université de TOURS

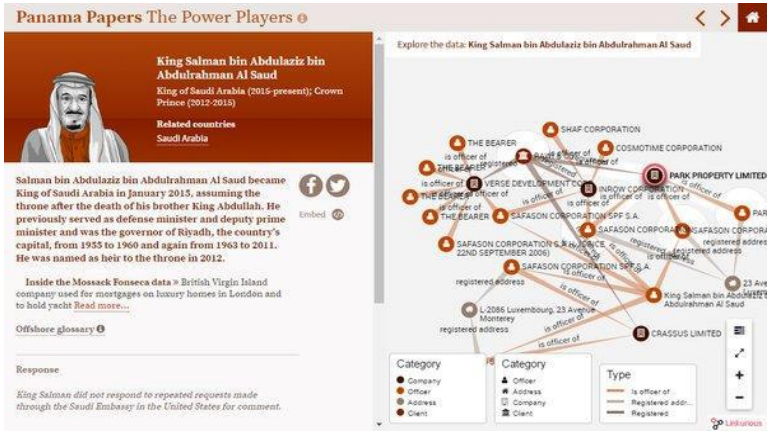
LIFAT EA 6300

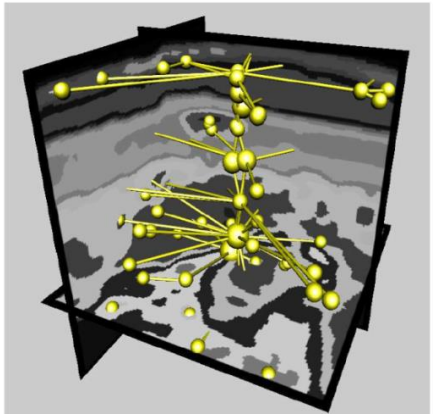
Laboratoire d'Informatique Fondamentale et Appliquée de Tours

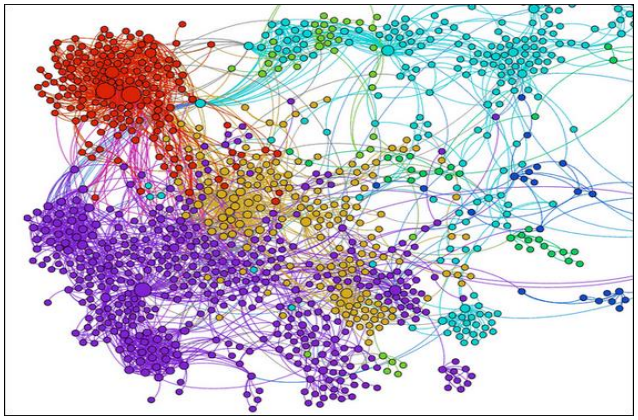
Advanced Data Mining

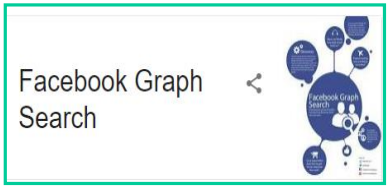
Part "Graphs for PR and Mining"

J.Y. RAMEL - 2021









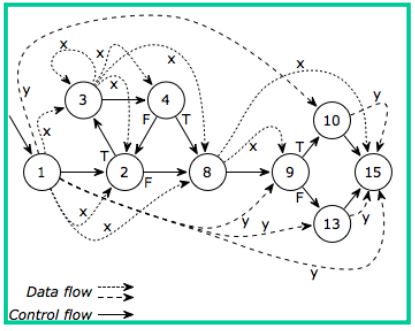
Teaching prepared in collaboration with **Romain Raveaux** (LIFAT – RFAI)

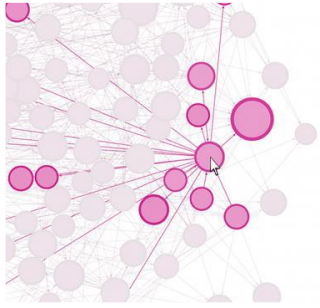
Questions?

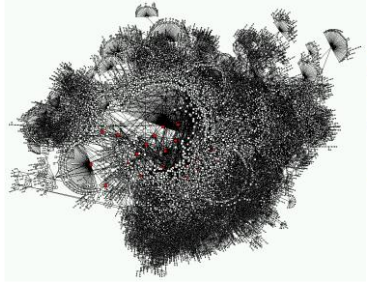
- Have you ever heard about Graphs?

