Universität Bielefeld

On Sorting Genomes with DCJ and Indels

Marília D. V. Braga

AG Genome Informatics - Bielefeld University

RECOMB-CG 2010

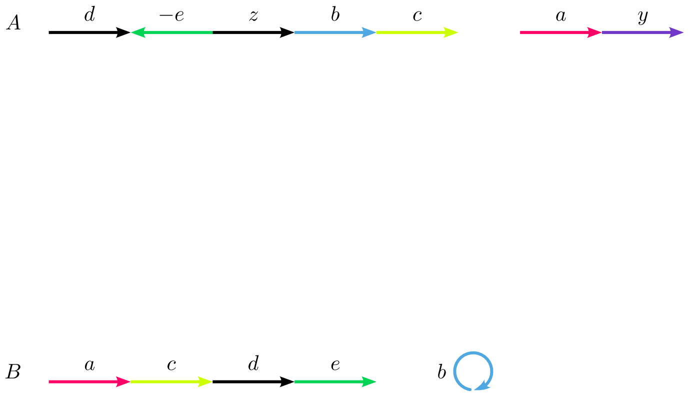
This work is an extension of **Genomic Distance with DCJ and Indels**
(M. D. V. Braga, E. Willing and J. Stoye), presented at *WABI*, 2010.

