On Sorting Genomes with DCJ and Indels

Marília D. V. Braga

AG Genome Informatics - Bielefeld University

RECOMB-CG 2010

This work is an extension of Genomic Distance with DCJ and Indels (M. D. V. Braga, E. Willing and J. Stoye), presented at WABI, 2010.
Overview

1 DCJ, Adjacency Graph, Indels

2 Sorting with DCJs and indels
   Accumulating runs
   Sorting one component individually
   Recombinations of two components

3 Sorting with minimum number of DCJs

4 Sorting with minimum number of indels
   Group runs with recombinations

5 Experiment and Discussion
DCJ, Adjacency Graph, Indels

Overview

1. DCJ, Adjacency Graph, Indels

2. Sorting with DCJs and indels
   - Accumulating runs
   - Sorting one component individually
   - Recombinations of two components

3. Sorting with minimum number of DCJs

4. Sorting with minimum number of indels
   - Group runs with recombinations

5. Experiment and Discussion
DCJ, Adjacency Graph, Indels

A: 

B: 

4 / 24 Marília Braga
Sorting with DCJ and Indels
DCJ, Adjacency Graph, Indels

A

\[ d \quad -e \quad z \quad b \quad c \]

\text{inversion}

B

\[ a \quad c \quad d \quad e \quad b \]
DCJ, Adjacency Graph, Indels

Diagram showing DCJ operations on sequences A and B.
DCJ, Adjacency Graph, Indels

A
\[
\begin{array}{cccc}
  d & -e & z & b & c \\
  d & e & z & b & c
\end{array}
\]

B
\[
\begin{array}{cccc}
  a & c & d & e \\
  a & c & d & e
\end{array}
\]

- inversion
- translocation

4 / 24

Marília Braga

Sorting with DCJ and Indels
DCJ, Adjacency Graph, Indels

A

\[ d \quad -e \quad z \quad b \quad c \]

\[ d \quad e \quad z \quad b \quad c \]

\[ a \quad y \quad z \quad b \quad c \]

\[ d \quad e \]

B

\[ a \quad c \quad d \quad e \]

\[ b \]

inversion

translocation
DCJ, Adjacency Graph, Indels

A

 inversion
translocation
fusion

B

 a c d e b
DCJ, Adjacency Graph, Indels

A

\[\begin{align*}
\text{inversion} & & \text{translocation} & & \text{fusion} \\
A & d & -e & z & b & c & a & y \end{align*}\]

B

\[\begin{align*}
B & a & c & d & e & b
\end{align*}\]
DCJ, Adjacency Graph, Indels

**A**

1. $d \rightarrow -e \rightarrow z \rightarrow b \rightarrow c$
2. $d \rightarrow e \rightarrow z \rightarrow b \rightarrow c$
3. $a \rightarrow y \rightarrow z \rightarrow b \rightarrow c$
4. $a \rightarrow y \rightarrow d \rightarrow e$
5. $a \rightarrow y \rightarrow z \rightarrow b \rightarrow c$
6. $a \rightarrow y \rightarrow z \rightarrow b \rightarrow c$
7. $d \rightarrow e$
8. $d \rightarrow e$

**B**

1. $a \rightarrow c \rightarrow d \rightarrow e$
2. $b$ (loop)

*Inversion, Translocation, Fusion, Excision*
DCJ, Adjacency Graph, Indels

A

\[
\begin{align*}
A & \quad d \quad -e \quad z \quad b \quad c \\
& \quad d \quad e \quad z \quad b \quad c \\
& \quad a \quad y \quad z \quad b \quad c \\
& \quad a \quad y \quad z \quad b \quad c \\
& \quad a \quad y \quad z \quad c \quad d \quad e \\
& \quad a \quad y \quad z \quad c \quad d \quad e
\end{align*}
\]

inversion
translocation
fusion
excision

B
\[
\begin{align*}
B & \quad a \quad c \quad d \quad e \\
& \quad a \quad c \quad d \quad e \\
& \quad a \quad c \quad d \quad e
\end{align*}
\]