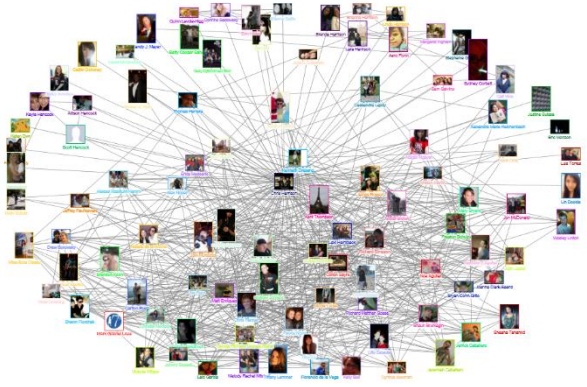
Graphs for PR and Mining

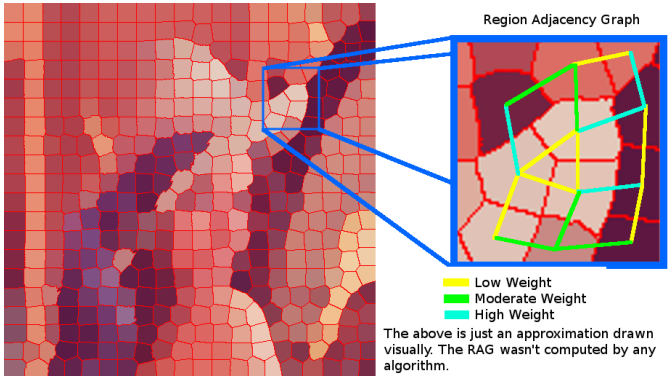
- **Outline**
  - Definitions, representation
    - Recall about graphs
    - Storing Graphs in memory
    - Types of problems (in PR, ML and DM)
  - Graph Analysis / Mining
    - Graph Characterization
    - Graph partitioning
    - Pattern detection
    - Graph indexing
  - Graphs for PR (in CV and ML)
    - Graph Matching
    - Graph comparison
    - Graph and deep learning







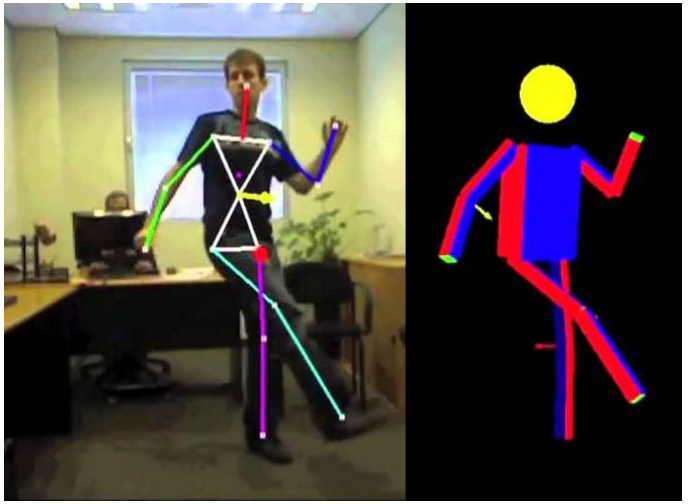


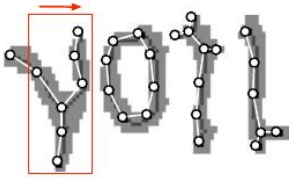


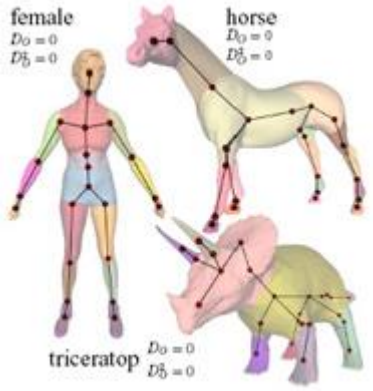
Why using graphs?