Overview

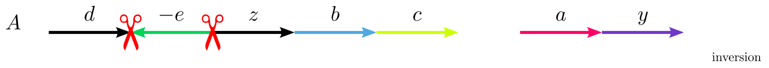
Universität Bielefeld

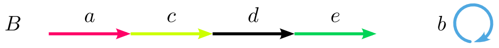
Overview

DCJ, Adjacency Graph, Indels

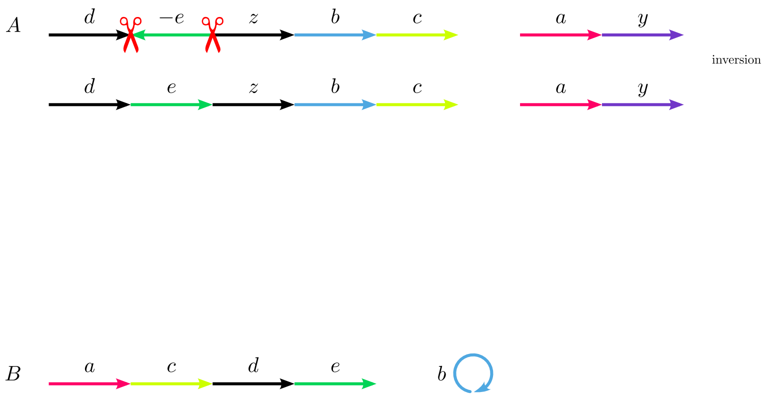


DCJ, Adjacency Graph, Indels

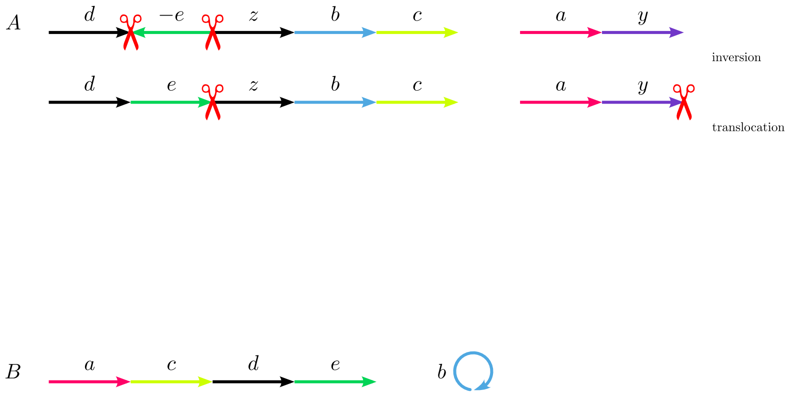




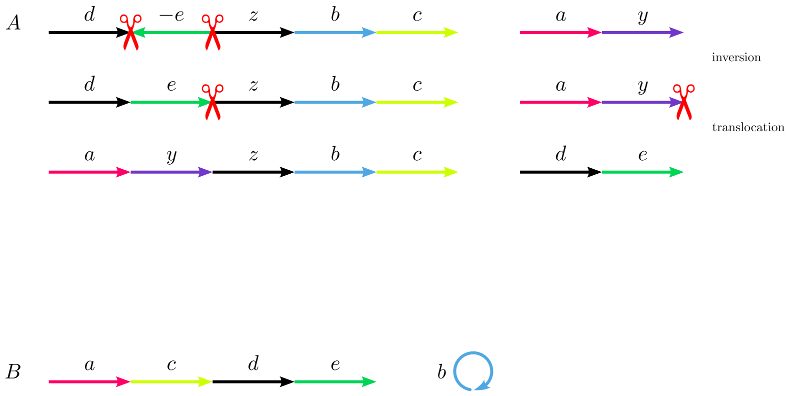
DCJ, Adjacency Graph, Indels



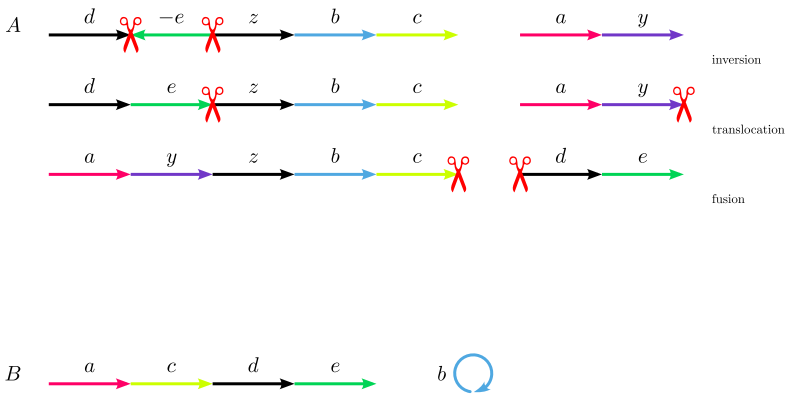
DCJ, Adjacency Graph, Indels



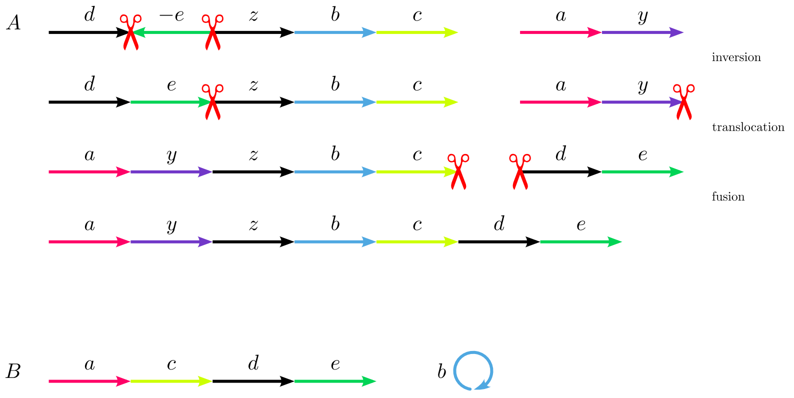
DCJ, Adjacency Graph, Indels



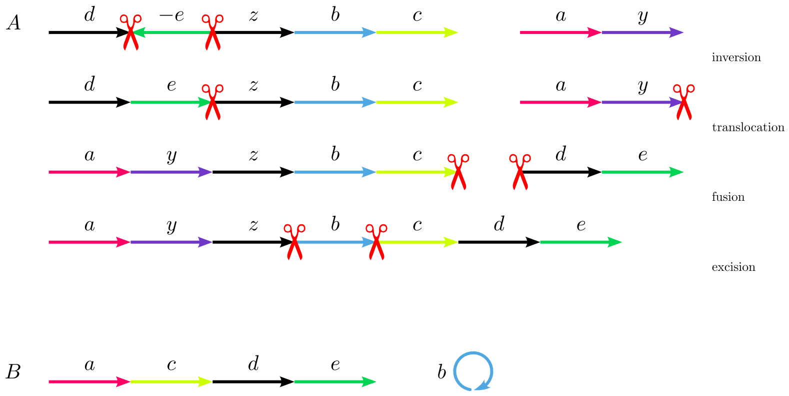
DCJ, Adjacency Graph, Indels



DCJ, Adjacency Graph, Indels

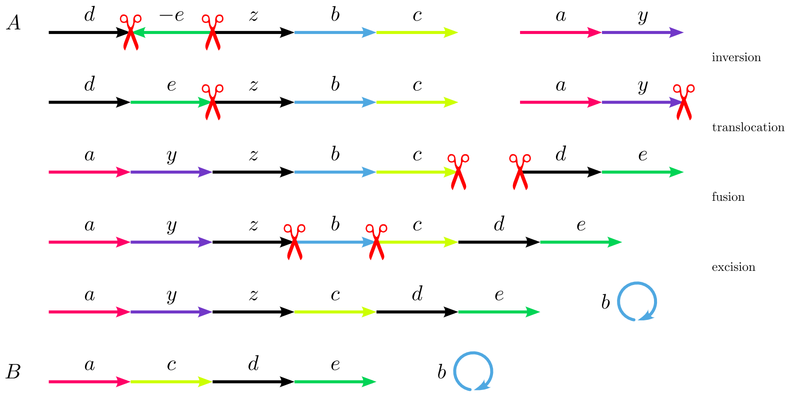


DCJ, Adjacency Graph, Indels



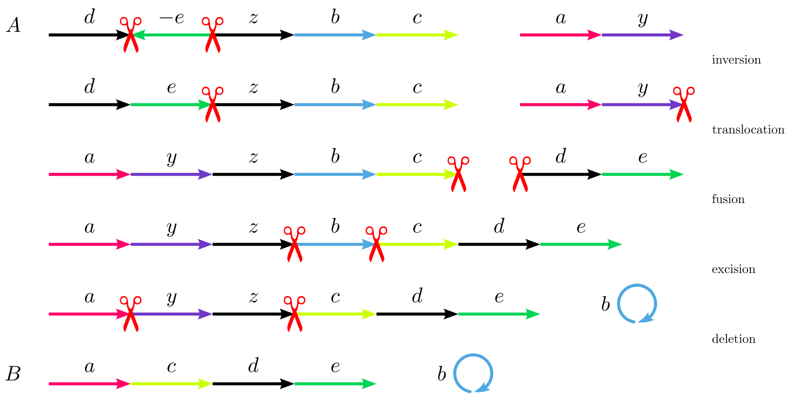
A
d $-e$ z b c a y
inversion
d e z b c a y
translocation
a y z b c d e
fusion
a y z b c d e
excision
B
a c d e
b

DCJ, Adjacency Graph, Indels



A
d $-e$ z b c a y
inversion
d e z b c a y
translocation
a y z b c d e
fusion
a y z b c d e
excision
a y z c d e b
B
a c d e b

DCJ, Adjacency Graph, Indels



A
d $-e$ z b c a y
inversion
d e z b c a y
translocation
a y z b c d e
fusion
a y z b c d e
excision
a y z c d e b
deletion
B
a c d e b

DCJ, Adjacency Graph, Indels

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]:

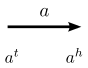
- allows DCJ operations, insertions and deletions (indels)
