Algorithms for computing the Family-Free Genomic Similarity under DCJ

Università di Pisa

Course: *Bioinformatics*

Prof: *Nadia Pisanti*

Student: *Antonio Sisbarra*

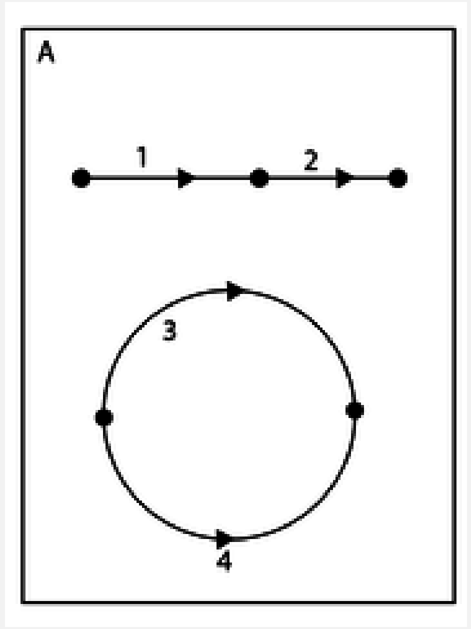
Outline

- Introduction
- ILP formulation
- Faster heuristics
- Experimental results
- Conclusions

Introduction

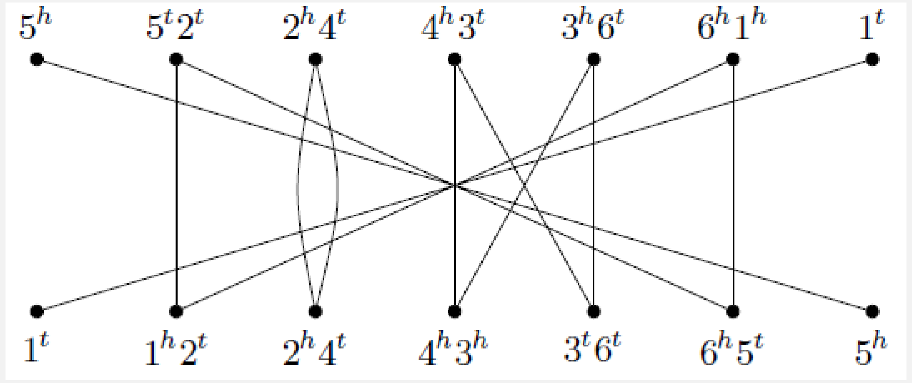
Introduction: DCJ model

- **Genes** represented by a **number** and an **orientation**
- **Global measure** for distance: number of genome rearrangements
- **Double cut and join** op.: *cut* a genome in two distinct positions and joining the four resultant open ends
- **Similarity**: more rearrangements needed to get equal genoms, less similarity



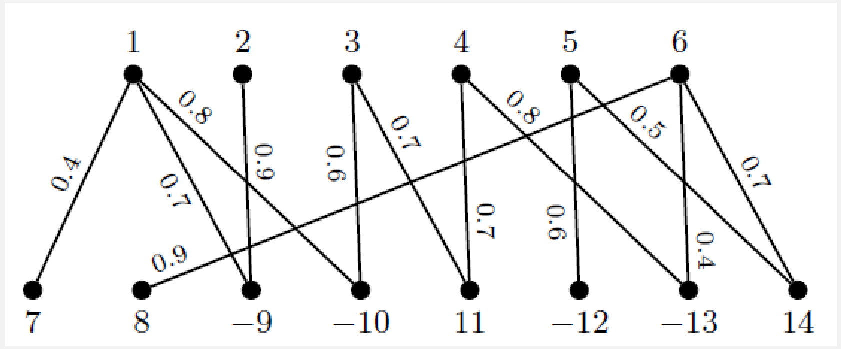
Introduction: Genes and Adjacency Graph

- Each **gene** *g* of a genome has two extremities *(tail* and *head)*: g^t and g^h
- **Adjacency:** Pair of ***consecutive*** gene extremities or extremity of a gene adjacent to a telomere
- **Telomere:** extremity of a linear chromosome
- **Adjacency Graph:**

