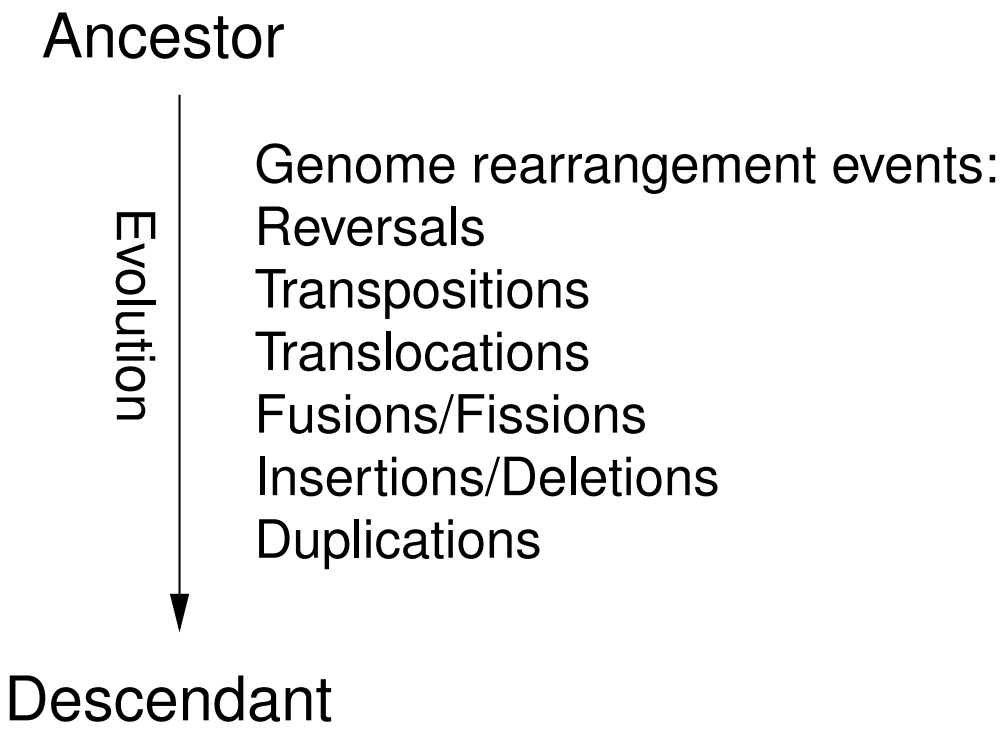
ulm university universität
uulm

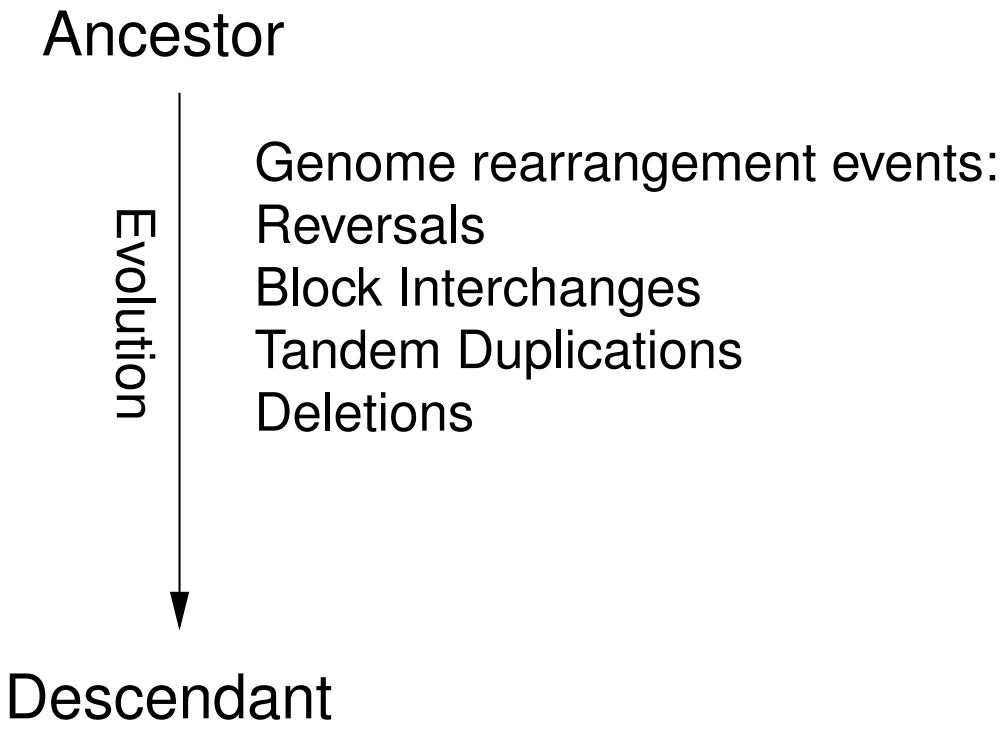
Sorting by Reversals, Block Interchanges, Tandem Duplications, and Deletions