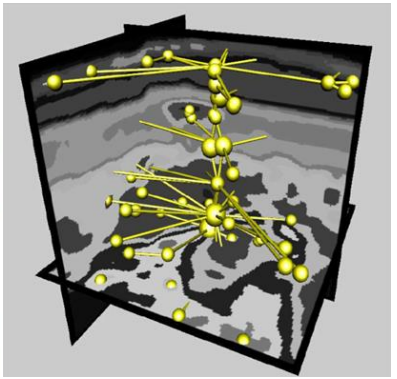
Structural Methods

- **Graphs by nature**
  - Relational data or structured data are designed as graphs
  - Non vectorial methods can guarantee to preserve the topological information
- **Combining sources of data**
  - An image and a knowledge graph for instance
- **Extended Euclidean data**
  - Pairwise features can be used to enrich an Euclidean representation.
  - i.e , a pixel can be connected to every pixel in the image and each relation can be enriched by a set of features







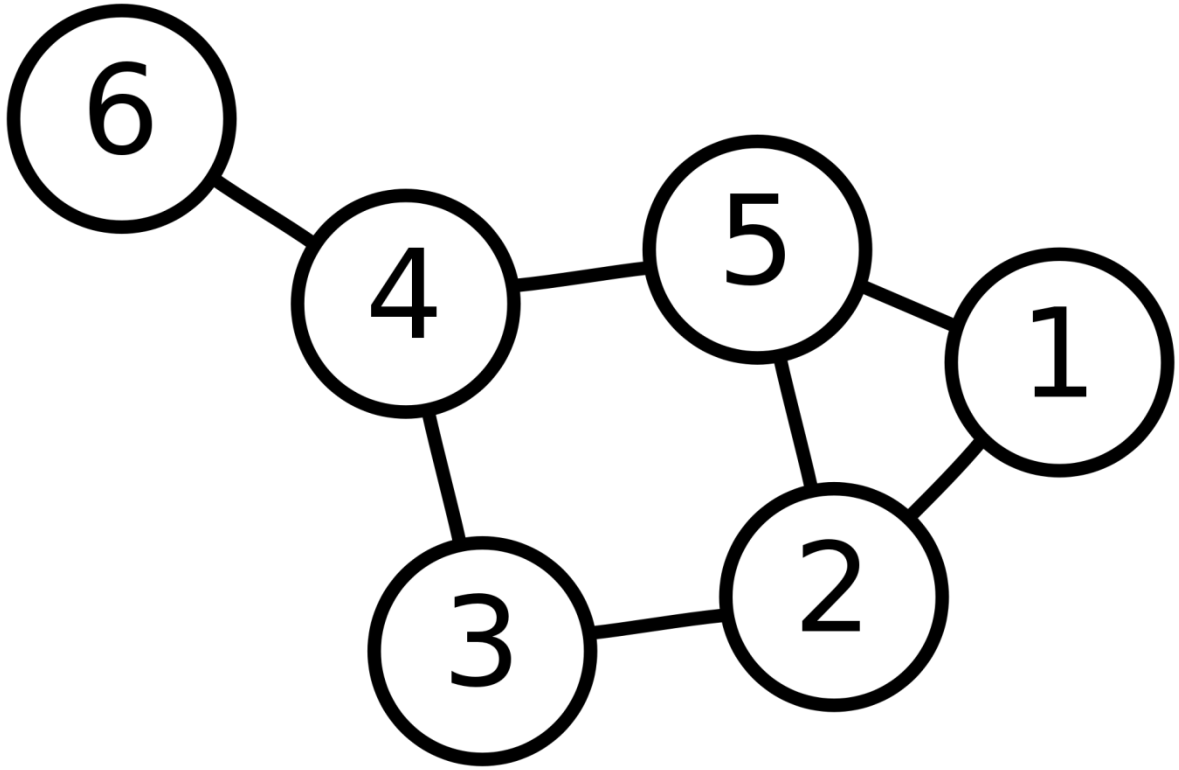


1. GRAPHS: Definitions & representations

Questions?

- What is a Graph?

Definition of « Graph » ?



Definitions

Initial definition

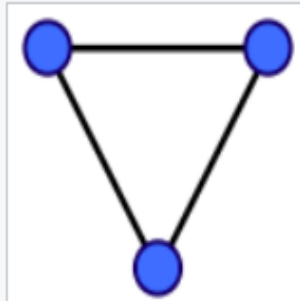
- A graph is a set of vertices linked by edges.
- Formally speaking:

A **graph** is an ordered pair $G = (V, E)$ where,

- V is the *vertex set* whose elements are the vertices, or *nodes* of the graph. This set is often denoted $V(G)$ or just V.
- E is the *edge set* whose elements are the edges, or connections between vertices, of the graph. This set is often denoted $E(G)$ or just E. If the graph is undirected, individual edges are unordered pairs $\{u, v\}$ where u and v are vertices in V. If the graph is directed, edges are ordered pairs (u, v).

