
# GOLDEN GENOME

## OR HOW TO CREATE FANCY GRAPH PROBLEMS

FABIAN EXTERNBRINK

SCADS & BIOINFORMATIK, LEIPZIG

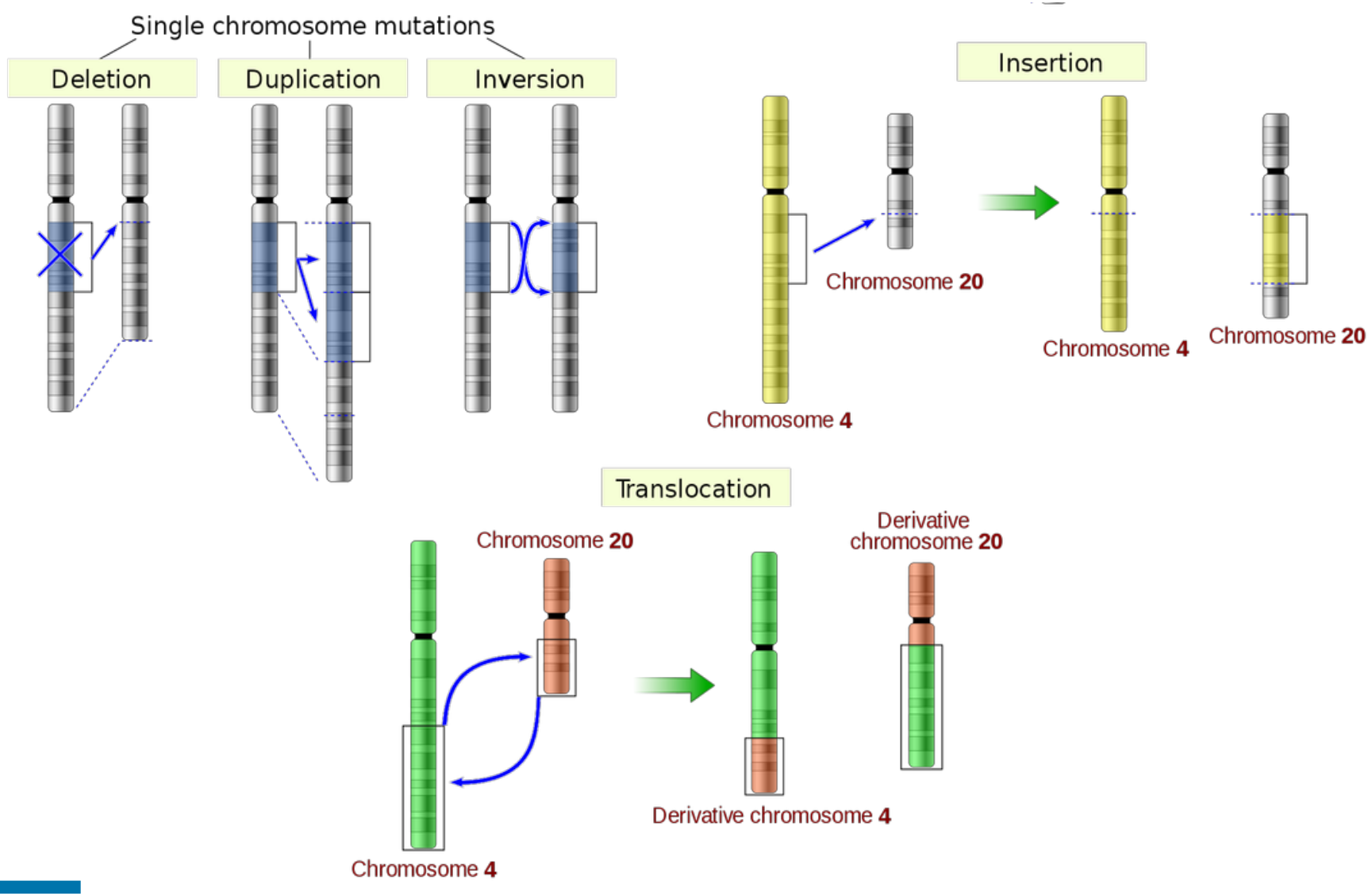
FABIAN@BIOINF.UNI-LEIPZIG.DE

ScaDS DRESDEN LEIPZIG

# GENOMIC DATA

- many new genomes sequenced in the past years
- current number of genomes in the NCBI database:
  - 2,579 eukaryotic genomes
  - 57,070 prokaryotic genomes
- Many studies target only one organism.
- However, similar/same data for different organisms already now available.
- In the next years, more studies between organisms
- Problem: How to compare data from different genomes?

# CHROMOSOME MUTATIONS



ScaDS DRESDEN LEIPZIG

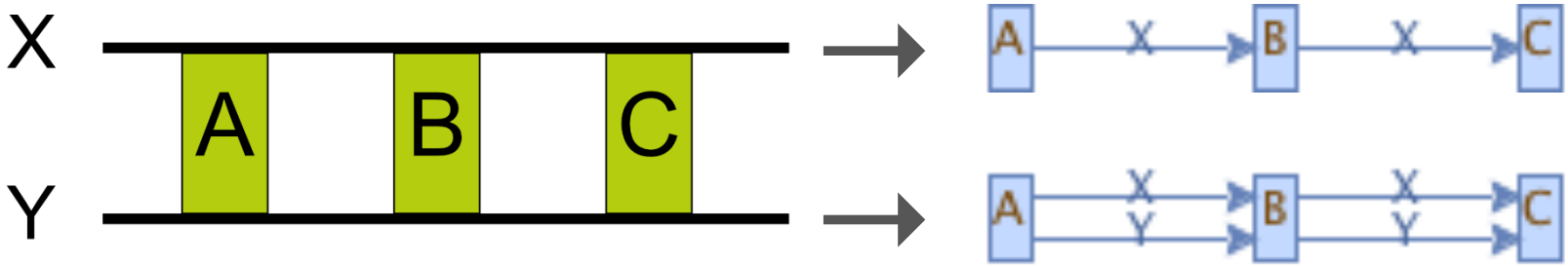
# GOLDEN GENOME

- creating a artificial “golden” genome that fulfill three conditions:
  1. is linear alignable to all contained genomes
  2. keep as much information as possible of all genomes
  3. has no redundancy
- roughly represents ancestral genome
- Starting point: local alignments
- Use pre-computed genome alignments in multiple alignment format (MAF)

ScaDS DRESDEN LEIPZIG

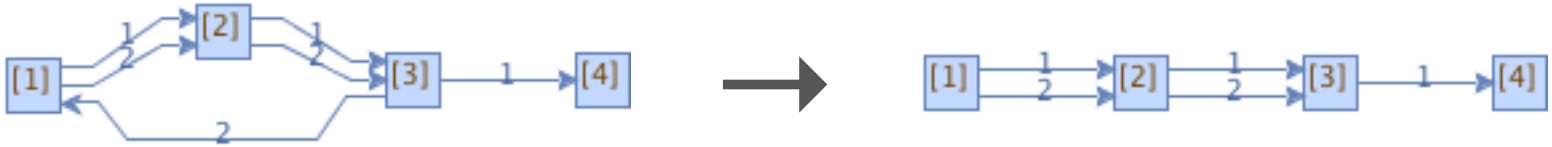
# GRAPH GENERATION

- One MAF-block is one local multiple alignment
- Every MAF-block is one vertex in the graph.
- Edges indicate the genomic order of the vertices for each species.
- If MAF-block B is genomic successor of MAF-block A in species X then there is a directed edge from A to B with label X.

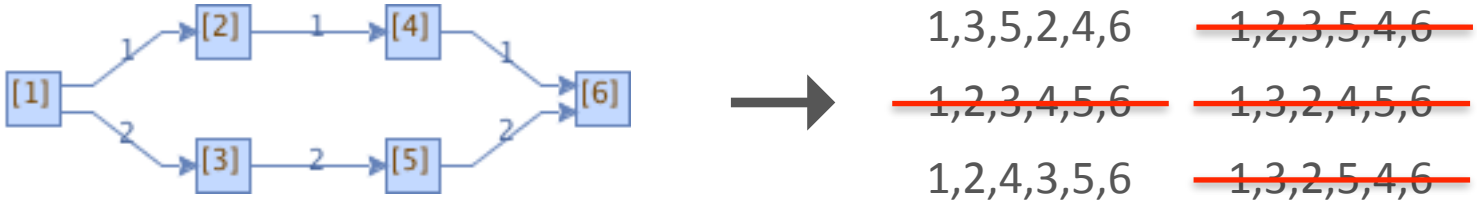


# GOLDEN GENOME TASK

- create a linear order of the vertices (golden genome order)
- keep most edges
  - Maximum Acyclic Subgraph (NP complete)




- keep neighborhood where it is possible
  - Topological sorting



ScaDS DRESDEN LEIPZIG

# MULTIGRAPH VS. WEIGHTED GRAPH

- Transform multigraph in a weighted graph
  - Count edges between to vertices.
  - If at least one exists add edge to weighted graph with the number of edges between the vertices as weight
- less edges in weighted graph, so many algorithm are faster
- Different views on golden genom graph with different information.
- Both have advantages

ScaDS DRESDEN LEIPZIG

# HOW TO DESCRIBE THE GRAPHS

- vertex and edge count multigraph
- edge count weighted graph
- In-Degree and Out-Degree multigraph
- In-Degree and Out-Degree weighted graph
- Harmonic Centrality weighted graph

