Universität Bielefeld

Technische Fakultät
Genominformatik

UFMS

Algorithms for Computing the Family-Free Genomic Similarity under DCJ

Jens Stoye

Technische Fakultät and CeBiTec, Universität Bielefeld, Germany

joint work with

Diego P. Rubert, Edna A. Hoshino, Marília D. V. Braga, Fábio V. Martinez

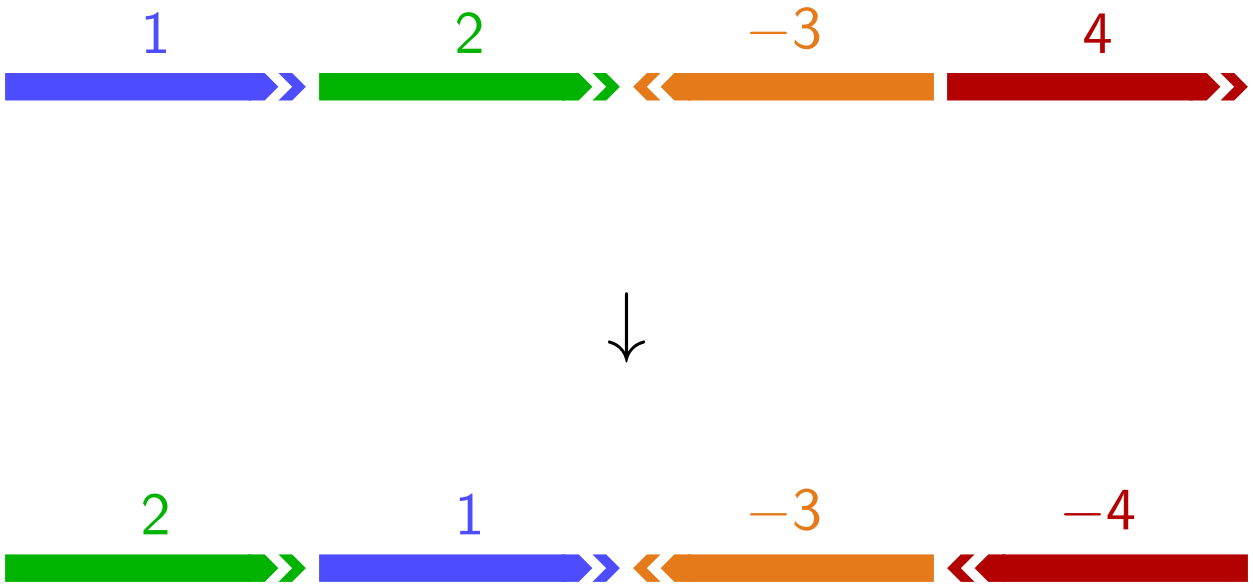
Genome rearrangements

- A central question in comparative genomics is the elucidation of similarities and differences between genomes
- Large-scale rearrangements change the number of chromosomes and/or the positions and orientations of genes (fusions, inversions, ...)
- Genomes are represented as sequences of oriented DNA fragments (genes)

- $A = (\circ\ 3\ -1\ 4\ 2\ -6\ 5\ \circ)$

Genome rearrangements

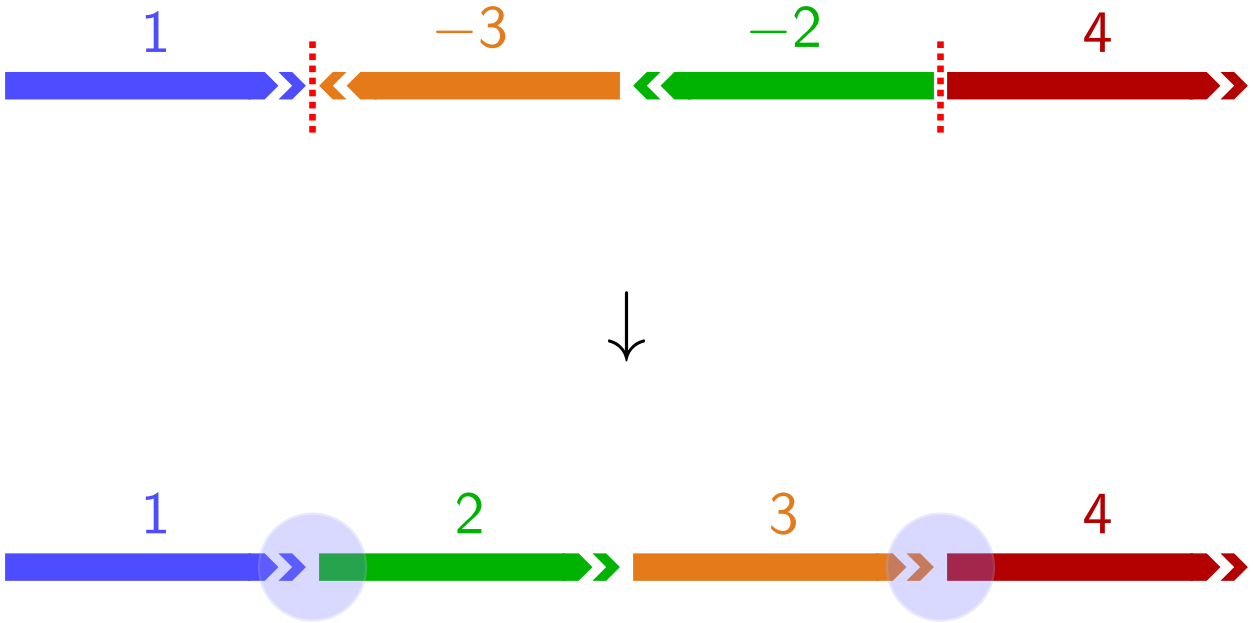
Classical problem: compute the rearrangement distance between two given genomes



The family-based setting

- Genes are grouped into *families*
- Without duplicate genes, several polynomial time algorithms are known to compute genomic distances and similarities
- With duplicate genes, problems become more intricate and many presented approaches are NP-hard