Two graphs G and H are considered equal when $V(G) = V(H)$ and $E(G) = E(H)$.

The **order** of a graph is the number of vertices in it, usually denoted $|V|$ or $|G|$ or sometimes n. The **size** of a graph is the number of edges in it, denoted $|E|$ or $\|G\|$, or sometimes m. If $n = 0$ and $m = 0$, the graph is called *empty* or *null*. If $n = 1$ and $m = 0$, the graph is considered *trivial*. If $1 \leq n$ and $m = 0$, the graph is called *discrete*.



A simple undirected graph with three vertices and three edges. Each vertex has degree two, so this is also a regular graph.

Adding labels and attributes

- **Updating the initial definition...**

Definition

Let L_V and L_E denote the set of node and edge labels, respectively. A labeled graph G is a 4-tuple $G = (V, E, \mu, \xi)$, where

- V is the set of nodes,
- $E \subseteq V \times V$ is the set of edges
- $\mu : V \rightarrow L_V$ is a function assigning labels to the nodes, and
- $\xi : E \rightarrow L_E$ is a function assigning labels to the edges.

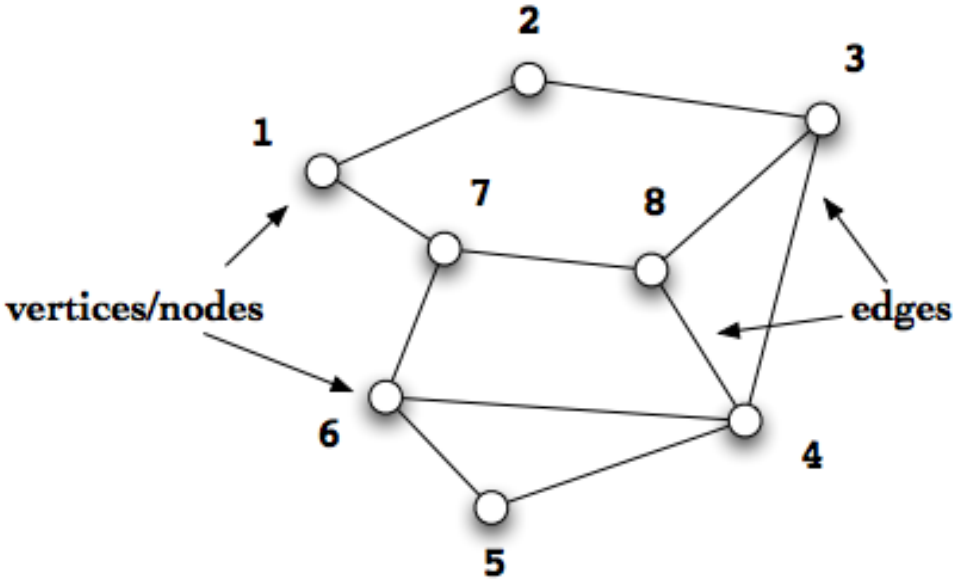
- Let $G_1 = (V_1, E_1, \mu_1, \xi_1)$ be the source graph
- And $G_2 = (V_2, E_2, \mu_2, \xi_2)$ the target graph
- With $V_1 = (u_1, ..., u_n)$ and $V_2 = (v_1, ..., v_m)$ respectively

Questions?

- Many complex or/and large graphs
- How to represent such graphs in computer?

Which data type?