- DCJ-indel distance can be computed in linear time [WABI 2010]

Universität Bielefeld

DCJ, Adjacency Graph, Indels

Definitions:

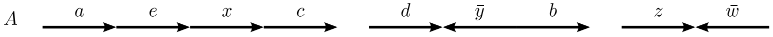
- *marker:* piece of DNA that has an orientation
- *$\mathcal{G}$ = small genome of A and B*: set of markers that occur once in *A* and once in *B* (no duplications)
- a marker $g \in \mathcal{G}$ has two extremities: head (g^h) and tail (g^t)
- $\mathcal{A}$ and $\mathcal{B}$: sets of *unique markers*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



DCJ, Adjacency Graph, Indels

Definitions:

- *marker:* piece of DNA that has an orientation
- *$\mathcal{G}$ = small genome of A and B*: set of markers that occur once in *A* and once in *B* (no duplications)
- a marker $g \in \mathcal{G}$ 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 $\mathcal{B}$)

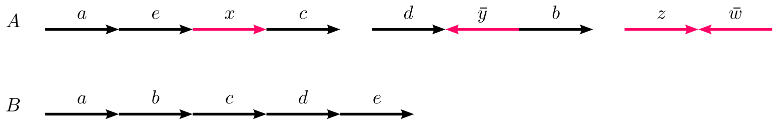


Universität Bielefeld

DCJ, Adjacency Graph, Indels

Definitions:

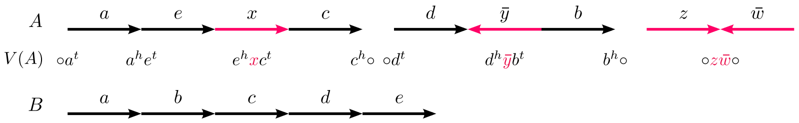
- *marker:* piece of DNA that has an orientation
- *$\mathcal{G}$ = small genome of A and B*: set of markers that occur once in *A* and once in *B* (no duplications)
- a marker $g \in \mathcal{G}$ has two extremities: head (g^h) and tail (g^t)
- $\mathcal{A}$ and $\mathcal{B}$: sets of *unique markers*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



