Querying Protein-Protein Interaction Networks

Guillaume Blin Florian Sikora[1] Stéphane Vialette[1]

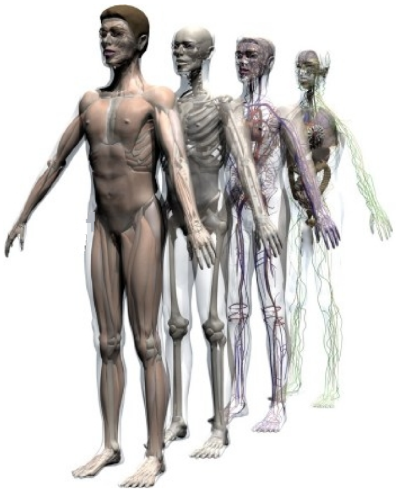
Université Paris-Est, LIGM - UMR CNRS 8049 - France
{gblin,sikora,vialette}@univ-mlv.fr

ISBRA May 2009

Outline

Outline

Motivations



25000 30000 45000

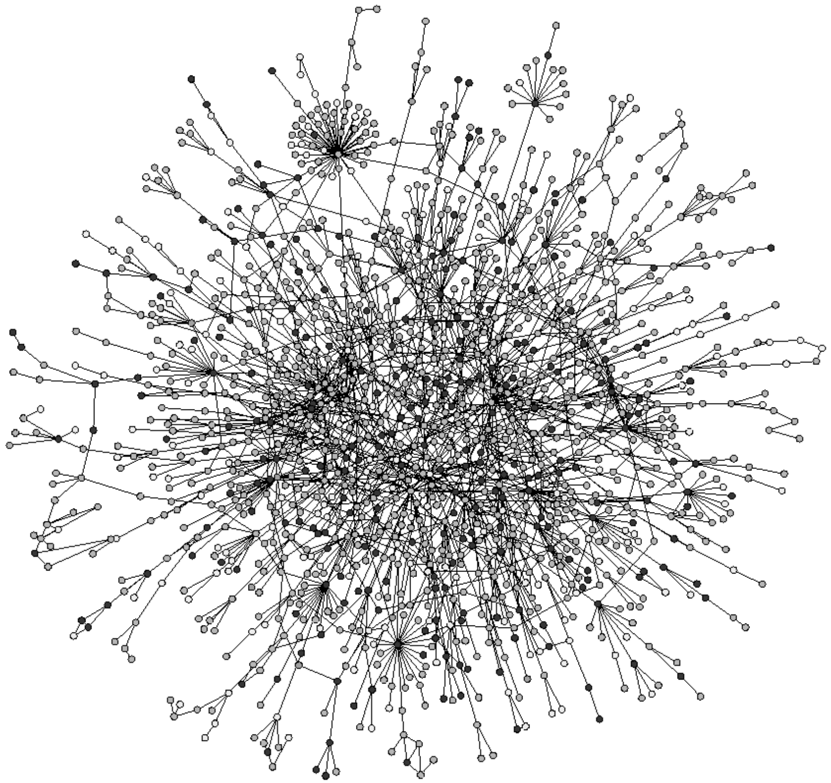
- Human complexity $\not\Leftrightarrow$ # of genes ?
- Human complexity $\Leftrightarrow$ proteins ?

Proteins..

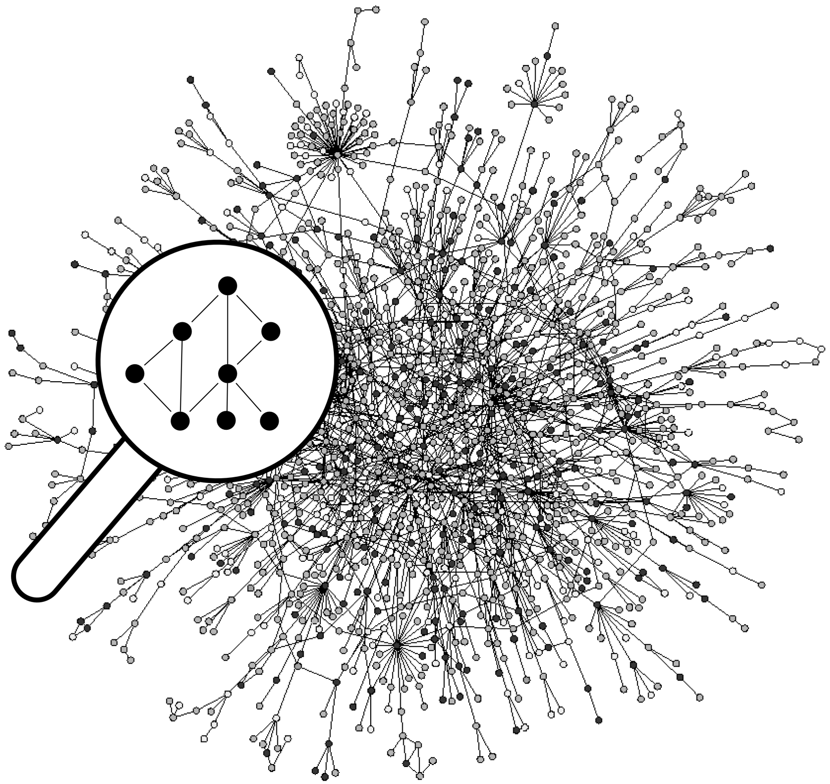
- New interest on proteins...
- ... and on their interactions: Protein-Protein Interactions (PPI)
- Biologically obtained... with noise !

Proteins network

- Proteins can interact with others proteins



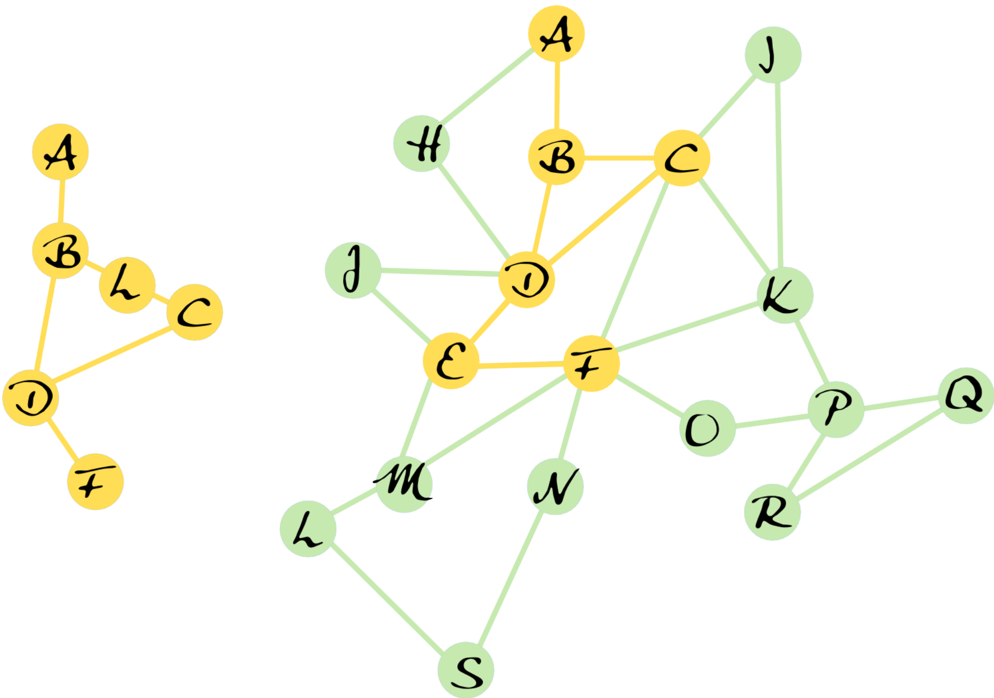
Proteins network



- Proteins are nodes
- Interactions are edges
- Edges can be weighed by interaction probability

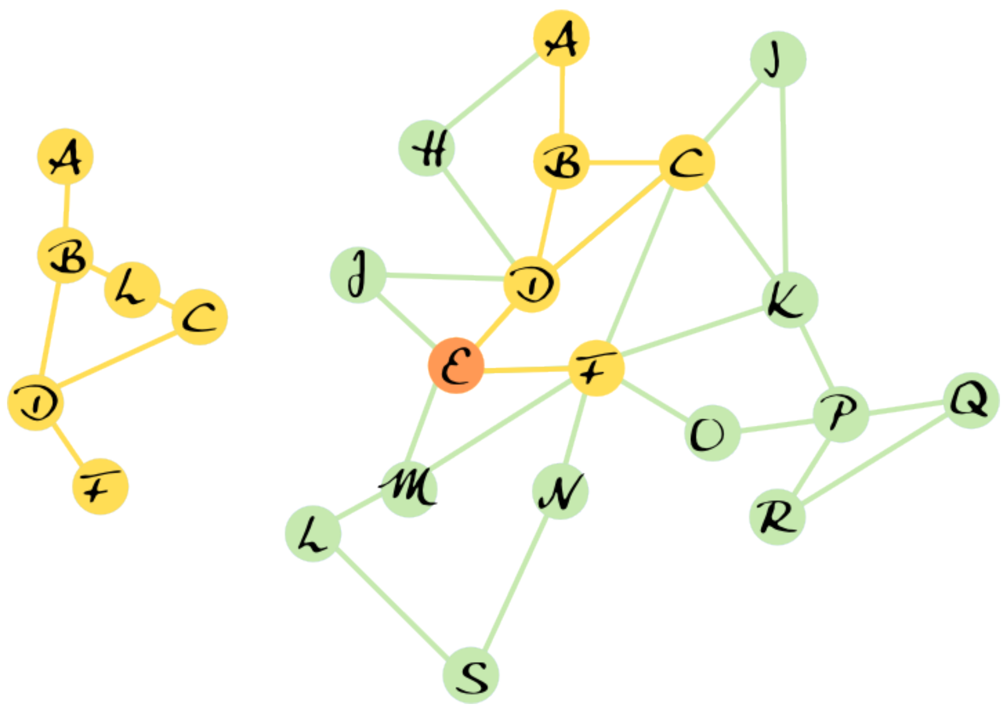
Searching patterns

- Searching patterns (set of proteins with a topology) in a PPI Network
- A protein is said to be **homologous** to another protein according to a BLAST sequence analysis



