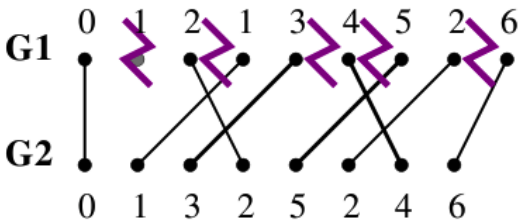
Gene Family Assignment-Free Comparative Genomics

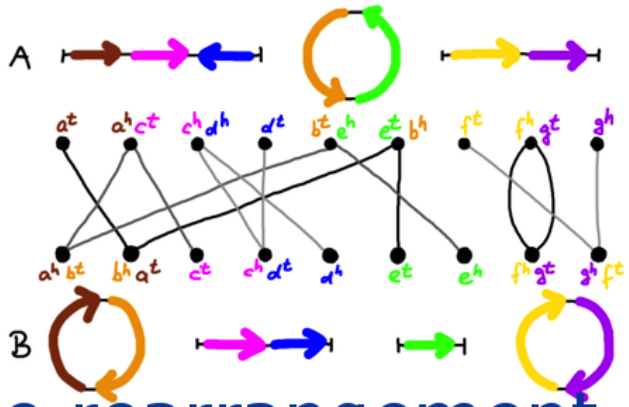
Daniel Doerr Annelyse Thévenin Jens Stoye

Genome Informatics, Faculty of Technology and
Institute for Bioinformatics, Center for Biotechnology (CeBiTec), Bielefeld University, Germany

Applications in Comparative Genomics



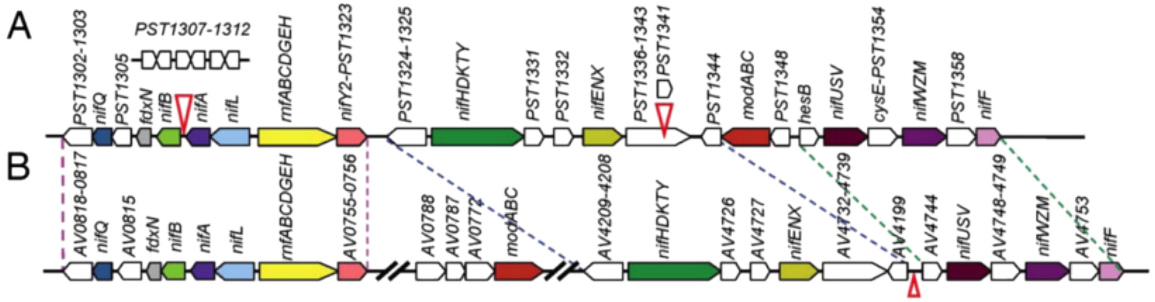
Gene order



Gene rearrangement

Comparative Genomics

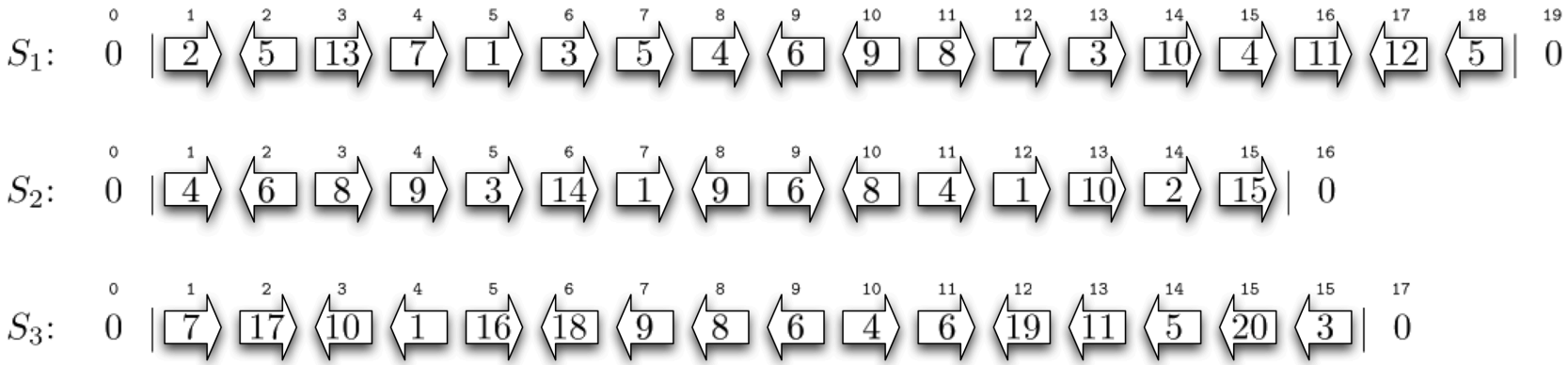
Gene cluster



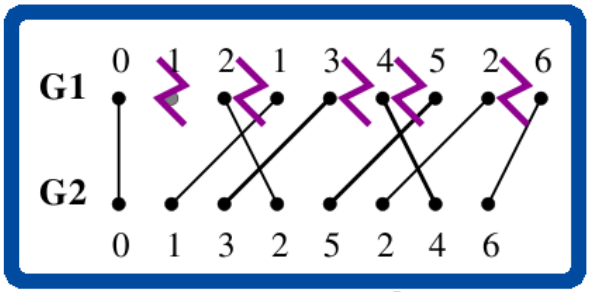
... and many more

Comparative genomics

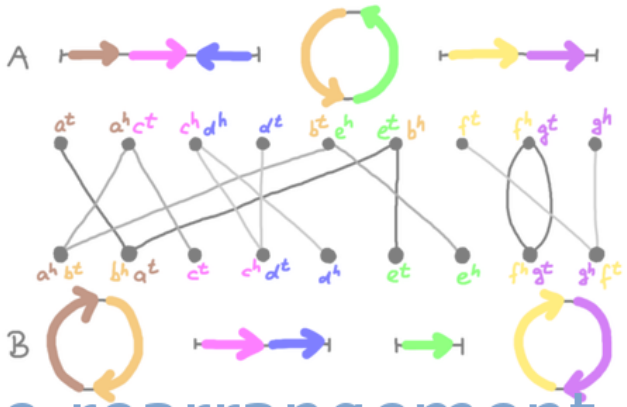
Prevalent abstract datastructure in comparative genomics is based on homology:



Comparing gene order of two genomes



Gene order



Gene rearrangement

Comparative Genomics

Gene cluster

... and many more

Comparing gene order of two genomes

Two genomes without duplication

G_1	$+0$	$+1$	$+2$	$+3$	$+4$	$+5$	$+6$	$+7$	$+8$	$+9$
G_2	$+0$	$+7$	$+3$	-5	-4	$+6$	$+1$	$+2$	-8	$+9$

A chromosome is a signed permutation.