DCJ, Adjacency Graph, Indels

Definitions:

- *marker:* piece of DNA that has an orientation
- *$\mathcal{G}$ = small genome of A and B*: set of markers that occur once in *A* and once in *B* (no duplications)
- a marker $g \in \mathcal{G}$ 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 $\mathcal{B}$)

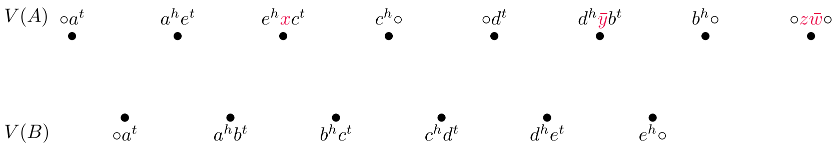


Universität Bielefeld

DCJ, Adjacency Graph, Indels

The adjacency graph with $\mathcal{G}$-adjacencies

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



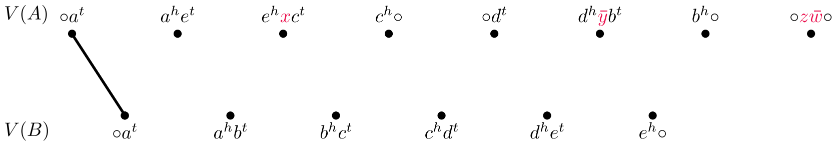
Genome *A* has four unique markers (*w*, *x*, *y* and *z*).

Universität Bielefeld

DCJ, Adjacency Graph, Indels

The adjacency graph with $\mathcal{G}$-adjacencies

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

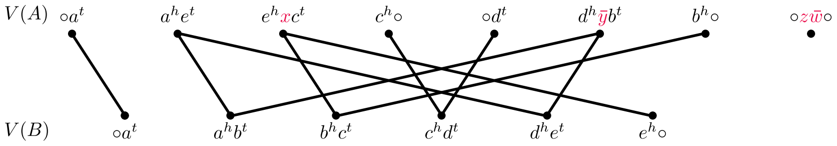


Genome *A* has four unique markers (*w*, *x*, *y* and *z*).

DCJ, Adjacency Graph, Indels

The adjacency graph with $\mathcal{G}$-adjacencies

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

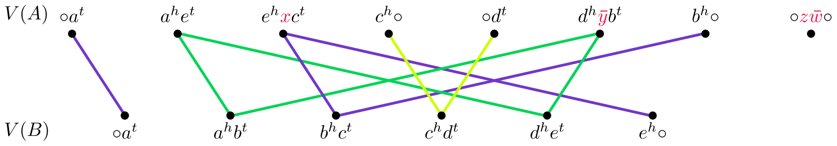


Genome *A* has four unique markers (*w*, *x*, *y* and *z*).

DCJ, Adjacency Graph, Indels

The adjacency graph with $\mathcal{G}$-adjacencies

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

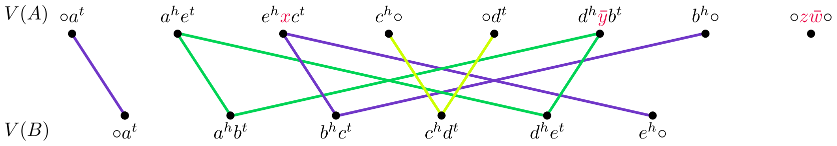


Components of AG(*A*, *B*): 1 cycle, 2 *AB*-paths and 2 *AA*-paths

DCJ, Adjacency Graph, Indels

The adjacency graph with $\mathcal{G}$-adjacencies

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



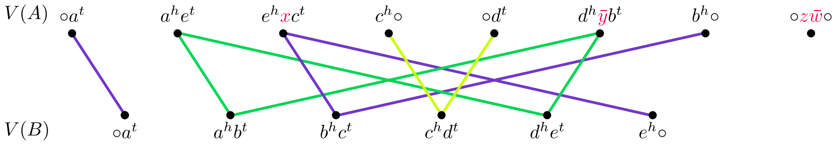
Components of AG(*A*, *B*): 1 cycle, 2 *AB*-paths and 2 *AA*-paths

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

DCJ, Adjacency Graph, Indels

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 ; $\boldsymbol{d_{\mathrm{DCJ}} = 5 - 1 - 2/2 = 3}$