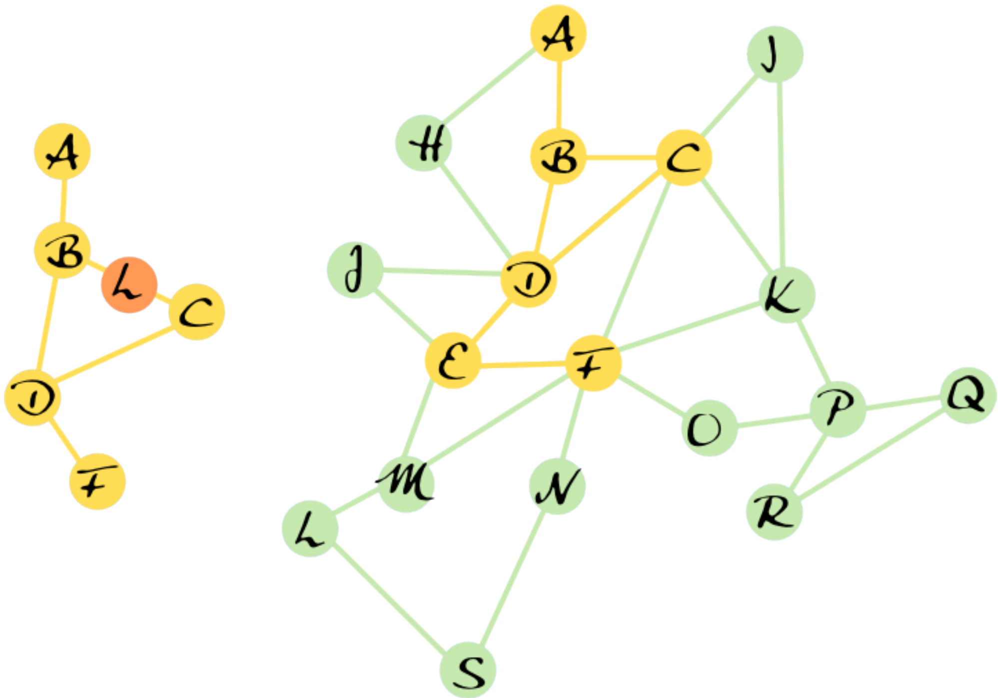
Searching patterns

- Searching patterns (set of proteins with a topology) in a PPI Network
- A protein is said to be **homologous** to another protein according to a BLAST sequence analysis
- Can allow a bounded number of **insertions** and deletions



Searching patterns

- Searching patterns (set of proteins with a topology) in a PPI Network
- A protein is said to be **homologous** to another protein according to a BLAST sequence analysis
- Can allow a bounded number of insertions and **deletions**

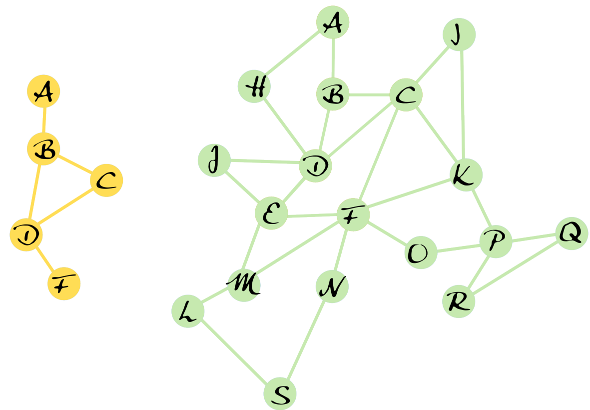


Motivation to search patterns

- Retrieve known functions
- **Deduce** information **from well-known** species **to less known** species

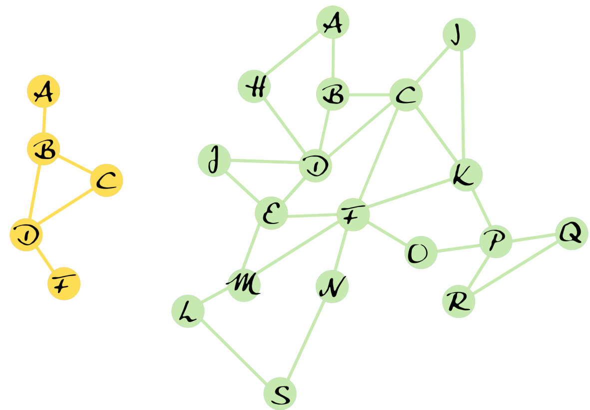
Graph Query

- Problem equivalent to **NP-hard** problem Graph Homeomorphism [GAREY&JOHNSON 1979]
- Exact solution → exponential runtime



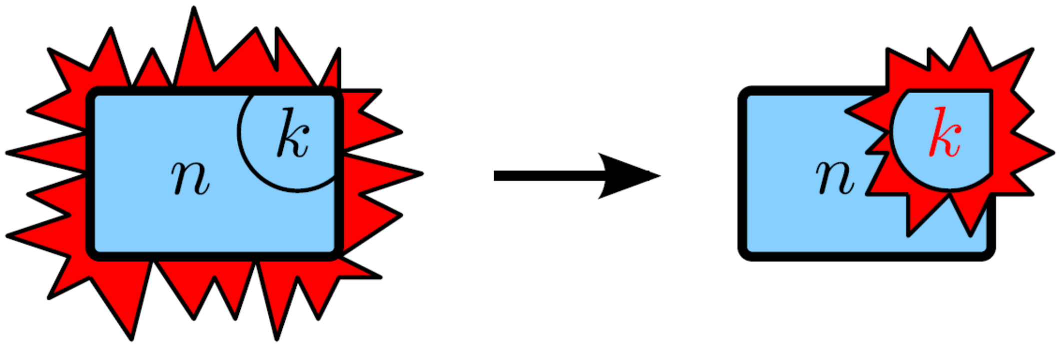
Graph Query

- Problem equivalent to **NP-hard** problem Graph Homeomorphism [GAREY&JOHNSON 1979]
- Exact solution → exponential runtime
- Idea: exploit the fact that **patterns are smaller** ($\sim 5 - 15$) than the network (e.g. $\sim$ 5.000 for the yeast)
- Restrict the exponential part to *k* instead of *n*: **parametrized complexity**



FPT Algorithms

- An FPT algorithm [DOWNEY & FELLOWS 1999]: exact algorithm **exponential** only in its **parameter k** (not in the input size *n*)
- $f(k).n^c$, with *c* a constant and *f* any function



State of the art

- When the pattern is a **path**:
  - Algorithm with a factorial complexity by [Kelley et al. 2003]
  - QPath, a faster FPT algorithm by [Shlomi et al. 2006]
- When the pattern is a **tree**
  - The network must be a forest [Pinter et al. 2005]
  - QNet, FPT algorithm [Dost et al. 2007]
- When the pattern is a **graph**
  - QNet [Dost et al. 2007]
    - Complexity exponential in the treewidth
    - Using the color-coding technique
    - Only theoretical result

Outline

Motivations and state of the art

Our Algorithm, PADA1

Experimental Results

PADA1 Outline

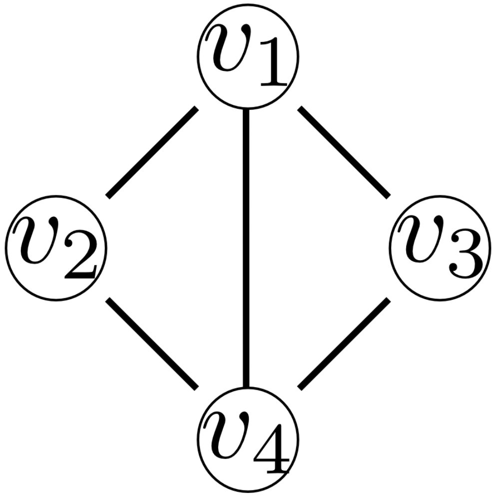
- Provide another pratical **exact** algorithm to query graphs in a PPI Network not depending on the treewidth

PADA1 Outline

- Provide another pratical **exact** algorithm to query graphs in a PPI Network not depending on the treewidth
- Adapt the FPT algorithm when the pattern is a tree
- 2 steps
  1. **Transform** the query in a tree without loss (with VFS and duplication)
  2. **Find** an occurence of that tree in the network by dynamic programming

Step 1: Graph to Tree

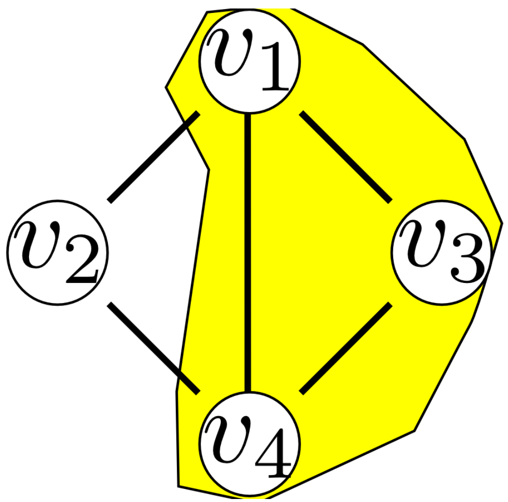
- As long as there are cycles in the graph, duplicate a node involved in the cycle (lossless)



- Original Graph

Step 1: Graph to Tree

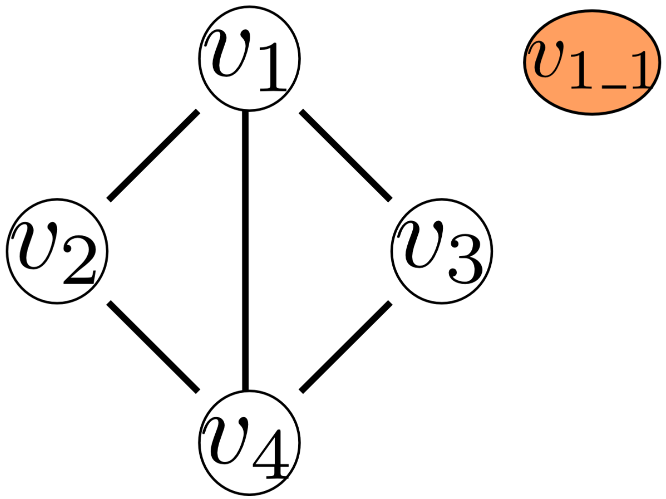
- As long as there are cycles in the graph, duplicate a node involved in the cycle (lossless)