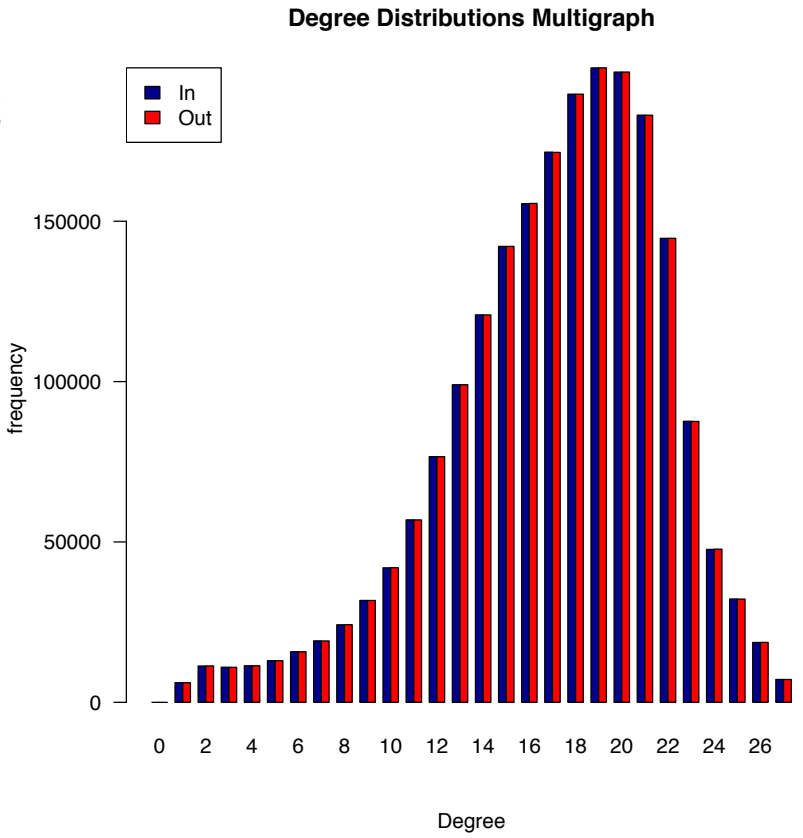
$$H(x)=\sum_{y\neq x}\frac{1}{d(y,x)}$$

- Betweenness Centrality weighted graph

$$B(x)=\sum_{s\neq x\neq t}\frac{\sigma_{st}(x)}{\sigma_{st}}$$

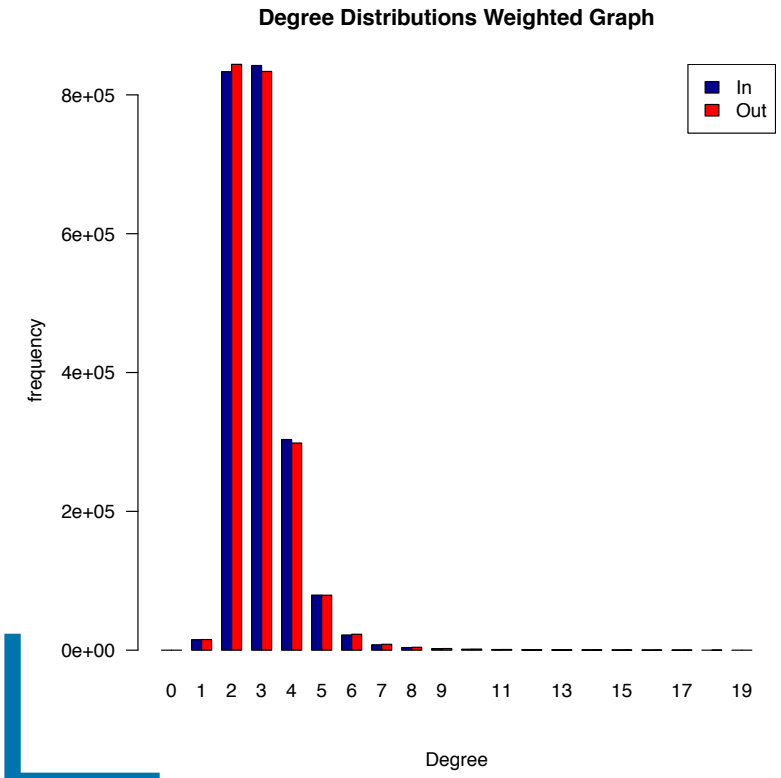
# 27 WAY INSECT

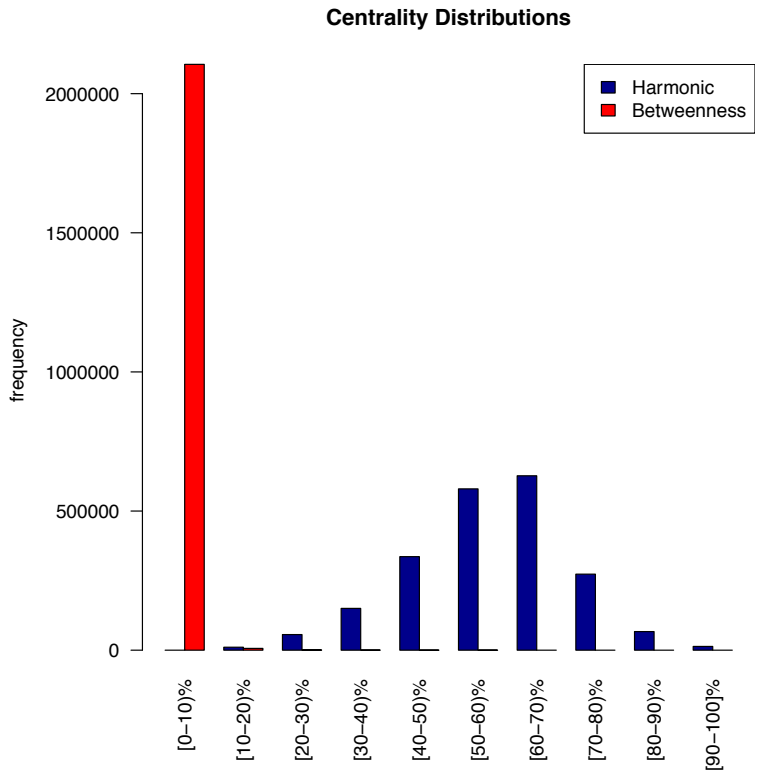
- 2,112,962 vertices
- 36,130,309 edges
- 1 connected component



# 27 WAY INSECT

- 2,112,962 vertices
- 6,097,044 edges



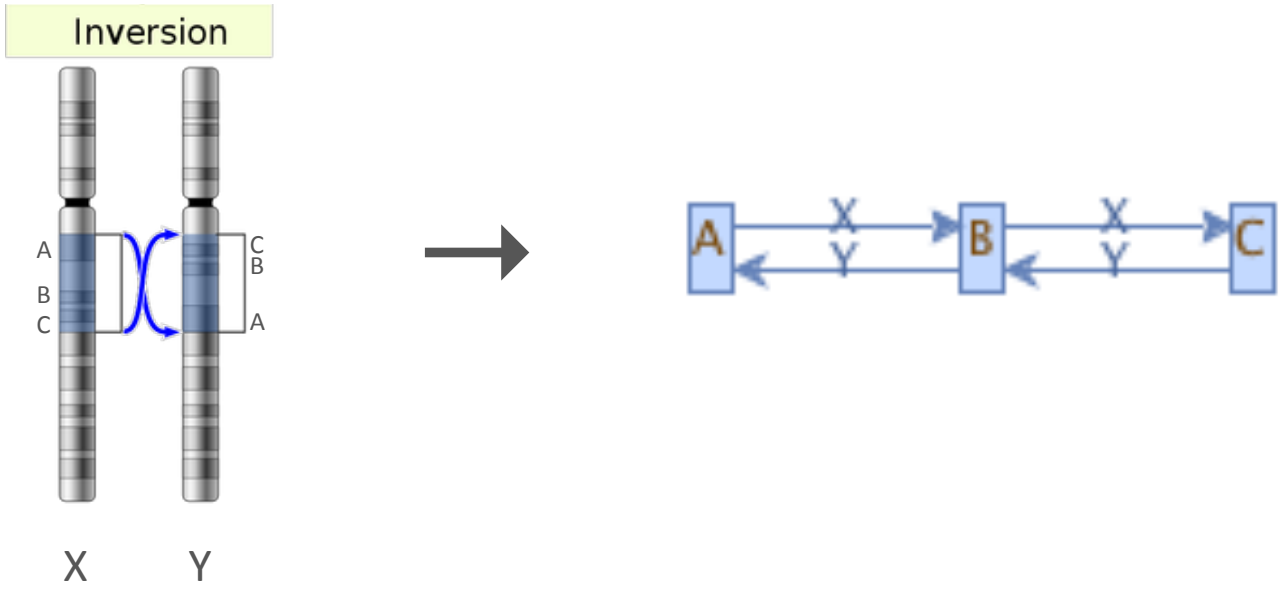
ScaDS DRESDEN LEIPZIG

# REDUCER

- reduce “noise” in the graph to make it easier to handle ( i.e. finding conserved regions)
- local inversions: remove simple cycles
- “almost” long, local alignments: collapse co-linear chain
- artificial sinks and sources due to incomplete genome sequences

# MINI CYCLE REMOVAL

- local inversions lead to mini cycles
- two adjacent vertices with edges in both directions
- Handle every mini cycle by removing all edges in one directions

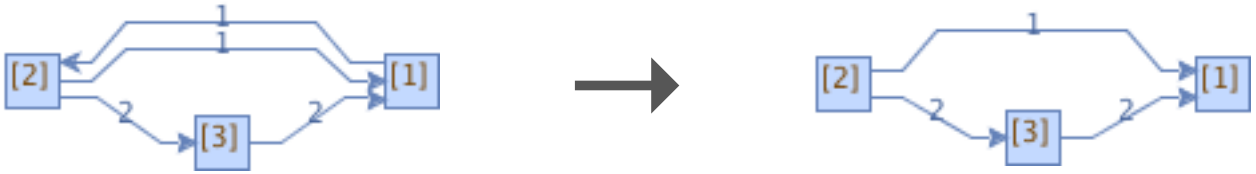


# MINI CYCLE REMOVAL

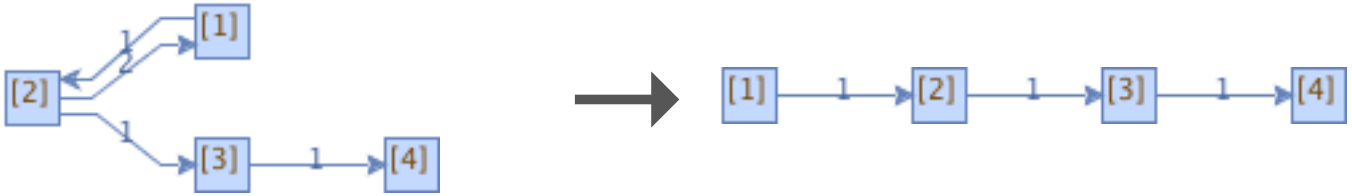
- Delete the direction that:

1. contains fewer edges


2. is part of another cycle



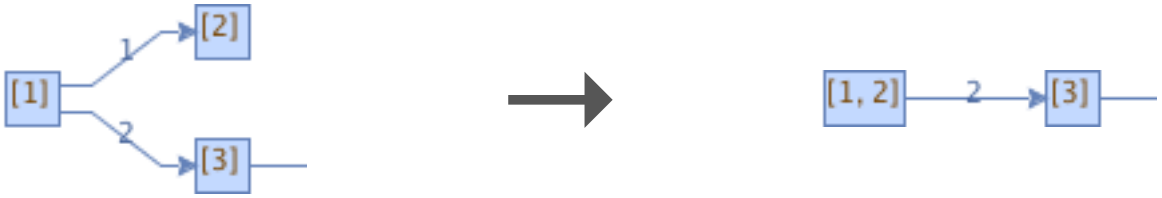

3. is not supported by alternative paths



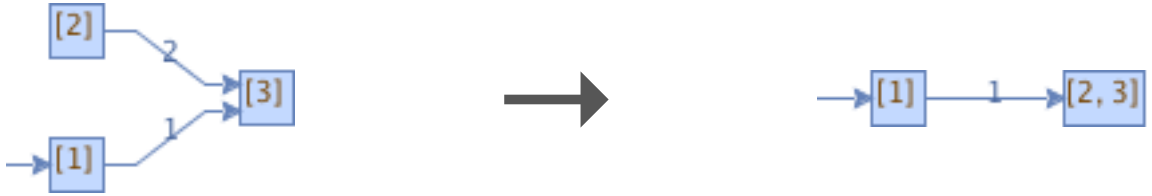
# SINK AND SOURCE REMOVAL

- Sinks with one predecessor can be collapsed into the predecessor.
- at most one neighborhood is destroyed
- The same hold for a source with one successor.

sink




source



ScaDS DRESDEN LEIPZIG

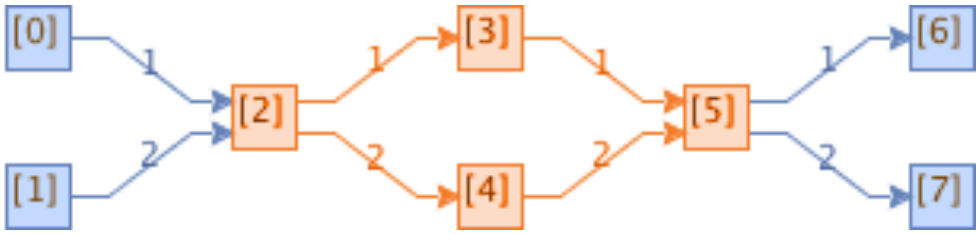
# 2 VERTEX REDUCER

- “almost” long, local alignments: collapse  co-linear chain
- If a vertex *x* has exactly one successor *y* and *x* is the only predecessor of *y* then the reducer is applied.
- Combine these two vertices to one vertex

ScaDS DRESDEN LEIPZIG

# CLOSED DAG REDUCER

- Closed DAG:
  1. Subgraph that is a **D**irected **A**cyclic **G**raph
  2. No edges from the subgraph to remaining graph except to one source and from one sink of the subgraph.
- Closed DAGs can be ordered independent from the rest of the graph.
- In any order a closed DAG is placed as an atomic component.