Number of adjacencies

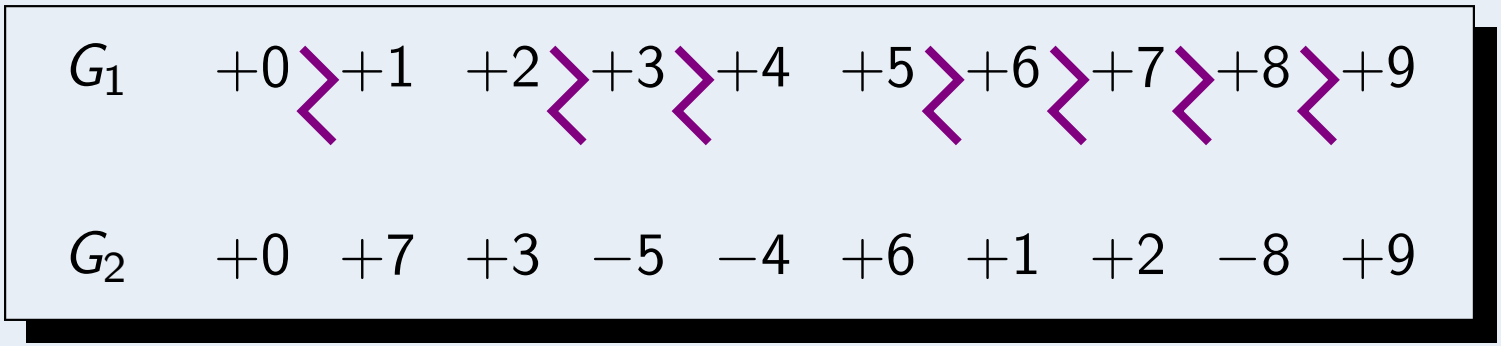
A measure of similarity

G_1	$+0$	$+1$	$+2$	$+3$	$+4$	$+5$	$+6$	$+7$	$+8$	$+9$
G_2	$+0$	$+7$	$+3$	-5	-4	$+6$	$+1$	$+2$	-8	$+9$

$\Rightarrow$ 2 adjacencies between G_1 and G_2.

Number of breakpoints [Sankoff and Blanchette, 1997]

A distance measure



Dual measure to the number of adjacencies:

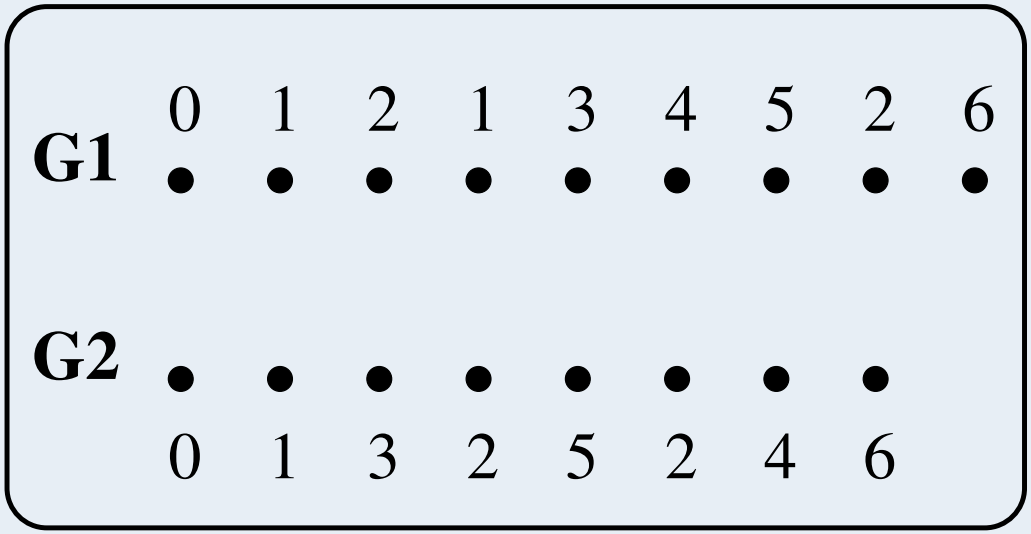
$$n = \#\mathtt{bkp}(G_1, G_2) + \#\mathtt{adj}(G_1, G_2) + 1$$

$\Rightarrow$ 7 breakpoints

Comparison of two genomes

Two genomes with duplications

A chromosome is a sequence over a set of signed characters.

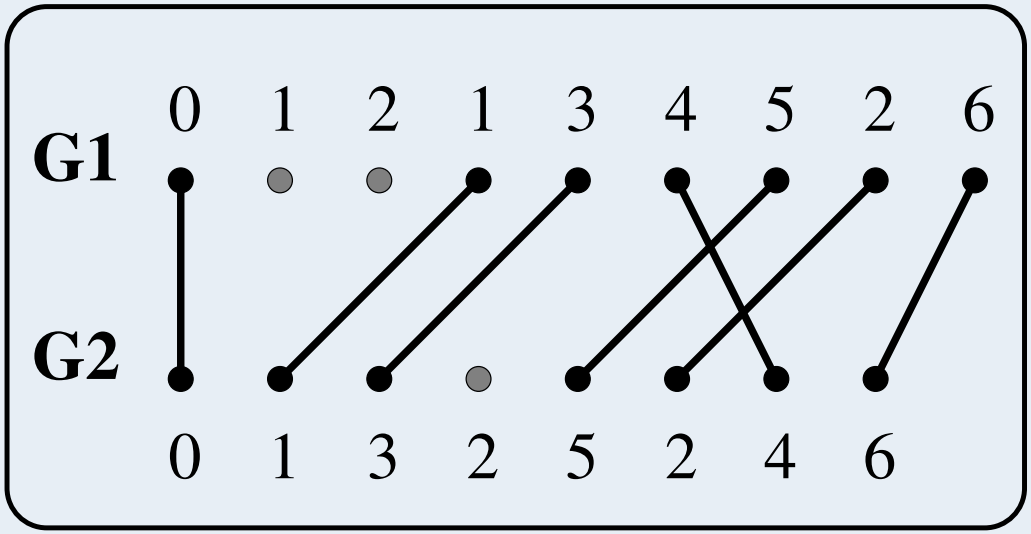


We need to find a matching between both genomes.

Comparison of two genomes

Two genomes with duplications

A chromosome is a sequence over a set of signed characters.



We need to find a matching between both genomes.

Matching models

There are 3 matching models:

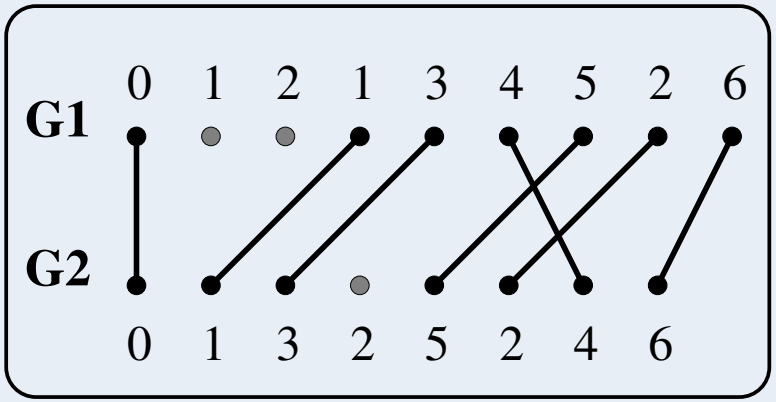
Exemplar - saturate exactly one gene of each gene family
[Sankoff, 1999]

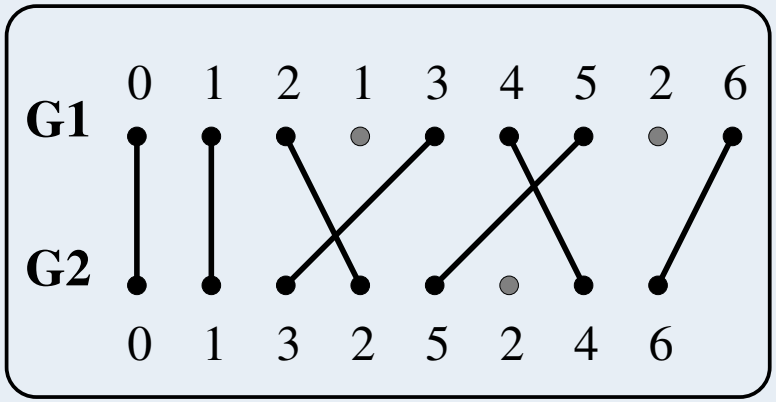
Maximum - saturate as many genes as possible of each gene family [Tang and Moret, 2003].