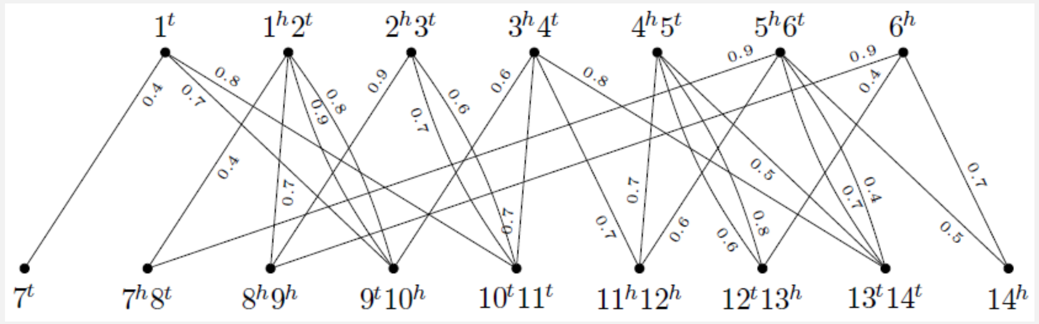


Introduction: Weighted Adjacency Graph

- **Family-Free** setting: genes represented by a unique (signed) symbol
- **No assumptions** on genes families
- **Distinct** genes: *normalized gene similarity [0,1]*
- With this setting vertices can have **degree greater than 2**