b

Simplification
DCJ, Adjacency Graph, Indels

A

\[\begin{align*}
  &d \quad -e \quad z \quad b \quad c \\
  &d \quad e \quad z \quad b \quad c \\
  &a \quad y \quad z \quad b \quad c \\
  &a \quad y \quad z \quad b \quad c \\
  &a \quad y \quad z \quad c \quad d \quad e \\
  &a \quad y \quad z \quad c \quad b \quad d \quad e \\
  &a \quad y \quad z \quad c \quad b \quad d \quad e \\
  &a \quad b \quad c \quad d \quad e \\
  &a \quad c \quad d \quad e \quad b
\end{align*}\]

B

\[\begin{align*}
  &a \quad c \quad d \quad e \quad b
\end{align*}\]

inversion
translocation
fusion
excision
deletion
The double cut and join model

(Traditional) DCJ [Yancopoulos et al. 2005, Bergeron et al. 2006]:
- cuts the genome twice and rejoins loose ends
- represents most genome rearrangement operations
  (same gene content, no duplications)

DCJ with indels [Yancopoulos et al. 2008]:
- allows DCJ operations, insertions and deletions (indels)
- DCJ-indel distance can be computed in linear time [WABI 2010]
Definitions:

- **marker**: piece of DNA that has an orientation
- \( G \) = small genome of \( A \) and \( B \): set of markers that occur once in \( A \) and once in \( B \) (no duplications)
- a marker \( g \in G \) has two extremities: head \( (g^h) \) and tail \( (g^t) \)
- \( A \) and \( B \): sets of unique markers of \( A \) resp. \( B \)
- \( G \)-adjacency: adjacency of markers from \( G \) (with labels from \( A \) or \( B \))
Definitions:

- **marker**: piece of DNA that has an orientation
- **$\mathcal{G} = \text{small genome of } A \text{ and } B****: set of markers that occur once in $A$ and once in $B$ (no duplications)
- a marker $g \in \mathcal{G}$ has two extremities: head ($g^h$) and tail ($g^t$)
- $\mathcal{A}$ and $\mathcal{B}$: sets of **unique markers** of $A$ resp. $B$
- **$\mathcal{G}$-adjacency**: adjacency of markers from $\mathcal{G}$ (with **labels** from $\mathcal{A}$ or $\mathcal{B}$)
Definitions:

- **marker**: piece of DNA that has an orientation
- **$\mathcal{G} = \text{small genome of } A \text{ and } B$$**: set of markers that occur once in $A$ and once in $B$ (no duplications)
- a marker $g \in \mathcal{G}$ has two extremities: head ($g^h$) and tail ($g^t$)
- $\mathcal{A}$ and $\mathcal{B}$: sets of **unique markers** of $A$ resp. $B$
- **$\mathcal{G}$-adjacency**: adjacency of markers from $\mathcal{G}$ (with **labels** from $\mathcal{A}$ or $\mathcal{B}$)
Definitions:

- **marker**: piece of DNA that has an orientation
- **$G$ = small genome of $A$ and $B$**: set of markers that occur once in $A$ and once in $B$ (no duplications)
- A marker $g \in G$ has two extremities: head ($g^h$) and tail ($g^t$)
- $A$ and $B$: sets of unique markers of $A$ resp. $B$
- **$G$-adjacency**: adjacency of markers from $G$ (with labels from $A$ or $B$)

**Diagram**:

```
A: a -> e -> x -> c -> d -> y -> b
V(A): a^t -> a^h e^t -> e^h x c^t -> c^h o d^t -> d^h y b^t -> b^h o
B: a -> b -> c -> d -> e
```

```
V(A): o a^t -> a^h e^t -> e^h x c^t -> c^h o d^t -> d^h y b^t -> b^h o
```
The adjacency graph AG(A, B) for genomes A and B:

Genome A has four unique markers (w, x, y and z).