Intermediate - saturate at least one gene of each gene family
[Angibaud *et al.*, 2008]

Number of adjacencies varies in matchings

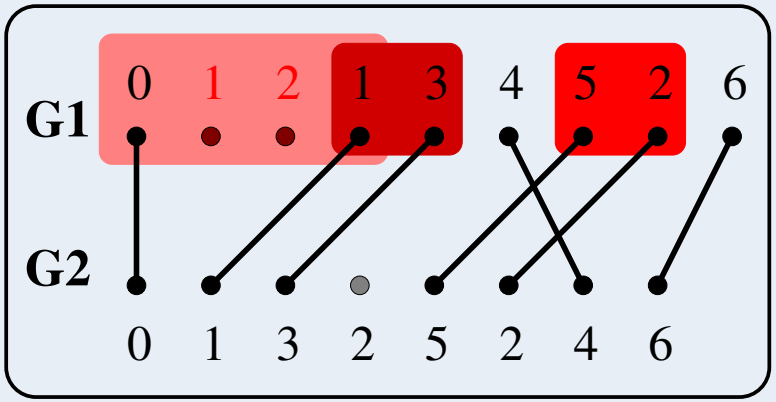
Example: Number of adjacencies in two different matchings under the **exemplar model**:



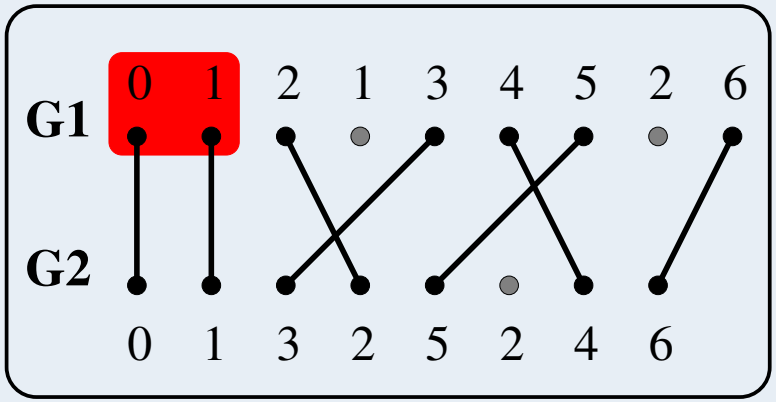


Number of adjacencies varies in matchings

Example: Number of adjacencies in two different matchings under the **exemplar model**:



3 adjacencies



1 adjacency

Current results

Maximizing the number of adjacencies under one of these three models is a **NP-hard** problem.

There exists **exact** (and realistic) **programs and heuristics** to resolve it. [Angibaud *et al.*, 2008].

Gene family consequences

Pros

+ Subsequent analyses produce **reasonable results**.
+ Facilitates **simple** but **powerful** datastructure.
+ Gene family information: Many **databases** and tools available.

Cons

– Wrong gene family assignments produce **incorrect results** in subsequent analyses.
– Datastructure: Strong and weak homology assumptions are **indifferent**.
– Gene family concept **not applicable** for all biological phenomena.

Gene family consequences

Pros

+ Subsequent analyses produce **reasonable results**.
+ Facilitates **simple** but **powerful** datastructure.
+ Gene family information: Many **databases** and tools available.

Cons

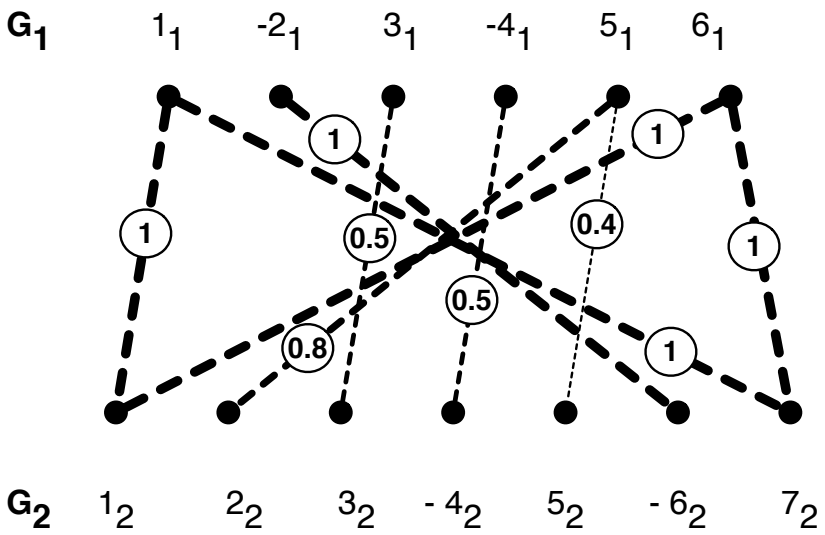
– Wrong gene family assignments produce **incorrect results** in subsequent analyses.
– Datastructure: Strong and weak homology assumptions are **indifferent**.
– Gene family concept **not applicable** for all biological phenomena.

New model

Gene family assignment-free

Gene family assignment free

Normalized similarity measure: $\sigma : G_1 \times G_2 \rightarrow [0, 1]$



Datastructure is an ordered weighted bipartite graph.

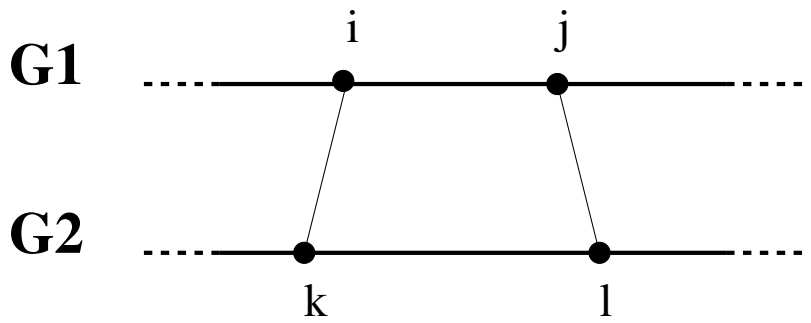
Conserved Adjacencies

Scoring scheme for adjacencies

Adjacencies in a matching $\mathcal{M}$ are scored according to the measure σ of the corresponding edges as follows:

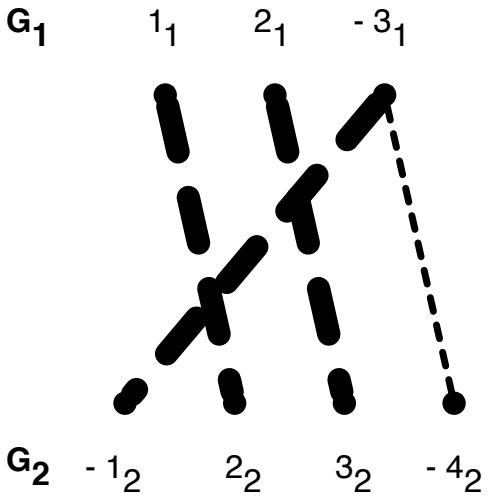
$$s(i,j,k,\ell) = \begin{cases} \sigma(G_1[i], G_2[k]) \cdot \sigma(G_1[j], G_2[\ell]) & \text{if adjacent } * \\ 0 & \text{otherwise} \end{cases}$$

*adjacent: $(G_1[i], G_1[j])$ and $(G_2[k], G_2[\ell])$ are saturated and consecutive (taking sign into account)



Adjacencies or edges?