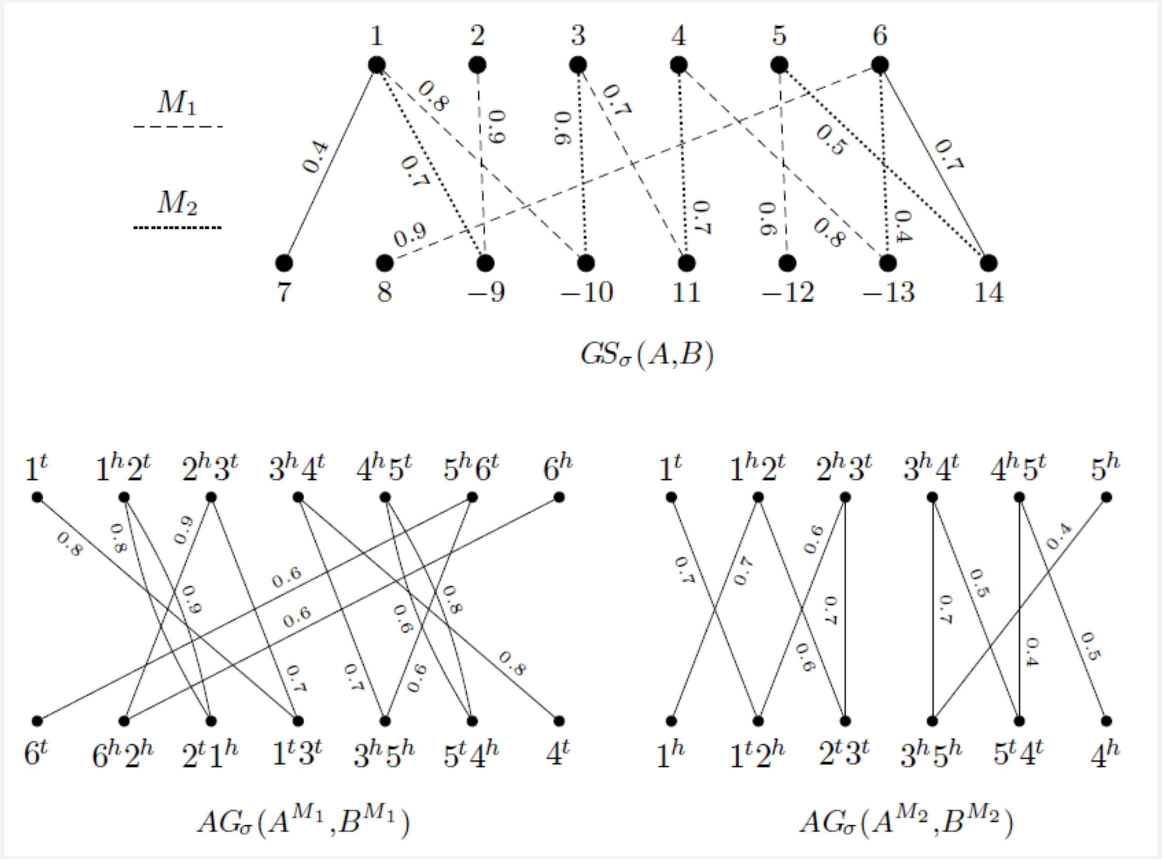
Gene similarity graph



Weighted adjacency graph

Introduction: Matchings

- **Matching:** set of edges without common vertices
- **Maximal matching(A^M, B^M):** if any edge not in M is added to M, it is no longer a matching
- **Decomposition** in components is simplified
- For each gene what is the **best connection** with the other genome?



M_1
M_2
1 2 3 4 5 6
7 8 −9 −10 11 −12 −13 14
$GS_\sigma(A,B)$
1^t 1^h2^t 2^h3^t 3^h4^t 4^h5^t 5^h6^t 6^h
6^t 6^h2^h 2^t1^h 1^t3^t 3^h5^h 5^t4^h 4^t
$AG_\sigma(A^{M_1},B^{M_1})$
1^t 1^h2^t 2^h3^t 3^h4^t 4^h5^t 5^h
1^h 1^t2^h 2^t3^t 3^h5^h 5^t4^t 4^h
$AG_\sigma(A^{M_2},B^{M_2})$

Introduction: FFDCJ Similarity

- **FFDCJ-SIM(A, B)**: given genomes *A* and *B* and their gene sim. σ, calculate their family-free DCJ similarity:

$$\boldsymbol{s_{FFDCJ}(A,B) = max_{M \in \mathcal{M}}\{s_\sigma(A^M, B^M)\}}$$

where *M* = *set of all maximal matchings in* $GS_{\boldsymbol{\sigma}}$***(A, B)***

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

  where P,I,C are *even paths, odd paths* and *cycles*
- **Cycles** are more important than **paths**

ILP formulation for an exact solution

ILP formulation: constraints

- **Integer Linear Programming (ILP):** mathematical optimization in which there are constraints and some or all variables are *integers*

- x_{ab} = 1 iff **edge** between *a* and *b* is included in the **solution**
- y_{ri} = 1 iff **vertex** v_r belongs to the cycle *i* in the **solution**
- E_G = set of edges in **Gene Adj. Graph**

- Core constraints (on binary variables):
  - **Valid edges:** $x_{a^h b^h} = x_{a^t b^t},\ \forall\, ab \in\ E_G$
  - **Maximal matching:** $x_{a^h a^t} + x_{b^h b^t}\ \leq 1,\ \ \forall\, ab \in\ E_G$ *(self-edge -> gene removed from solution)*
  - **Each vertex can only belong to one cycle:** $\sum_{i=1}^{r} y_{ri} \leq 1, \forall\, v_r \in V_H$

ILP formulation: objective function

- We have to:

$$\textbf{maximize} \sum_{i=1}^{k} \sum_{l \in L} \sum_{e \in E_m} \frac{w_e x_{ei}^{l}}{l}$$