- Find a cycle

Step 1: Graph to Tree

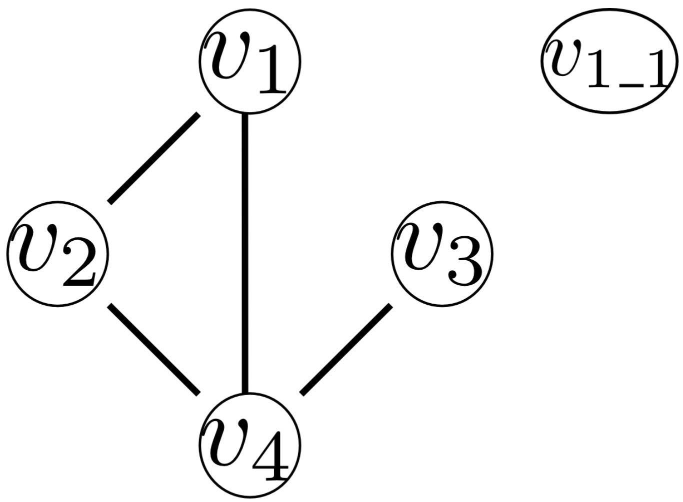
- As long as there are cycles in the graph, duplicate a node involved in the cycle (lossless)



- Duplicate a node

Step 1: Graph to Tree

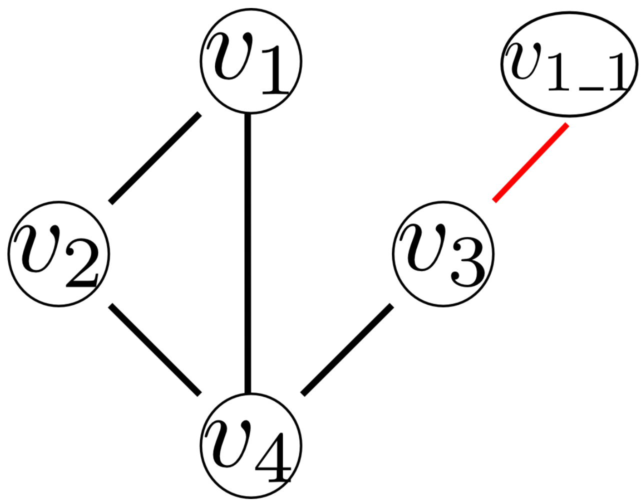
- As long as there are cycles in the graph, duplicate a node involved in the cycle (lossless)



- Delete an edge which create a cycle

Step 1: Graph to Tree

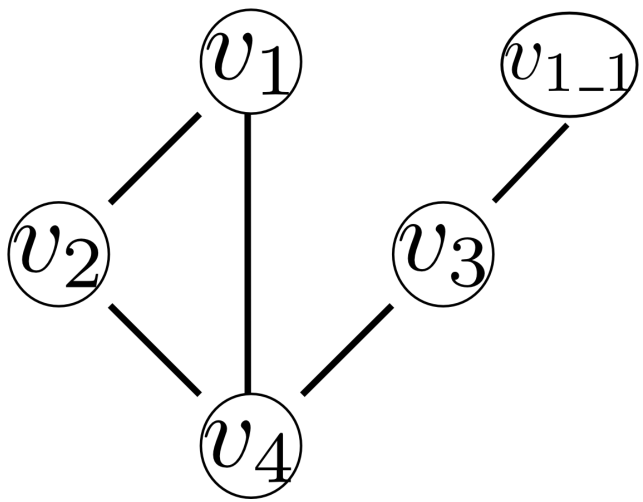
- As long as there are cycles in the graph, duplicate a node involved in the cycle (lossless)



- Add a new edge to the new node

Step 1: Graph to Tree

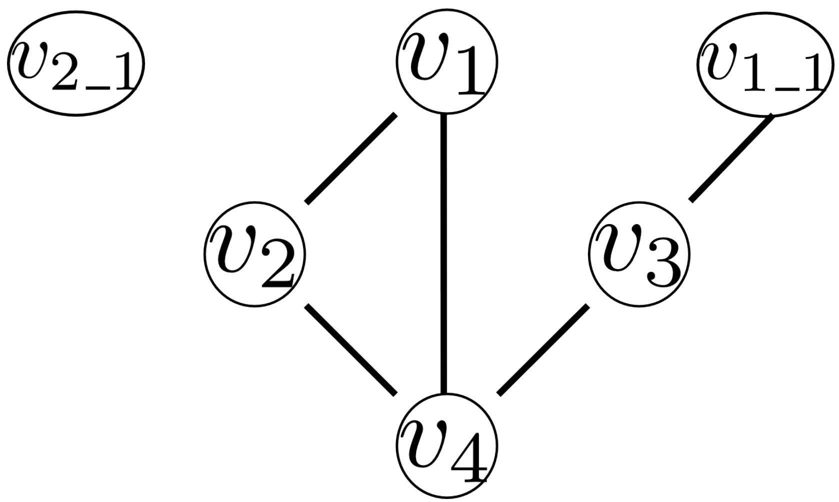
- As long as there are cycles in the graph, duplicate a node involved in the cycle (lossless)



- Repeat for all cycles

Step 1: Graph to Tree

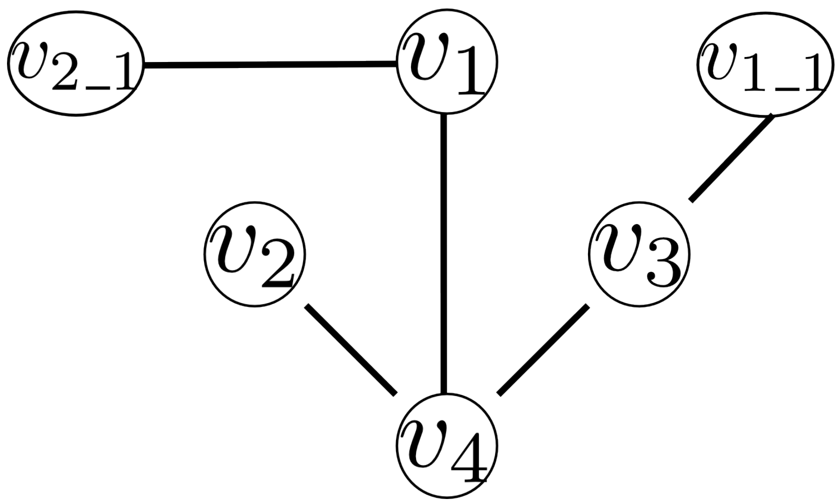
- As long as there are cycles in the graph, duplicate a node involved in the cycle (lossless)



- Repeat for all cycles

Step 1: Graph to Tree

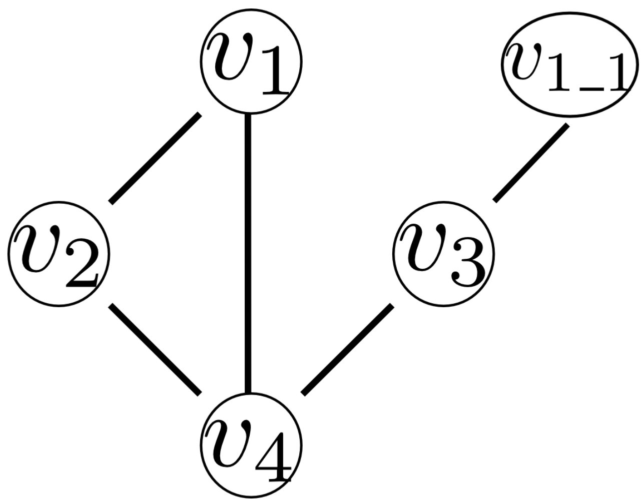
- As long as there are cycles in the graph, duplicate a node involved in the cycle (lossless)



- Repeat for all cycles

Step 1: Graph to Tree

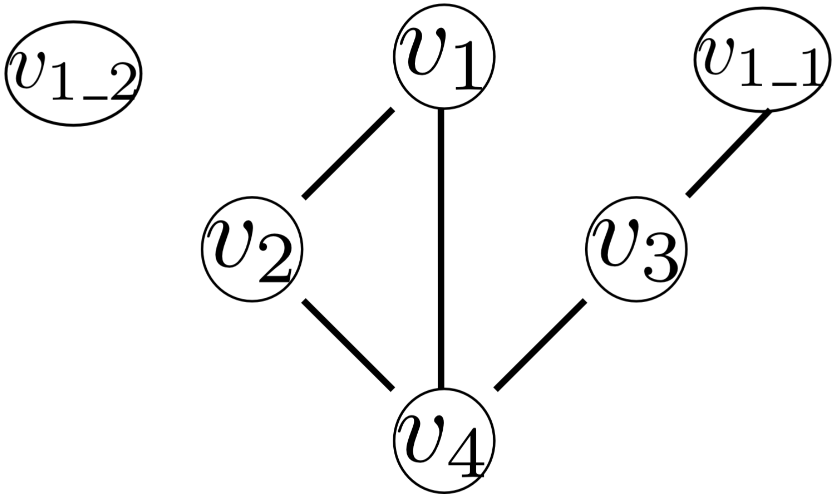
- As long as there are cycles in the graph, duplicate a node involved in the cycle (lossless)



- Can choose a different node

Step 1: Graph to Tree

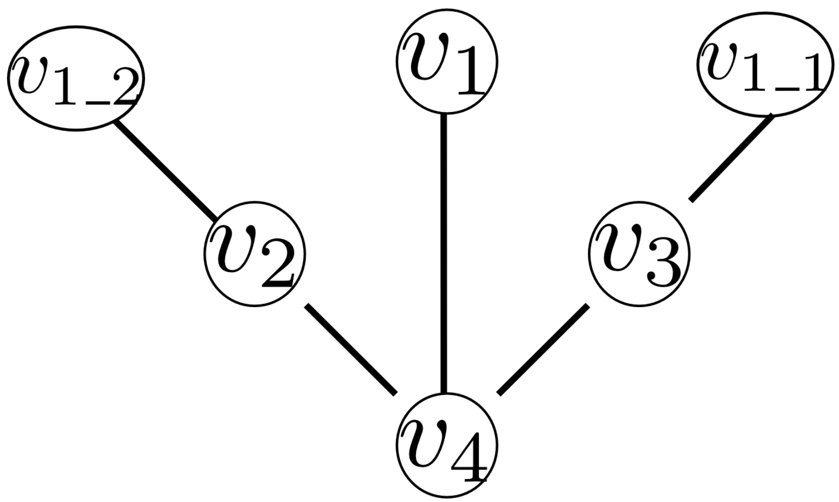
- As long as there are cycles in the graph, duplicate a node involved in the cycle (lossless)