# REDUCER

- every reduction can create new options for further reduction
- all four reducers are applied until a fixed point is reached
- running time of all reducers is $O(|V|)$
- in the worst case the outer loop is repeated $|V|-1$ times
- in total $O(|V|^2)$ running time

ScaDS DRESDEN LEIPZIG

# MAXIMUM ACYCLIC SUBGRAPH

- Given a directed graph create the DAG with the maximum number of edges.
- $G' = (V,E')$ with $E' \subseteq E$ and $|E'|$ maximal
- one of Richard M. Karp's 21 NP-complete problems (1972)
- many heuristics are published
- each has different properties (choose one that fits our problem best)

ScaDS DRESDEN LEIPZIG

# EADES ALGORITHM (1993)

- Genome graphs are sparse.
- running time: O(|E|)
- performance: |E'|>|E|/2+|V|/6
- Simple, greedy algorithm
- Creates a vertex order by removing all sinks and sources.
- when no sink or source exists, remove the vertex with the maximum of: out-degree - in-degree
- Transform the vertex order in a subgraph by keeping all edges that goes from a smaller to a bigger vertex in the order

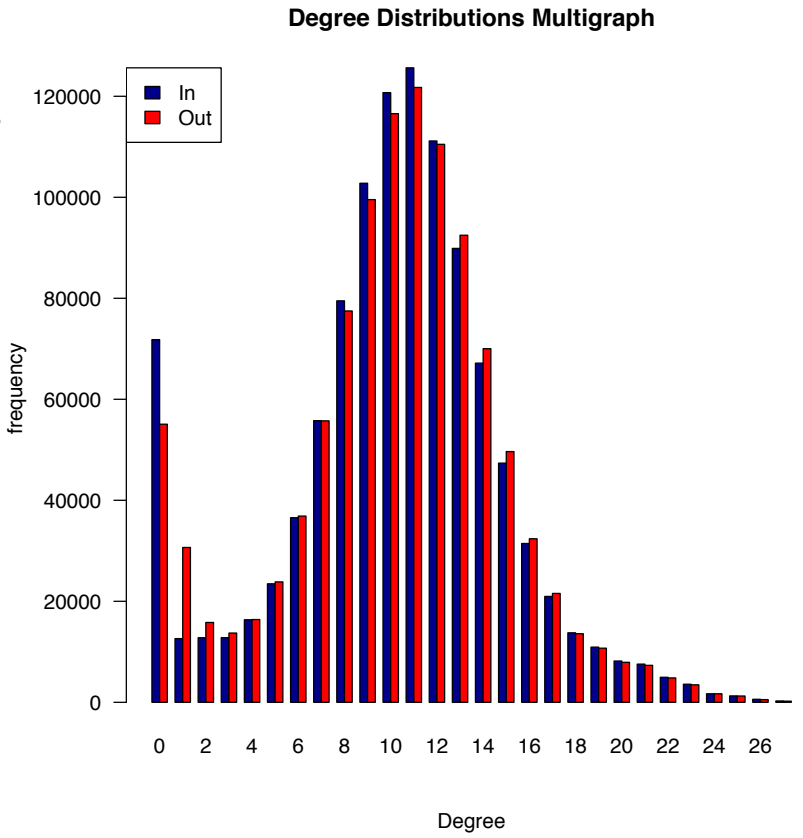
ScaDS DRESDEN LEIPZIG

# POST CYCLE REMOVAL

- new possibilities for reduction
- a DAG is cycle-free: cycle removal not required
- the other three reducers are reapplied to the DAG
- reduce until fixed point is reached

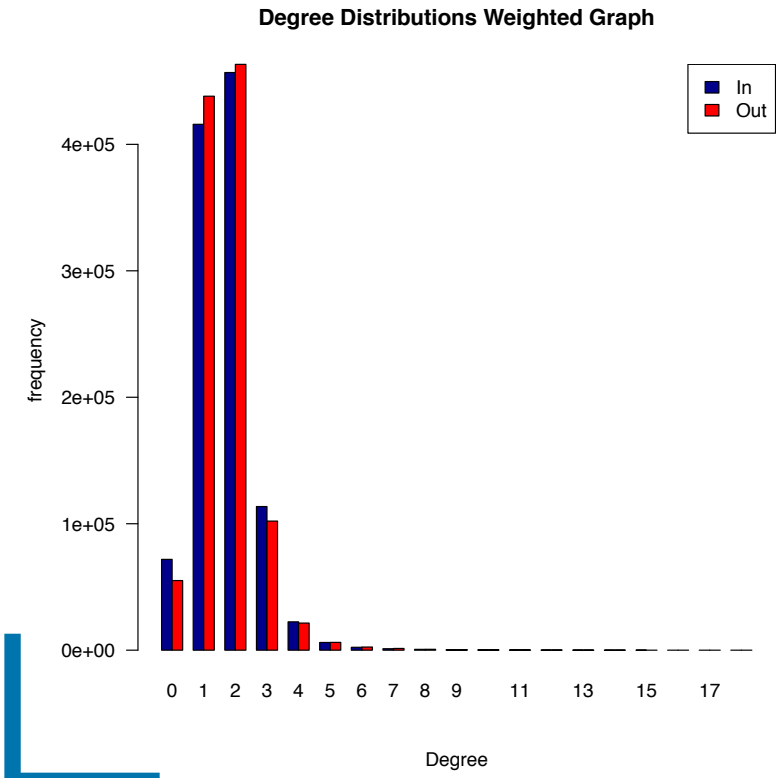
# 27 WAY INSECT NO CYCLE REDUCED

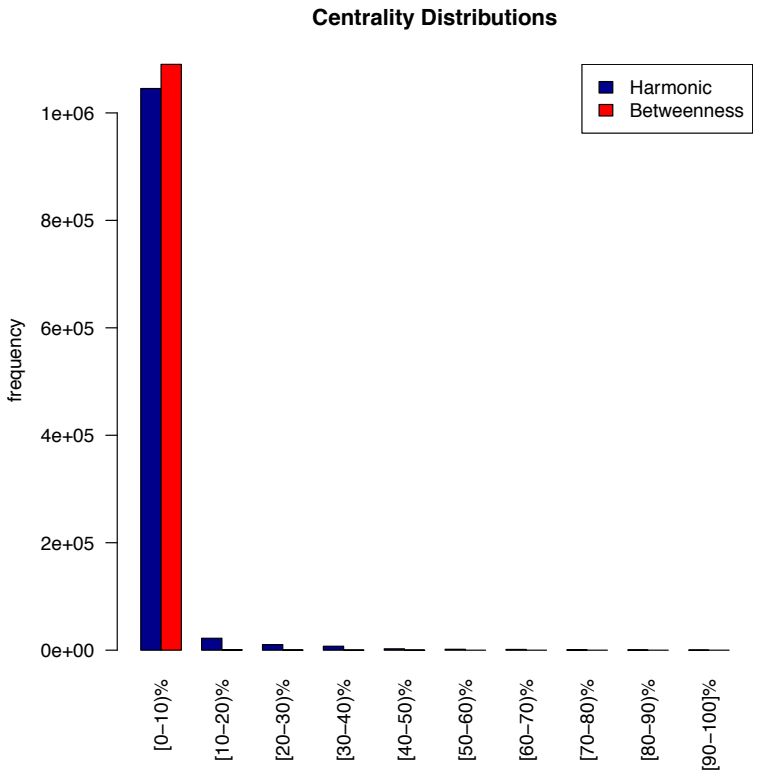
- 1,091,540 vertices
- 11,095,894 edges
- 1 connected component



# 27 WAY INSECT NO CYCLE REDUCED

- 1,091,540 vertices
- 1,826,626 edges



# LINEAR ORDER

theory:

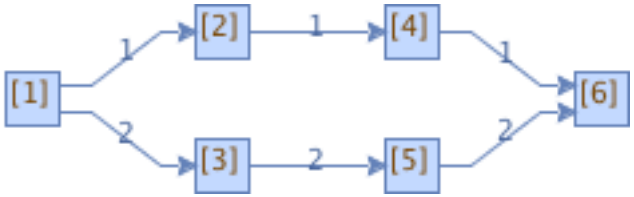
- genomes in higher species are linear
- reduced DAG can be transformed into a linear order using topological sorting

implementation:

- special constraint to keep the neighborhood
- use a modified version of Kahn's algorithm (1962)

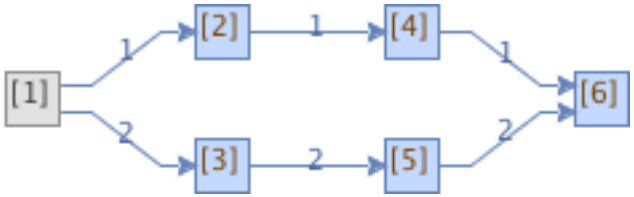
# MODIFIED KAHN'S ALGORITHM

- for each source v:
  - insert v to the order at the end
  - save successors of v and remove v
  - for all successors s of v:
    - if s is now a source
      - continue with s as new source
- after the algorithm: either no edge is left or graph was not a DAG



# MODIFIED KAHN'S ALGORITHM

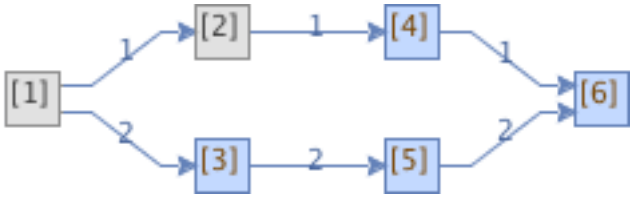
- for each source v:
  - insert v to the order at the end
  - save successors of v and remove v
  - for all successors s of v:
    - if s is now a source
      - continue with s as new source
- after the algorithm: either no edge is left or graph was not a DAG




1

# MODIFIED KAHN'S ALGORITHM

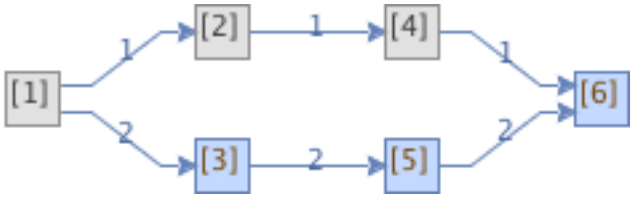
- for each source v:
  - insert v to the order at the end
  - save successors of v and remove v
  - for all successors s of v:
    - if s is now a source
      - continue with s as new source
- after the algorithm: either no edge is left or graph was not a DAG




1,2

# MODIFIED KAHN'S ALGORITHM

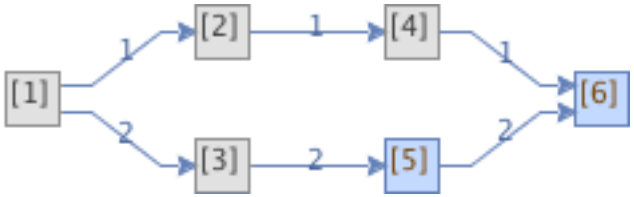
- for each source v:
  - insert v to the order at the end
  - save successors of v and remove v
  - for all successors s of v:
    - if s is now a source
      - continue with s as new source
- after the algorithm: either no edge is left or graph was not a DAG




1,2 ,4

# MODIFIED KAHN'S ALGORITHM

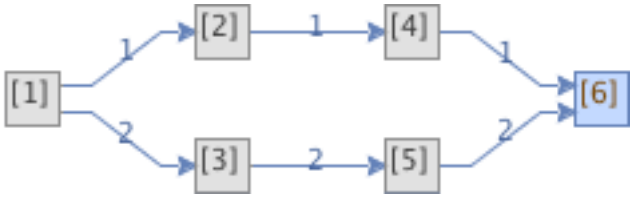
- for each source v:
  - insert v to the order at the end
  - save successors of v and remove v
  - for all successors s of v:
    - if s is now a source
      - continue with s as new source
- after the algorithm: either no edge is left or graph was not a DAG