- $\boldsymbol{E_m}$ = matching edges set
- $\boldsymbol{k}$ = number of vertices in Adjacency Graph
- $\boldsymbol{l} \in \boldsymbol{L}$ = every possible length for the cycles
- $\boldsymbol{x_{ei}^{l}}$ = 1 if edge $\boldsymbol{e}$ ∈ cycle $\boldsymbol{i}$ with length $\boldsymbol{l}$

ILP formulation: complexity

- We have to:

$$\textbf{maximize} \sum_{i=1}^{k} \sum_{l \in L} \sum_{e \in E_m} \frac{w_e x_{ei}^l}{l}$$

- $O(N^4)$ variables
- $N = |A| + |B|$
- Solving an ILP problem is **NP-hard**

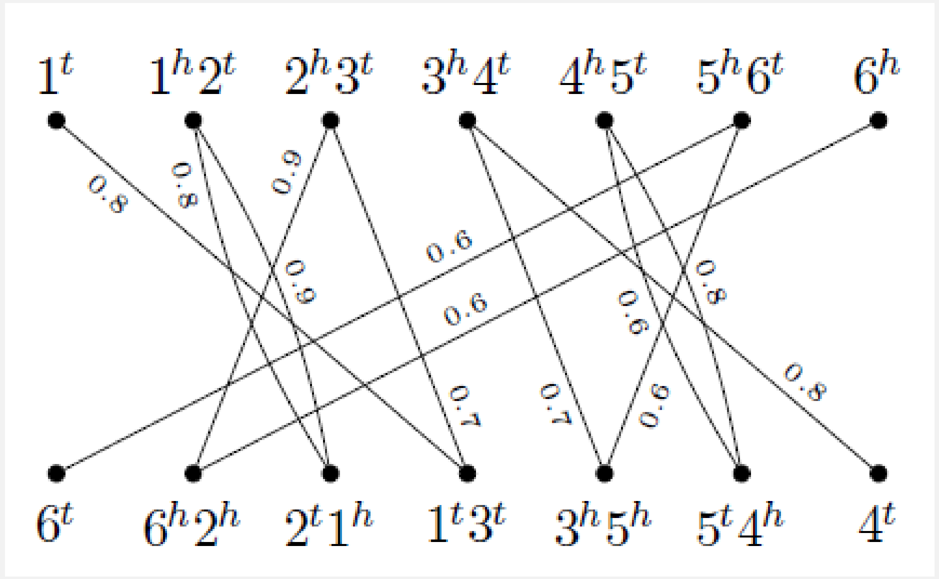
Faster heuristic algorithms

Heuristic algorithms: Best Density

- **First step**: generate all cycles in weighted adjacency graph
- **Second step**: Store all cycles C of $AG_{\boldsymbol{\sigma}}(A, B)$ in decreasing order of their ***density*** $\frac{w(C)}{|C|^2}$
- **Third step**: *while* it is possible select and add to M the best density *consistent* cycle, remove it from AG, *loop*
- **Final step**: return $\boldsymbol{s_{\sigma}(A, B)} = \sum_{C\in \mathrm{M}}(\frac{w(C)}{|C|})$

Heuristic algorithms: Best Density example

- **First step**: two cycles of length 2
- **Second step**: <c_{12}, 0.425> , < c_{45}, 0.350>
- **Final result**: $\boldsymbol{s_\sigma(A,B)} = \sum_{C\in M}\left(\frac{w(C)}{|C|}\right) = \left(\frac{0.9+0.8}{2}\right) + \left(\frac{0.8+0.6}{2}\right)$ **= 1.55**
- *Max value for similarity is the number of components in the matching*



Heuristic algorithms: Best Length

- **First step**: generate all cycles in weighted adjacency graph
- **Second step**: Store all cycles C of $AG_{\boldsymbol{\sigma}}(A, B)$ in increasing order of their ***lengths*** $|C|$
- **Third step**: *while* it is possible select and add to M *consistent* cycle of length 2, then of length 4 and so on, *until* there are no more *consistent* cycles.
- **Final step**: return $\boldsymbol{s_{\sigma}(A, B)} = \sum_{C \in \mathrm{M}} (\frac{w(C)}{|C|})$