Martin Bader | January 15, 2009

Genome Rearrangement Problems



Genome Rearrangement Problems

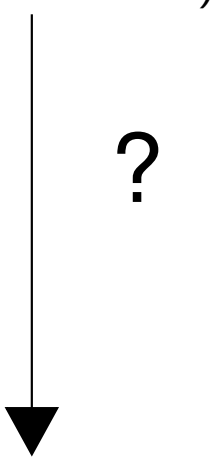


- Further restrictions: unichromosomal genomes, ancestor has no duplicated genes

Example

Sort $(\overrightarrow{1}\ \overrightarrow{2}\ \overrightarrow{3}\ \overrightarrow{4}\ \overrightarrow{5})$ into $(\overrightarrow{4}\ \overleftarrow{2}\ \overleftarrow{5}\ \overrightarrow{5}\ \overrightarrow{2}\ \overrightarrow{3})$

- $(\overrightarrow{1}\ \overrightarrow{2}\ \overrightarrow{3}\ \overrightarrow{4}\ \overrightarrow{5})$



- $(\overrightarrow{4}\ \overleftarrow{2}\ \overleftarrow{5}\ \overrightarrow{5}\ \overrightarrow{2}\ \overrightarrow{3})$

Example

Sort $(\overrightarrow{1}\ \overrightarrow{2}\ \overrightarrow{3}\ \overrightarrow{4}\ \overrightarrow{5})$ into $(\overrightarrow{4}\ \overleftarrow{2}\ \overleftarrow{5}\ \overrightarrow{5}\ \overrightarrow{2}\ \overrightarrow{3})$

- $(\overrightarrow{1}\ \underline{\overrightarrow{2}\ \overrightarrow{3}\ \overrightarrow{4}\ \overrightarrow{5}})$ transposition
- $(\overrightarrow{1}\ \overrightarrow{4}\ \overrightarrow{5}\ \overrightarrow{2}\ \overrightarrow{3})$
- $(\overrightarrow{4}\ \overleftarrow{2}\ \overleftarrow{5}\ \overrightarrow{5}\ \overrightarrow{2}\ \overrightarrow{3})$

Example

Sort $(\overrightarrow{1}\ \overrightarrow{2}\ \overrightarrow{3}\ \overrightarrow{4}\ \overrightarrow{5})$ into $(\overrightarrow{4}\ \overleftarrow{2}\ \overleftarrow{5}\ \overrightarrow{5}\ \overrightarrow{2}\ \overrightarrow{3})$

- $(\overrightarrow{1}\ \overrightarrow{2}\ \overrightarrow{3}\ \overrightarrow{4}\ \overrightarrow{5})$ transposition
- $(\overrightarrow{1}\ \overrightarrow{4}\ \overrightarrow{5}\ \overrightarrow{2}\ \overrightarrow{3})$ tandem duplication
- $(\overrightarrow{1}\ \overrightarrow{4}\ \overrightarrow{5}\ \overrightarrow{2}\ \overrightarrow{5}\ \overrightarrow{2}\ \overrightarrow{3})$

- $(\overrightarrow{4}\ \overleftarrow{2}\ \overleftarrow{5}\ \overrightarrow{5}\ \overrightarrow{2}\ \overrightarrow{3})$