- Can choose a different node

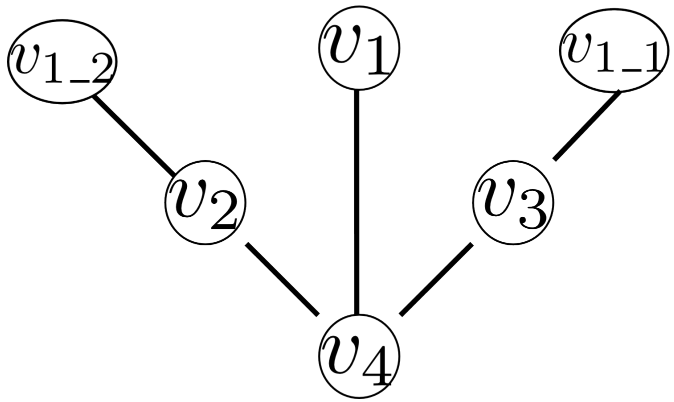
Step 1: Graph to Tree

- As long as there are cycles in the graph, duplicate a node involved in the cycle (lossless)



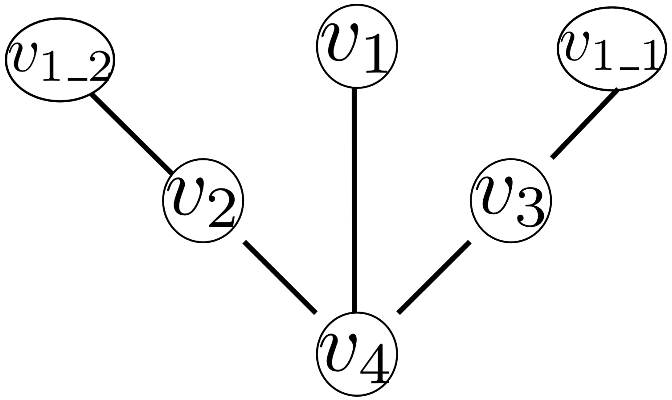
- Can choose a different node

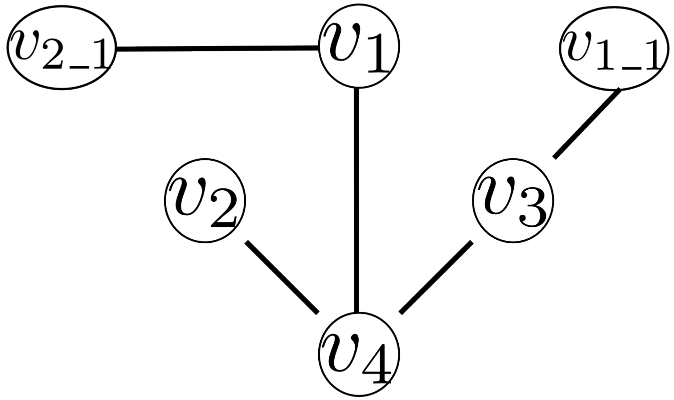
Graph to Tree



- v_1, v_{1_1} and v_{1_2} represents the same node v_1

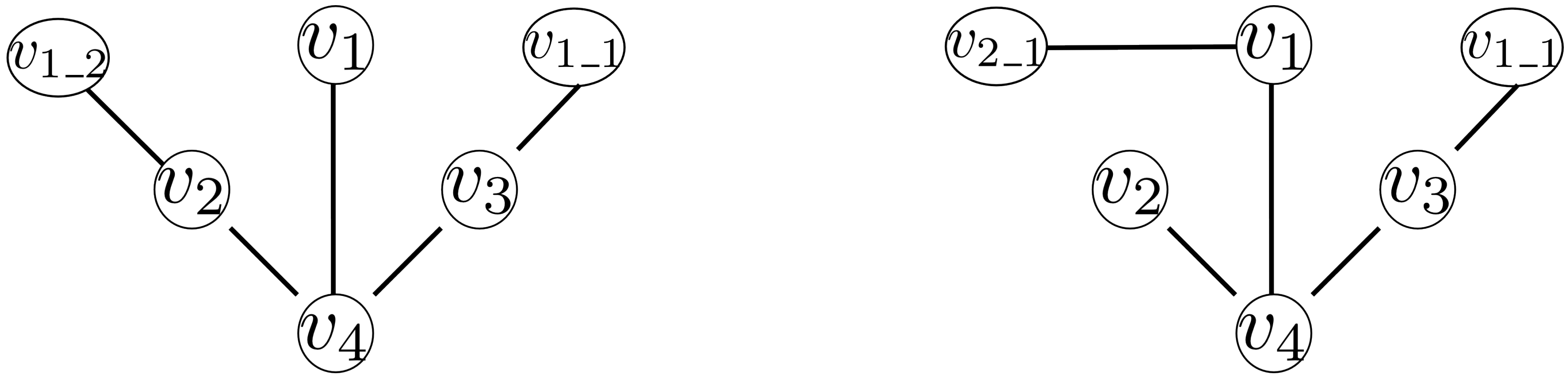
Graph to Tree





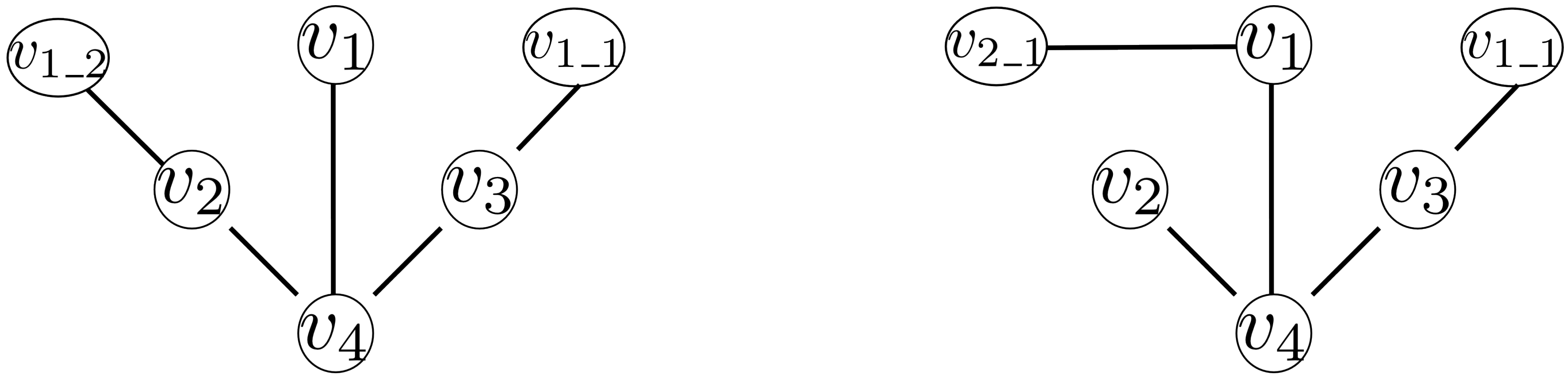
- v_1, v_{1_1} and v_{1_2} represents the same node v_1
- F is the set of nodes which have been duplicate at the end

Graph to Tree



- v_1, v_{1_1} and v_{1_2} represents the same node v_1
- F is the set of nodes which have been duplicate at the end
- On the left side, #duplicated nodes = 2 **but** $|F| = 1$ ($F = [v_1]$)

Graph to Tree



- v_1, v_{1_1} and v_{1_2} represents the same node v_1
- F is the set of nodes which have been duplicate at the end
- On the left side, #duplicated nodes = 2 **but** $|F| = 1$ ($F = [v_1]$)
- Overall complexity depends only of $|F|$
- To **minimize** this set: compute the **Vertex Feedback Set** and duplicate only these nodes

Step 2: Tree matching

- Now, the query is a tree (with duplicated nodes)
- General problem is NP-hard, exact algorithms are exponential

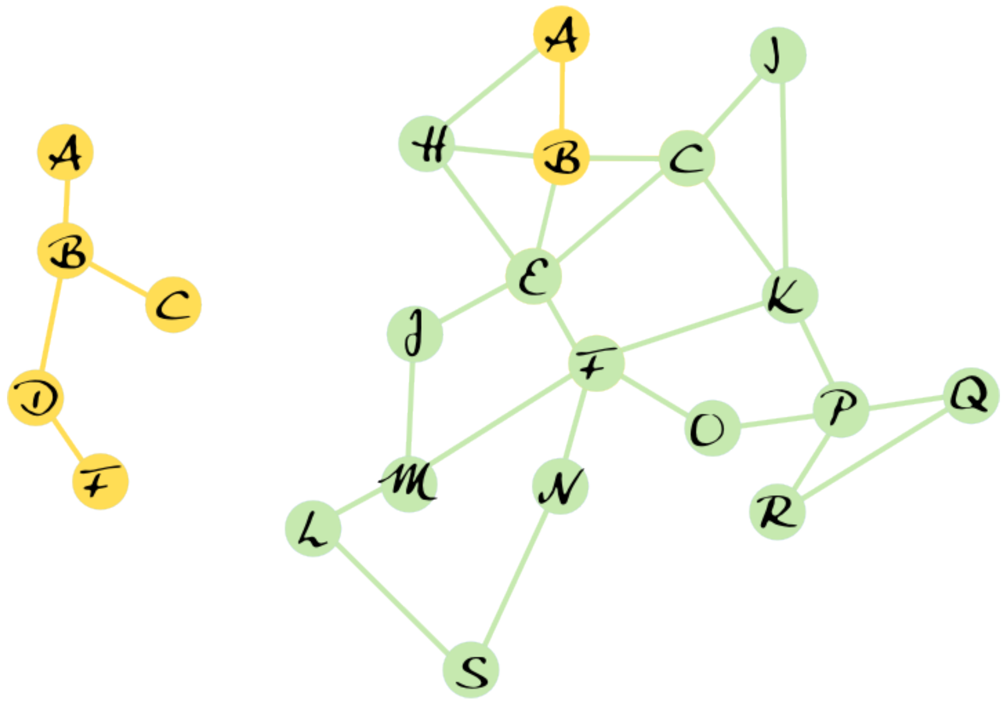
Step 2: Tree matching

- Now, the query is a tree (with duplicated nodes)
- General problem is NP-hard, exact algorithms are exponential
- Problem FPT with the color-coding technique [Alon et al. 1995]