Quantifying the **quality of a matching** $\mathcal{M}$: Adjacencies vs edges.



$$adj(\mathcal{M}) = \sum_{\substack{0 \le i < j \le |G_1|, \\ 0 \le k, \ell \le |G_2|}} s(i,j,k,\ell)$$

$$edg(\mathcal{M}) = \sum_{e \in \mathcal{M}} \sigma(e)^2$$

Studied problem

Family-free(FF)-Adjacencies problem

Given two genomes G_1 and G_2 and some $\alpha \in]0, 1]$, find an intermediate matching $\mathcal{M}$ such that at least one edge per connected component is covered and the following formula is maximized:

$$\mathcal{F}_\alpha(\mathcal{M}) = \alpha \cdot adj(\mathcal{M}) + (1 - \alpha) \cdot edg(\mathcal{M}) .$$

Generalization of intermediate (and maximum) matching

Lemma (Generalization scheme)

Let the graph between two genomes be such that all edges have edge weight 1 and all connected components are fully connected.

Then for maximizing $\mathcal{F}_\alpha$ *with*

- $\alpha = 1$ *is equivalent maximizing the number of adjacencies under the* **intermediate matching** *model;*
- $\alpha \to 0$ *is equivalent maximizing the number of adjacencies under the* **maximum matching** *model*

with gene family assignment.

$\Rightarrow$ **NP-hard** problem.

Our strategy

Goal: Family-free comparative genome analysis.

Strategy: Resolve a particular case: For a given pair of genomes G_1 and G_2, find optimal solution for FF-adjacencies problem.

Method: Exact algorithm and heuristic.

Algorithms

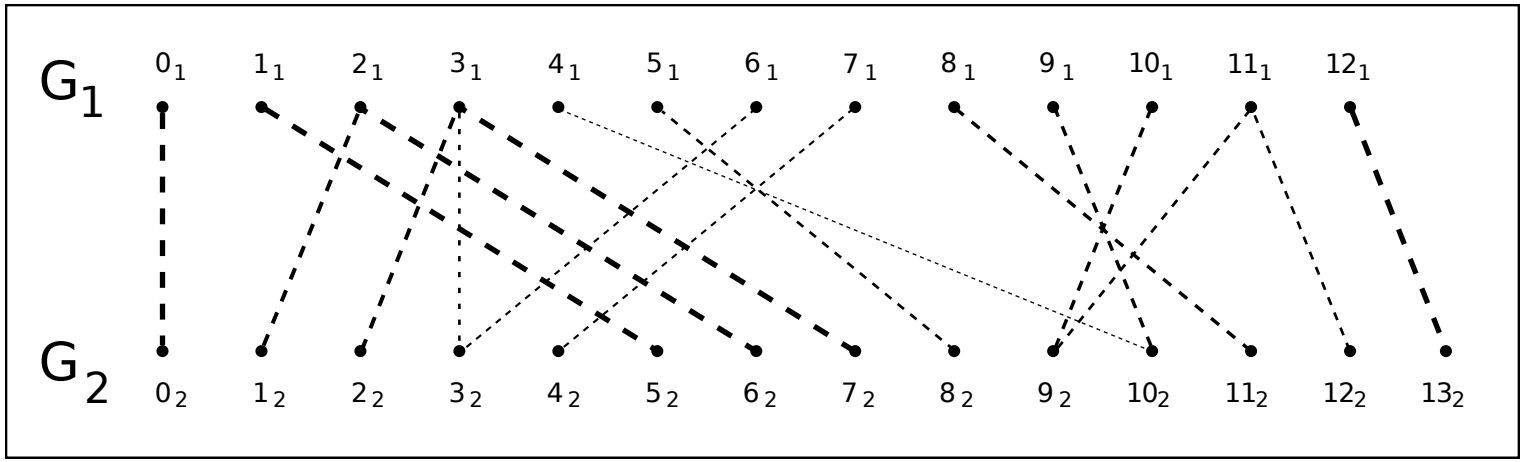
Algorithms

FFAdj-Int Exact algorithm, implemented as pseudo-boolean program, based on previous work [Angibaud *et al.*, 2008]

FFAdj-MCS Heuristic, based on LCS - Longest Common Substring [Marron *et al.*, 2004]

Heuristic FFAdj-MCS

Based on **LCS** - Longest Common Substring [Marron *et al.*, 2004]

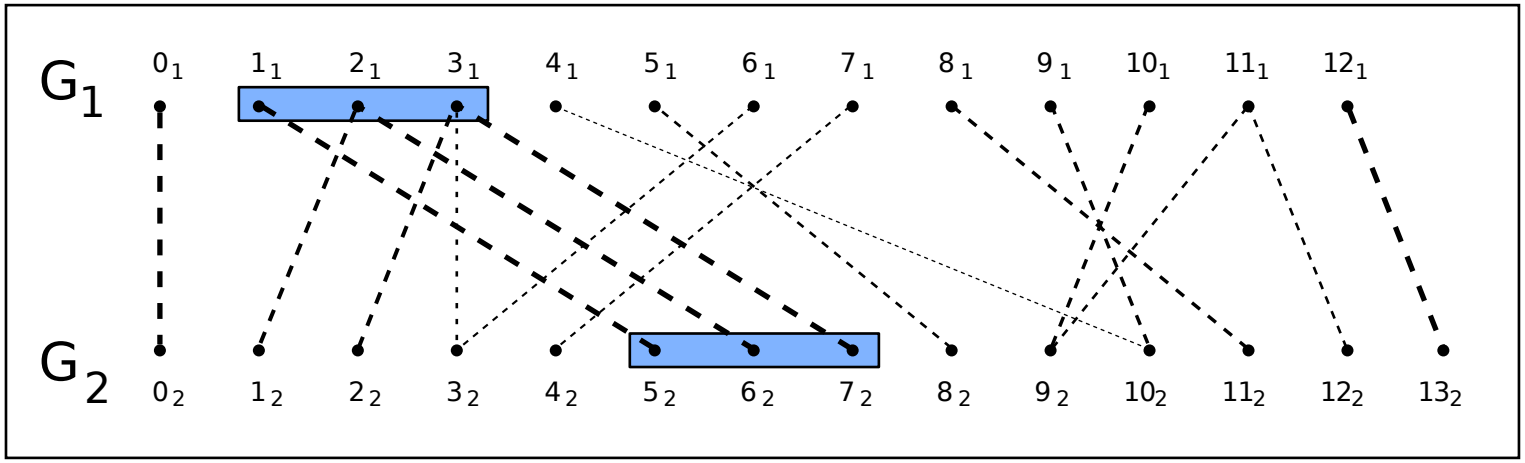


Heuristic FFAdj-MCS

1. Compute S: The Maximum Common Substring (MCS) up to a reversal.
2. Map all the genes of S accordingly.
3. Remove genes that cannot be matched any longer according to the model. Grown up previous MCS if possible.
4. Iterate the process until saturation.