Example

Sort $(\overrightarrow{1}\ \overrightarrow{2}\ \overrightarrow{3}\ \overrightarrow{4}\ \overrightarrow{5})$ into $(\overrightarrow{4}\ \overleftarrow{2}\ \overleftarrow{5}\ \overrightarrow{5}\ \overrightarrow{2}\ \overrightarrow{3})$

- $(\overrightarrow{1}\ \overrightarrow{2}\ \overrightarrow{3}\ \overrightarrow{4}\ \overrightarrow{5})$ transposition
- $(\overrightarrow{1}\ \overrightarrow{4}\ \overrightarrow{5}\ \overrightarrow{2}\ \overrightarrow{3})$ tandem duplication
- $(\overrightarrow{1}\ \overrightarrow{4}\ \overrightarrow{5}\ \overrightarrow{2}\ \overrightarrow{5}\ \overrightarrow{2}\ \overrightarrow{3})$ reversal
- $(\overrightarrow{1}\ \overrightarrow{4}\ \overleftarrow{2}\ \overleftarrow{5}\ \overrightarrow{5}\ \overrightarrow{2}\ \overrightarrow{3})$
- $(\overrightarrow{4}\ \overleftarrow{2}\ \overleftarrow{5}\ \overrightarrow{5}\ \overrightarrow{2}\ \overrightarrow{3})$

Example

Sort $(\overrightarrow{1}\ \overrightarrow{2}\ \overrightarrow{3}\ \overrightarrow{4}\ \overrightarrow{5})$ into $(\overrightarrow{4}\ \overleftarrow{2}\ \overleftarrow{5}\ \overrightarrow{5}\ \overrightarrow{2}\ \overrightarrow{3})$

- $(\overrightarrow{1}\ \overrightarrow{2}\ \overrightarrow{3}\ \overrightarrow{4}\ \overrightarrow{5})$ transposition
- $(\overrightarrow{1}\ \overrightarrow{4}\ \overrightarrow{5}\ \overrightarrow{2}\ \overrightarrow{3})$ tandem duplication
- $(\overrightarrow{1}\ \overrightarrow{4}\ \overrightarrow{5}\ \overrightarrow{2}\ \overrightarrow{5}\ \overrightarrow{2}\ \overrightarrow{3})$ reversal
- $(\overrightarrow{1}\ \overrightarrow{4}\ \overleftarrow{2}\ \overleftarrow{5}\ \overrightarrow{5}\ \overrightarrow{2}\ \overrightarrow{3})$ deletion
- $(\overrightarrow{4}\ \overleftarrow{2}\ \overleftarrow{5}\ \overrightarrow{5}\ \overrightarrow{2}\ \overrightarrow{3})$

Algorithm: Outline

- Simulate Reversals and Block Interchanges by DCJs
- Start with π, sort backwards to id

  $\Rightarrow$ apply inverse operations
- Define a lower bound on $d(\pi, id)$ based on the Breakpoint Graph
- Find operations on π that decrement the lower bound
- Apply the “best” of them (Greedy algorithm)
- If no such operation exists, use additional heuristics

The Breakpoint Graph

- Invented by Bafna and Pevzner for genomes without duplicates

The Breakpoint Graph

- Invented by Bafna and Pevzner for genomes without duplicates
- Write the identity genome on a straight line

$\overrightarrow{1}$ $\overrightarrow{2}$ $\overrightarrow{3}$ $\overrightarrow{4}$ $\overrightarrow{5}$

Example: $\pi = (\overleftarrow{3}\ \overleftarrow{2}\ \overrightarrow{1}\ \overrightarrow{4}\ \overleftarrow{5})$

The Breakpoint Graph

- Invented by Bafna and Pevzner for genomes without duplicates
- Write the identity genome on a straight line
- Replace $\overrightarrow{x}$ by $-x\ +x$

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

Example: $\pi = (\overleftarrow{3}\ \overleftarrow{2}\ \overrightarrow{1}\ \overrightarrow{4}\ \overleftarrow{5})$

The Breakpoint Graph

- Invented by Bafna and Pevzner for genomes without duplicates
- Write the identity genome on a straight line
- Replace $\overrightarrow{x}$ by $-x\ +x$
- Add boundary elements $+0$ and $-(n+1)$