1,2 ,4,3

# MODIFIED KAHN'S ALGORITHM

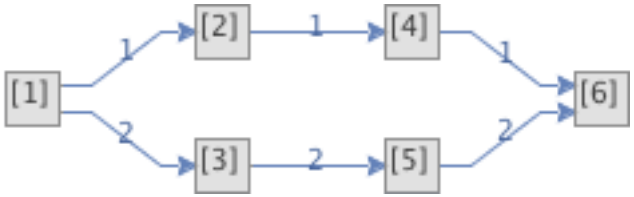
- for each source v:
  - insert v to the order at the end
  - save successors of v and remove v
  - for all successors s of v:
    - if s is now a source
      - continue with s as new source
- after the algorithm: either no edge is left or graph was not a DAG




1,2 ,4,3 ,5

# MODIFIED KAHN'S ALGORITHM

- for each source v:
  - insert v to the order at the end
  - save successors of v and remove v
  - for all successors s of v:
    - if s is now a source
      - continue with s as new source
- after the algorithm: either no edge is left or graph was not a DAG




1,2 ,4,3 ,5,6

ScaDS DRESDEN LEIPZIG

# FURTHER WORK

- near future: optimize implementation
- investigate additional heuristics
- alternative order for different views on the golden genome
- analyze evolutionary events with the help of the golden genome

ScaDS
DRESDEN LEIPZIG

# THANK YOU FOR YOUR ATTENTION

ScaDS DRESDEN LEIPZIG

# DATASETS

- from UCSC
- small test data
  - 4 Way Bacteria, 6,222 blocks
- medium data
  - 27 Way Insect, 2,115,903 blocks
- big data
  - 100 Way Vertebrata, 109,850,411 blocks

ScaDS DRESDEN LEIPZIG

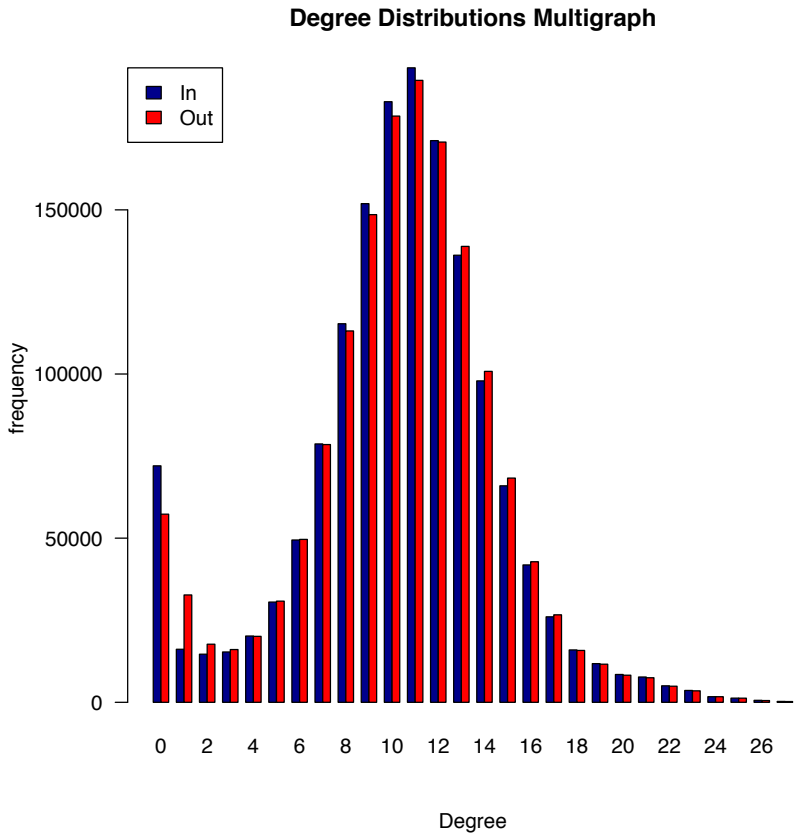
# MAF-FILES

- used by all major databases
- multiple alignment format
- contains MAF-blocks
- One MAF-block is one local multiple alignment

```
a score=3870.000000
s dm6.chr2L             2724 60 + 23513712 TCTTATTTTACCGCAAACCCAAatcgacaatgcacgacaga----ggaa-gcagaacagatattt
s droSim1.chr2L         1448 60 + 22036055 TCTTATATTACCGCAAGCCCAAAAtgacaacgcacgacaag----gaga-gcaagagagatagtc
s droSec1.super_14      1380 65 +  2068291 tctctctttagCGACTACTTAGGGTCGCAATATGGAATAAAGGCTGAGACGCAAATTAAATATTT
```

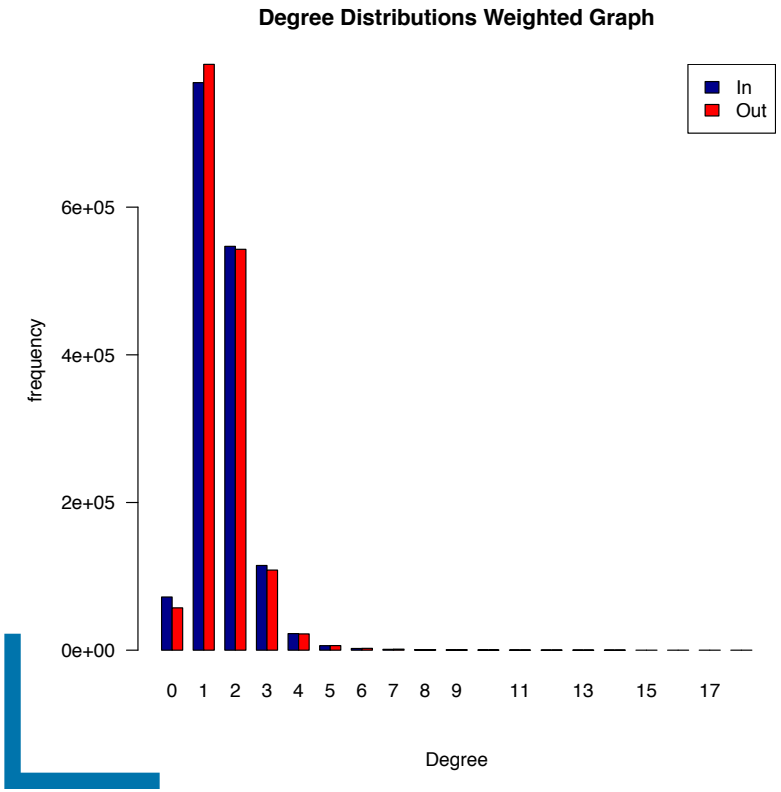
# 27 WAY INSECT NO CYCLE

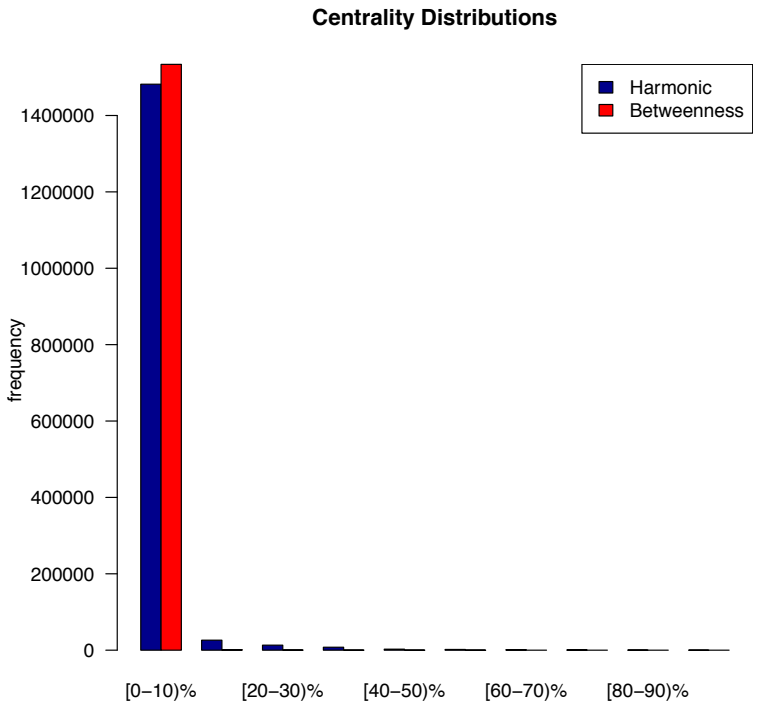
- 1,535,811 vertices
- 15,843,948 edges
- 1 connected component



