation - Type (c) example



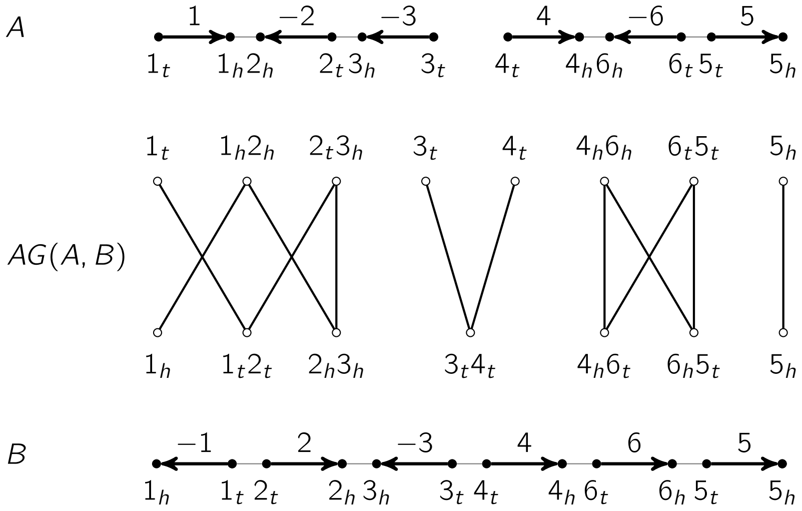
- Join: 2_h3_t
- **DCJ operation**: $\{2_h, 3_t\} \rightarrow \{2_h3_t\}$

Adjacency Graph

- The **adjacency graph** was proposed by Bergeron, Mixtacki and Stoye in 2006.
- Similarly to the BP graph, it is very useful for solving rearrangement problems.

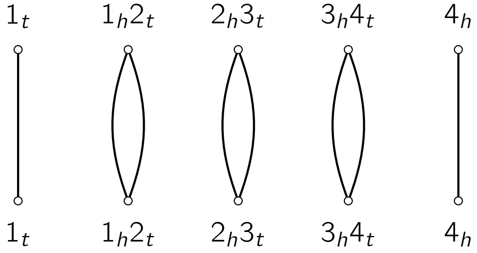
- The **adjacency graph** $AG(A, B)$ is a graph where:
  - **Vertices** are the adjacencies and telomeres of A and B.
  - **Edges** connect corresponding extremities of A and B.

Adjacency Graph



DCJ distance with the Adjacency Graph

- When A and B are the same, in $AG(A, B)$ there are only:
  - Cycles of length 2 (common adjacencies)
  - Paths of length 1 (common telomeres).



DCJ Distance

Lemma (Bergeron, Mixtacki, Stoye, 2006)

Genomes A and B are the same $\iff N = C + I/2$, where N is the number of genes, C is the number of cycles and I the number of odd paths in $AG(A, B)$.

Corollary: when A and B are different, $N > C + I/2$.

Proof?

Effect of a DCJ Operation in $AG(A, B)$

The application of one DCJ operation can change the graph $AG(A, B)$ in the following ways:

- # of odd paths by -2, 0 or $+2$. $\Delta I = -2, 0, +2$
- # of cycles by -1, 0 or $+1$. $\Delta C = -1, 0, +1$
- No DCJ changes odd paths and cycles *at the same time*.

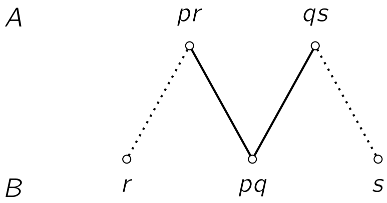
Therefore, we have: $\Delta(C + I/2) = -1, 0, +1$.

When two genomes are the same, we have that $N - (C + I/2) = 0$, which results in the following lower bound:

$$d_{\mathrm{DCJ}}(A, B) \geq N - (C + I/2)$$

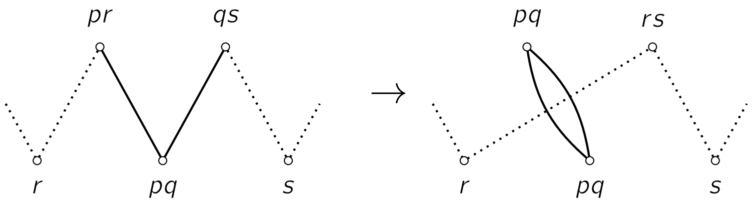
Increasing Cycles and Odd Paths in $AG(A, B)$

- If an adjacency pq in B is not present in A, then in $AG(A, B)$ the vertex pq in B will be connected to two different vertices in A.