The double-cut-and-join (DCJ) operation



Family-based DCJ distance

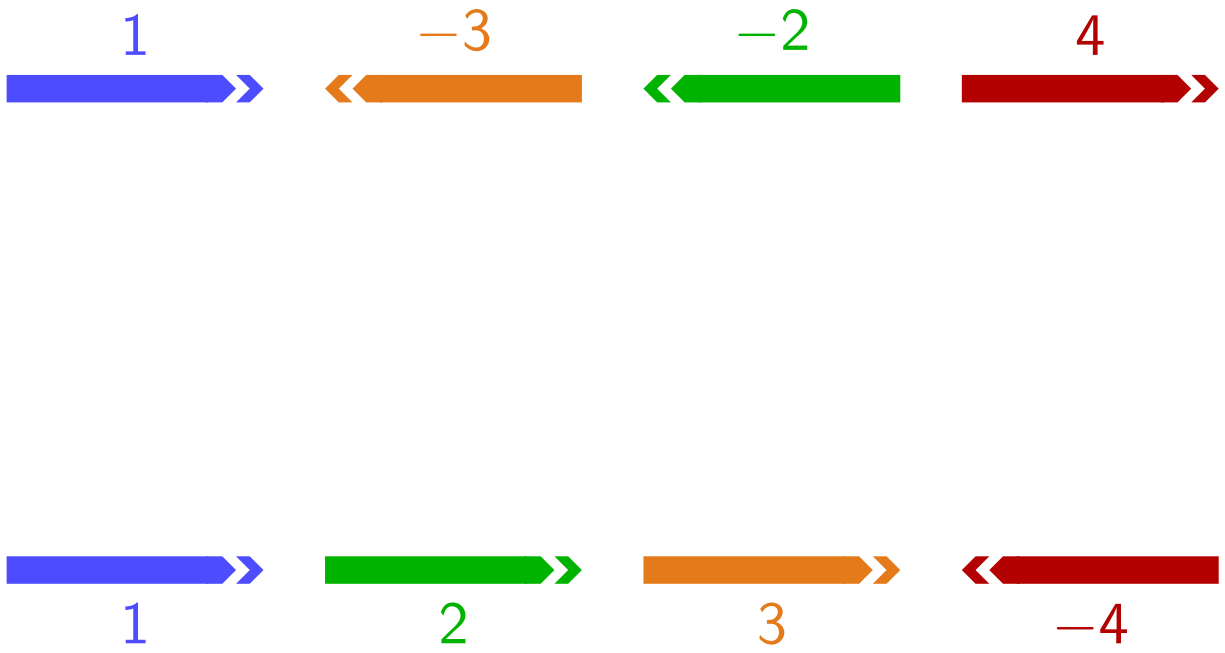
- Computing the family-based DCJ distance between two genomes A and B is easy.

Family-based DCJ distance

- Computing the family-based DCJ distance between two genomes A and B is easy.
- Adjacency Graph $AG(A, B)$:

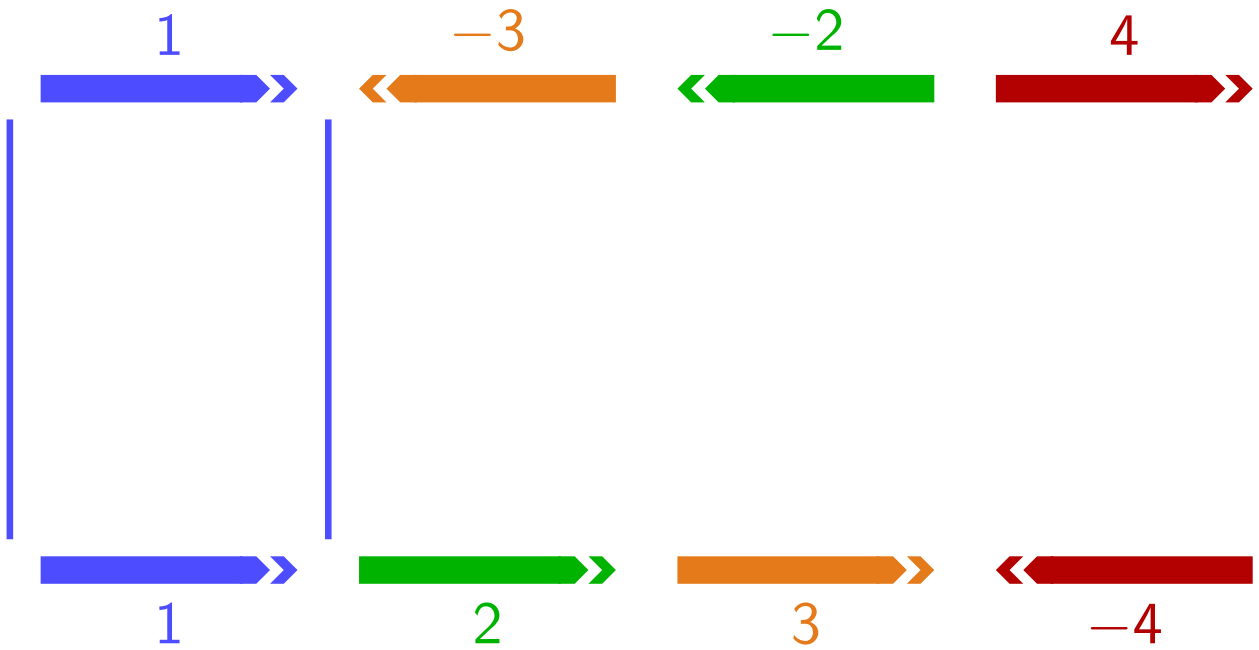
Family-based DCJ distance

- Computing the family-based DCJ distance between two genomes A and B is easy.
- Adjacency Graph $AG(A, B)$:



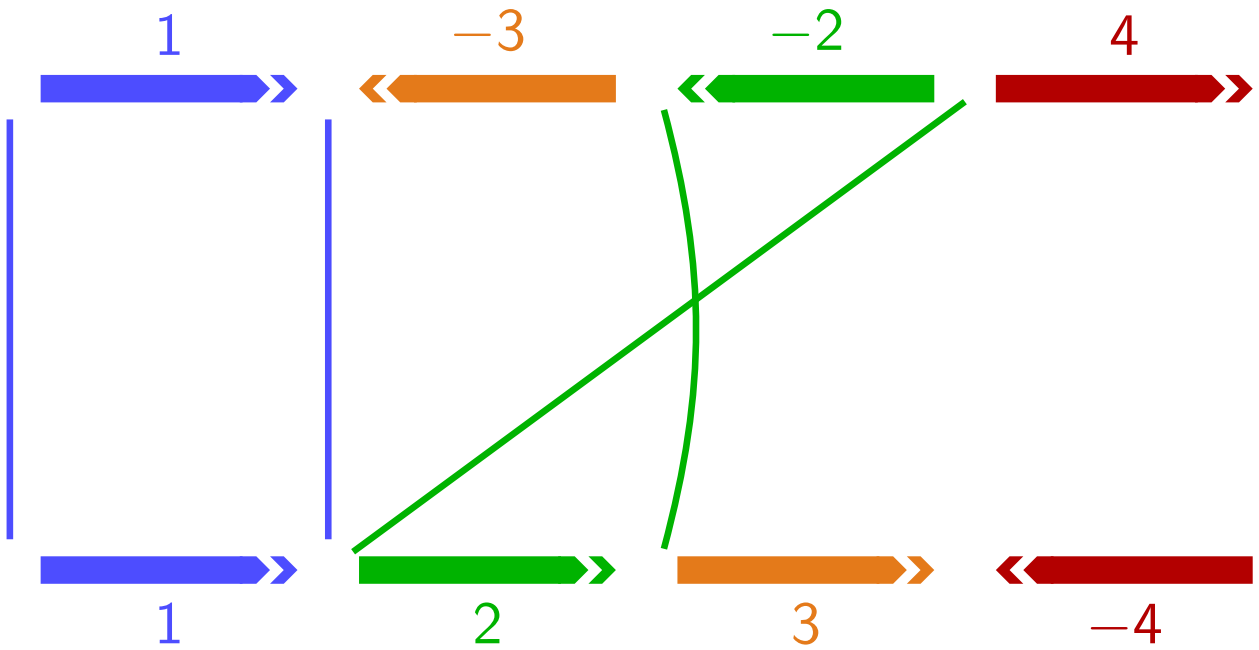
Family-based DCJ distance

- Computing the family-based DCJ distance between two genomes A and B is easy.
- Adjacency Graph $AG(A, B)$:



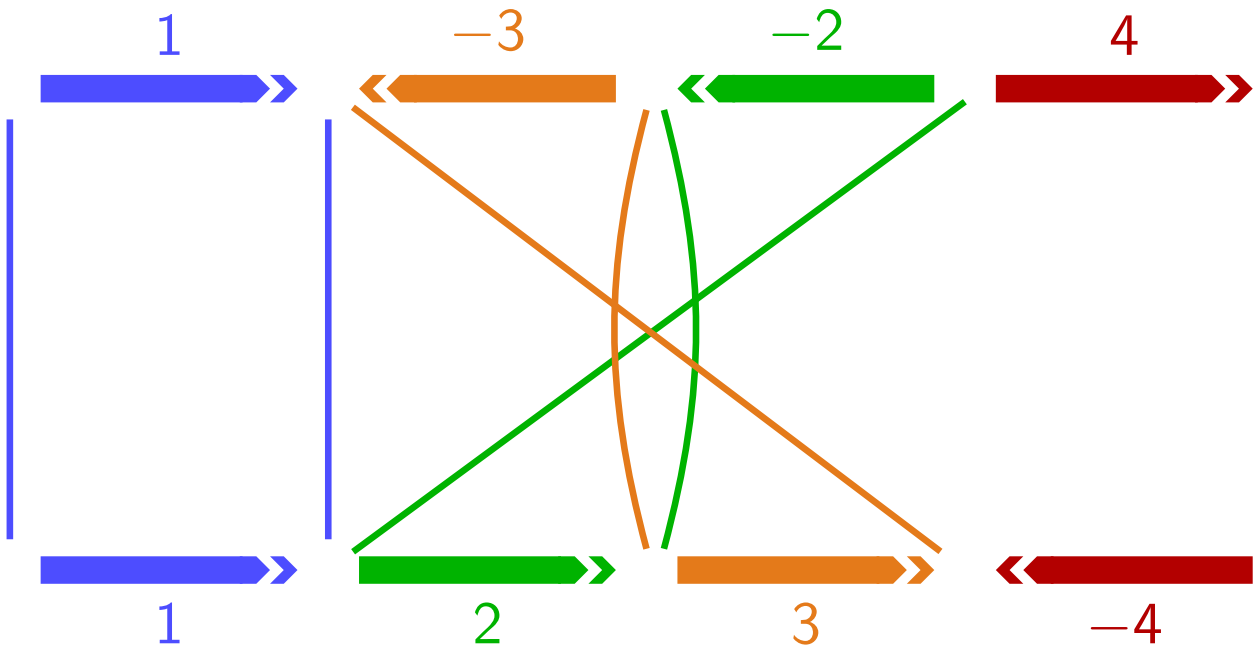
Family-based DCJ distance

- Computing the family-based DCJ distance between two genomes A and B is easy.
- Adjacency Graph $AG(A, B)$:



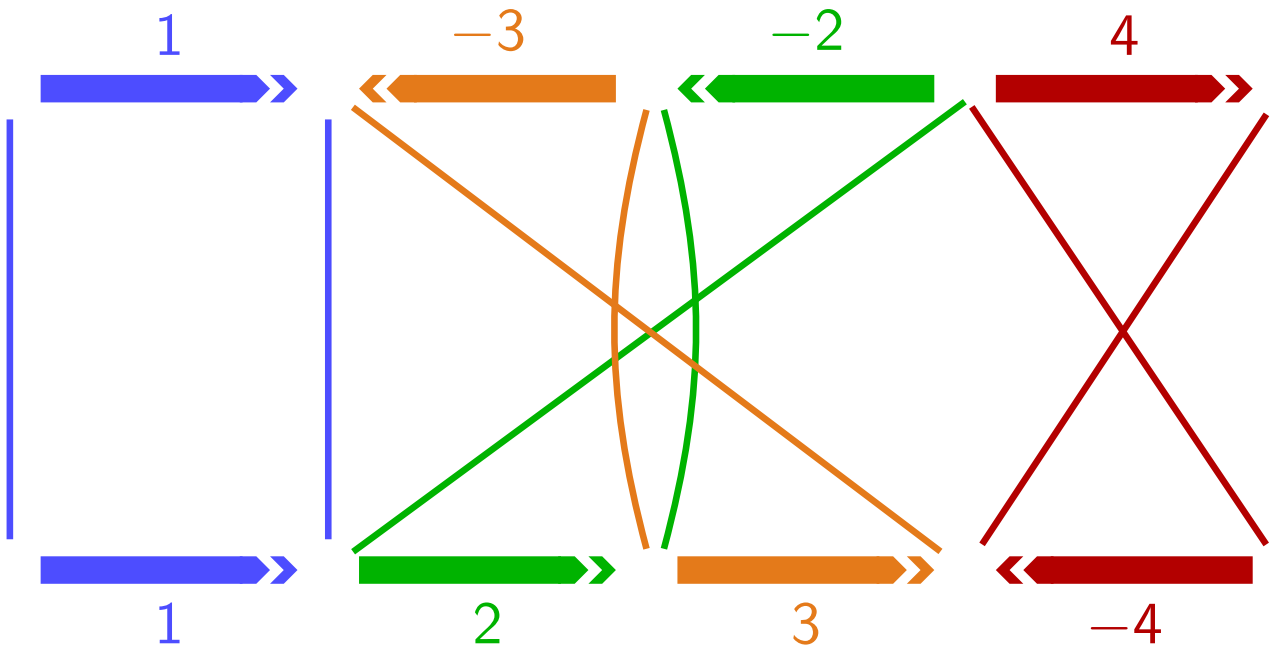
Family-based DCJ distance

- Computing the family-based DCJ distance between two genomes A and B is easy.
- Adjacency Graph $AG(A, B)$:



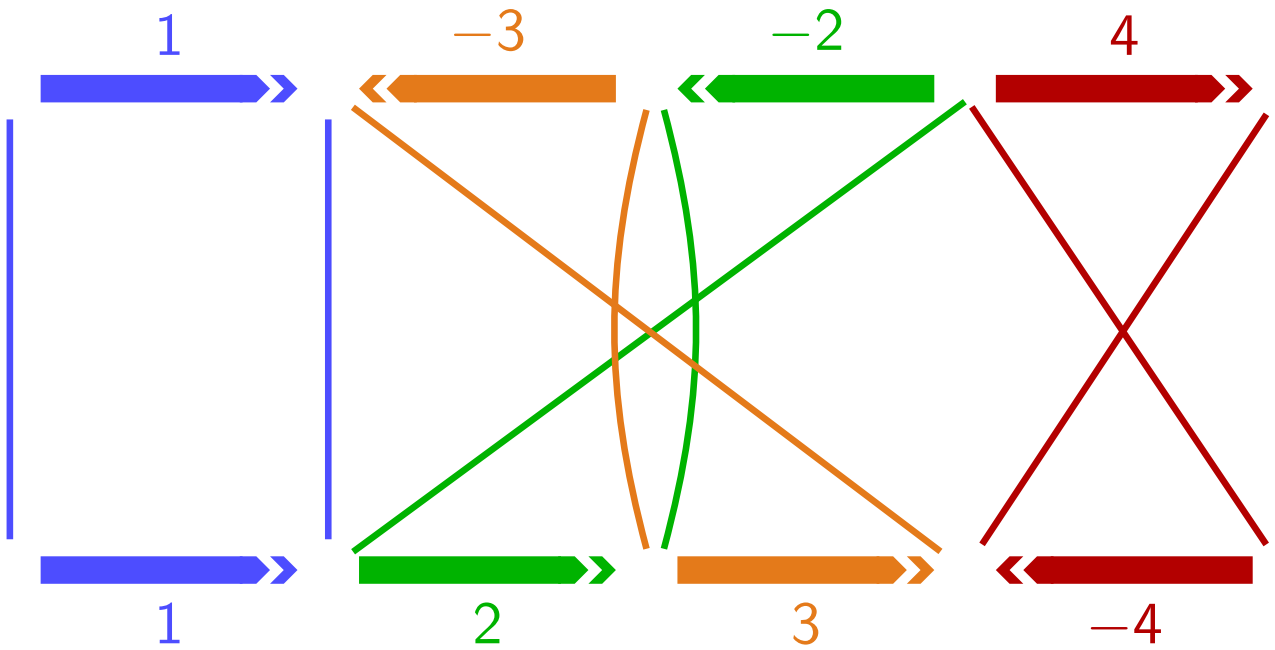
Family-based DCJ distance

- Computing the family-based DCJ distance between two genomes A and B is easy.
- Adjacency Graph $AG(A, B)$:



Family-based DCJ distance

- Computing the family-based DCJ distance between two genomes A and B is easy.
- Adjacency Graph $AG(A, B)$:



- Family-based DCJ distance: $\mathrm{d}_{\mathrm{DCJ}}(A, B) = n - c - \frac{i}{2}$

Family-based similarity

- In some contexts, similarity measures are more flexible
- Family-based DCJ similarity (Martinez *et al.*, AMB 2015):

$$\begin{aligned} s_{\mathrm{DCJ}}(A,B) &= \sum_{C\in\mathcal{P}}\left(\frac{|C|}{|C|+2}\right)+\sum_{C\in\mathcal{I}}\left(\frac{|C|}{|C|+1}\right)+\sum_{C\in\mathcal{C}}\left(\frac{|C|}{|C|}\right) \\ &= \sum_{C\in\mathcal{P}}\left(\frac{|C|}{|C|+2}\right)+\sum_{C\in\mathcal{I}}\left(\frac{|C|}{|C|+1}\right)+c \end{aligned}$$

  where $\mathcal{P}$, $\mathcal{I}$, $\mathcal{C}$ are the even paths, odd paths, cycles in $AG(A,B)$

- Can be computed as efficiently as the DCJ distance

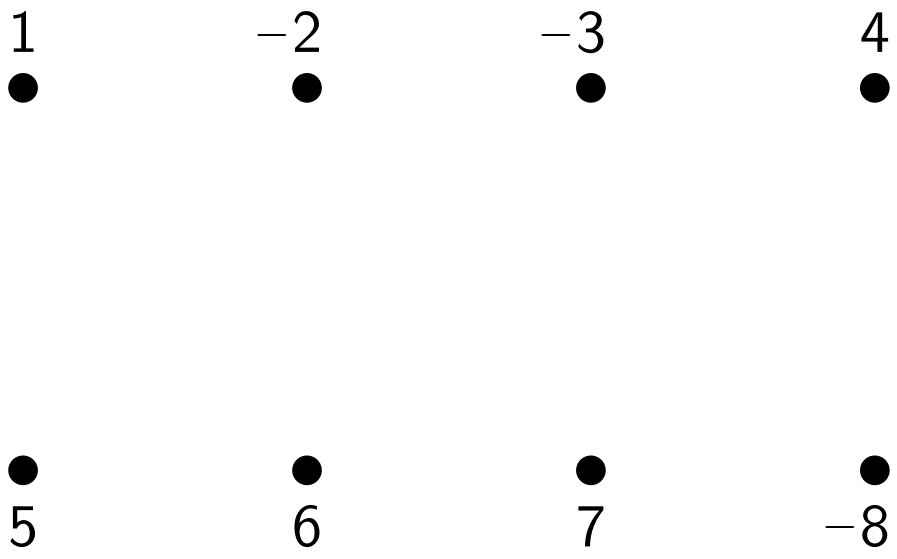
The family-based setting

However,

- Family assignments are most of the time made automatically
- Even in the absence of errors, there may be ambiguities

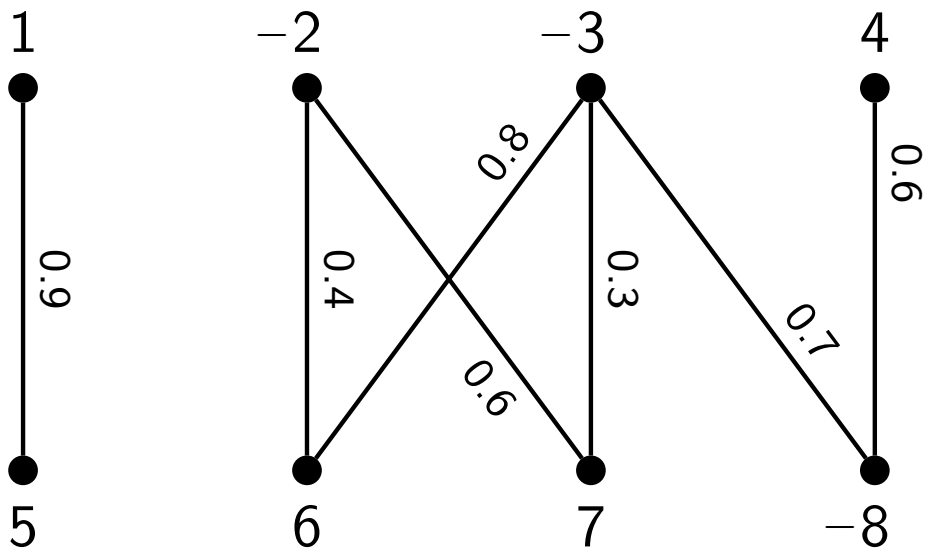
The family-free setting

- Each gene in each genome is represented by a unique (signed) symbol
- *Normalized gene similarities* with respect to some function σ are represented in the *gene similarity graph* $GS_\sigma(A, B)$