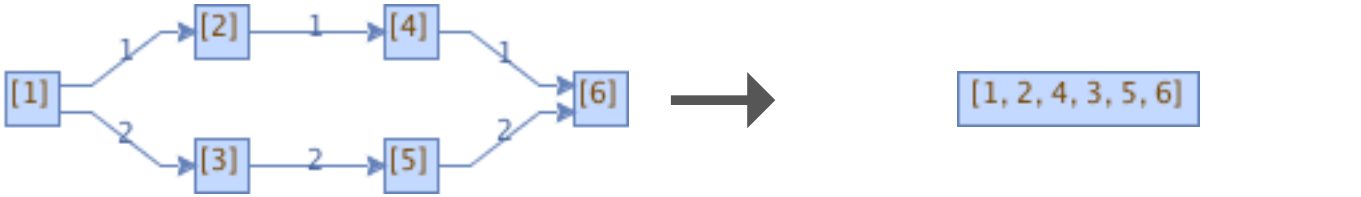
# 27 WAY INSECT NO CYCLE

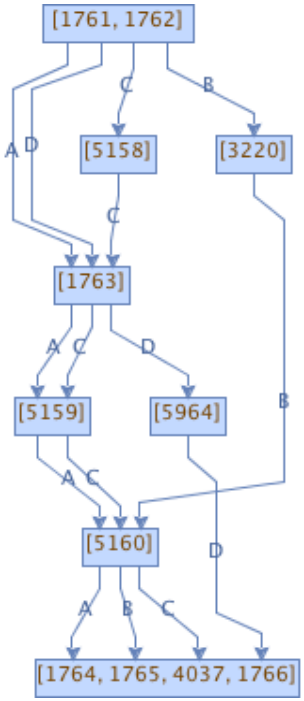
- 1,535,811 vertices
- 2,363,118 edges

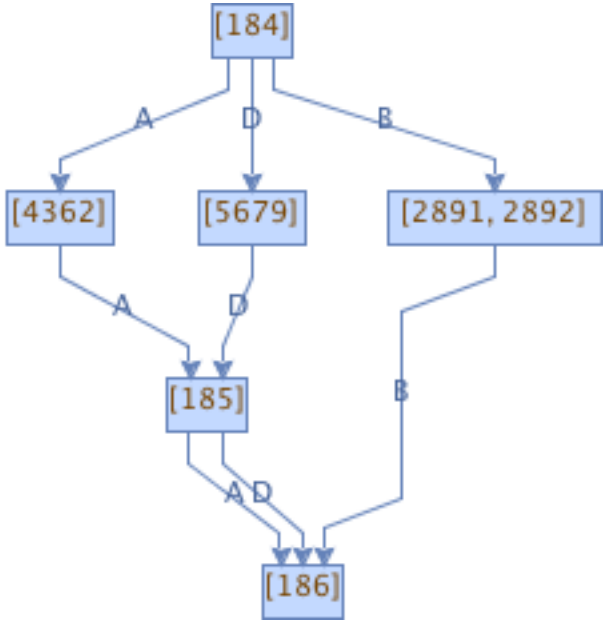


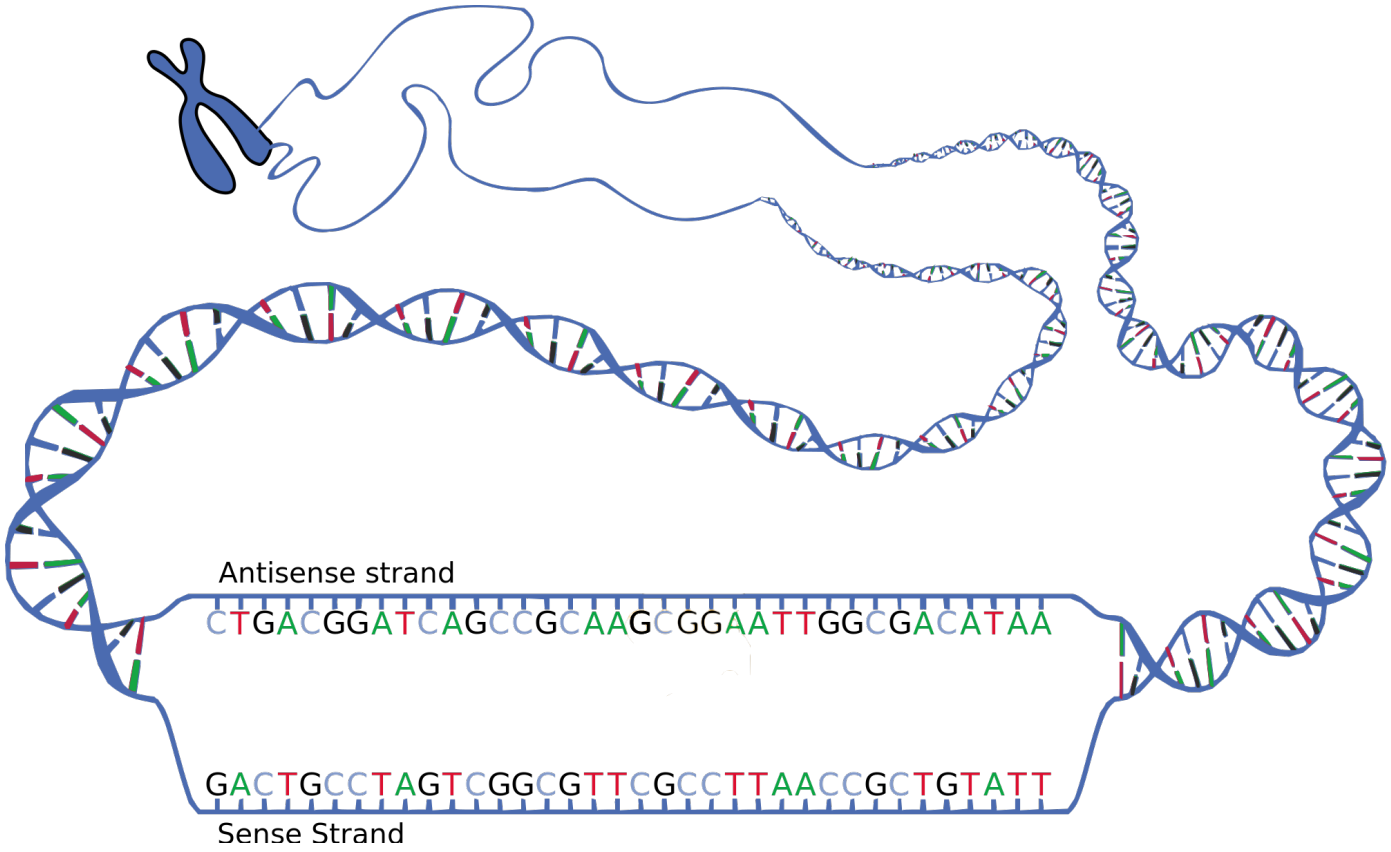
# CLOSED DAG REDUCER



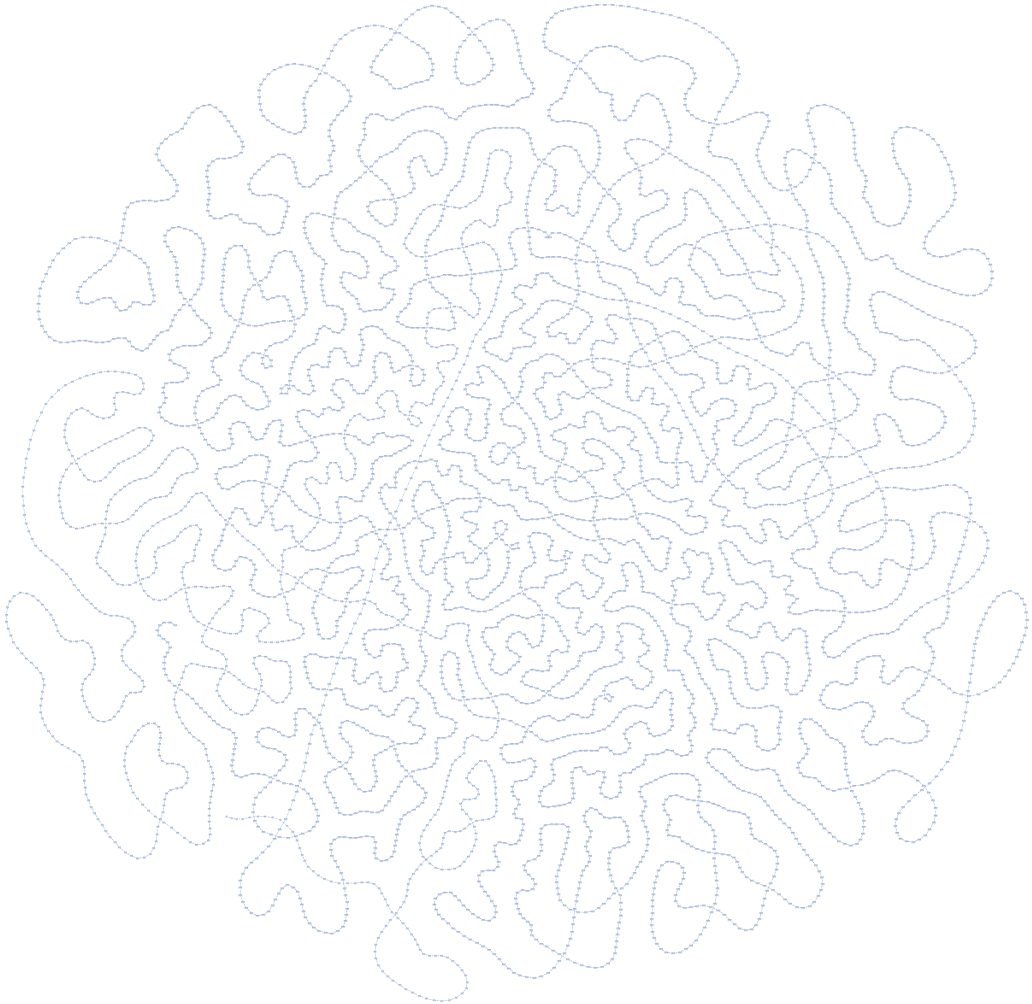



# GENOME


Antisense strand
CTGACGGATCAGCCGCAAGCGGAATTGGCGACATAA
GACTGCCTAGTCGGCGTTCGCCTTAACCGCTGTATT
Sense Strand