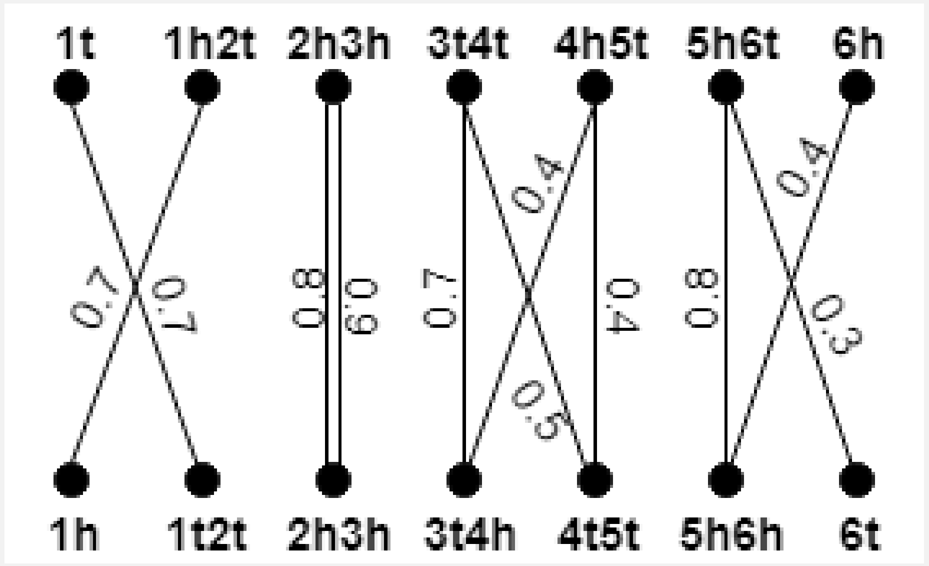
Heuristic algorithms: Best Length example

- **First step**: one 2-cycle and one 4-cycle
- **Second step**: <c_{23}, 2, 0.425> , < c_{45}, 4, 0.125>
- **Final result**: $\boldsymbol{s_\sigma(A, B)}$ =

$$\sum_{C\in \mathrm{M}}\left(\frac{w(C)}{|C|}\right) = \left(\frac{0.9+0.8}{2}\right)$$

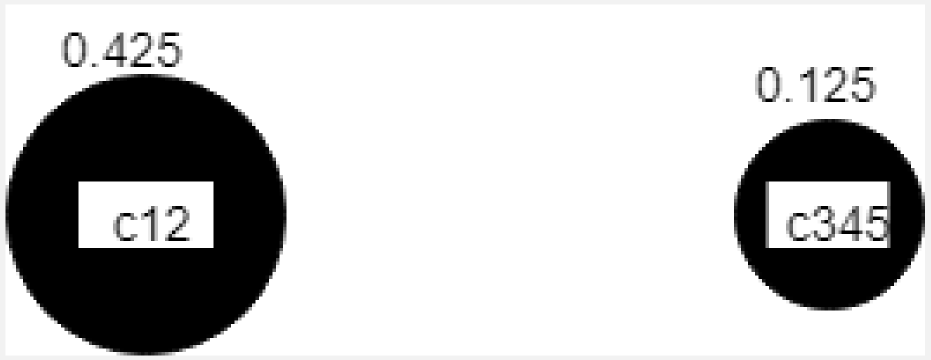
$$+\left(\frac{0.7+0.4+0.4+0.5}{4}\right) = \mathbf{1.35}$$

- *Lower similarity than before: 4-cycles are less important*



Heuristic algorithms: Best Length with Max Independent Set

- **Cycle Graph**: vertex is a cycle, edge if cycles are *inconsistent*
- **Weight of vertex**: *density* of cycle
- **Algorithm**: find an *indipendent set* with the greatest weight in the cycle graph