The family-free setting

- Each gene in each genome is represented by a unique (signed) symbol
- *Normalized gene similarities* with respect to some function σ are represented in the *gene similarity graph* $GS_\sigma(A, B)$



The family-free similarity

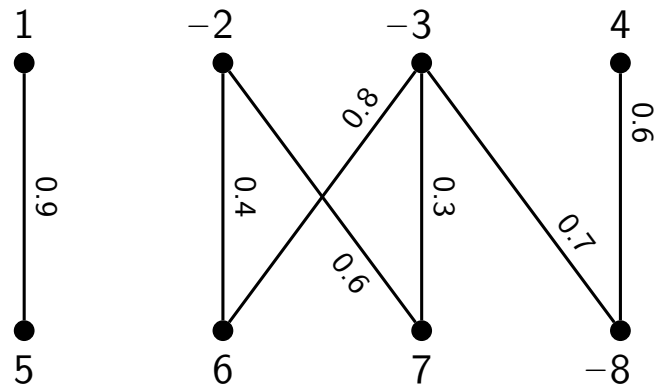
- Given a maximal matching M of the genes in A and the genes in B, inducing *reduced genomes* A^M and B^M, the family-free DCJ similarity is defined by:

$$s_\sigma(A^M, B^M) = \sum_{C \in \mathcal{P}} \left(\frac{w(C)}{|C| + 2} \right) + \sum_{C \in \mathcal{I}} \left(\frac{w(C)}{|C| + 1} \right) + \sum_{C \in \mathcal{C}} \left(\frac{w(C)}{|C|} \right)$$

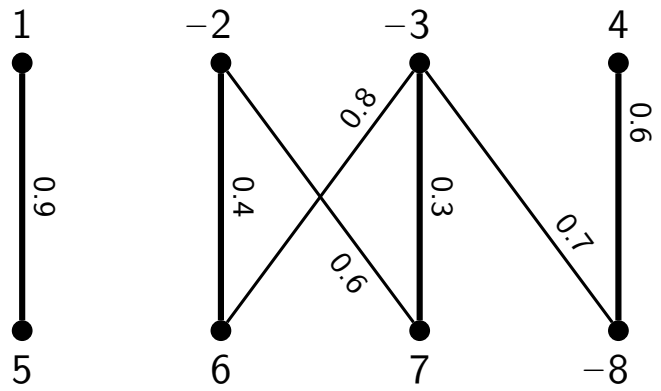
  where $\mathcal{P}$, $\mathcal{I}$, $\mathcal{C}$ are the even paths, odd paths, cycles in $AG(A^M, B^M)$

- The *family-free DCJ similarity* is the highest score $s_\sigma(A^M, B^M)$ possible for any maximal matching M in $GS_\sigma(A, B)$.