$+0 \quad -1 \quad +1 \quad -2 \quad +2 \quad -3 \quad +3 \quad -4 \quad +4 \quad -5 \quad +5 \quad -6$

Example: $\pi = (\overleftarrow{3}\ \overleftarrow{2}\ \overrightarrow{1}\ \overrightarrow{4}\ \overleftarrow{5})$

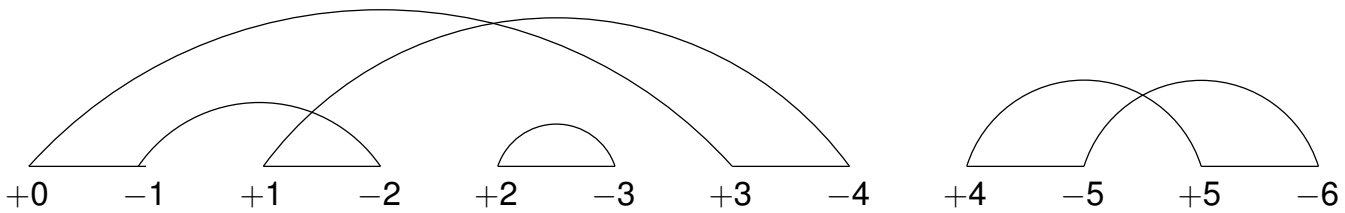
The Breakpoint Graph

- Invented by Bafna and Pevzner for genomes without duplicates
- Write the identity genome on a straight line
- Replace $\overrightarrow{x}$ by $-x\ +x$
- Add boundary elements $+0$ and $-(n+1)$
- Add reality edges from $+x$ to $-(x+1)$

Example: $\pi = (\overleftarrow{3}\ \overleftarrow{2}\ \overrightarrow{1}\ \overrightarrow{4}\ \overleftarrow{5})$

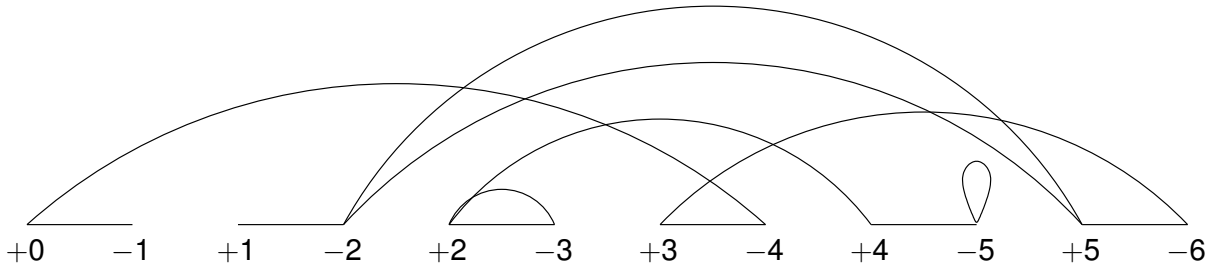
The Breakpoint Graph

- Invented by Bafna and Pevzner for genomes without duplicates
- Write the identity genome on a straight line
- Replace $\overrightarrow{x}$ by $-x\ +x$
- Add boundary elements $+0$ and $-(n+1)$
- Add reality edges from $+x$ to $-(x+1)$
- Add desire edges according to adjacencies in π



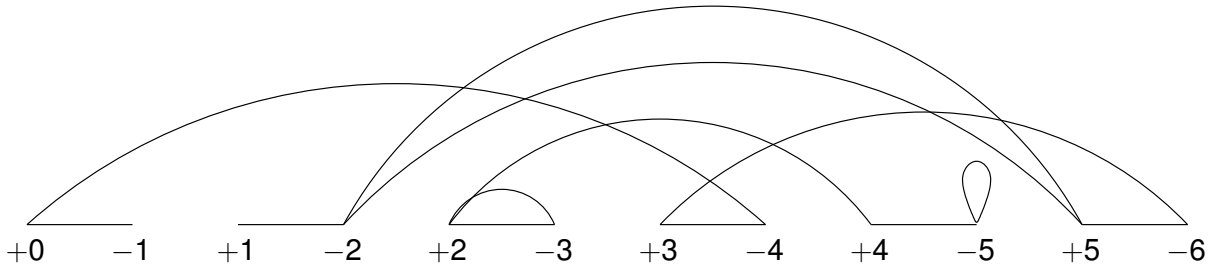
Example: $\pi = (\overleftarrow{3}\ \overleftarrow{2}\ \overrightarrow{1}\ \overrightarrow{4}\ \overleftarrow{5})$