ScaDS DRESDEN LEIPZIG

# PERFECT GRAPH

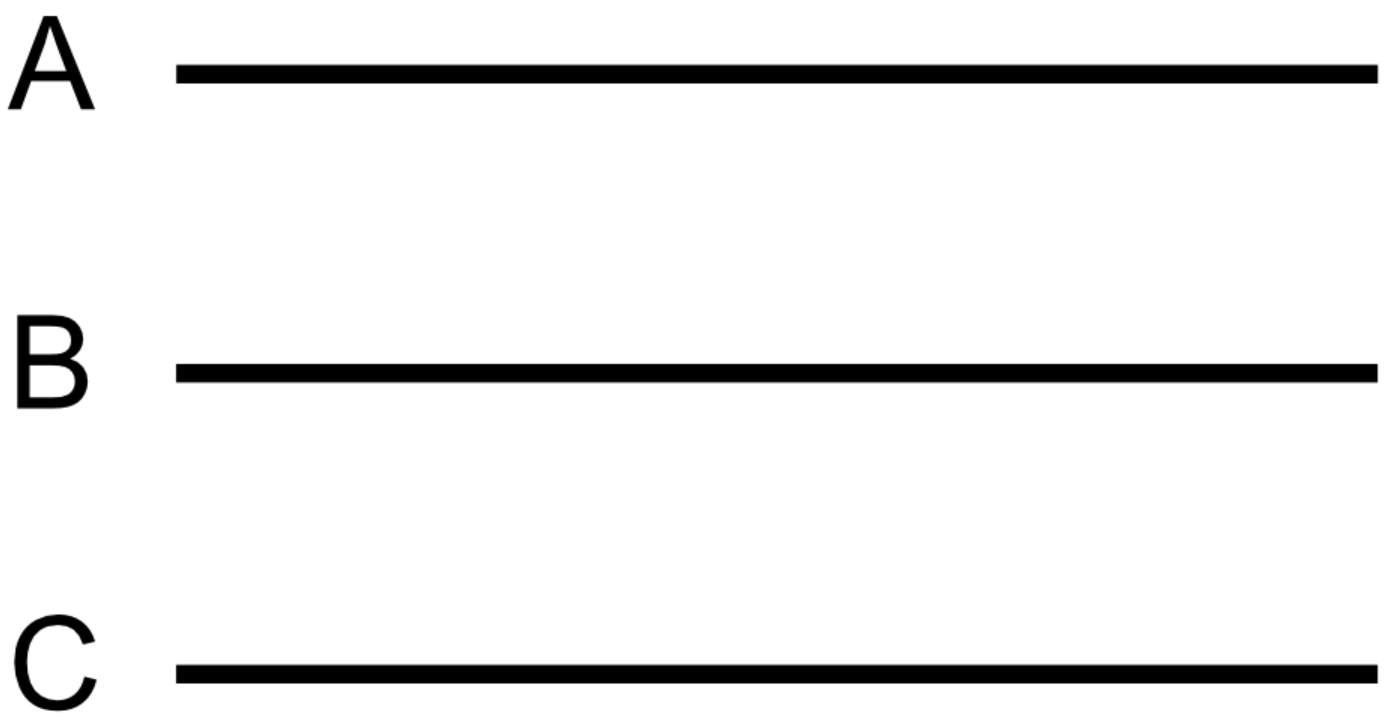


ScaDS DRESDEN LEIPZIG

# LOCAL MULTIPLE ALIGNMENTS

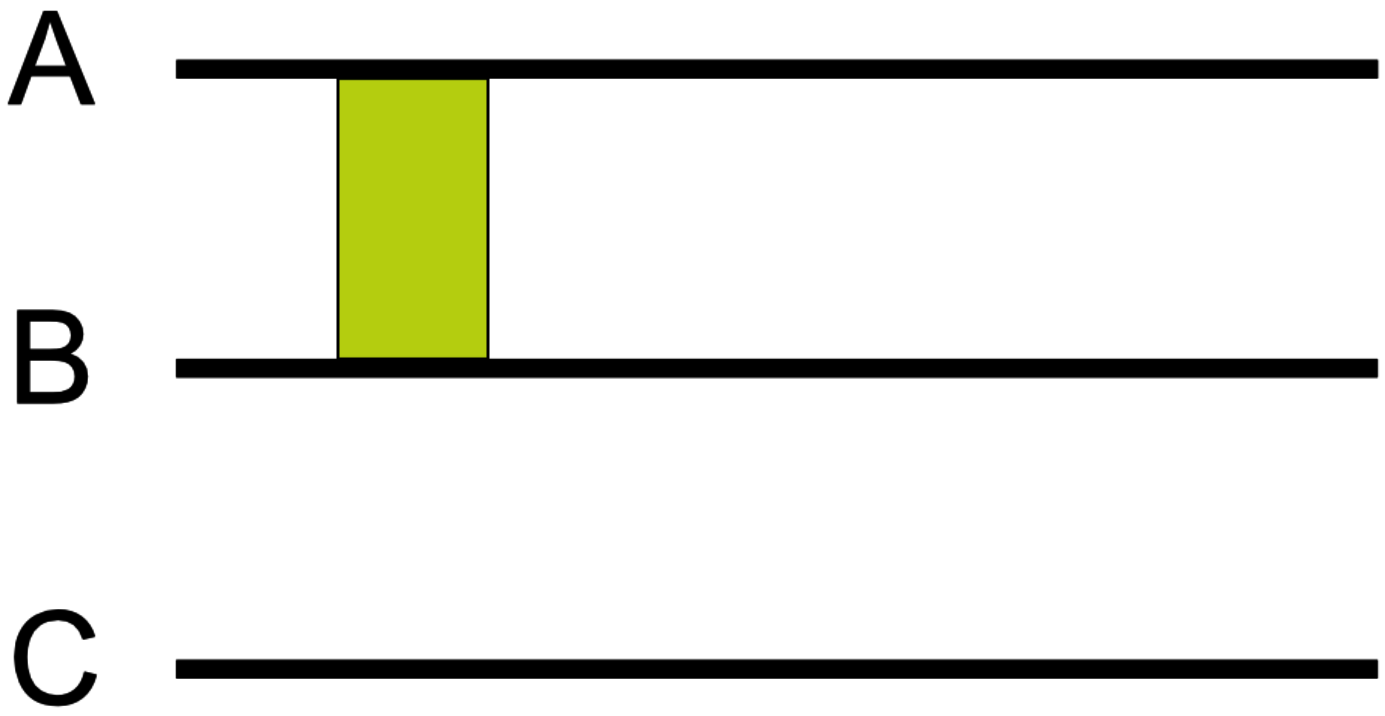
ScaDS DRESDEN LEIPZIG

# LOCAL MULTIPLE ALIGNMENTS



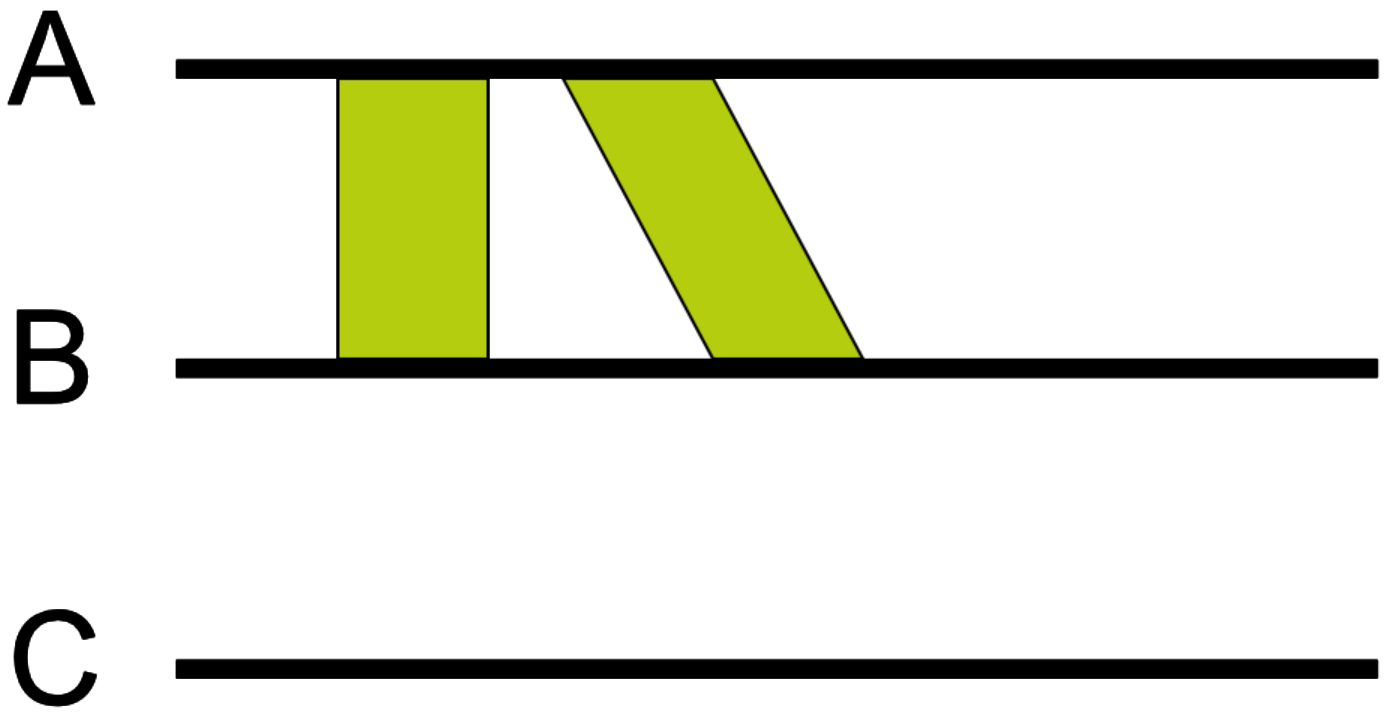
ScaDS DRESDEN LEIPZIG

# LOCAL MULTIPLE ALIGNMENTS



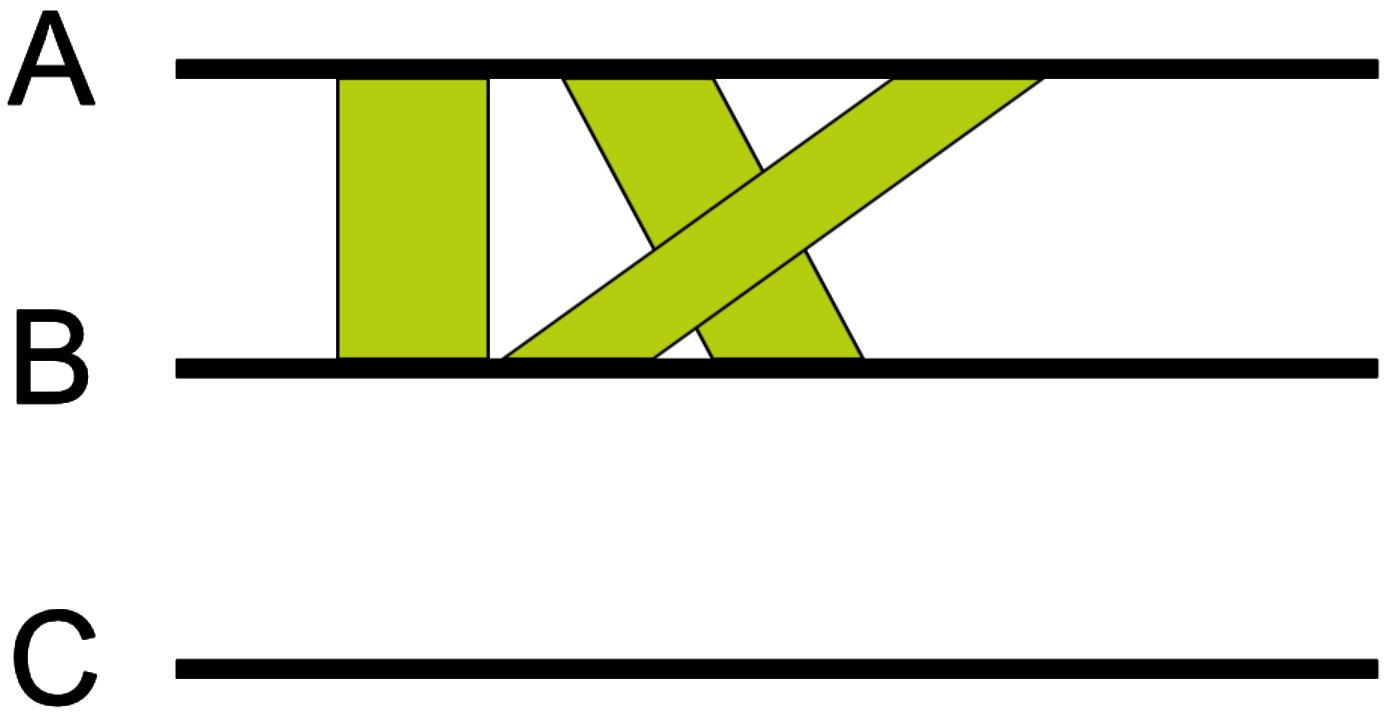
ScaDS DRESDEN LEIPZIG

# LOCAL MULTIPLE ALIGNMENTS



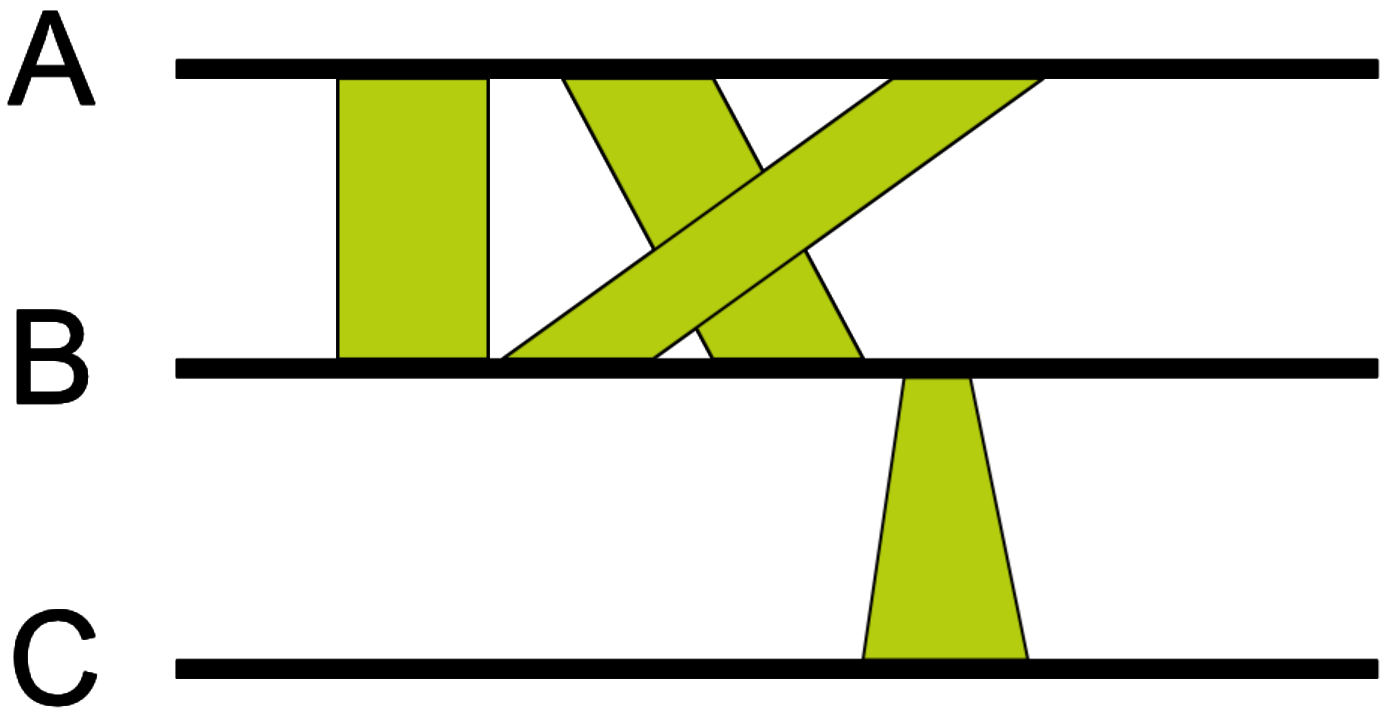