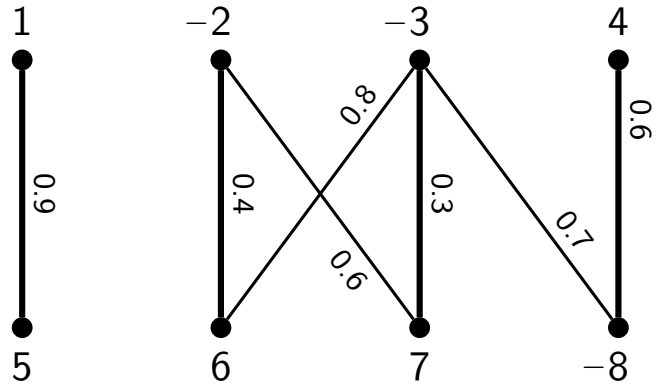
The family-free similarity

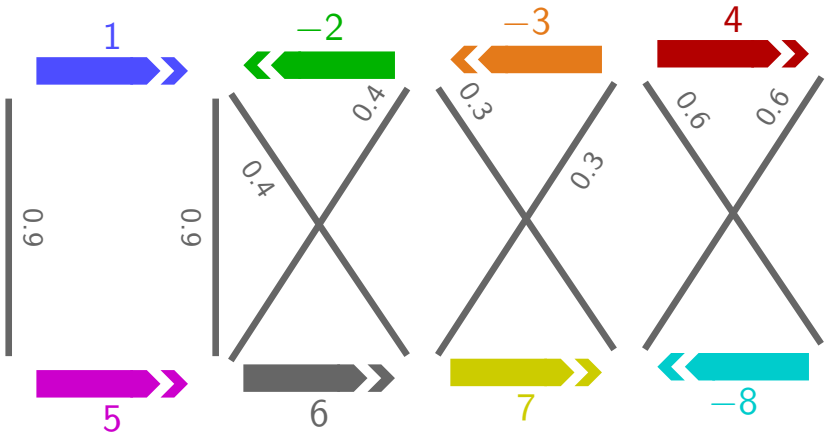


The family-free similarity



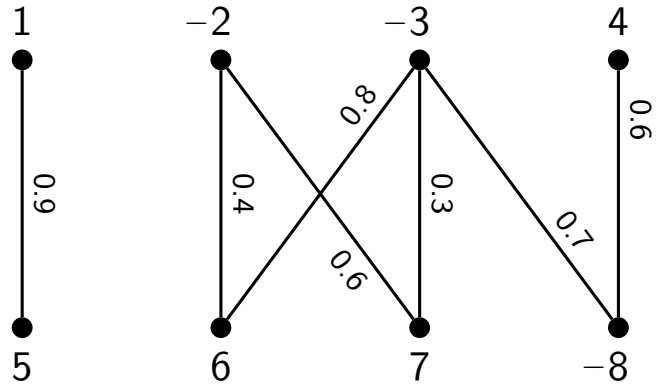
The family-free similarity

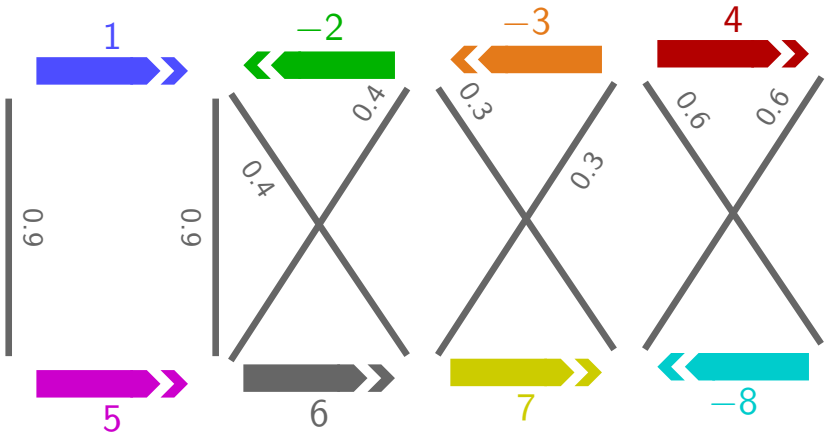




Score: $\frac{0.9}{1+1} + \frac{3.5}{7+1} = 0.8875$

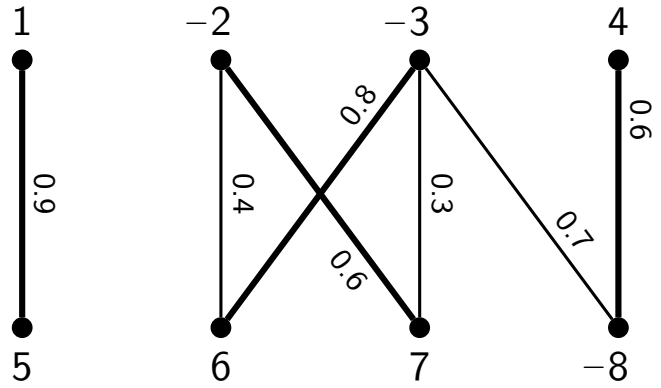
The family-free similarity

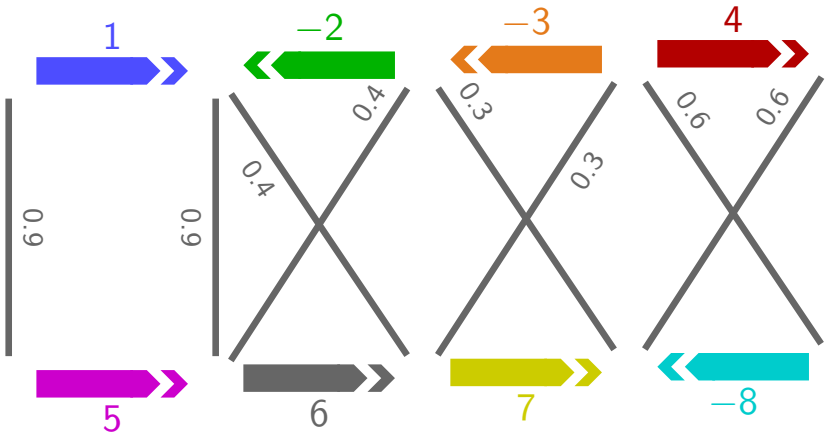




Score: $\frac{0.9}{1+1} + \frac{3.5}{7+1} = 0.8875$

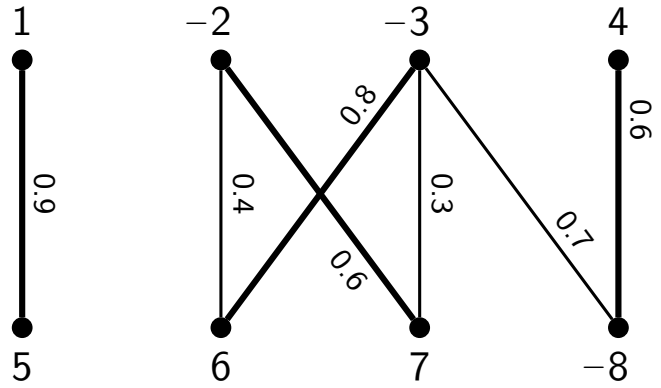
The family-free similarity

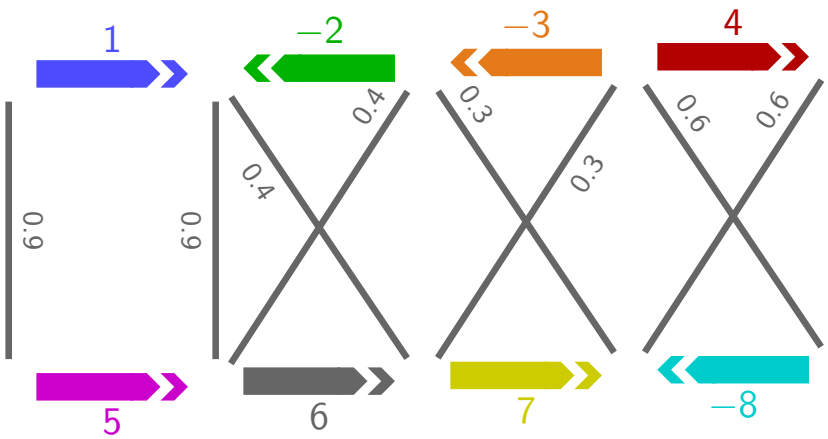




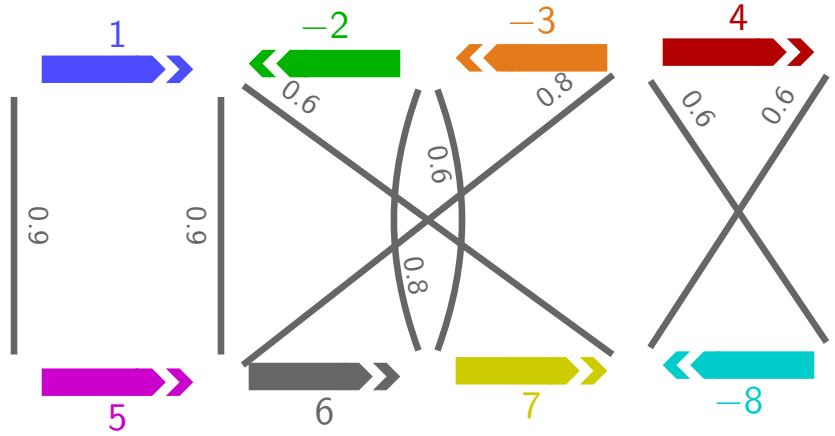
Score: $\frac{0.9}{1+1} + \frac{3.5}{7+1} = 0.8875$

The family-free similarity





Score: $\frac{0.9}{1+1} + \frac{3.5}{7+1} = 0.8875$



Score: $\frac{0.9}{1+1} + \frac{3.5}{5+1} + \frac{1.4}{2} = 1.733$

Inapproximability of FFDCJ-SIMILARITY

- Previously shown: FFDCJ-SIMILARITY is NP-complete (Martinez *et al.*, AMB 2015)

Theorem

FFDCJ-SIMILARITY *is APX-hard and cannot be approximated with approximation ratio better than* $22/21 = 1.0476\ldots$, *unless* $P = NP$.

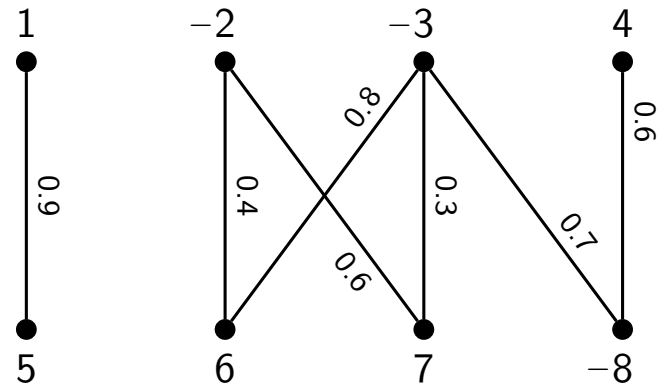
- Reduction from MAX-2SAT3 and MAX-2SAT.