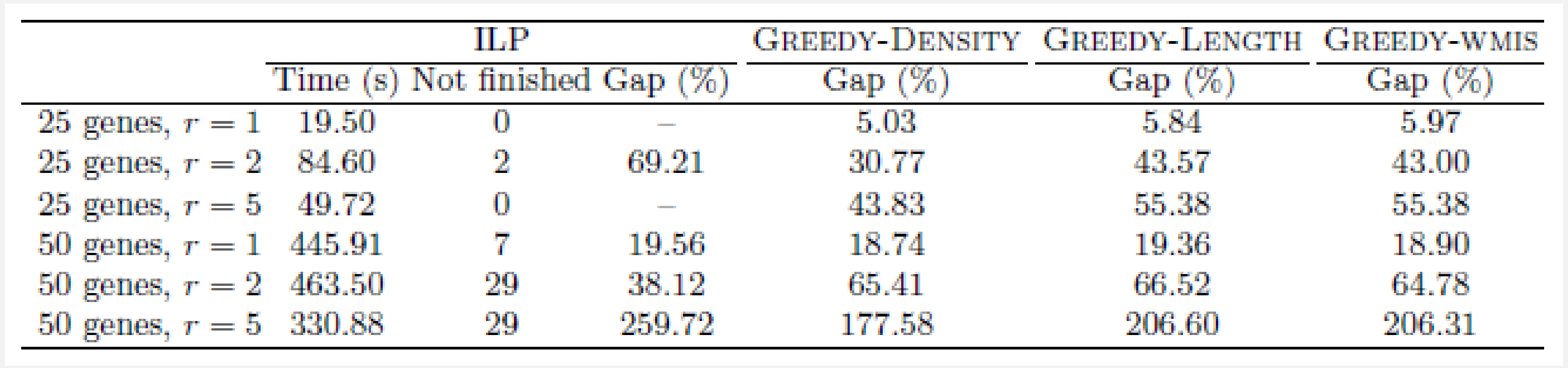


Experimental results

Experimental results: tests characteristics

- **Single machine**: 16GB memory, *gurobi* library for ILP computation, 1800sec time limit
- Each dataset with **10 simulated genome samples**: each genome of sizes around 25, 50 or 1000 genes (only used for heuristics)
- **45 comparisons** of pairs per dataset
- Tests on different proportions of **genome level events**: **r-fold** with rates 1, 2 and 5
- $\boldsymbol{Gap} = \left(\frac{upper\ bound}{best\ solution} - 1\right) * 100$