Heuristic FFAdj-MCS

Based on **LCS** - Longest Common Substring [Marron *et al.*, 2004]

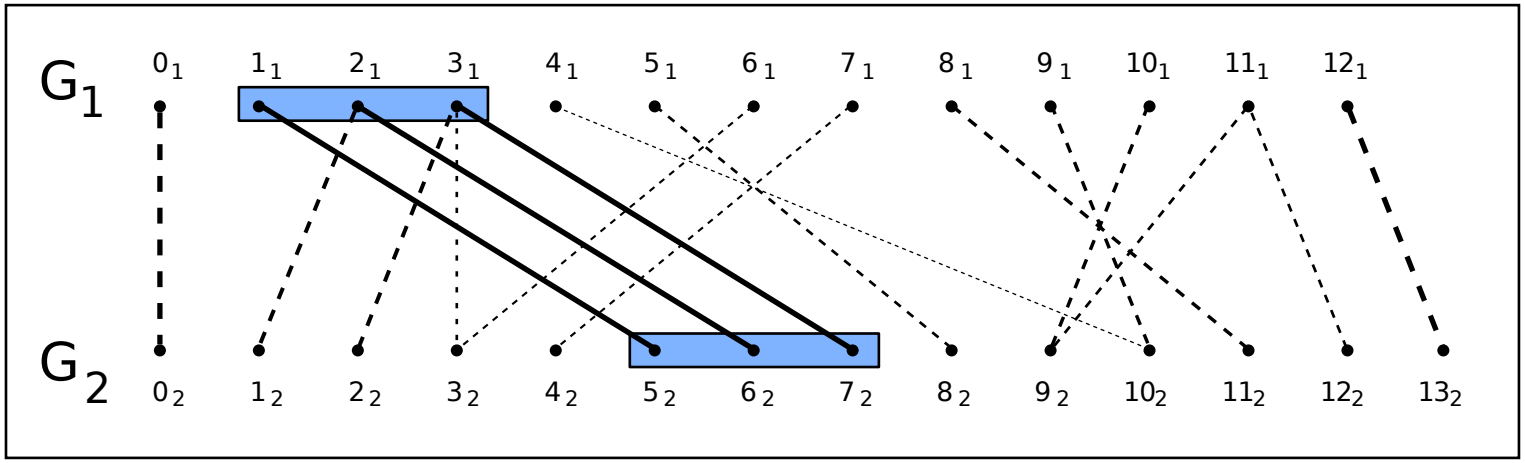


Heuristic FFAdj-MCS

1. Compute S: The Maximum Common Substring (MCS) up to a reversal.
2. Map all the genes of S accordingly.
3. Remove genes that cannot be matched any longer according to the model. Grown up previous MCS if possible.
4. Iterate the process until saturation.

Heuristic FFAdj-MCS

Based on **LCS** - Longest Common Substring [Marron *et al.*, 2004]

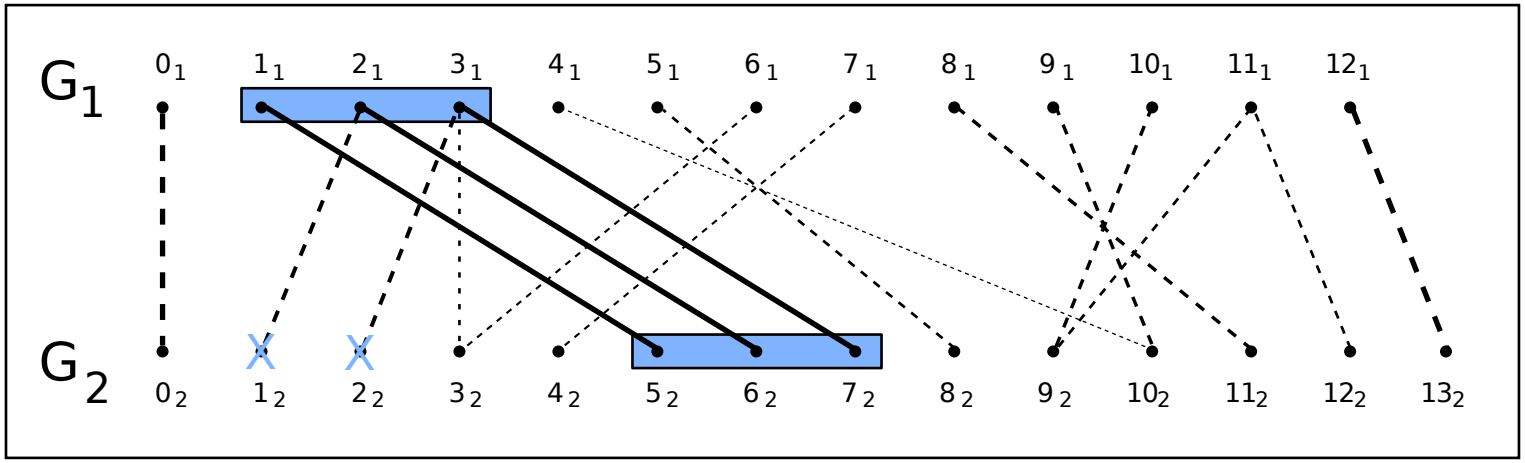


Heuristic FFAdj-MCS

1. Compute S: The Maximum Common Substring (MCS) up to a reversal.
2. Map all the genes of S accordingly.
3. Remove genes that cannot be matched any longer according to the model. Grown up previous MCS if possible.
4. Iterate the process until saturation.

Heuristic FFAdj-MCS

Based on **LCS** - Longest Common Substring [Marron *et al.*, 2004]

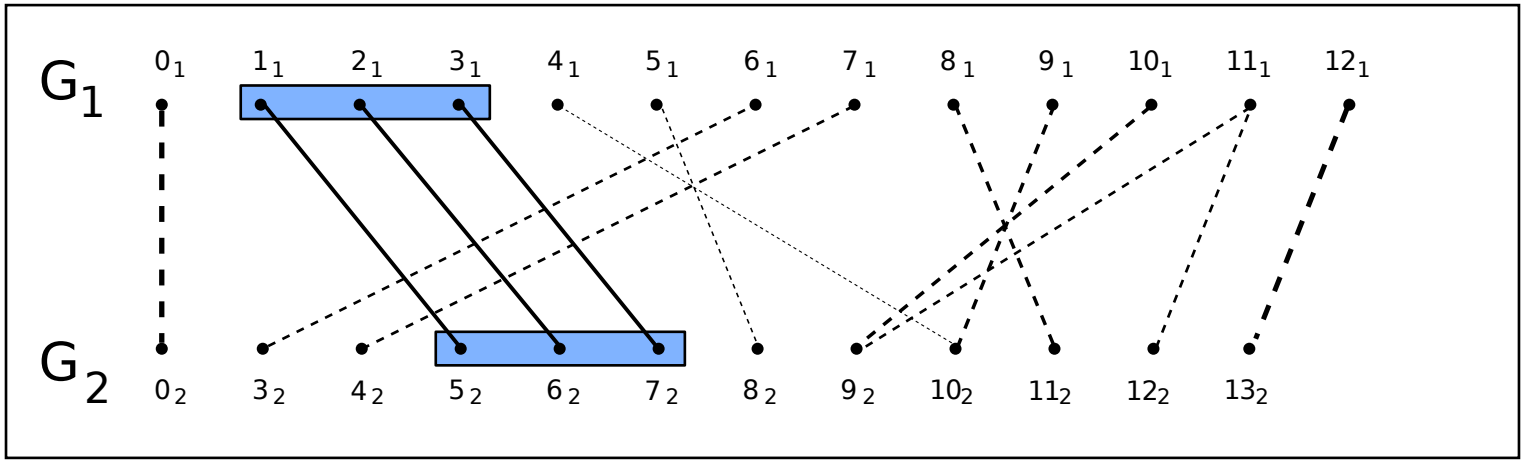


Heuristic FFAdj-MCS

1. Compute S: The Maximum Common Substring (MCS) up to a reversal.
2. Map all the genes of S accordingly.
3. Remove genes that cannot be matched any longer according to the model. Grown up previous MCS if possible.
4. Iterate the process until saturation.