Exact algorithm

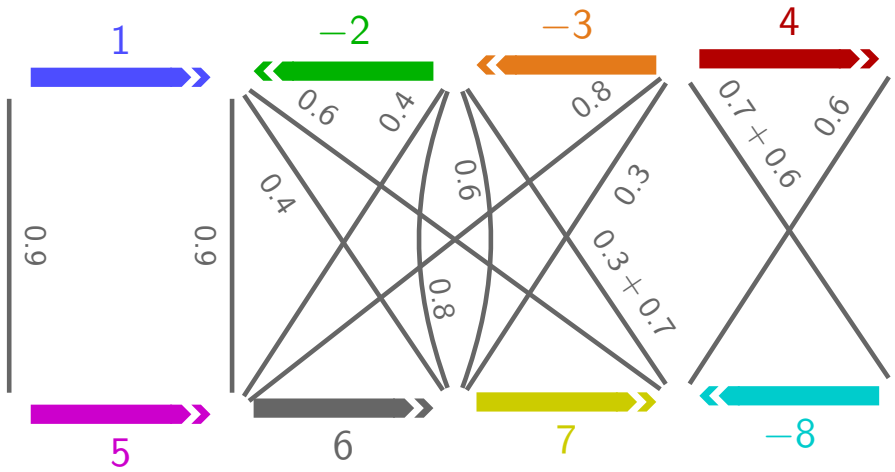
- We propose an integer linear program (ILP) formulation
- Similar to the one for the family-free DCJ distance (Martinez *et al.*, AMB 2015), based on an approach by Shao *et al.* (JCB 2015) to compute the family-based DCJ distance with gene duplications
- Has $O(N^4)$ variables and $O(N^3)$ constraints, where $N = |A| + |B|$

Heuristics

- Gene similarity graph $GS_\sigma(A, B)$:

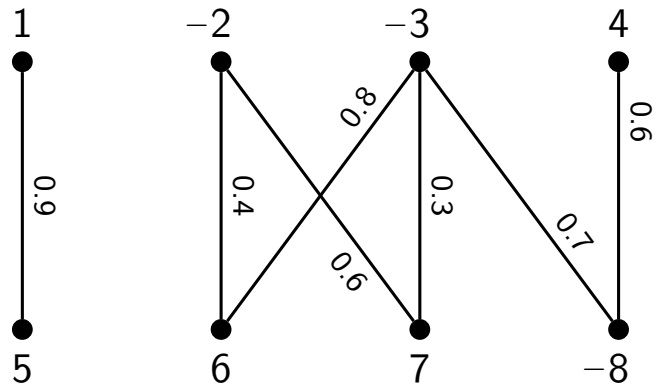


- Weighted adjacency graph $AG_\sigma(A, B)$:

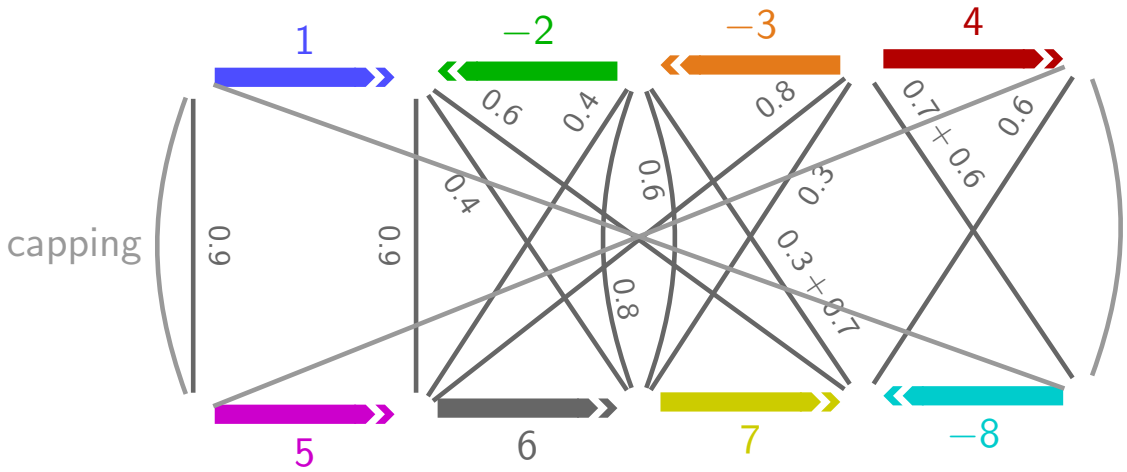


Heuristics

- Gene similarity graph $GS_\sigma(A, B)$:



- Weighted adjacency graph $AG_\sigma(A, B)$:



Heuristics

Goal: Pick set of consistent cycles maximizing the DCJ similarity score.

We have three heuristics:

GREEDY-DENSITY: prioritizes cycles in $AG_\sigma(A, B)$ with higher densities, where the *density* of some cycle C is given by $w(C)/|C|^2$

Heuristics

Goal: Pick set of consistent cycles maximizing the DCJ similarity score.

We have three heuristics:

GREEDY-DENSITY: prioritizes cycles in $AG_\sigma(A, B)$ with higher densities, where the *density* of some cycle C is given by $w(C)/|C|^2$

GREEDY-LENGTH: prioritizes 2-cycles in $AG_\sigma(A, B)$ with higher weights, then 4-cycles with higher weights, then 6-cycles...

Heuristics

Goal: Pick set of consistent cycles maximizing the DCJ similarity score.

We have three heuristics:

GREEDY-DENSITY: prioritizes cycles in $AG_\sigma(A, B)$ with higher densities, where the *density* of some cycle C is given by $w(C)/|C|^2$

GREEDY-LENGTH: prioritizes 2-cycles in $AG_\sigma(A, B)$ with higher weights, then 4-cycles with higher weights, then 6-cycles...

GREEDY-WMIS: tries to select a set of 2-cycles in $AG_\sigma(A, B)$ with the highest sum of weights by an WMIS algorithm, then a set of 4-cycles...