Minimum number of DCJs necessary to sort A into B (ignoring indels):

\[ d_{\text{DCJ}}(A, B) = |G| - (c + b^2) \]

where \( c \) = number of cycles, \( b \) = number of AB-paths [Bergeron et al. 2006]
The adjacency graph AG(A, B) for genomes A and B:

Genome A has four unique markers (w, x, y and z).
The adjacency graph with $\mathcal{G}$-adjacencies

The adjacency graph $AG(A, B)$ for genomes $A$ and $B$:

Genome $A$ has four unique markers ($w, x, y$ and $z$).
The adjacency graph with $g$-adjacencies

The adjacency graph $AG(A, B)$ for genomes $A$ and $B$:

Components of $AG(A, B)$: 1 cycle, 2 $AB$-paths and 2 $AA$-paths
The adjacency graph with $\mathcal{S}$-adjacencies

The adjacency graph $AG(A, B)$ for genomes $A$ and $B$:

Components of $AG(A, B)$: 1 cycle, 2 $AB$-paths and 2 $AA$-paths

Minimum number of DCJs necessary to sort $A$ into $B$ (ignoring indels):

$$d_{DCJ}(A, B) = |\mathcal{S}| - \left( c + \frac{b}{2} \right),$$
where $c = \#$ cycles, $b = \# AB$-paths [Bergeron et al. 2006]
The adjacency graph with \( \mathcal{G} \)-adjacencies

The adjacency graph \( AG(A, B) \) for genomes \( A \) and \( B \):

Components of \( AG(A, B) \): 1 cycle, 2 \( AB \)-paths and 2 \( AA \)-paths; \( d_{DCJ} = 5 - 1 - 2/2 = 3 \)

Minimum number of DCJs necessary to sort \( A \) into \( B \) (ignoring indels):
\[
d_{DCJ}(A, B) = |\mathcal{G}| - (c + \frac{b}{2}) \]
where \( c = \# \) cycles, \( b = \# \) \( AB \)-paths [Bergeron et al. 2006]
Overview

1 DCJ, Adjacency Graph, Indels

2 Sorting with DCJs and indels
   - Accumulating runs
   - Sorting one component individually
   - Recombinations of two components

3 Sorting with minimum number of DCJs

4 Sorting with minimum number of indels
   - Group runs with recombinations

5 Experiment and Discussion
Saving indel operations

3 steps (trivial)

2 steps
Sorting with DCJs and indels

Saving indel operations

Types of DCJ operation:

<table>
<thead>
<tr>
<th>DCJ</th>
<th>effect on AG(A, B)</th>
<th>weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>optimal</td>
<td>increase c or b</td>
<td>0</td>
</tr>
<tr>
<td>neutral</td>
<td>c and b unchanged</td>
<td>+1</td>
</tr>
<tr>
<td>counter-optimal</td>
<td>decrease c or b</td>
<td>+2</td>
</tr>
</tbody>
</table>
The optimal sorting scenarios can have different compositions with respect to the number of DCJ and indel operations.

### 3 DCJs + 3 indels

1. Translocation
   - $a \ x \ e \ c \ \Downarrow$
   - $a \ x \ y \ b$
2. Deletion
   - $a \ b \ \Downarrow$
3. Translocation
   - $a \ b \ c \ \Downarrow$
4. Fusion
   - $a \ b \ c \ \Downarrow$
5. Insertion
   - $a \ u \ b \ c \ \Downarrow$
6. Insertion
   - $a \ u \ b \ c \ \Downarrow$

### 4 DCJs + 2 indels

1. Translocation
   - $a \ x \ e \ c \ \Downarrow$
2. Deletion
   - $a \ x \ y \ d \ \Downarrow$
3. Fusion
   - $a \ d \ \Downarrow$
4. Fusion
   - $a \ d \ c \ \Downarrow$
5. Insertion
   - $a \ u \ v \ d \ \Downarrow$
6. Inversion
   - $a \ u \ v \ d \ \Downarrow$
7. Inversion
   - $a \ u \ b \ c \ \Downarrow$
Accumulating runs

An optimal DCJ accumulates labels in a single vertex:
Accumulating runs

An optimal DCJ *accumulates* labels in a single vertex:

Runs of a component $C$:

Optimal DCJs accumulate the labels of one run in a single $\mathcal{G}$-adjacency
$
\Rightarrow \text{only 1 indel per run is necessary}
$
A DCJ can merge at most two $A$-runs and two $B$-runs:

```
\ell_1 \quad \ell_3 \quad \ell_5 \quad \ell_2
```

5 runs

```
\ell_1 \quad \ell_5 \quad \ell_3
```

1 + 2 runs

Ignoring indels (labels), it is possible to sort a component individually with optimal DCJs only:

$$d(C) = \text{number of DCJs required to sort } C$$

Indel-potential of a component $C$:

$$\lambda(C) = \Lambda(C) + \frac{\|\Lambda(C)\|}{2}$$

A component $C$ can be sorted with $d(C)$ DCJs + $\lambda(C)$ indels.
Sorting with DCJs and indels

Sorting one component individually

A DCJ can merge at most two $A$-runs and two $B$-runs:

Ignoring indels (labels), it is possible to sort a component individually with optimal DCJs only: $d(C) =$ number of DCJs required to sort $C$

Indel-potential of a component $C$ [WABI 2010]

Minimum number of runs obtained by sorting $C$ with optimal DCJs:

$$\lambda(C) = \left\lceil \frac{\Lambda(C) + 1}{2} \right\rceil$$
Sorting with DCJs and indels

Sorting one component individually

A DCJ can merge at most two $A$-runs and two $B$-runs:

Ignoring indels (labels), it is possible to sort a component individually with optimal DCJs only: $d(C) =$ number of DCJs required to sort $C$

Indel-potential of a component $C$ [WABI 2010]

Minimum number of runs obtained by sorting $C$ with optimal DCJs:

$$\lambda(C) = \left\lceil \frac{\Lambda(C) + 1}{2} \right\rceil$$

<table>
<thead>
<tr>
<th>$\Lambda(C)$</th>
<th>$\lambda(C)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Sorting with DCJs and indels

Sorting one component individually

A DCJ can merge at most two $A$-runs and two $B$-runs:

5 runs $\rightarrow$ 1 + 2 runs

Ignoring indels (labels), it is possible to sort a component individually with optimal DCJs only: $d(C) = \text{number of DCJs required to sort } C$

Indel-potential of a component $C$ [WABI 2010]

Minimum number of runs obtained by sorting $C$ with optimal DCJs:

$$\lambda(C) = \left\lceil \frac{\Lambda(C) + 1}{2} \right\rceil$$

A component $C$ can be sorted with $d(C)$ DCJs + $\lambda(C)$ indels.
Neutral DCJs in components with $\lambda \geq 3$ save one indel:

$$\Lambda = 7; \lambda = 4$$
Neutral DCJs in components with $\lambda \geq 3$ save one indel:

\[
\begin{align*}
\Lambda &= 7; \; \lambda = 4 \\
\Lambda &= 5; \; \lambda = 3
\end{align*}
\]
Sorting with DCJs and indels

Sorting one component individually

Neutral DCJs in components with $\lambda \geq 3$ save one indel:

\[ \Lambda = 7; \lambda = 4 \]
\[ \Lambda = 5; \lambda = 3 \]
\[ \Lambda = 3; \lambda = 2 \]
Neutral DCJs in components with $\lambda \geq 3$ save one indel:

$\Lambda = 7; \lambda = 4 \quad \rightarrow \quad \Lambda = 5; \lambda = 3 \quad \rightarrow \quad \Lambda = 3; \lambda = 2$
Sorting with DCJs and indels

Sorting one component individually

**Neutral** DCJs in components with $\lambda \geq 3$ save one indel:

\[
\begin{align*}
\Lambda &= 7; \; \lambda = 4 & \Lambda &= 5; \; \lambda = 3 & \Lambda &= 3; \; \lambda = 2 \\
\ell_1 &\quad \ell_3 \\
\ell_2 \quad \ell_4 \\
\ell_5 \quad \ell_7 \\
\end{align*}
\]

A component $C$ can be sorted with $d(C) + \lambda(C)$ operations:

<table>
<thead>
<tr>
<th>DCJs</th>
<th>indels</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d(C)$</td>
<td>$\lambda(C)$</td>
</tr>
<tr>
<td>$d(C) + 1$</td>
<td>$\lambda(C) - 1$</td>
</tr>
<tr>
<td>$d(C) + 2$</td>
<td>$\lambda(C) - 2$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$d(C) + \lambda(C) - 2$</td>
<td>2 (min indels)</td>
</tr>
</tbody>
</table>
Recombination of two components

\[ \Delta d = \Delta d_{\text{DCJ}} + \Delta \lambda \]

Neutral:

\[ BB + AB \rightarrow BB + AB \]

\[ \Lambda: 4 + 2 \rightarrow 1 + 3 \quad (\Delta \Lambda = -2) \]

\[ \lambda: 3 + 2 \rightarrow 1 + 2 \quad (\Delta \lambda = -2) \]
Recombination of two components

\[ \Lambda: 4 + 2 \rightarrow 1 + 3 \quad (\Delta \Lambda = -2) \]

\[ \lambda: 3 + 2 \rightarrow 1 + 2 \quad (\Delta \lambda = -2) \]
Recombination of two components

\[
\begin{array}{c}
\Lambda: \quad 3 + 2 \rightarrow 1 + 2 \quad (\Delta \Lambda = -2) \\
\lambda: \quad 2 + 2 \rightarrow 1 + 2 \quad (\Delta \lambda = -1)
\end{array}
\]
Sorting with DCJs and indels

Recombination of two components

\[ \Delta d = \Delta d_{\text{DCJ}} + \Delta \lambda_{\text{neutral}} \]

\[ \Lambda: \quad \begin{array}{c} \ell_1 \ell_2 \ell_3 \ 
\ell_5 \ell_6 \ell_7 \quad \rightarrow \quad \ell_1 \ell_6 \\
\ell_2 \ell_5 \ell_7 \quad \rightarrow \quad \ell_5 \ell_7 \\
\end{array} \]

\[ \lambda: \quad \begin{array}{c} \ell_1 \ell_2 \ell_3 \ 
\ell_5 \ell_6 \ell_7 \quad \rightarrow \quad \ell_5 \ell_7 \\
\ell_2 \ell_5 \ell_7 \quad \rightarrow \quad \ell_5 \ell_7 \\
\end{array} \]

<table>
<thead>
<tr>
<th>\Lambda(C)</th>
<th>\lambda(C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
</tr>
</tbody>
</table>

\[ \Delta \Lambda = -2 \]

\[ \Delta \lambda = 0 \]

...
Recombination of two components

\[ \Delta d = \Delta d_{\text{DCJ}} + \Delta \lambda \]

Examples:

1. $\Lambda : 4 + 2 \rightarrow 1 + 3$ \( (\Delta \Lambda = -2) \)
2. $\lambda : 3 + 2 \rightarrow 1 + 2$ \( (\Delta \lambda = -2) \)
Recombination of two components

Change in the DCJ-indel distance: $\Delta d = \Delta d_{\text{DCJ}} + \Delta \lambda$

<table>
<thead>
<tr>
<th>DCJ</th>
<th>$\Delta d_{\text{DCJ}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>optimal</td>
<td>0</td>
</tr>
<tr>
<td>neutral</td>
<td>1</td>
</tr>
<tr>
<td>counter-opt</td>
<td>+2</td>
</tr>
</tbody>
</table>
Recombination of two components

Change in the DCJ-indel distance: \( \Delta d = \Delta d_{\text{DCJ}} + \Delta \lambda \)

neutral: \( BB + AB \rightarrow BB + AB \)

<table>
<thead>
<tr>
<th>DCJ</th>
<th>( \Delta d_{\text{DCJ}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>optimal</td>
<td>0</td>
</tr>
<tr>
<td>neutral</td>
<td>+1</td>
</tr>
<tr>
<td>counter-opt</td>
<td>+2</td>
</tr>
</tbody>
</table>
Recombination of two components