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

Universität Bielefeld

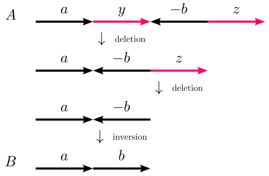
Sorting with DCJs and indels

Overview

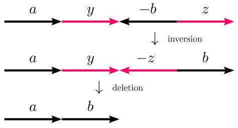
Universität Bielefeld

Sorting with DCJs and indels

Saving indel operations



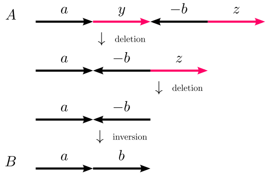
3 steps (trivial)



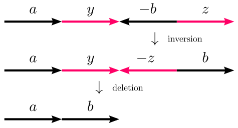
2 steps

Sorting with DCJs and indels

Saving indel operations



3 steps (trivial)



2 steps

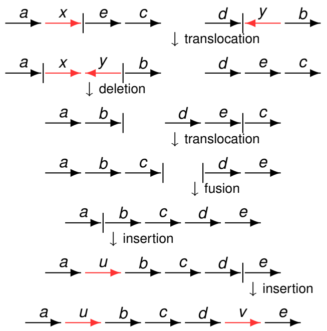
Types of DCJ operation:

DCJ	effect on AG(A, B)	weight
optimal	increase c or b	0
neutral	c and b unchanged	+1
counter-optimal	decrease c or b	+2

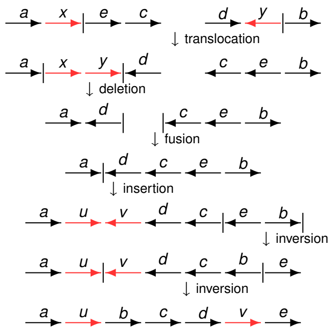
Sorting with DCJs and indels

The optimal sorting scenarios can have different compositions with respect to the number of DCJ and indel operations.

3 DCJs + 3 indels



4 DCJs + 2 indels

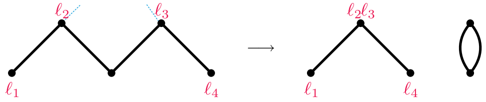


Universität Bielefeld

Sorting with DCJs and indels

Accumulating runs

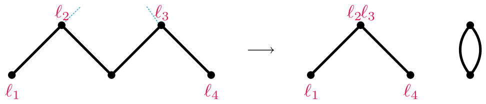
An optimal DCJ *accumulates* labels in a single vertex:



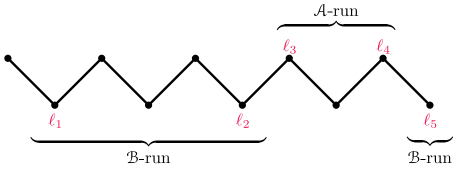
Sorting with DCJs and indels

Accumulating runs

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



Runs of a component C:



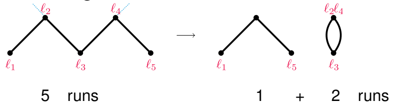
$\Lambda(C) = 3$

Optimal DCJs accumulate the labels of one run in a single $\mathcal{G}$-adjacency
$\Rightarrow$ **only 1 indel per run is necessary**

Sorting with DCJs and indels

Sorting one component individually

A DCJ can merge at most two $\mathcal{A}$-runs and two $\mathcal{B}$-runs:

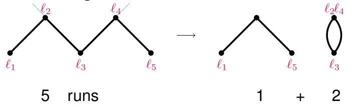


Universität Bielefeld

Sorting with DCJs and indels

Sorting one component individually

A DCJ can merge at most two $\mathcal{A}$-runs and two $\mathcal{B}$-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]

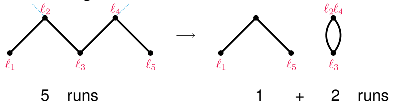
Minimum number of runs obtained by sorting C with **optimal** DCJs:

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

Sorting with DCJs and indels

Sorting one component individually

A DCJ can merge at most two $\mathcal{A}$-runs and two $\mathcal{B}$-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

$\Lambda(C)$	$\lambda(C)$
0	0
1	1
2	2
3	2
4	3
5	3
$\vdots$	$\vdots$

Indel-potential of a component C [WABI 2010]

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

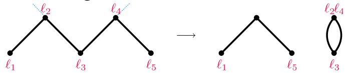
$$\lambda(C) = \left\lceil \frac{\Lambda(C)+1}{2} \right\rceil$$

Universität Bielefeld

Sorting with DCJs and indels

Sorting one component individually

A DCJ can merge at most two $\mathcal{A}$-runs and two $\mathcal{B}$-runs:



5 runs 1 + 2 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:

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

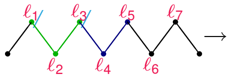
A component C can be sorted with $\boldsymbol{d(C)}$ **DCJs** $+\boldsymbol{\lambda(C)}$ **indels**.