Experimental results: time and gap comparison

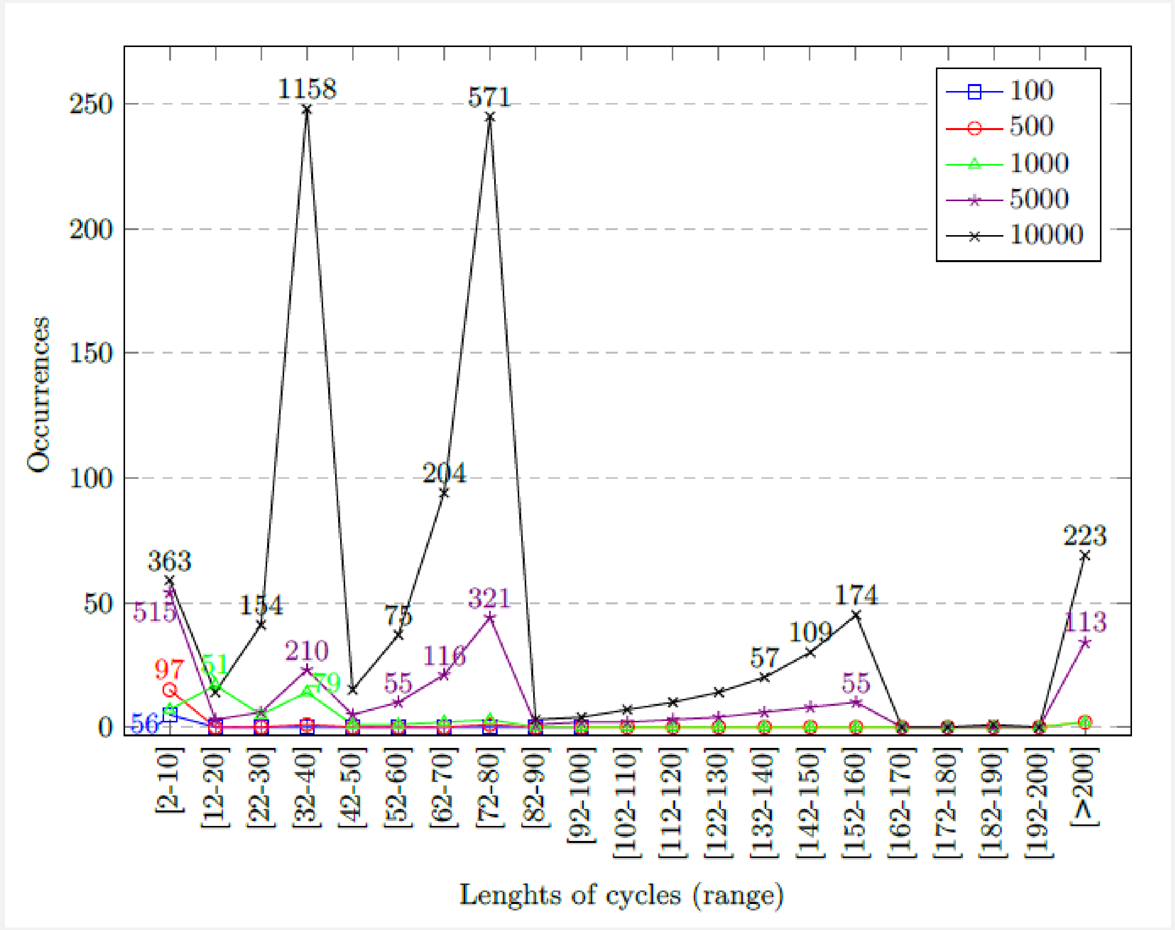


	ILP			Greedy-Density	Greedy-Length	Greedy-wmis
	Time (s)	Not finished	Gap (%)	Gap (%)	Gap (%)	Gap (%)
25 genes, $r = 1$	19.50	0	–	5.03	5.84	5.97
25 genes, $r = 2$	84.60	2	69.21	30.77	43.57	43.00
25 genes, $r = 5$	49.72	0	–	43.83	55.38	55.38
50 genes, $r = 1$	445.91	7	19.56	18.74	19.36	18.90
50 genes, $r = 2$	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

- **Gap** increases proportionally to the rate of reversals and translocations (association of genes is subject to this bias)
- **Greedy-density** provides almost every time the best solutions among the heuristics (balance between normalized weight and cycles length)

Experimental results: cycles distribution

- **5-fold** instances running the **Greedy-Density** algorithm on genomes with different size
- **Most of the cycles** found are **short** compared to the genome sizes
- That's why **heuristics are faster** (2896 sec on 10000 genes)
- Number of cycles could be **exponential** in theory → enumerate is bad!



Conclusion

Conclusion: ILP vs Heuristics

- **ILP** program is fast and accurate for smaller instances

- **Greedy-density** heuristic is probably the best choice for general use on larger istances

- **Drawback**: FF DCJ similarity values cannot be compared easily: value is between 0 and $|M|$

- **Best normalization**: divide sim. value by $|M|$

Thank you for your attention!

Reference: Diego P. Rubert, Gabriel L. Medeiros, Edna A. Hoshino, Marilia D.V. Braga, Jens Stoye, and Fabio V. Martinez

Any further info on slides: *sisbarra@gmail.com*