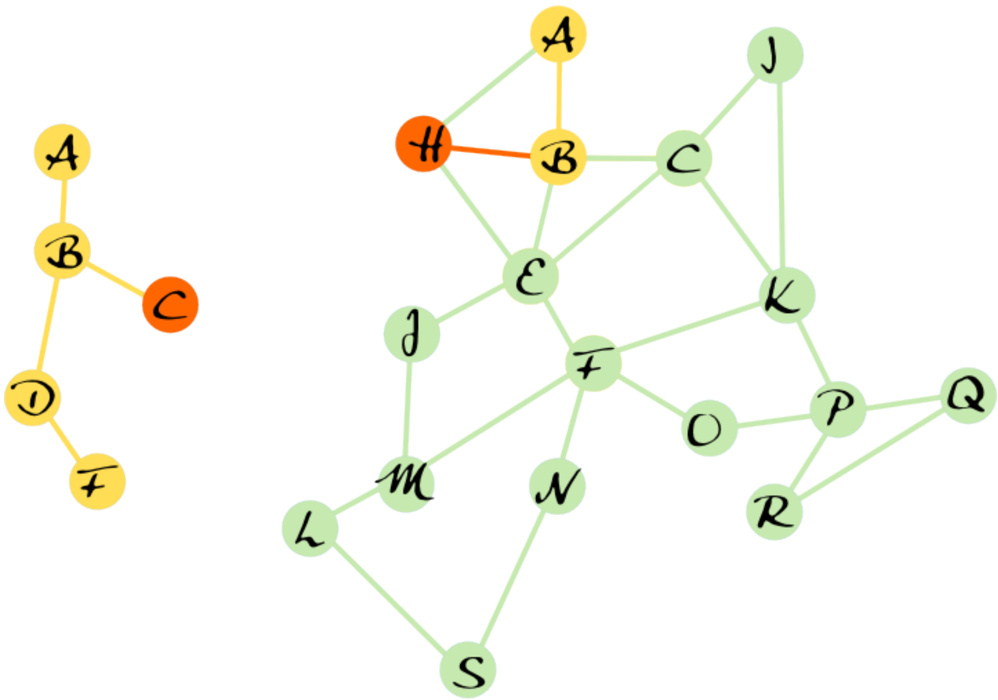
Tree matching – Color-coding [Alon et al. 1995]

- Query with k nodes, network of n nodes
- Exhaustive search : n^k potential subgraphs
- Color-coding idea :
  - Randomly color the graph with k different colors
  - Look for a colorful (one occurence for each color) result in $\mathcal{O}(2^k)$
  - Colorful probability of $\frac{k!}{k^k}$
  - Repeat until it is reasonably certain that the result appears
- Exponential part of the runtime only depends on k

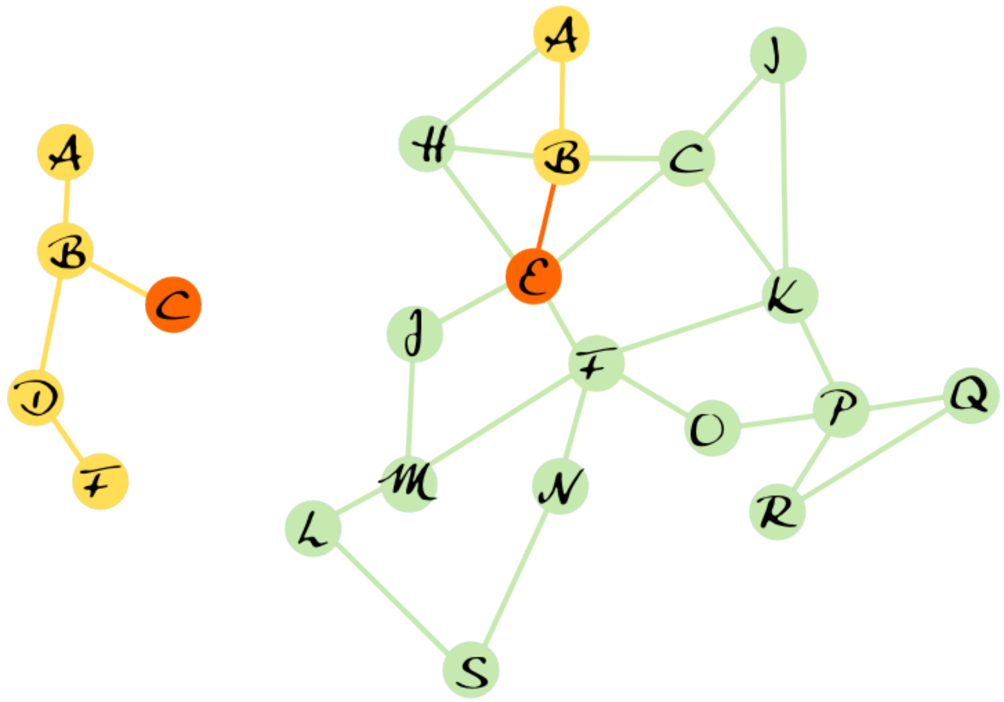
Tree Matching – Idea of the Dynamic Programming



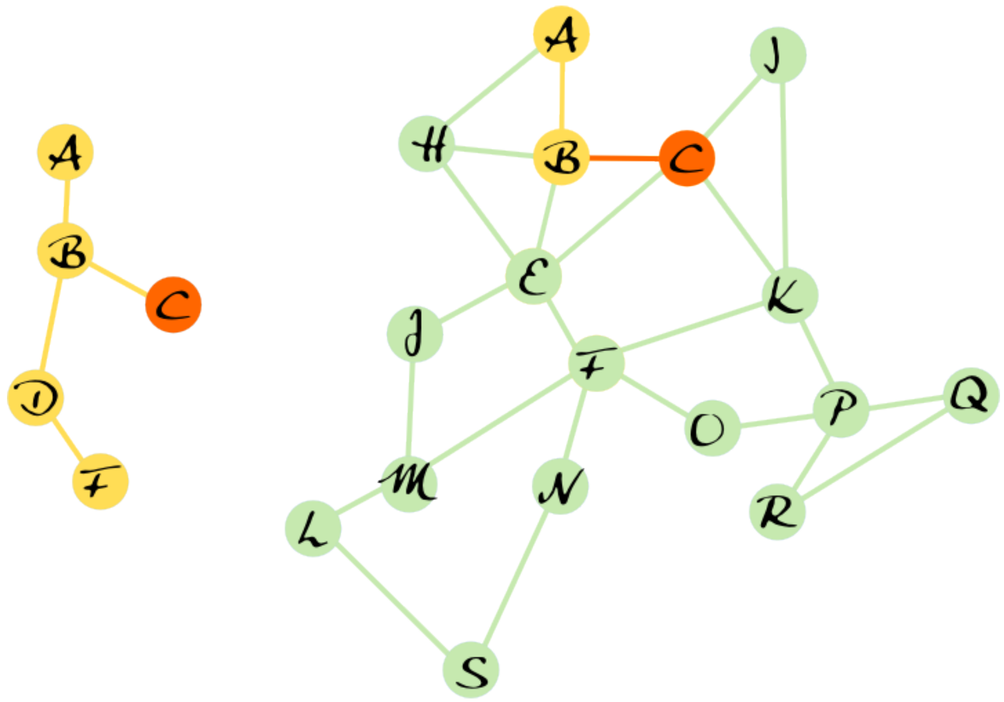
Tree Matching – Idea of the Dynamic Programming



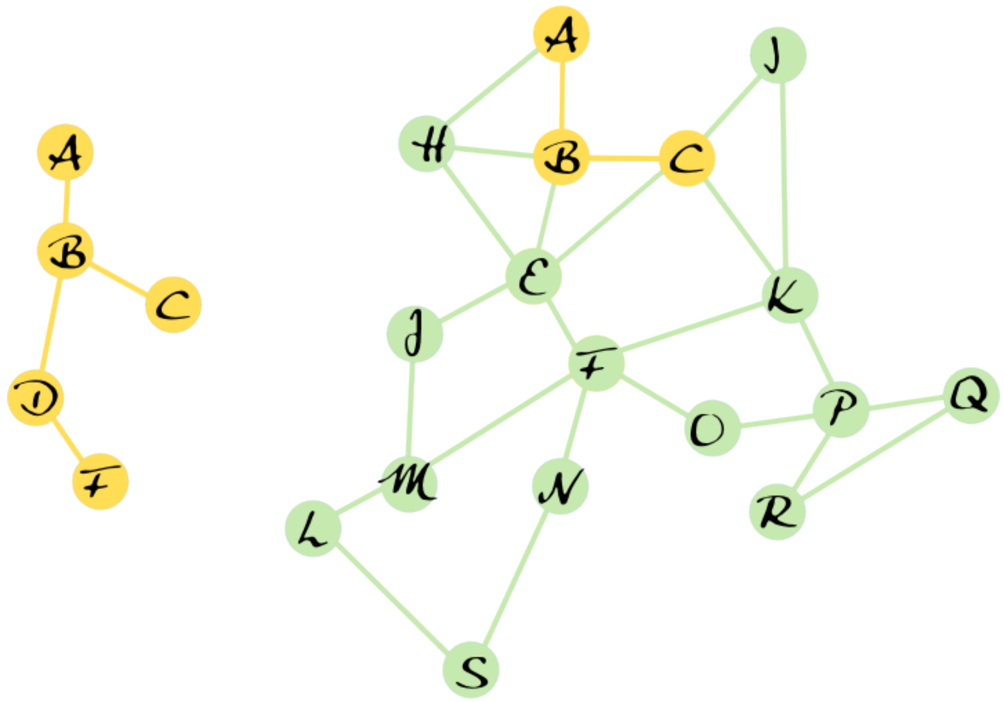
Tree Matching – Idea of the Dynamic Programming