Universität Bielefeld

Sorting with DCJs and indels

Sorting one component individually

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

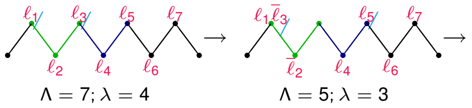


$\Lambda = 7; \lambda = 4$

Sorting with DCJs and indels

Sorting one component individually

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

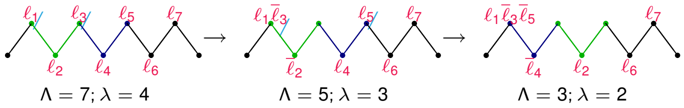


Universität Bielefeld

Sorting with DCJs and indels

Sorting one component individually

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

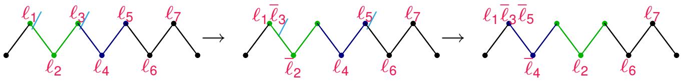


ℓ_1 ℓ_3 ℓ_5 ℓ_7
ℓ_2 ℓ_4 ℓ_6
$\Lambda = 7; \lambda = 4$
$\longrightarrow$
$\ell_1\overline{\ell}_3$ ℓ_5 ℓ_7
$\overline{\ell}_2$ ℓ_4 ℓ_6
$\Lambda = 5; \lambda = 3$
$\longrightarrow$
$\ell_1\overline{\ell}_3\overline{\ell}_5$ ℓ_7
$\overline{\ell}_4$ ℓ_2 ℓ_6
$\Lambda = 3; \lambda = 2$

Sorting with DCJs and indels

Sorting one component individually

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



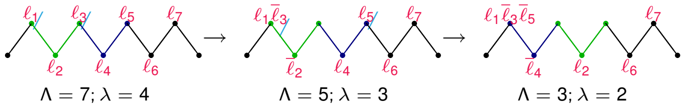
$\Lambda = 7; \lambda = 4$ $\Lambda = 5; \lambda = 3$ $\Lambda = 3; \lambda = 2$

$\Lambda(C)$	$\lambda(C)$
0	0
1	1
2 3	2 2
4 5	3 3
⋮	⋮

Sorting with DCJs and indels

Sorting one component individually

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



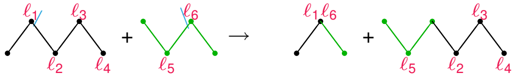
ℓ_1 ℓ_3 ℓ_5 ℓ_7
ℓ_2 ℓ_4 ℓ_6
$\Lambda = 7; \lambda = 4$
$\rightarrow$
$\ell_1 \overline{\ell}_3$ ℓ_5 ℓ_7
$\overline{\ell}_2$ ℓ_4 ℓ_6
$\Lambda = 5; \lambda = 3$
$\rightarrow$
$\ell_1 \overline{\ell}_3 \overline{\ell}_5$ ℓ_7
$\overline{\ell}_4$ ℓ_2 ℓ_6
$\Lambda = 3; \lambda = 2$

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

DCJs	**indels**	
$d(C)$	$\lambda(C)$	(min DCJs)
$d(C) + 1$	$\lambda(C) - 1$	
$d(C) + 2$	$\lambda(C) - 2$	
$\vdots$	$\vdots$	
$d(C) + \lambda(C) - 2$	2	(min indels)

Sorting with DCJs and indels

Recombination of two components

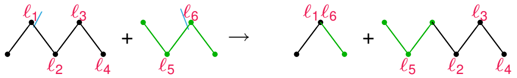


Λ:	4	+	2	$\longrightarrow$	1	+	3	$(\Delta\Lambda = -2)$
λ:	3	+	2	$\longrightarrow$	1	+	2	$(\Delta\lambda = -2)$

Universität Bielefeld

Sorting with DCJs and indels

Recombination of two components



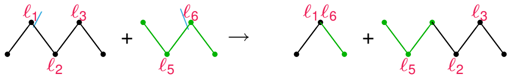
Λ: 4 + 2 $\longrightarrow$ 1 + 3 $(\Delta\Lambda = -2)$