Heuristic FFAdj-MCS

Based on **LCS** - Longest Common Substring [Marron *et al.*, 2004]

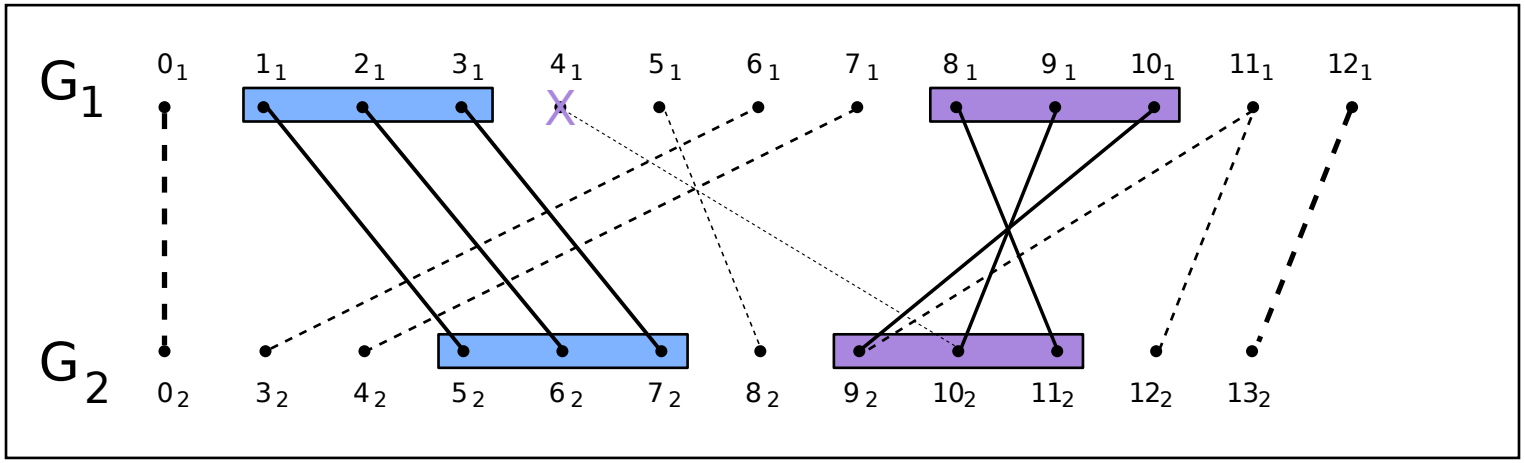


Heuristic FFAdj-MCS

1. Compute S: The Maximum Common Substring (MCS) up to a reversal.
2. Map all the genes of S accordingly.
3. Remove genes that cannot be matched any longer according to the model. Grown up previous MCS if possible.
4. Iterate the process until saturation.

Heuristic FFAdj-MCS

Based on **LCS** - Longest Common Substring [Marron *et al.*, 2004]

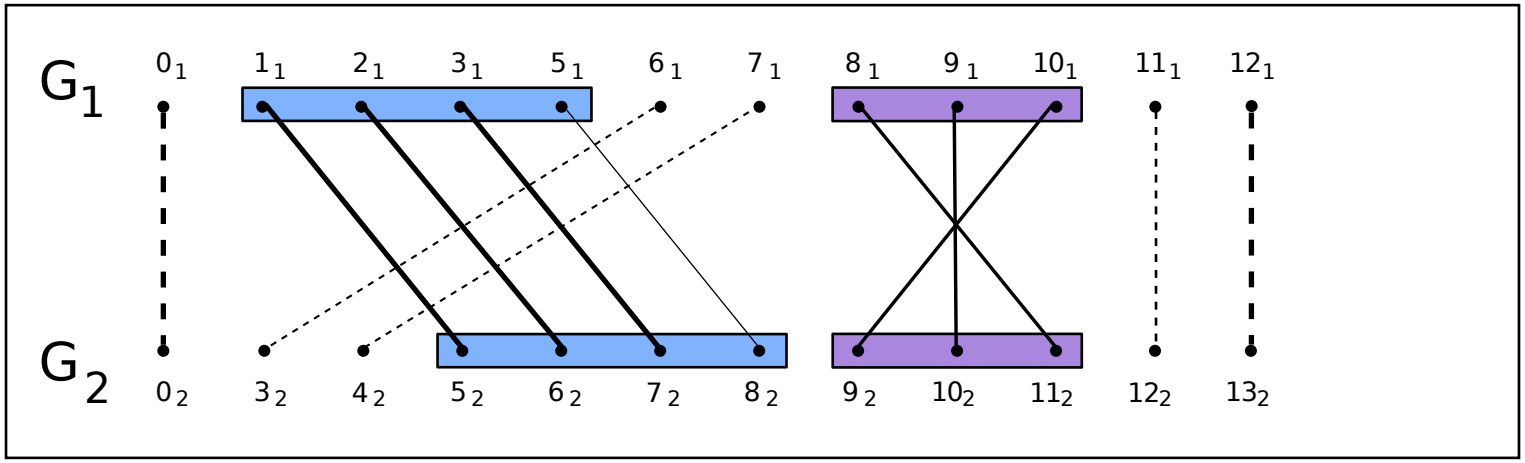


Heuristic FFAdj-MCS

1. Compute S: The Maximum Common Substring (MCS) up to a reversal.
2. Map all the genes of S accordingly.
3. Remove genes that cannot be matched any longer according to the model. Grown up previous MCS if possible.
4. Iterate the process until saturation.

Heuristic FFAdj-MCS

Based on **LCS** - Longest Common Substring [Marron *et al.*, 2004]

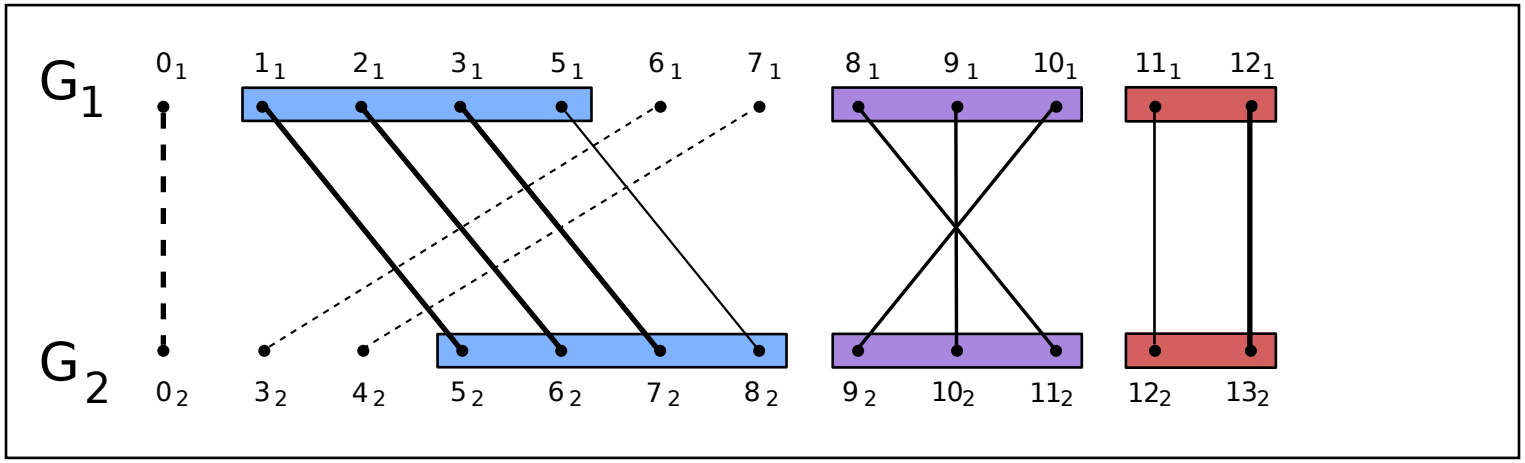


Heuristic FFAdj-MCS

1. Compute S: The Maximum Common Substring (MCS) up to a reversal.
2. Map all the genes of S accordingly.
3. Remove genes that cannot be matched any longer according to the model. Grown up previous MCS if possible.
4. Iterate the process until saturation.