Change in the DCJ-indel distance: \( \Delta d = \Delta d_{DCJ} + \Delta \lambda \)

\[ \begin{align*}
\text{neutral:} & \quad BB + AB \rightarrow BB + AB \\
\Lambda: & \quad 4 + 2 \rightarrow 1 + 3 \quad (\Delta \Lambda = -2) \\
\lambda: & \quad 3 + 2 \rightarrow 1 + 2 \quad (\Delta \lambda = -2)
\end{align*} \]

\[ \Delta d = +1 - 2 = -1 \]
A recombination saves at most 2 indels

Possible ways to achieve $\Delta d \leq -1$:

<table>
<thead>
<tr>
<th></th>
<th>$\Delta d_{DCJ}$</th>
<th>$\Delta \lambda$</th>
<th>$\Delta d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>optimal</td>
<td>0</td>
<td>-2</td>
<td>-2</td>
</tr>
<tr>
<td>neutral</td>
<td>+1</td>
<td>-2</td>
<td>-1</td>
</tr>
</tbody>
</table>
A recombination saves at most 2 indels

Possible ways to achieve $\Delta d \leq -1$:

<table>
<thead>
<tr>
<th></th>
<th>$\Delta d_{\text{DCJ}}$</th>
<th>$\Delta \lambda$</th>
<th>$\Delta d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>optimal</td>
<td>0</td>
<td>-2</td>
<td>-2</td>
</tr>
<tr>
<td>optimal</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>neutral</td>
<td>+1</td>
<td>-2</td>
<td>-1</td>
</tr>
</tbody>
</table>

Considering all components of $\text{AG}(A, B)$, find a sequence of recombinations $S$ such that the weight $w(S) = \sum_{\rho \in S} \Delta d(\rho)$ is minimum: solved in linear time [WABI 2010]
A recombination saves at most 2 indels

Possible ways to achieve $\Delta d \leq -1$:

<table>
<thead>
<tr>
<th></th>
<th>$\Delta d_{DCJ}$</th>
<th>$\Delta \lambda$</th>
<th>$\Delta d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>optimal</td>
<td>0</td>
<td>-2</td>
<td>-2</td>
</tr>
<tr>
<td>optimal</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>neutral</td>
<td>+1</td>
<td>-2</td>
<td>-1</td>
</tr>
</tbody>
</table>

Considering all components of $AG(A, B)$, find a sequence of recombinations $S$ such that the weight $w(S) = \sum_{\rho \in S} \Delta d(\rho)$ is minimum: solved in linear time [WABI 2010]
Overview

1. DCJ, Adjacency Graph, Indels

2. Sorting with DCJs and indels
   - Accumulating runs
   - Sorting one component individually
   - Recombinations of two components

3. Sorting with minimum number of DCJs

4. Sorting with minimum number of indels
   - Group runs with recombinations

5. Experiment and Discussion
Algorithm 1: Sorting genome $A$ into $B$ with minimum number of DCJs

1. Apply all recombinations in $S$.
2. For each component $C \in \text{AG}(A, B)$:
   
   2.1 Split $C$ with optimal DCJs (that have $\Delta \lambda = 0$) until only components that have at most 2 runs are obtained and the total number of runs in all new components is equal to $\lambda(C)$.
   
   2.2 Accumulate all runs in the smaller components derived from $C$ with optimal DCJ operations (that have $\Delta \lambda = 0$).
   
   2.3 Apply optimal DCJ operations (that have $\Delta \lambda = 0$) in the smaller components derived from $C$ until only DCJ-sorted components exist.
   