λ: 3 + 2 $\longrightarrow$ 1 + 2 $(\Delta\lambda = -2)$

$\Lambda(C)$	$\lambda(C)$
0	0
1	1
2	2
3	2
4	3
5	3
⋮	⋮

Sorting with DCJs and indels

Recombination of two components

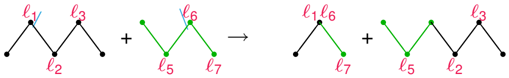


Λ:	3	+	2	$\longrightarrow$	1	+	2	$(\Delta\Lambda = -2)$
λ:	2	+	2	$\longrightarrow$	1	+	2	$(\Delta\lambda = -1)$

$\Lambda(C)$	$\lambda(C)$
0	0
1	1
2	2
3	2
4	3
5	3
$\vdots$	$\vdots$

Sorting with DCJs and indels

Recombination of two components



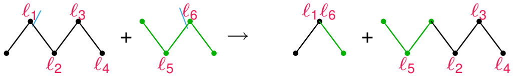
Λ:	3	+	3	$\longrightarrow$	2	+	2 $(\Delta\Lambda = -2)$
λ:	2	+	2	$\longrightarrow$	2	+	2 $(\Delta\lambda = 0)$

$\Lambda(C)$	$\lambda(C)$
0	0
1	1
2	2
3	2
4	3
5	3
$\vdots$	$\vdots$

Universität Bielefeld

Sorting with DCJs and indels

Recombination of two components



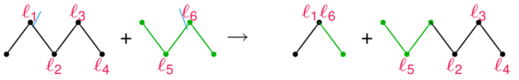
Λ: 4 + 2 $\longrightarrow$ 1 + 3 $(\Delta\Lambda = -2)$

λ: 3 + 2 $\longrightarrow$ 1 + 2 $(\Delta\lambda = -2)$

Sorting with DCJs and indels

Recombination of two components

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



Λ:	4	+	2	$\longrightarrow$	1	+	3	$(\Delta\Lambda = -2)$
λ:	3	+	2	$\longrightarrow$	1	+	2	$(\Delta\lambda = -2)$

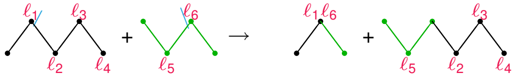
DCJ	Δd_{DCJ}
optimal	0
neutral	$+1$
counter-opt	$+2$

Sorting with DCJs and indels

Recombination of two components

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

neutral: $BB + AB \rightarrow BB + AB$



Λ: 4 + 2 $\rightarrow$ 1 + 3 $(\Delta\Lambda = -2)$

λ: 3 + 2 $\rightarrow$ 1 + 2 $(\Delta\lambda = -2)$

DCJ	Δd_{DCJ}
optimal	0
neutral	$+1$
counter-opt	$+2$

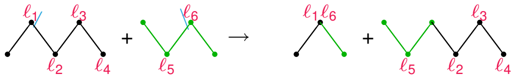
Universität Bielefeld

Sorting with DCJs and indels

Recombination of two components

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

neutral: $BB + AB \rightarrow BB + AB$



Λ: $4 + 2 \rightarrow 1 + 3$ $(\Delta\Lambda = -2)$

λ: $3 + 2 \rightarrow 1 + 2$ $(\Delta\lambda = -2)$

$\boldsymbol{\Delta d = +1 - 2 = -1}$

DCJ	Δd_{DCJ}
optimal	0
neutral	$+1$
counter-opt	$+2$

Universität Bielefeld

Sorting with DCJs and indels

A recombination saves
at most 2 indels

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

	Δd_{DCJ}	$\Delta\lambda$		Δd
optimal	0	$-\ 2$	$=$	-2
optimal	0	$-\ 1$	$=$	-1
neutral	$+1$	$-\ 2$	$=$	-1

Universität Bielefeld

Sorting with DCJs and indels

A recombination saves at most 2 indels

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

	Δd_{DCJ}	$\Delta\lambda$		Δd
optimal	0	$-\ 2$	$=$	-2
optimal	0	$-\ 1$	$=$	-1
neutral	$+1$	$-\ 2$	$=$	-1

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]

Sorting with DCJs and indels

A recombination saves at most 2 indels

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

	Δd_{DCJ}	$\Delta\lambda$		Δd
optimal	0	-2	$=$	-2
optimal	0	-1	$=$	-1
neutral	$+1$	-2	$=$	-1