Heuristic FFAdj-MCS

Based on **LCS** - Longest Common Substring [Marron *et al.*, 2004]

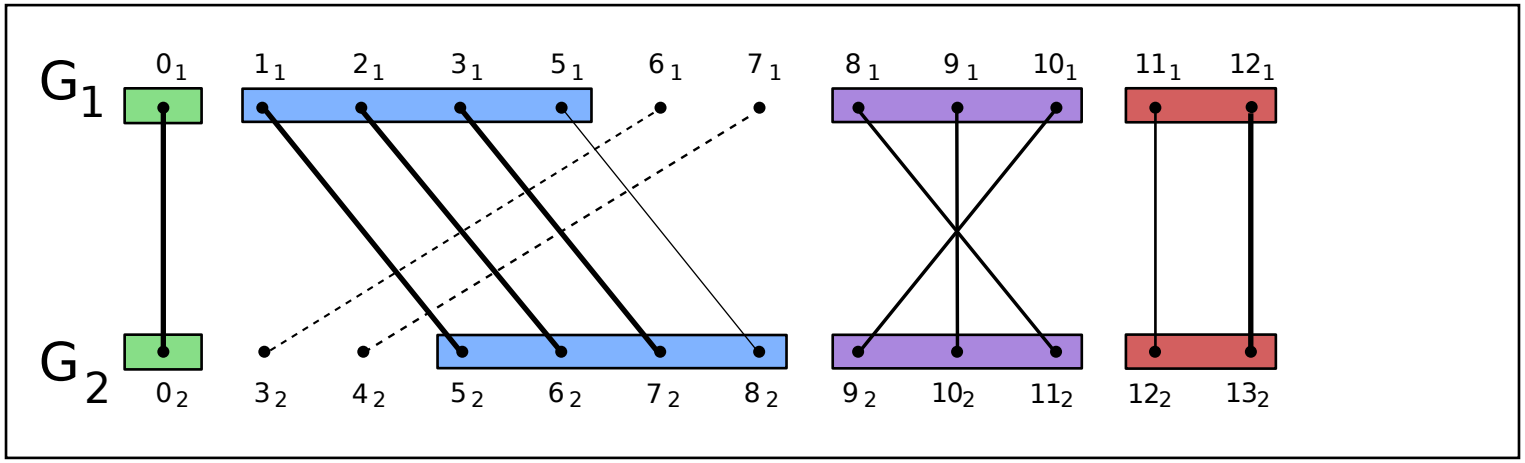


Heuristic FFAdj-MCS

1. Compute S: The Maximum Common Substring (MCS) up to a reversal.
2. Map all the genes of S accordingly.
3. Remove genes that cannot be matched any longer according to the model. Grown up previous MCS if possible.
4. Iterate the process until saturation.

Heuristic FFAdj-MCS

Based on **LCS** - Longest Common Substring [Marron *et al.*, 2004]

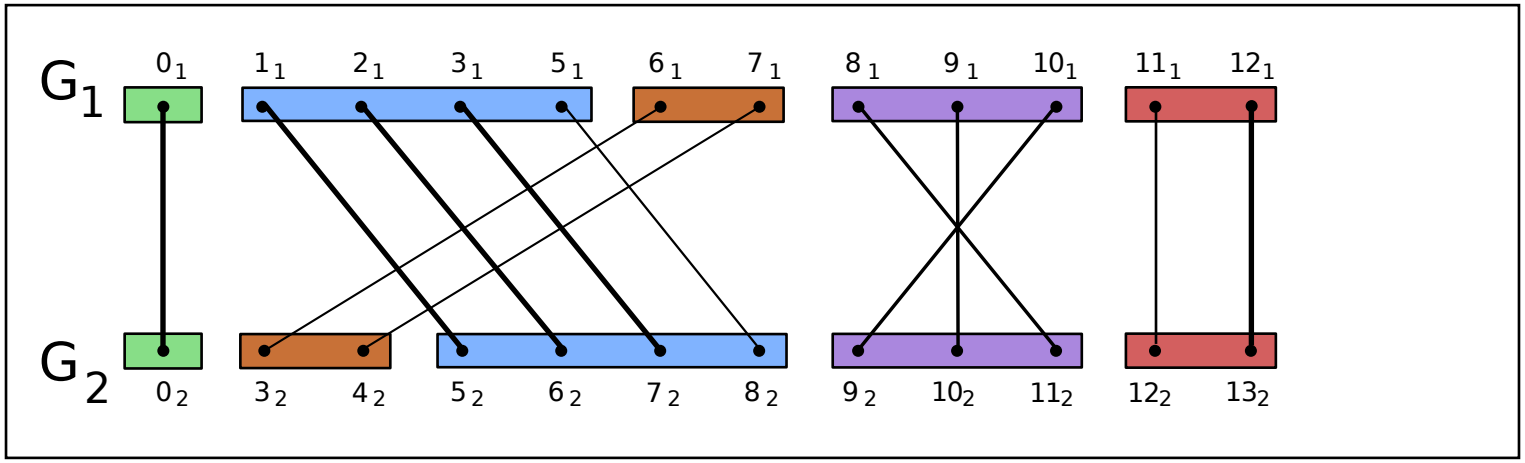


Heuristic FFAdj-MCS

1. Compute S: The Maximum Common Substring (MCS) up to a reversal.
2. Map all the genes of S accordingly.
3. Remove genes that cannot be matched any longer according to the model. Grown up previous MCS if possible.
4. Iterate the process until saturation.

Heuristic FFAdj-MCS

Based on **LCS** - Longest Common Substring [Marron *et al.*, 2004]

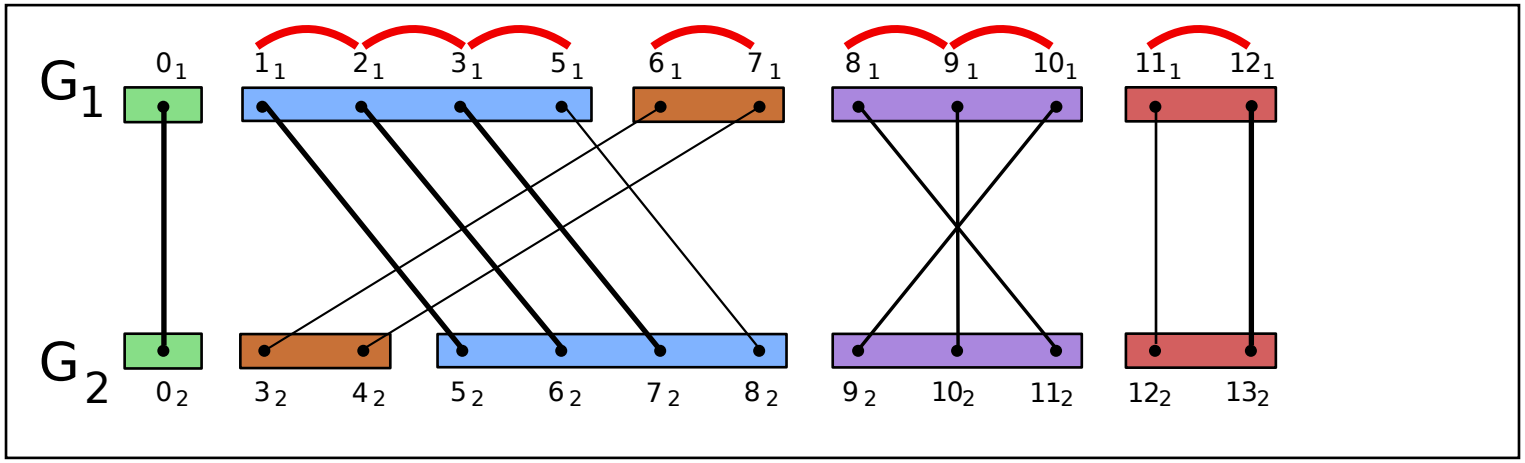


Heuristic FFAdj-MCS

1. Compute S: The Maximum Common Substring (MCS) up to a reversal.
2. Map all the genes of S accordingly.
3. Remove genes that cannot be matched any longer according to the model. Grown up previous MCS if possible.
4. Iterate the process until saturation.