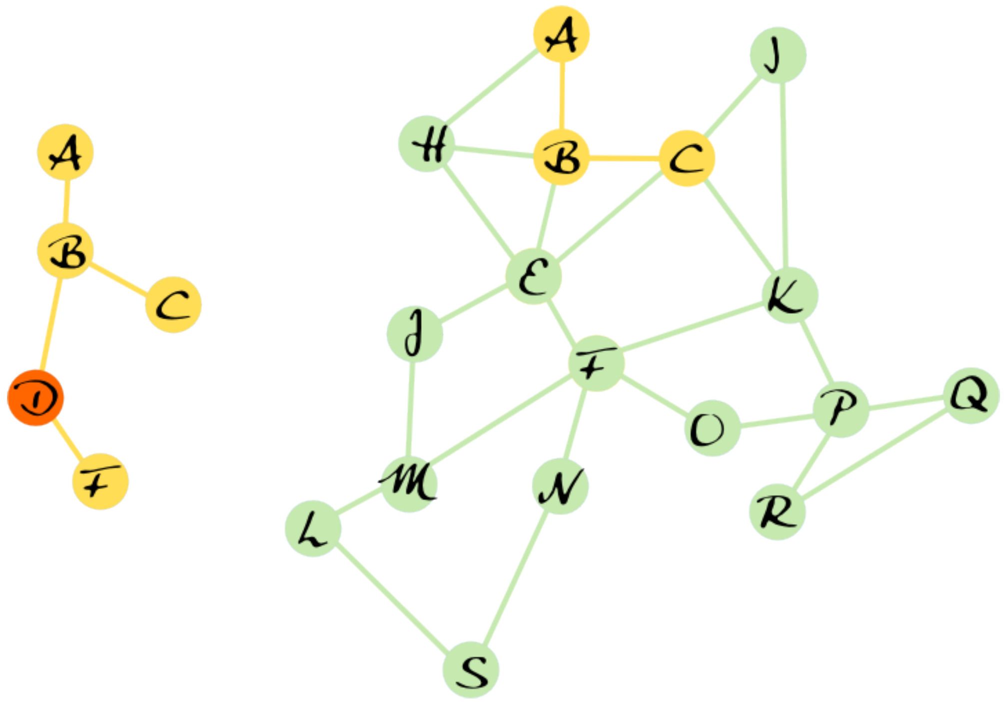
Tree Matching – Idea of the Dynamic Programming



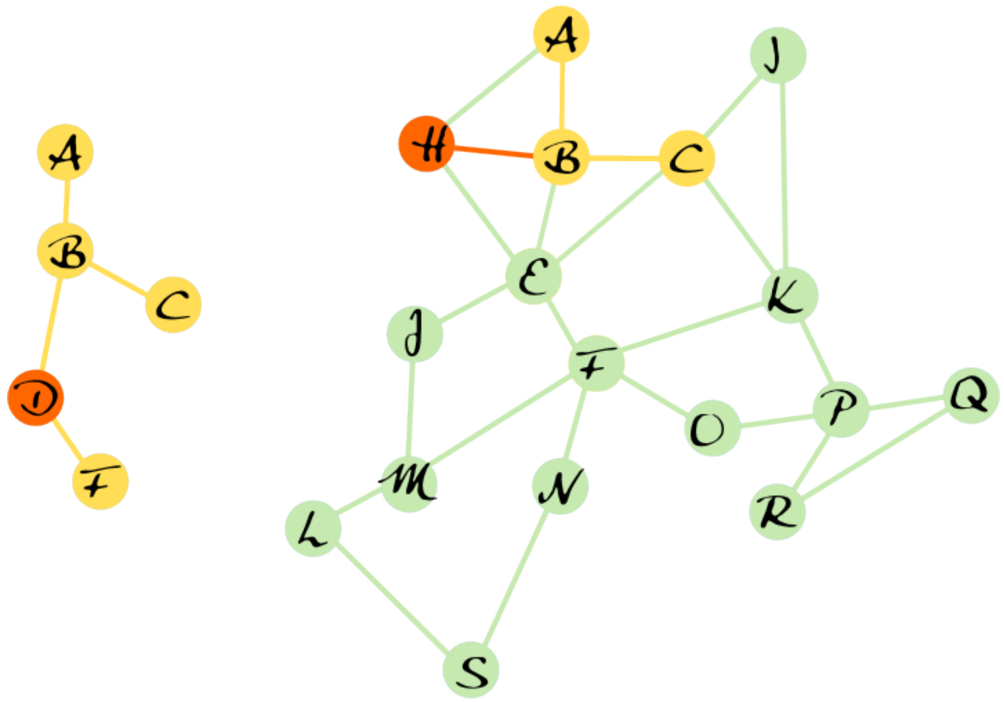
Tree Matching – Idea of the Dynamic Programming



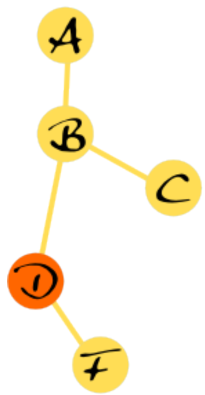
Tree Matching – Idea of the Dynamic Programming