ScaDS DRESDEN LEIPZIG

# LOCAL MULTIPLE ALIGNMENTS



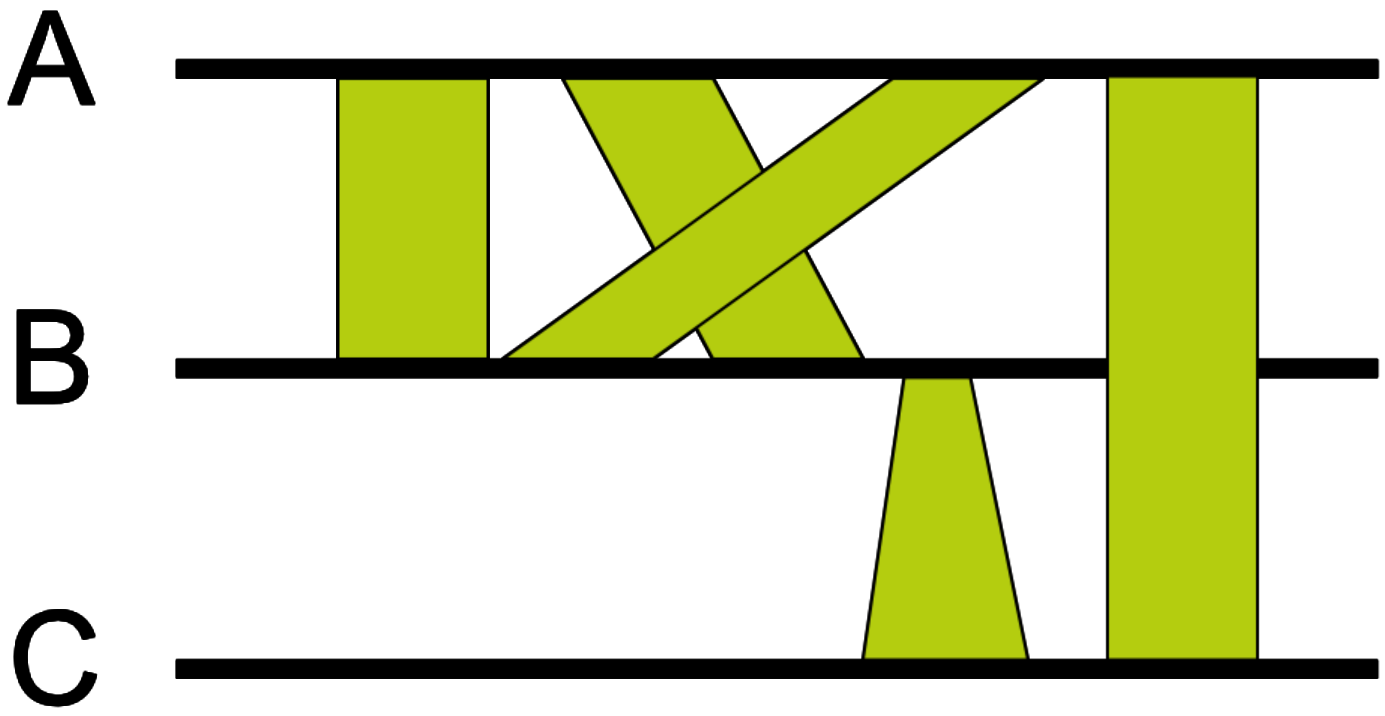
ScaDS DRESDEN LEIPZIG

# LOCAL MULTIPLE ALIGNMENTS



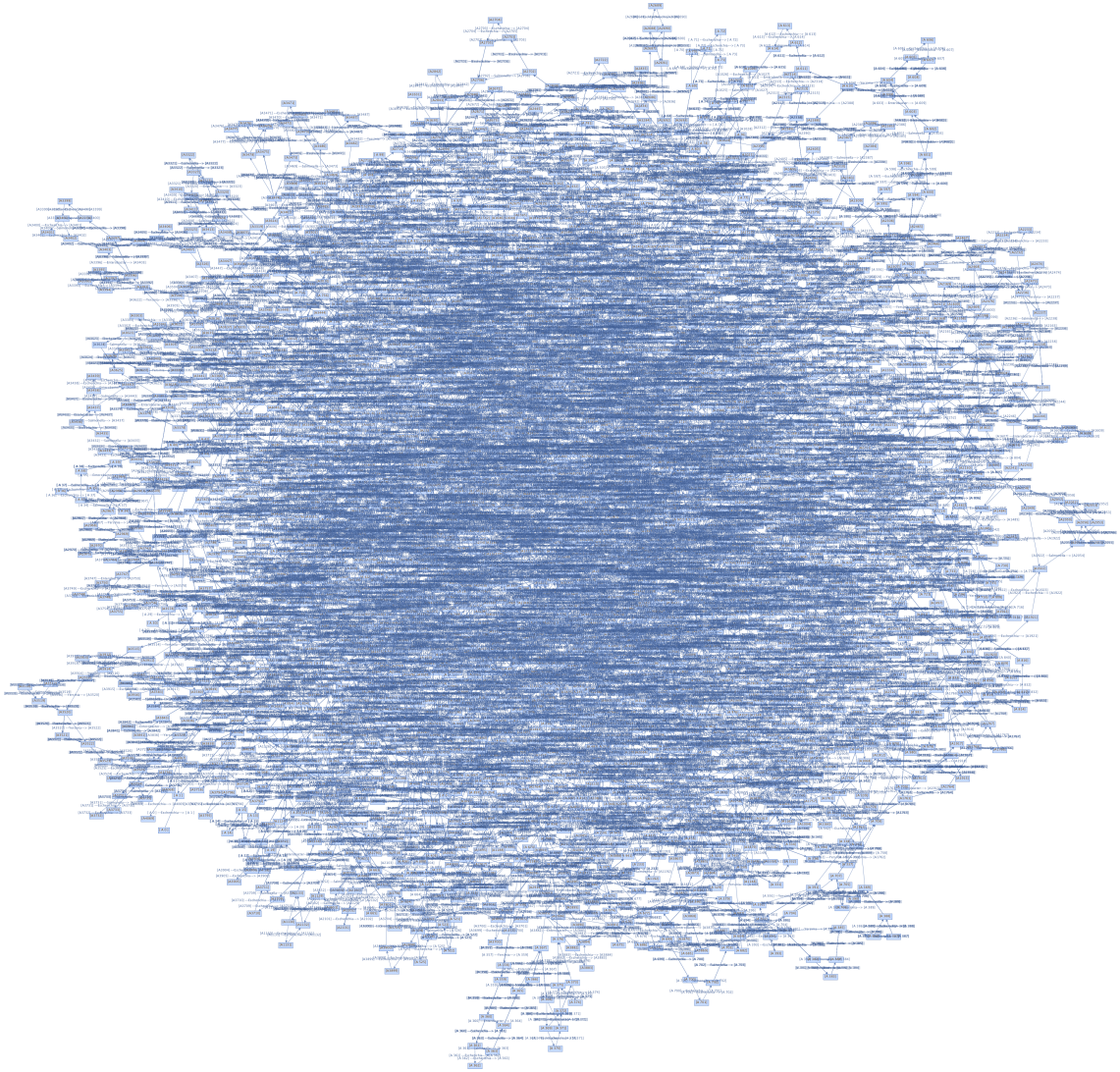
ScaDS DRESDEN LEIPZIG

# LOCAL MULTIPLE ALIGNMENTS



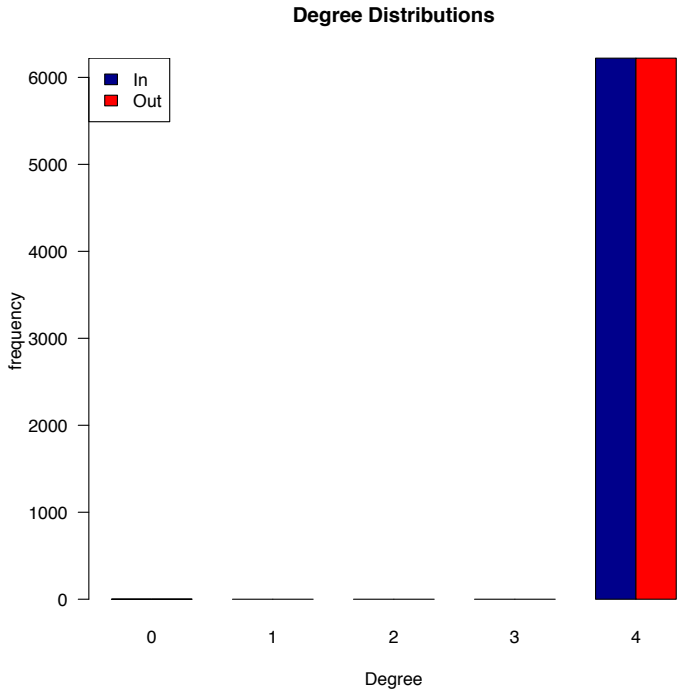
ScaDS DRESDEN LEIPZIG

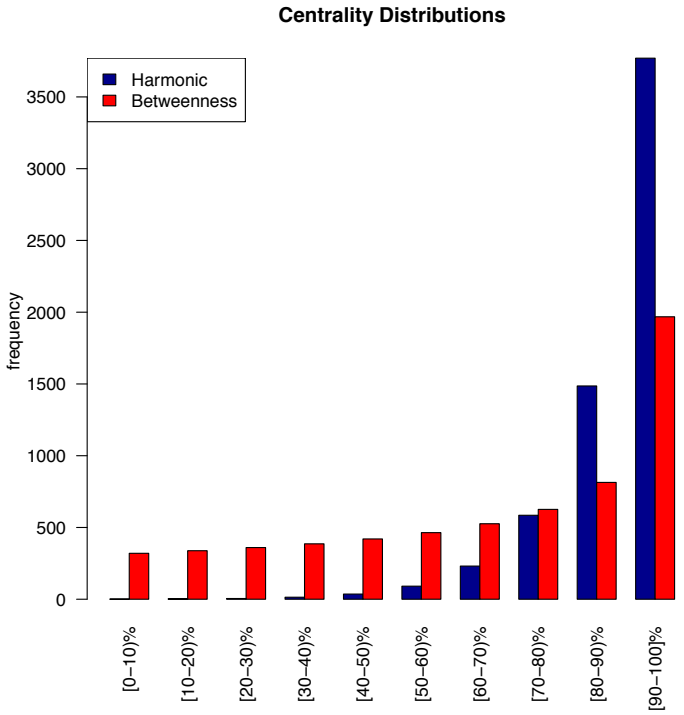
# 4 WAY BACTERIA



# PERFECT GRAPH

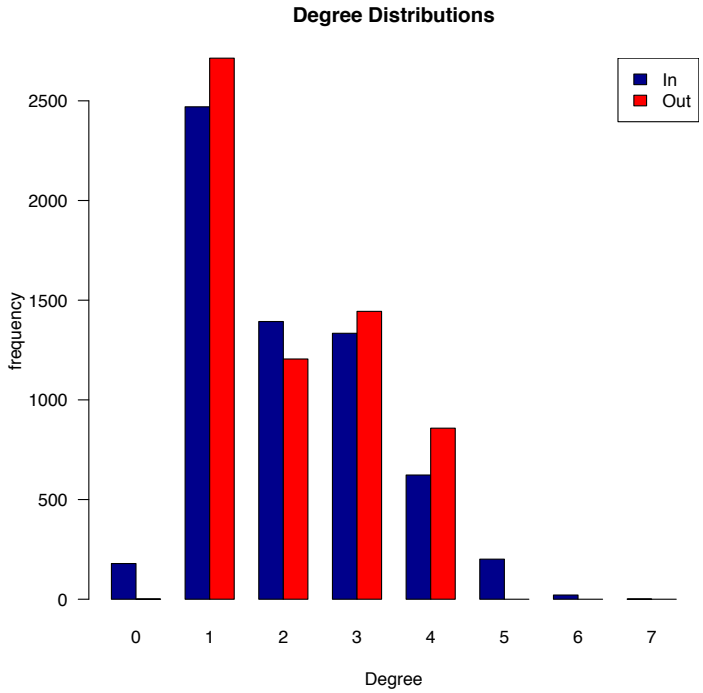
- 6,222 vertices
- 24,884 edges
- 1 connected component