   2.4 Delete/insert all runs in the DCJ-sorted components derived from $C$. 
Overview

1. DCJ, Adjacency Graph, Indels
2. Sorting with DCJs and indels
   - Accumulating runs
   - Sorting one component individually
   - Recombinations of two components
3. Sorting with minimum number of DCJs
4. Sorting with minimum number of indels
   - Group runs with recombinations
5. Experiment and Discussion
Neutral operations in components with $\lambda \geq 3$ saves one indel

$\rightarrow$ **Grouping runs in one component first is good!**
Neutral operations in components with $\lambda \geq 3$ saves one indel

$\rightarrow$ **Grouping runs in one component first is good!**

$\lambda : \quad 2 + 2$
Neutral operations in components with $\lambda \geq 3$ saves one indel

→ **Grouping runs in one component first is good!**
Neutral operations in components with $\lambda \geq 3$ saves one indel
→ Grouping runs in one component first is good!
Algorithm 2: Sorting genome $A$ into $B$ with minimum number of indels

1. Apply all recombinations in $S$.

2. (*) Group runs of two components in one component using counter-optimal recombinations with $\Delta \lambda = -2$, neutral recombinations with $\Delta \lambda = -1$ and optimal recombinations with $\Delta \lambda = 0$.

   [After this step, there is at most one component with 2 or more runs; the others have at most one run.]

3. For each component $C \in \text{AG}(A, B)$:
   
   3.1 (*) While $\lambda(C) \geq 3$, apply a neutral DCJ on $C$ with $\Delta \lambda = -1$.
   
   3.2 If $\Lambda(C) = 3$ ($C$ is a path), merge the last and the first runs of $C$ extracting a cycle with all runs (optimal DCJ with $\Delta \lambda = 0$).
   
   3.3 Accumulate all runs in the smaller components derived from $C$ with optimal DCJ operations that have $\Delta \lambda = 0$.
   
   3.4 Apply optimal DCJ operations in the smaller components derived from $C$ until only DCJ-sorted components exist (these DCJs have $\Delta \lambda = 0$).
   
   3.5 Delete/insert all runs in the DCJ-sorted components derived from $C$. 
Overview

1. DCJ, Adjacency Graph, Indels

2. Sorting with DCJs and indels
   - Accumulating runs
   - Sorting one component individually
   - Recombinations of two components

3. Sorting with minimum number of DCJs

4. Sorting with minimum number of indels
   - Group runs with recombinations

5. Experiment and Discussion
Comparing species of *Rickettsia*

Data from Blanc *et al.* PLoS Genet. 2007

<table>
<thead>
<tr>
<th>species</th>
<th>Mbp</th>
<th>$d_{id}^{DCJ}$</th>
<th>MIN DCJs DCJs+indels</th>
<th>MIN indels DCJs+indels</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>R. conorii</em></td>
<td>1.27</td>
<td>414</td>
<td>261 + 153</td>
<td>313 + 101</td>
</tr>
<tr>
<td><em>R. africæ</em></td>
<td>1.28</td>
<td>426</td>
<td>260 + 166</td>
<td>322 + 104</td>
</tr>
<tr>
<td><em>R. massilæ</em></td>
<td>1.36</td>
<td>448</td>
<td>276 + 172</td>
<td>340 + 108</td>
</tr>
<tr>
<td><em>R. felis</em></td>
<td>1.55</td>
<td>493</td>
<td>312 + 181</td>
<td>389 + 104</td>
</tr>
<tr>
<td><em>R. typhi</em></td>
<td>1.11</td>
<td>309</td>
<td>195 + 114</td>
<td>212 + 97</td>
</tr>
<tr>
<td><em>R. prowazæki</em></td>
<td>1.11</td>
<td>314</td>
<td>197 + 117</td>
<td>216 + 98</td>
</tr>
</tbody>
</table>

$\Delta = 67$  $\Delta = 11$
Experiment and Discussion

Summary: Sorting with DCJs and indels
► Minimizing number of DCJs
► Minimizing number of indels

Future work
► Study the whole space of solutions of the problem
► Incorporate substitutions in the model?
Experiment and Discussion

Summary: Sorting with DCJs and indels

▶ Minimizing number of DCJs
▶ Minimizing number of indels

Future work

▶ Study the whole space of solutions of the problem
▶ Incorporate substitutions in the model?
Acknowledgments

- Eyla Willing
- Jens Stoye
- Sophia Yancopoulos
Acknowledgments

- Eyla Willing
- Jens Stoye
- Sophia Yancopoulos

Thank you for your attention!
References