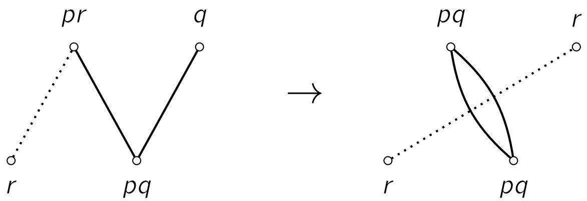
- Can we apply a DCJ operation in A that creates the adjacency pq, also increasing the number of cycles or odd paths in $AG(A, B)$?

DCJ in $AG(A,B)$, Type (a)



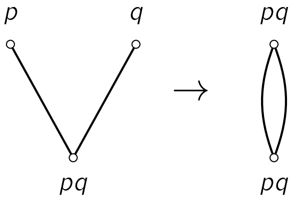
- Type (a) operation: $\{pr, qs\} \rightarrow \{pq, rs\}$
- $\Delta C = +1$, and the other component mantains the type and parity.

DCJ in $AG(A, B)$, Type (b)



- Type (b) operation: $\{pr, q\} \rightarrow \{pq, r\}$
- $\Delta C = +1$, and the original path mantains its parity.

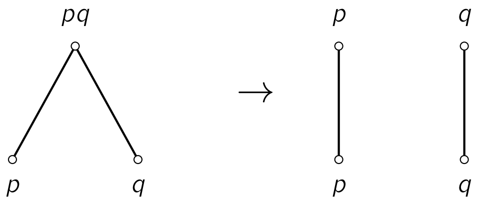
DCJ in $AG(A, B)$, Type (c)



- Type (c) operation: $\{p, q\} \rightarrow \{pq\}$
- $\Delta C = +1$, and the original even path is gone.

Another type (c) operation

- If all adjacencies of B exist in A, there is still one last possible case:



- Type (c) operation: $\{pq\} \rightarrow \{p, q\}$
- $\Delta I = +2$, and the original even path is gone.

Building a DCJ Algorithm

- Since in all cases we can always find a DCJ that increases $(C + I/2)$ by 1, this can be used to build a greedy algorithm that performs these kind of operations until A in transformed into B.

Algorithm 1 (Greedy sorting by DCJ)

```
1: for each adjacency $\{p, q\}$ in genome $B$ do
2:     let $u$ be the element of genome $A$ that contains $p$
3:     let $v$ be the element of genome $A$ that contains $q$
4:     if $u \neq v$ then
5:         replace $u$ and $v$ in $A$ by $\{p, q\}$ and $(u \setminus \{p\}) \cup (v \setminus \{q\})$
6:     end if
7: end for
8: for each telomere $\{p\}$ in genome $B$ do
9:     let $u$ be the element of genome $A$ that contains $p$
10:    if $u$ is an adjacency then
11:        replace $u$ in $A$ by $\{p\}$ and $(u \setminus \{p\})$
12:    end if
13: end for
```

Bergeron, Mixtacki & Stoye, 2006

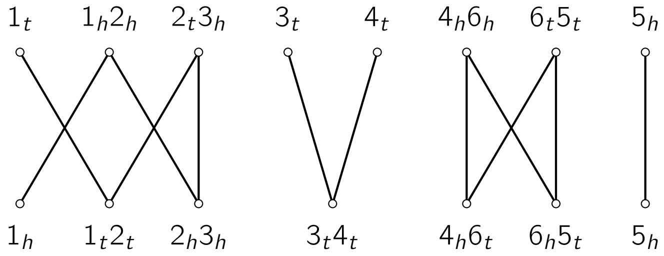
DCJ Distance

- Is this algorithm optimal?
- Since it can always increase $(C + I/2)$ by one at each step, it is not difficult to show that it always transforms A into B in $N - (C + I/2)$ steps, which is the lower bound.
- That means that the algorithm is optimal, and the DCJ distance is given by

$$d_{\mathrm{DCJ}}(A, B) = N - (C + I/2)$$

Example

$AG(A, B)$



Examples of sorting DCJ operations from A to B:

- Type (a): $\{1_h2_h, 2_t3_h\} \rightarrow \{2_h3_h, 1_h2_t\}$
- Type (b): $\{1_t, 2_t3_h\} \rightarrow \{1_t2_t, 3_h\}$
- Type (c): $\{3_t, 4_t\} \rightarrow \{3_t4_t\}$