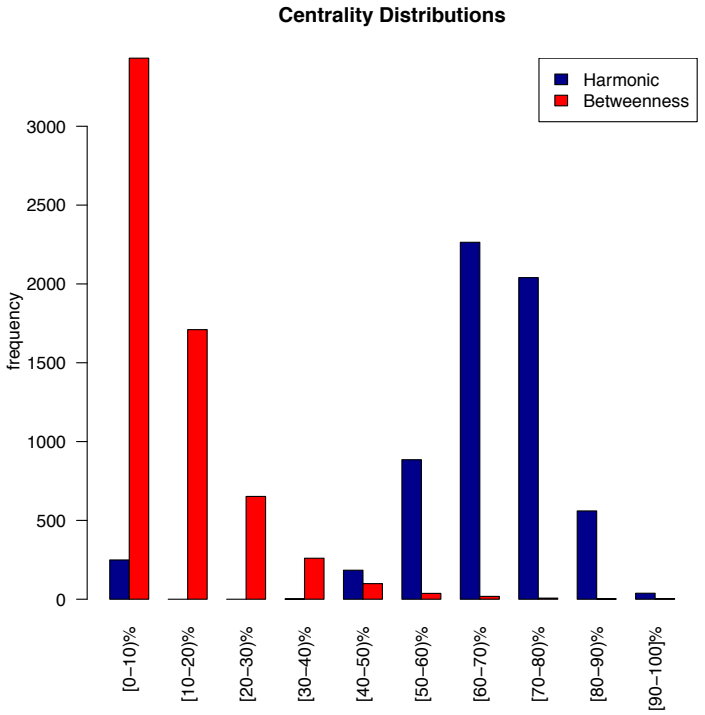


# 4 WAY BACTERIA

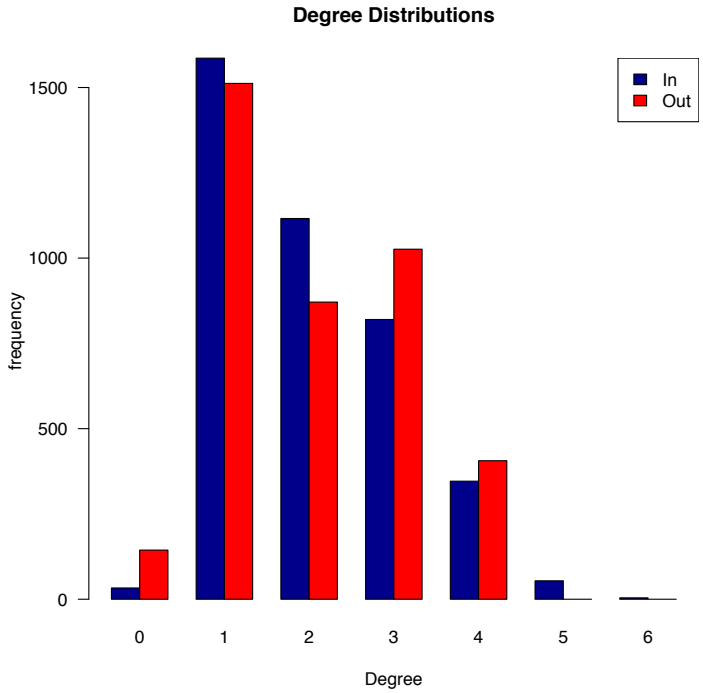
- 6,222 vertices
- 12,888 edges
- 1 connected component

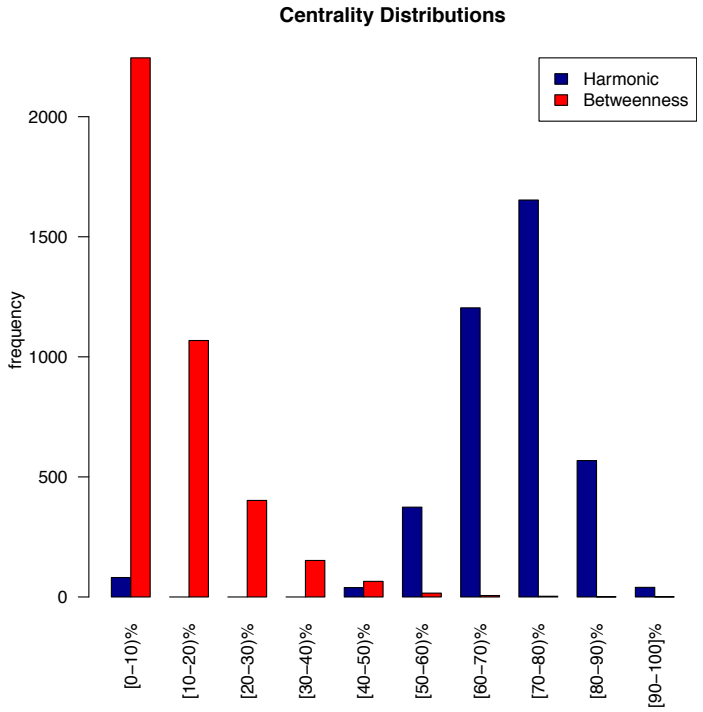


# 4 WAY BACTERIA REDUCED

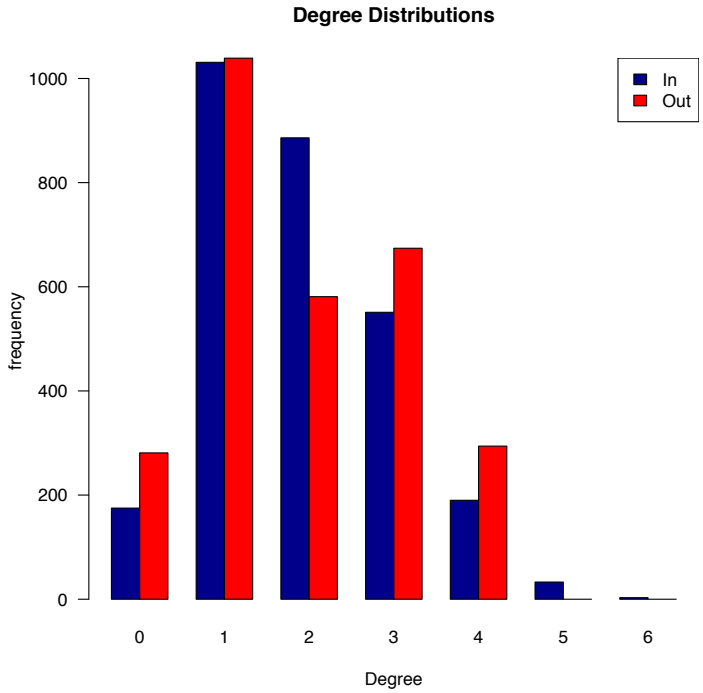
- 3,959 vertices
- 7,956 edges
- 1 connected component

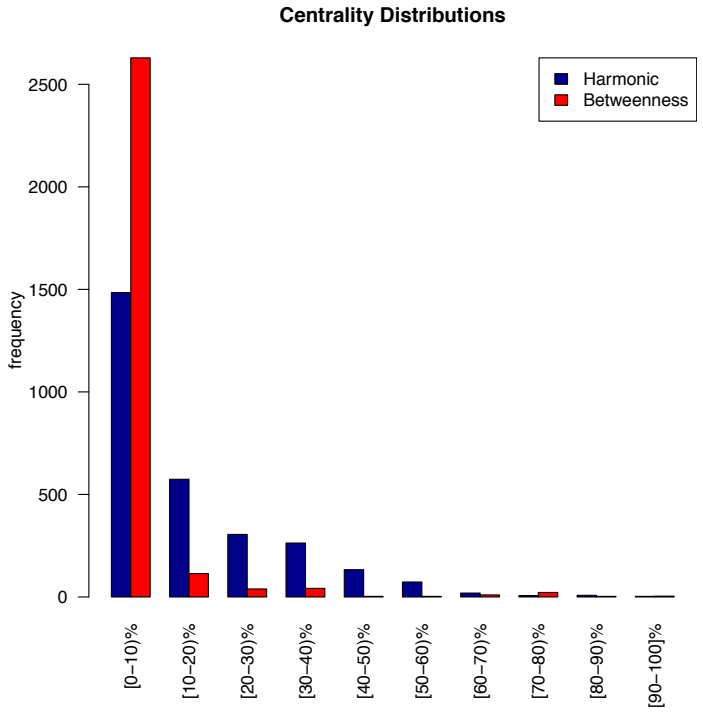


# 4 WAY BACTERIA NO CYCLE REDUCED

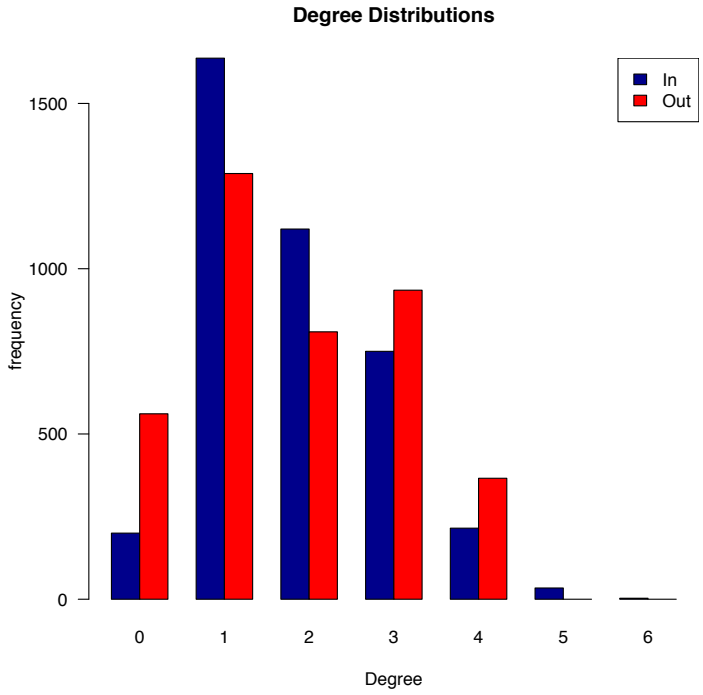
- 2,869 vertices
- 5,399 edges
- 1 connected component

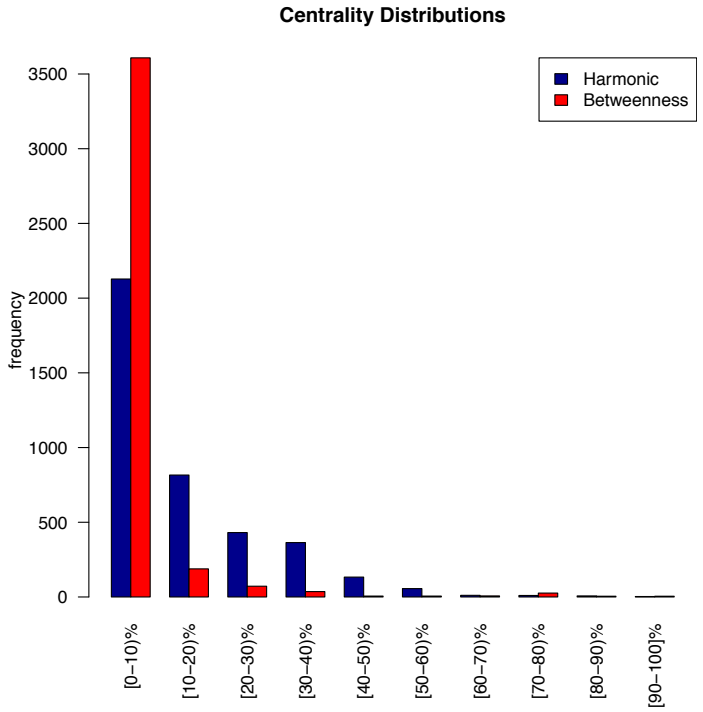


# 4 WAY BACTERIA NO CYCLE

- 3,959 vertices
- 7,175 edges
- 1 connected component



ScaDS DRESDEN LEIPZIG

# GRAPH GENERATION

- Graph Generation:
  1. Parsing
  2. Sorting
  3. Vertex insertion
  4. Edge insertion
     - Strand fixing
     - Insert edge depending on strand

ScaDS DRESDEN LEIPZIG

# LOCAL ALIGNMENTS

"In this letter we extend the above ideas to find a pair of segments, one from each of two long sequences, such that there is no other pair of segments with greater similarity (homology). "

*Smith and Waterman (1981)*