The Breakpoint Graph revisited



Example: $\pi = (\overrightarrow{4}\ \overleftarrow{2}\ \overleftarrow{5}\ \overrightarrow{5}\ \overrightarrow{2}\ \overrightarrow{3})$

The Breakpoint Graph revisited

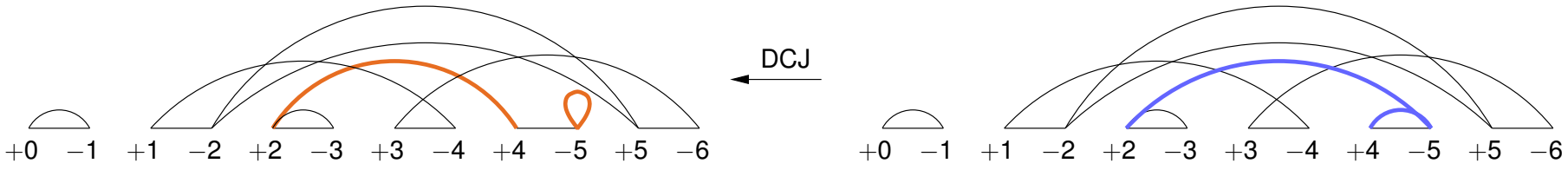


Example: $\pi = (\overrightarrow{4}\ \overleftarrow{2}\ \overleftarrow{5}\ \overrightarrow{5}\ \overrightarrow{2}\ \overrightarrow{3})$

- Multiplicity of an element x: number of occurrences of x in π
- Multiplicity of a desire edge (v, w): number of desire edges (v, w) in the breakpoint graph
- Loop: Desire edge (v, v)
- Component: Connected component (graph theory)
- 1-bridge: Desire edge that can be removed to increase the number of components
- 2-bridge: Pair of desire edges that can be removed to increase the number of components

Effects of Operations: DCJ

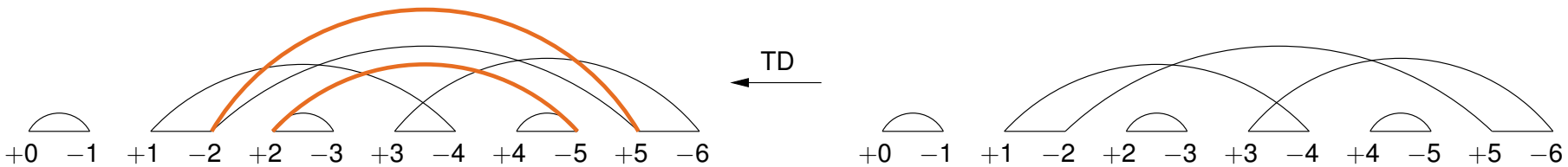
- Cuts two desire edges and rejoins the ends
- Can split a component with a 2-bridge or two 1-bridges
- Can remove up to two loops



$$(\overrightarrow{1}\ \overrightarrow{4}\ \overleftarrow{2}\ \overleftarrow{5}\ \overrightarrow{5}\ \overrightarrow{2}\ \overrightarrow{3}) \leftarrow (\overrightarrow{1}\ \overrightarrow{4}\ \overrightarrow{5}\ \overrightarrow{2}\ \overrightarrow{5}\ \overrightarrow{2}\ \overrightarrow{3})$$