Heuristic FFAdj-MCS

Based on **LCS** - Longest Common Substring [Marron *et al.*, 2004]



Heuristic FFAdj-MCS

1. Compute S: The Maximum Common Substring (MCS) up to a reversal.
2. Map all the genes of S accordingly.
3. Remove genes that cannot be matched any longer according to the model. Grown up previous MCS if possible.
4. Iterate the process until saturation.

Evaluation of our methods

Experimentation

Dataset

- 12 γ-proteobacteria complete genomes,
- Size: Between 564 and 5571 genes,
- Already used in [Angibaud *et al.*, 2008],
- 7.6% of duplicated genes.

- The parameter α is in $\{0.001, 0.3, 0.5, 0.8, 1\}$.
- Pairwise normalized similarities σ were obtained using the Relative Reciprocal BLAST Score (RRBS)
- The solver used is CPLEX http://www.ilog.com/products/cplex.

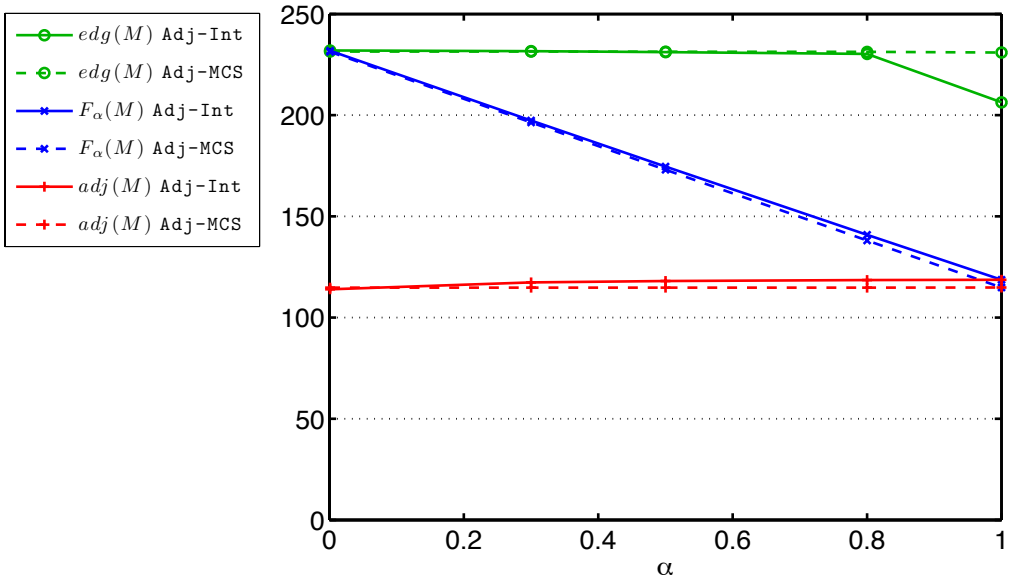
Exact algorithm results

- For **40 out of 66 pairs of genomes** we could solve the pseudo-boolean program for all values of α (few seconds by comparison)
- We do **not obtain all exact results** because to the lack of sufficient memory, in particular for similar and long genomes

Results of the heuristic FFAdj-MCS

Quality: The heuristic FFAdj-MCS deviates in the objective by **less than 3%** (between 0.2% for $\alpha = 0.001$ and 2.9% for $\alpha = 1$).

Time: Few minutes.



Comparison with previous results

With or without gene family assign

Although the number of adjacencies is artificially increased in the gene family assignment study (the genes are unsigns), we observed **the same number of adjacencies** relative to the size of the matching (which increase) in the results of `FFAdj-Int` (for $\alpha = 1$).

Definition

Distance in phylogenetic reconstruction

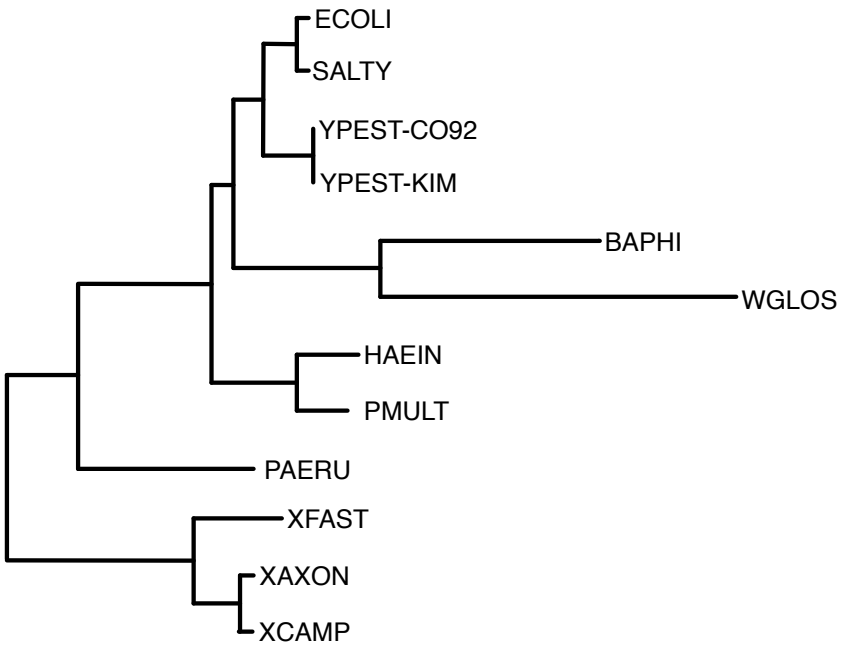
Normalized distance based on previous observations:

$$\Delta(\mathcal{M}) = \frac{edg(\mathcal{M}) - adj(\mathcal{M}) - 1}{edg(\mathcal{M})}$$

We used **Neighbor Joining** method for inferring phylogenetic trees.

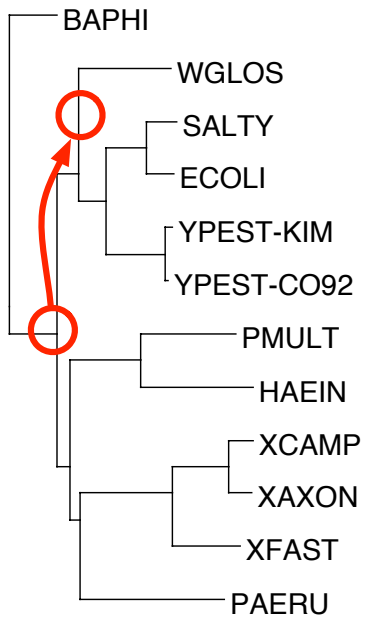
Reconstructed trees

True phylogeny [Lerat, 2003]



Reconstructed tree

(Robinson-Foulds distance: 2)



This branch is known to be particularly hard to reconstruct since the two organisms diverged far from each other.

Conclusions

Conclusions

Idea: **Direct analysis** of genomes without prior assignment of gene families,

Practical case: Find a **matching optimizing similarity** between two genomes,

Results: **Exact algorithm**, efficient for small genomes, and a good **heuristic**,

Evaluation: Free family assignment gene **improve the comparison**.

Future works

- **Improve the exact algorithm** to reduce the required memory
- Develop a **hybrid** heuristic
- Deep study of the **measure** σ
- Apply gene family assignment-free strategy to **other practical cases**

Acknowledgments

Unterstützt von / Supported by

Alexander von Humboldt
Stiftung/Foundation