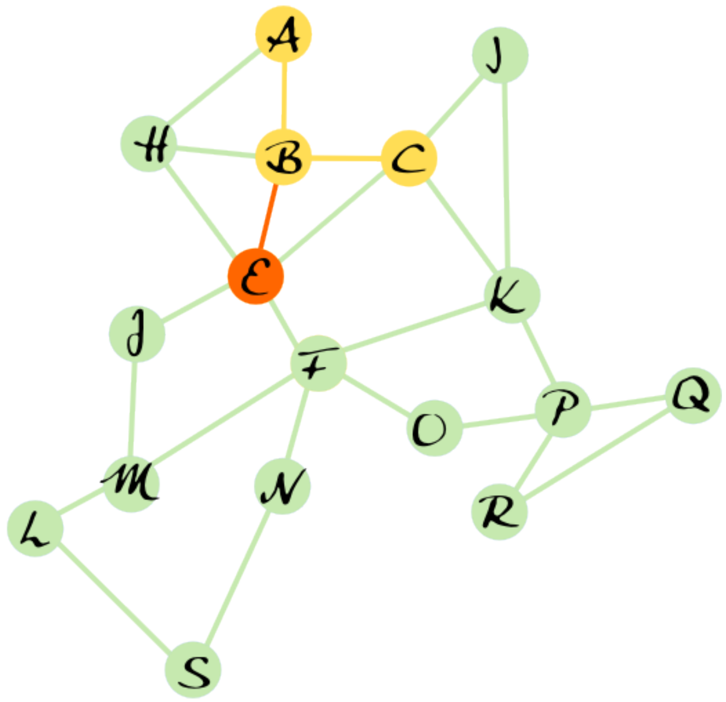


Tree Matching – Idea of the Dynamic Programming

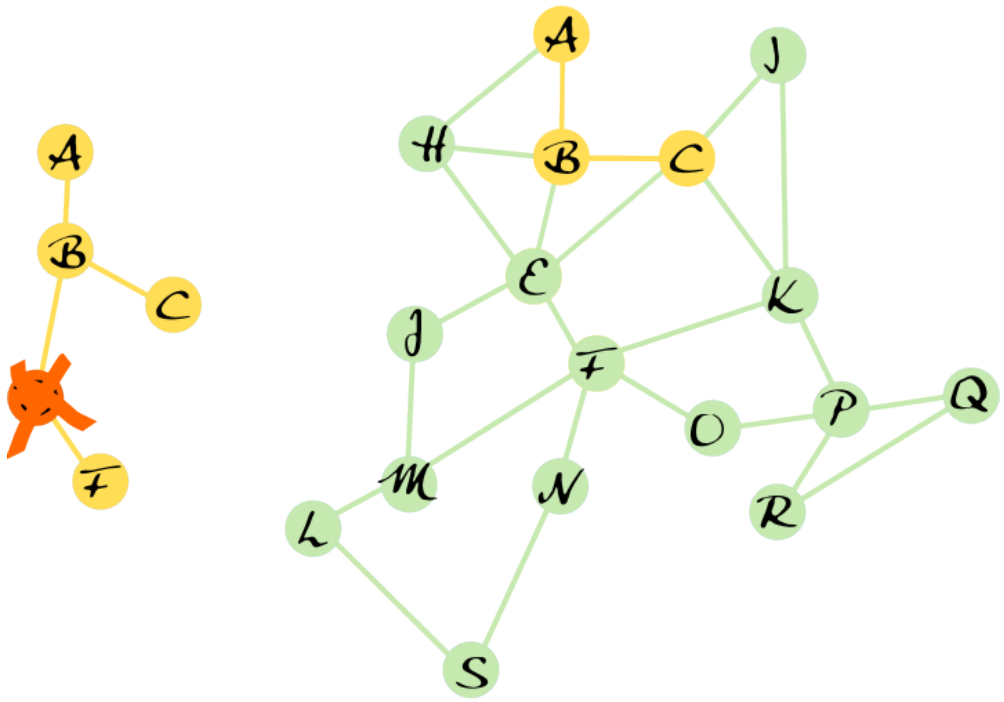


Tree Matching – Idea of the Dynamic Programming

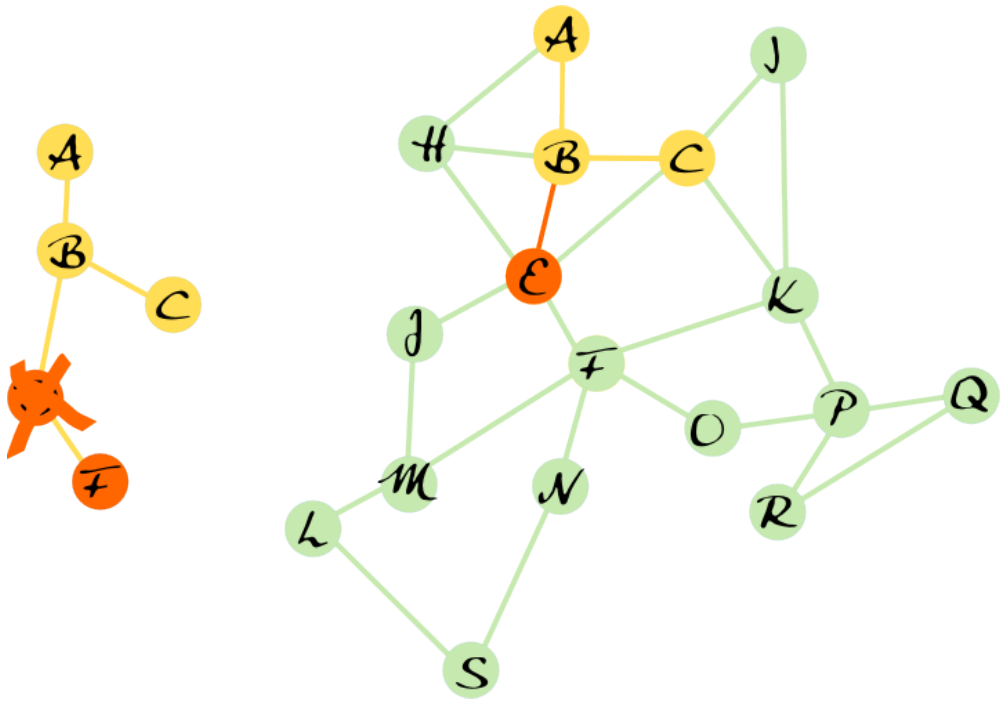




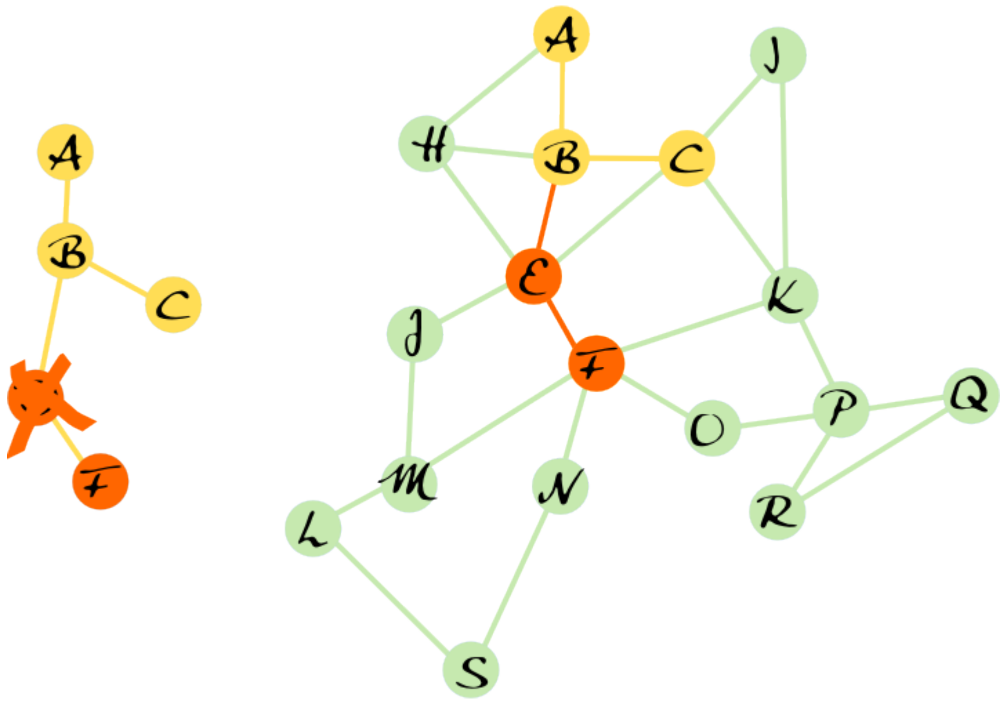
Tree Matching – Idea of the Dynamic Programming



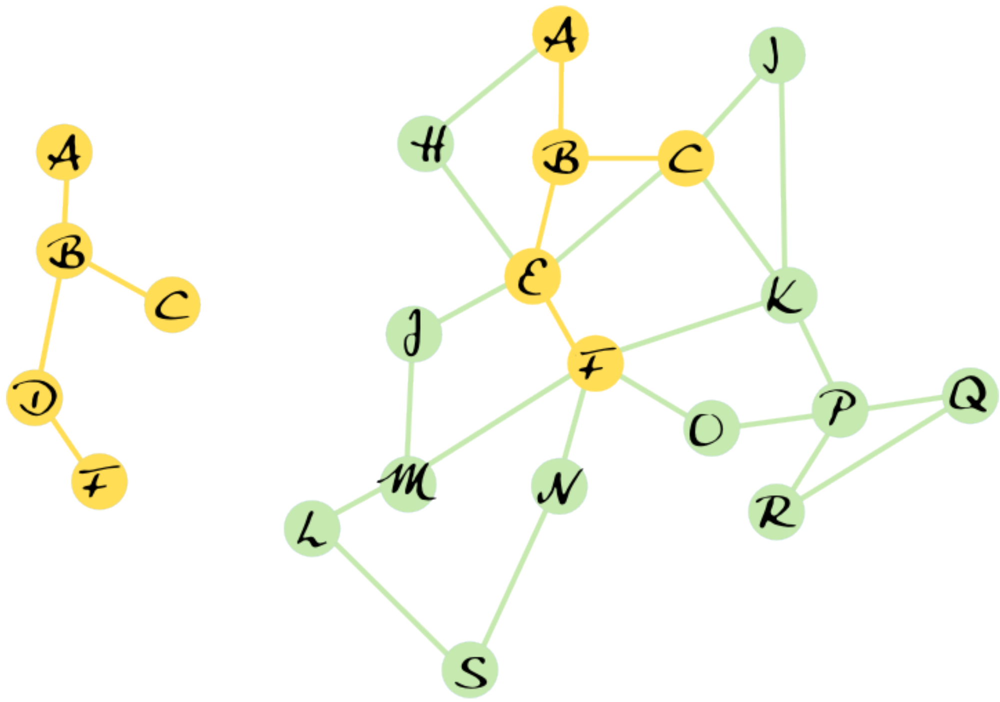
Tree Matching – Idea of the Dynamic Programming



Tree Matching – Idea of the Dynamic Programming



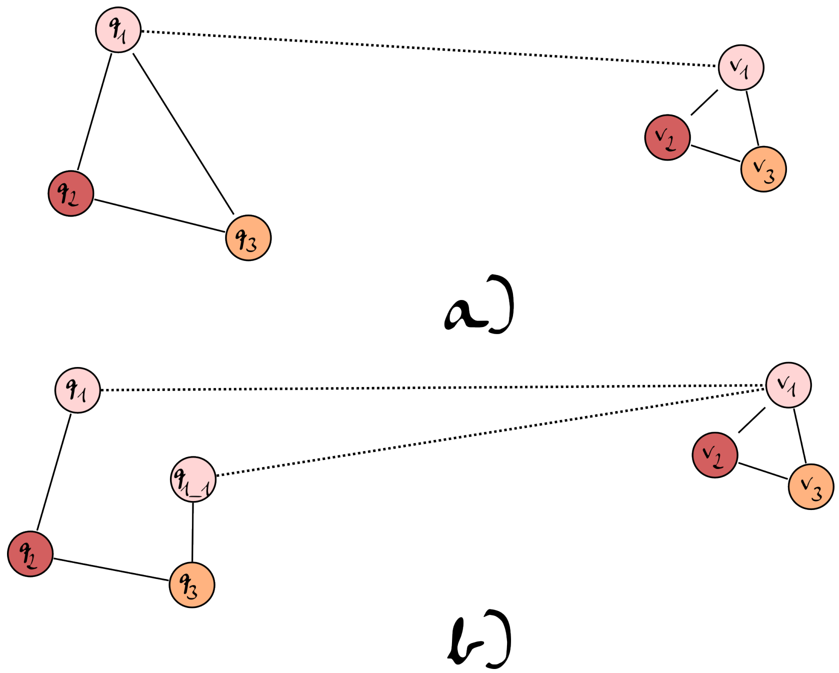
Tree Matching – Idea of the Dynamic Programming



- Initialization has to be done with each homologous node of the network

Tree Matching – Dealing with duplicated nodes

- **Constraint**: All nodes of a same family (i.e. they are duplicated from the same node) must perform the same action (i.e., match or deletion), since they represent the same protein

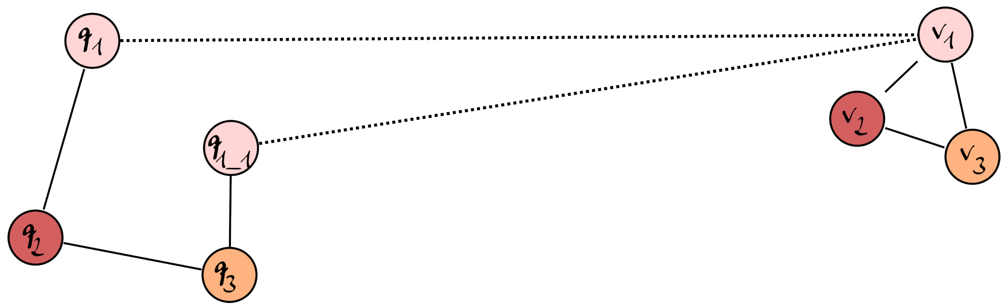


Tree Matching – Dealing with duplicated nodes

- **Solution**: Pre-compute the action for each families before running the dynamic programming algorithm
- We launch the dynamic programming algorithm with each possibilities (a family with each nodes of the network)

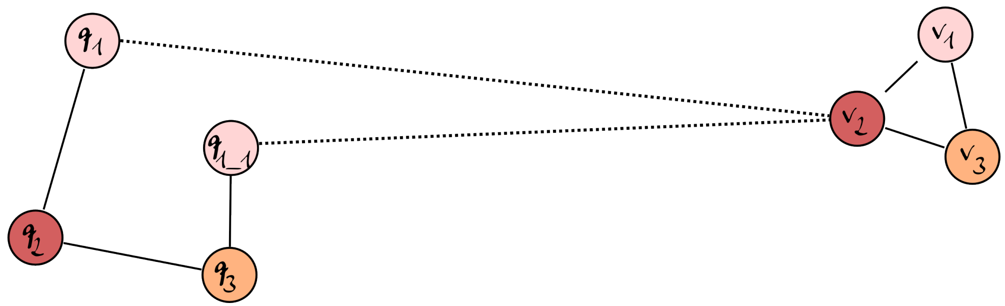
Tree Matching – Dealing with duplicated nodes

- **Solution**: Pre-compute the action for each families before running the dynamic programming algorithm
- We launch the dynamic programming algorithm with each possibilities (a family with each nodes of the network)