Effects of Operations: Inverse Tandem Duplication

- Removes desire edge between segment end and segment start
- Removes desire edges inside the segment
- The latter desire edges have a multiplicity ≥ 2
- Splits a component if and only if the former desire edge is a 1-bridge
- Can remove one loop
- Precondition: Two consecutive identical segments



$$(\overrightarrow{1}\ \overrightarrow{4}\ \overrightarrow{5}\ \overrightarrow{2}\ \overrightarrow{5}\ \overrightarrow{2}\ \overrightarrow{3}) \leftarrow (\overrightarrow{1}\ \overrightarrow{4}\ \overrightarrow{5}\ \overrightarrow{2}\ \overrightarrow{3})$$

Effects of Operations: Inverse Deletion

- Removes one desire edge
- Inserts arbitrary desire edges
- Can split a component if the removed desire edge is a 1-bridge
- Can remove one loop

$$(\overrightarrow{4}\ \overleftarrow{2}\ \overleftarrow{5}\ \overrightarrow{5}\ \overrightarrow{2}\ \overrightarrow{3}) \leftarrow (\overrightarrow{1}\ \overrightarrow{4}\ \overleftarrow{2}\ \overleftarrow{5}\ \overrightarrow{5}\ \overrightarrow{2}\ \overrightarrow{3})$$

A lower bound

- The breakpoint graph of id has $n+1$ components and no loops
- Thus, the distance $d(\pi, id)$ can be bounded by

$$d(\pi, id) \geq lb(\pi) = n + 1 - C(\pi) + \sum_{Components} \lceil \frac{S_i}{2} \rceil$$

  where $C(\pi)$ is the number of components and S_i is the number of vertices with a loop in component C_i
- $lb(\pi) = 0$ if and only if $\pi = id$, otherwise $lb(\pi) > 0$

Additional Heuristics

Search for

- Tandem duplications that do not change the lower bound
- Reversals that create adjacencies
- Sequences for elements with multiplicity ≥ 3
- Sequences for the few remaining cases

Additional Heuristics

Search for

- Tandem duplications that do not change the lower bound
- Reversals that create adjacencies
- Sequences for elements with multiplicity ≥ 3
- Sequences for the few remaining cases

Which of those is the best?

- Maximize the number of adjacencies
- Bring multiplicity of elements close to 1

Additional Heuristics

Search for

- Tandem duplications that do not change the lower bound
- Reversals that create adjacencies
- Sequences for elements with multiplicity ≥ 3
- Sequences for the few remaining cases

Which of those is the best?

- Maximize the number of adjacencies
- Bring multiplicity of elements close to 1

$$\Rightarrow \text{Maximize}$$

$$\tau(\pi) := \#\textit{adjacencies} - 2 \cdot (\#\textit{missing elements} + \#\textit{duplicated elements})$$