Experimental results

- Simulated data generated by the *Artificial Life Simulator* (ALF)
- *Gurobi* solver for the ILPs, 4 threads, time limit of 1800 s
- Heuristics implemented in C++
- Genomes of sizes around 25, 50, and 1000 (heuristics only)
- 1-, 2-, and 5-fold increase in rearrangement rates (r)

Experimental results

	ILP			Greedy-Density	Greedy-Length	Greedy-wmis
	Time (s)	Not finished	Gap (%)	Δ (%)	Δ (%)	Δ (%)
25 genes, $r = 1$	19.50	0	–	5.03	5.84	5.97
25 genes, $r = 2$	84.60	2	69.21	30.77	43.57	43.00
25 genes, $r = 5$	49.72	0	–	43.83	55.38	55.38
50 genes, $r = 1$	445.91	7	19.56	18.74	19.36	18.90
50 genes, $r = 2$	463.50	29	38.12	65.41	66.52	64.78
50 genes, $r = 5$	330.88	29	259.72	177.58	206.60	206.31

- Running time for heuristics was negligible
- Average relative delta of heuristics increases proportionally to the rate of reversals and translocations $\Rightarrow$ higher normalized weights on longer cycles

Experimental results

Observations:

- For some larger instances the relative delta for heuristics is very close to the values obtained by the ILP solver, suggesting the use of heuristics:
  - may be a good alternative for some classes of instances
  - could help the solver finding lower bounds quickly.
- Greedy-Density found solutions with delta $< 1\%$ for 38% of the instances with 25 genes
- For heuristics and genomes of size 1000, avg. running time is 9 s for $r = 1$ and 68 s $r = 5$, Greedy-Density was the best most of times

How can our heuristics be fast?

- The total number of cycles may be exponential

How can our heuristics be fast?

- The total number of cycles may be exponential
- But when finding longer cycles, the heuristics do not try to find cycles composed by adjacencies in $AG_\sigma(A, B)$ already covered by shorter cycles chosen previously

How can our heuristics be fast?

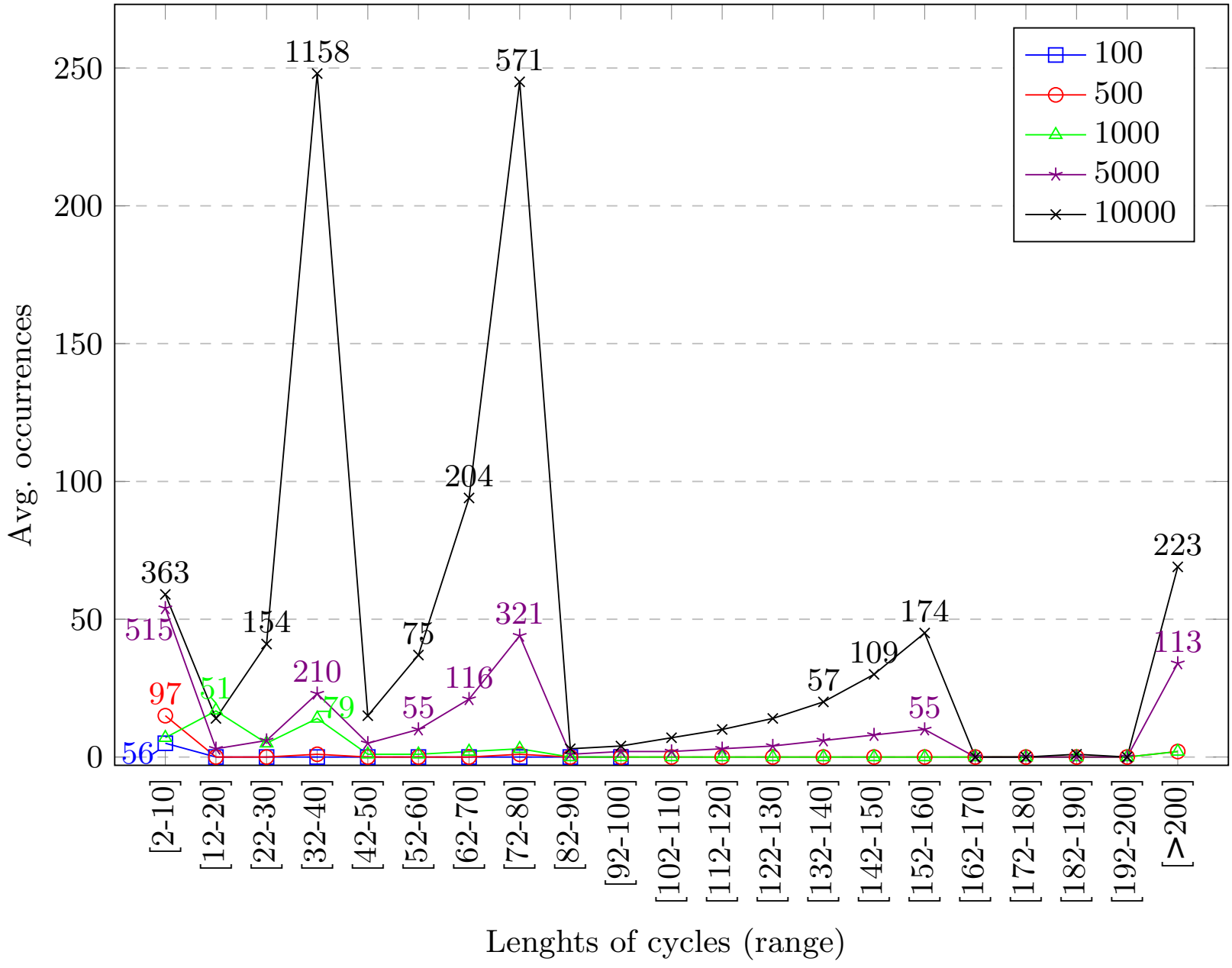
- The total number of cycles may be exponential
- But when finding longer cycles, the heuristics do not try to find cycles composed by adjacencies in $AG_\sigma(A, B)$ already covered by shorter cycles chosen previously
- To better understand how cycles scale, we generated 5-fold instances with 100, 500, 1000, 5000, and 10000 genes, running the Greedy-Density, avg. running time was 0.008 s, 0.667 s, 1.98 s, 508 s and 2896 s

How can our heuristics be fast?

- The total number of cycles may be exponential
- But when finding longer cycles, the heuristics do not try to find cycles composed by adjacencies in $AG_\sigma(A, B)$ already covered by shorter cycles chosen previously
- To better understand how cycles scale, we generated 5-fold instances with 100, 500, 1000, 5000, and 10000 genes, running the GREEDY-DENSITY, avg. running time was 0.008 s, 0.667 s, 1.98 s, 508 s and 2896 s
- Results (next figure) show that most of the cycles found are of short lengths compared to the genome sizes

How can our heuristics be fast?

- The total number of cycles may be exponential
- But when finding longer cycles, the heuristics do not try to find cycles composed by adjacencies in $AG_\sigma(A, B)$ already covered by shorter cycles chosen previously
- To better understand how cycles scale, we generated 5-fold instances with 100, 500, 1000, 5000, and 10000 genes, running the GREEDY-DENSITY, avg. running time was 0.008 s, 0.667 s, 1.98 s, 508 s and 2896 s
- Results (next figure) show that most of the cycles found are of short lengths compared to the genome sizes
- Even the maximum number of longer cycles found for any instance is reasonably small



(on top of some marks is shown the maximum number of cycles)

Thank you!