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]

DCJ-indel distance formula [WABI 2010]

$$d^{id}_{\text{DCJ}}(A, B) = d_{\text{DCJ}}(A, B) + \sum_{C \in \text{AG}(A,B)} \lambda(C) - w(S)$$

Universität Bielefeld

Sorting with minimum number of DCJs

Overview

Sorting with minimum number of DCJs

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 $\boldsymbol{\Delta\lambda = 0}$) 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 $\boldsymbol{\Delta\lambda = 0}$).
   - 2.3 Apply **optimal** DCJ operations (that have $\boldsymbol{\Delta\lambda = 0}$) 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.

Universität Bielefeld

Sorting with minimum number of indels

Overview

Sorting with minimum number of indels

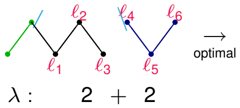
Group runs with recombinations

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

Sorting with minimum number of indels

Group runs with recombinations

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

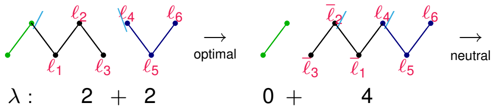


Universität Bielefeld

Sorting with minimum number of indels

Group runs with recombinations

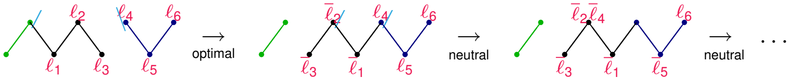
Neutral operations in components with $\lambda \geq 3$ saves one indel
$\rightarrow$ **Grouping runs in one component first is good!**



Sorting with minimum number of indels

Group runs with recombinations

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



λ : 2 + 2 0 + 4 0 + 3 ...

Sorting with minimum number of indels

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 $\boldsymbol{\Delta\lambda = -1}$.
   - 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 $\boldsymbol{\Delta\lambda = 0}$).
   - 3.3 Accumulate all runs in the smaller components derived from *C* with **optimal** DCJ operations that have $\boldsymbol{\Delta\lambda = 0}$.
   - 3.4 Apply **optimal** DCJ operations in the smaller components derived from *C* until only DCJ-sorted components exist (these DCJs have $\boldsymbol{\Delta\lambda = 0}$).
   - 3.5 **Delete/insert** all runs in the DCJ-sorted components derived from *C*.

Experiment and Discussion

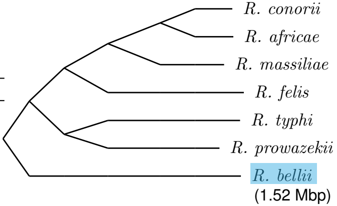
Overview

Experiment and Discussion

Comparing species of *Rickettsia*

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

species	Mbp	d^{id}_{DCJ}	MIN DCJs DCJs+indels	MIN indels DCJs+indels
R. conorii	1.27	414	261 + 153	313 + 101
R. africae	1.28	426	260 + 166	322 + 104
R. massiliae	1.36	448	276 + 172	340 + 108
R. felis	1.55	493	312 + 181	389 + 104
R. typhi	1.11	309	195 + 114	212 + 97
R. prowazekii	1.11	314	197 + 117	216 + 98
			$\Delta = 67$	$\Delta = 11$



Experiment and Discussion

Summary: Sorting with DCJs and indels

- Minimizing number of DCJs
- Minimizing number of indels

Universität Bielefeld

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
- Incoporporate substitutions in the model?

Acknowledgments

- Eyla Willing
- Jens Stoye
- Sophia Yancopoulos

Acknowledgments

- Eyla Willing
- Jens Stoye
- Sophia Yancopoulos

Thank you for your attention!

Universität Bielefeld

References

- A. Bergeron, J. Mixtacki and J. Stoye. A unifying view of genome rearrangements. In Proceedings of *WABI*, 2006.
- M. D. V. Braga and J. Stoye. The solution space of sorting by DCJ. To appear in *Journal of Computational Biology*, 2010.
- M. D. V. Braga, E. Willing and J. Stoye. Genomic distance with DCJ and indels. In Proceedings of *WABI*, 2010.
- S. Yancopoulos and R. Friedberg. Sorting Genomes with Insertions, Deletions and Duplications by DCJ. In Proceedings of *RECOMB-CG*, 2008.
- S. Yancopoulos, O. Attie and R. Friedberg. Efficient sorting of genomic permutations by translocation, inversion and block interchange. *Bioinformatics*, 2005.