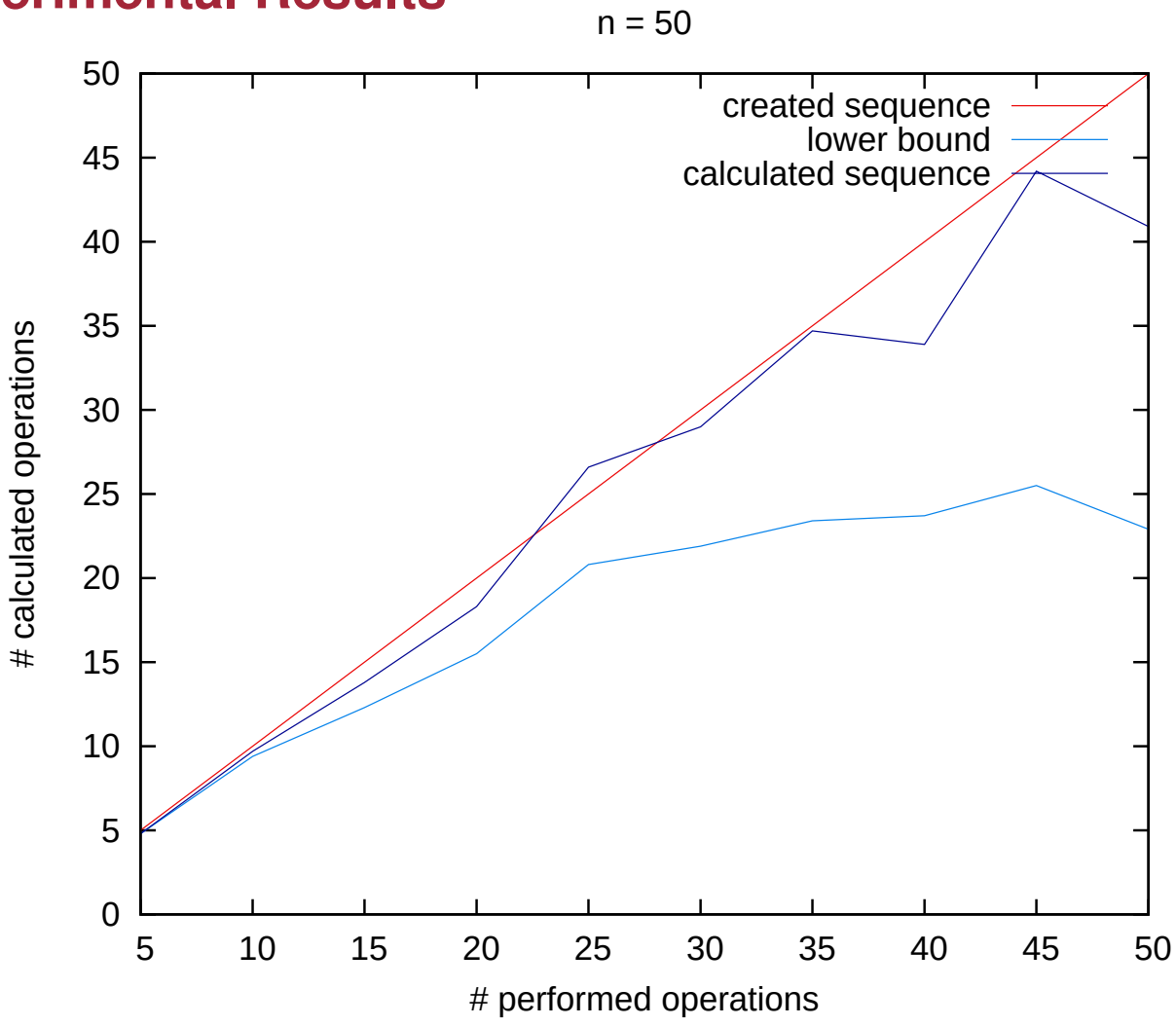
Algorithm: Pseudocode

while $\pi \neq id$ **do**
  Find all operations that decrease $lb(\pi)$
  if operation found **then**
    apply an operation that maximizes $\tau(\pi)$
  else
    find tandem duplications
    find sequences for segments with multiplicity ≥ 3
    find operations that create adjacencies
    find sequences for the remaining cases
    apply a sequence that maximizes $\tau(\pi)$
  end if
end while

Experimental Results

- Start with *id* of size n ($n \in \{20, 50, 80, 100\}$)
- Apply αn random operations ($\alpha \in [0, 1]$))
- Use algorithm to reconstruct this sequence
- Compare # applied operations to # calculated operations

Experimental Results



Conclusion and Future work

- Algorithm works well for small values of n and α
- Possible improvements:
  - Tighter lower bound
  - Finding an upper bound
  - Improving the heuristics
  - Extending the algorithm to multichromosomal genomes

Acknowledgements

- Thanks to Sophia Yancopoulos for the initial idea of combining DCJ and duplications
- Thanks to Michal Ozery-Flato for invaluable discussion

Thanks!

Thank you for your attention!

Algorithm: Completeness

- Define

$$\tau(\pi) := \#\textit{adjacencies} - 2 \cdot (\#\textit{missing elements} + \#\textit{duplicated elements})$$

- $\tau(\pi)$ is maximized for $\pi = id$
- All additional heuristics increase $\tau(\pi)$ and do not decrease $lb(\pi)$
- Between two operations that decrease the lower bound, only a finite number of operations can be applied
- Only a finite number of operations that decrease the lower bound can be applied

The Double Cut and Join Operator (DCJ)

- Invented by Yancopoulos et al. (2005)
- Cuts the genome at two positions, and rejoins the ends
- Reversals can be simulated by one DCJ
- Block interchanges can be simulated by two DCJs (via circular intermediate)
- Circular intermediates must be absorbed by the next operation