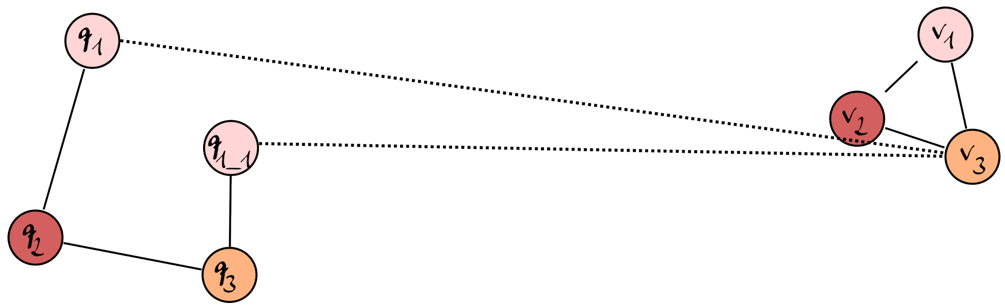
Tree Matching – Dealing with duplicated nodes

- **Solution**: Pre-compute the action for each families before running the dynamic programming algorithm
- We launch the dynamic programming algorithm with each possibilities (a family with each nodes of the network)



Tree Matching – Dealing with duplicated nodes

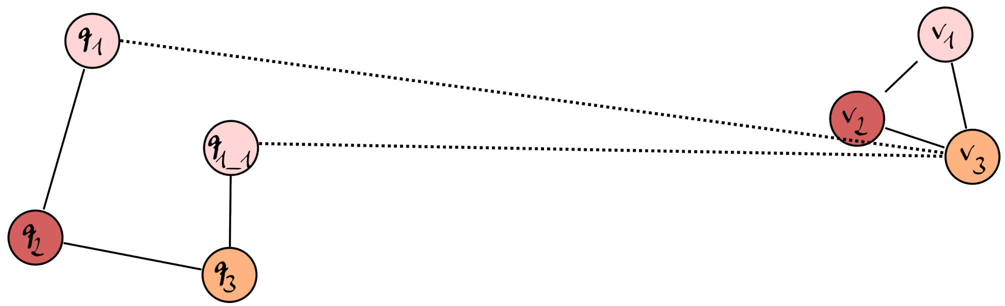
- **Solution**: Pre-compute the action for each families before running the dynamic programming algorithm
- We launch the dynamic programming algorithm with each possibilities (a family with each nodes of the network)



- $n^{|F|}$ possibilities

Tree Matching – Dealing with duplicated nodes

- **Solution**: Pre-compute the action for each families before running the dynamic programming algorithm
- We launch the dynamic programming algorithm with each possibilities (a family with each nodes of the network)



- $n^{|F|}$ possibilities
- The dynamic programming algorithm must respect the pre-computation

Tree Matching – Complexity

- There are $n^{|F|}$ pre-computed assignments
- For each one, we launch the $2^{\mathcal{O}(k)}.m$ time complexity dynamic programming
- On the whole, $\mathcal{O}(n^{|F|}2^{\mathcal{O}(k)}.m)$

Comparison with QNet

PADA1

- $\mathcal{O}(n^{|\mathbf{F}|}2^{\mathcal{O}(k)}.m)$
- Exponential in $|F|$, $|F|$ is the size of the Vertex Feedback Set

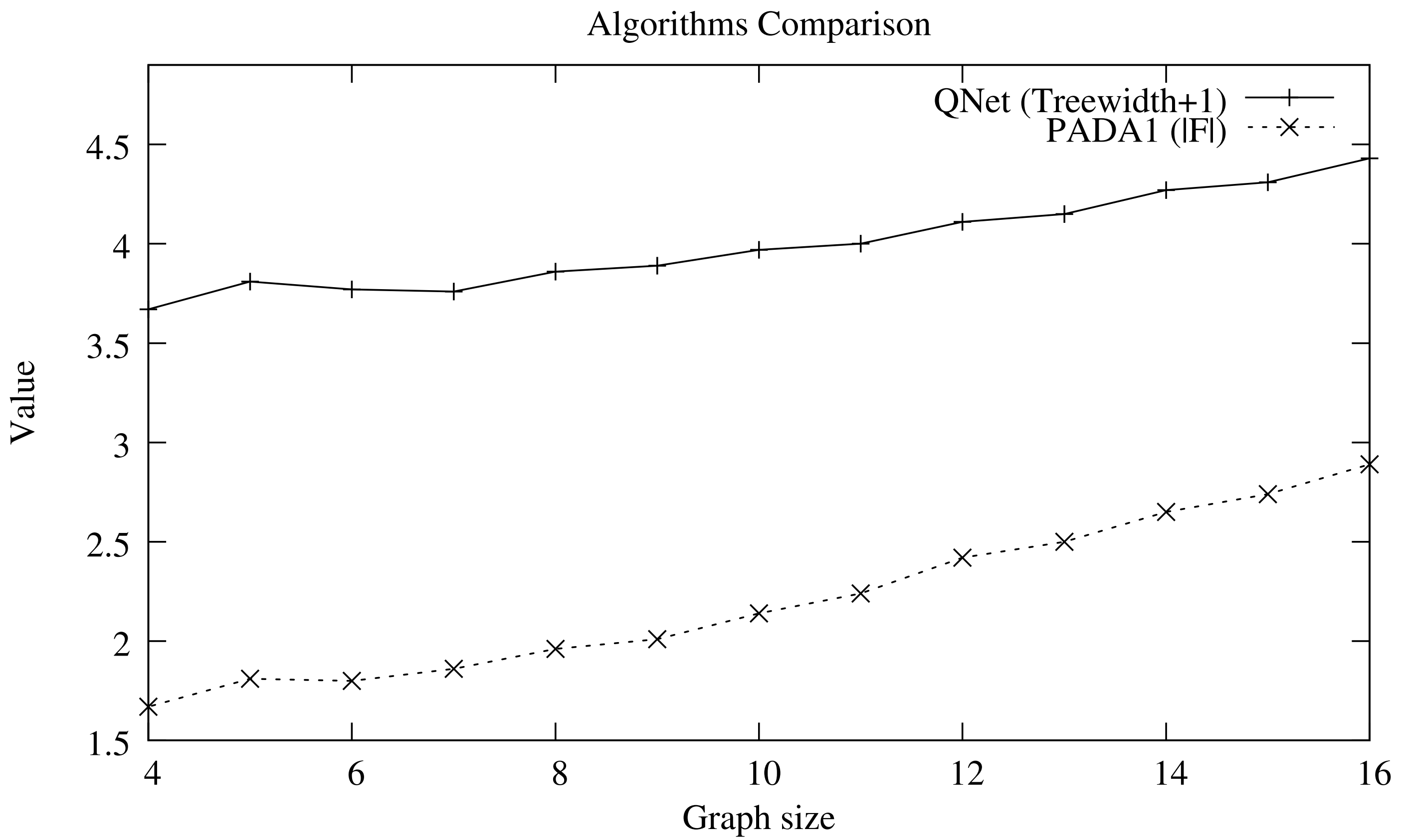
QNet

- $\mathcal{O}(n^{\mathbf{t+1}}2^{\mathcal{O}(k)})$
- Exponential in $t+1$, t is the treewidth

- No theoretical results between the **treewidth** value and **Vertex Feedback Set** size

Comparison with QNet

- Experimental comparison on random graphs between treewidth and Vertex Feedback Set size



Comparison with QNet

- Our bound of $n^{|F|}$ is over-estimated
- Launch the dynamic programming algorithm only if the nodes of F are homologous to the network node in the pre-computed assignation
- For example, if
  - a protein is homologous with in average 10 proteins
  - $|F| = 3$
  - then, $n^3 \ll 10^3$ (very blast dependant)

Outline

Motivations and state of the art

Our Algorithm, PADA1

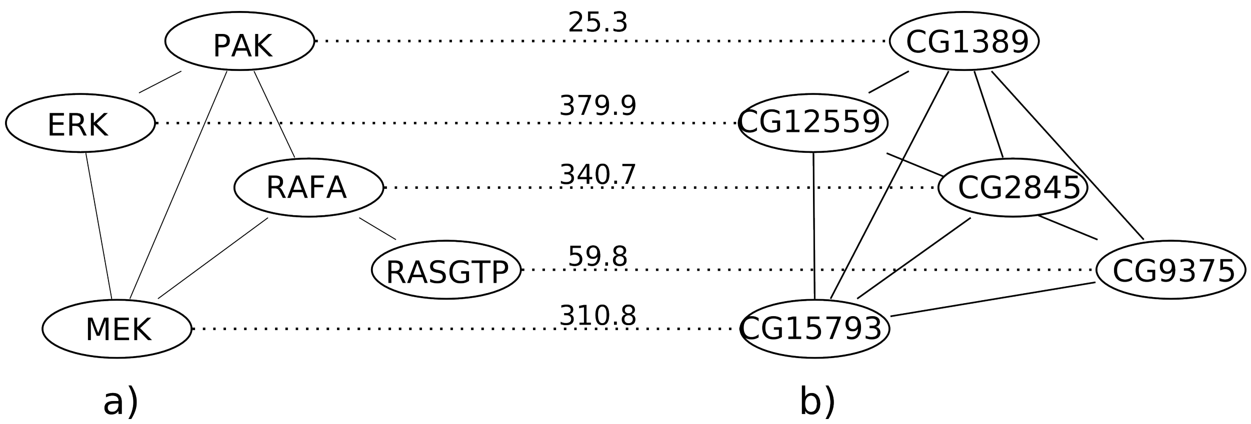
Experimental Results

Experimental tests

- Perform a Python implementation
- Cannot compare experimentaly (no implementation of QNet with graphs)
- Tests on real data : retrieve known yeast patterns

Experimental tests

- Cross specie experimentation
- Search human MAPK pattern in fly network



Conclusion

- Algorithm to retrieve graphs pattern in a PPI network using the Vertex Feedback Set, node duplication and color-coding technique
- Find an average time complexity ?
- Abstraction of the pattern's topology (GRAPHMOTIF problem [LACROIX ET AL. 2006], [FELLOWS ET AL. 2007], [BETZLER ET AL. 2008], [DONDI ET AL. 2007, 2009], [BRUCKNER ET AL. 2009])

Questions on Querying Protein-Protein Interaction Networks ?

Guillaume Blin Florian Sikora[1] Stéphane Vialette[1]

Université Paris-Est, LIGM - UMR CNRS 8049 - France
{gblin,sikora,vialette}@univ-mlv.fr

ISBRA May 2009