Adjacency matrix



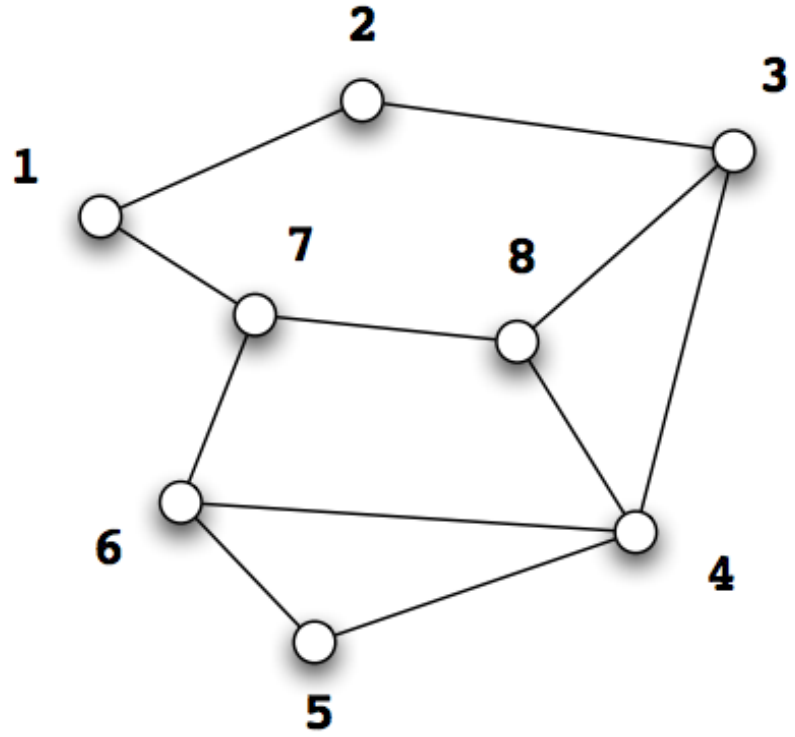
Undirected Graph G(V, E)

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

sub-matrix of A = a subgraph of G

Which data type?

Degree matrix

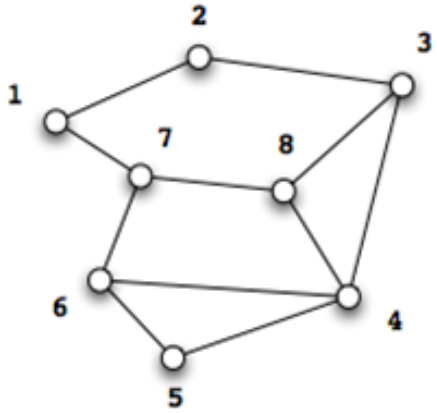


$$D = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

Normalized Adjacency matrix $\tilde{A} = D^{-1}A$ is a stochastic matrix (each row sums to one)

Which data type?

Laplacian Matrix



$$L = \begin{bmatrix} 2 & -1 & 0 & 0 & 0 & 0 & -1 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 3 & -1 & 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 4 & -1 & -1 & 0 & -1 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & 3 & -1 & 0 \\ -1 & 0 & 0 & 0 & 0 & -1 & 3 & -1 \\ 0 & 0 & -1 & -1 & 0 & 0 & -1 & 3 \end{bmatrix}$$

$$L = D - A$$

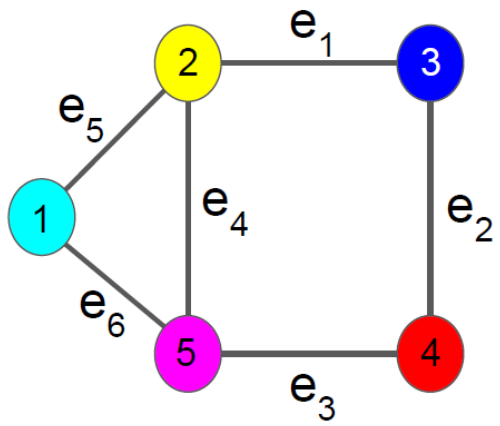
Normalized version

$$\tilde{L} = D^{-\frac{1}{2}}(D - A)D^{-\frac{1}{2}}$$

Which data type?

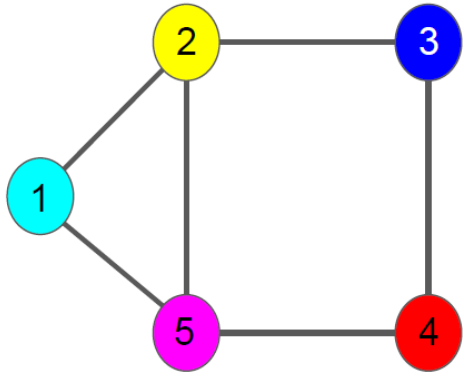
How graphs are represented in computer memory?

- Incidence matrix



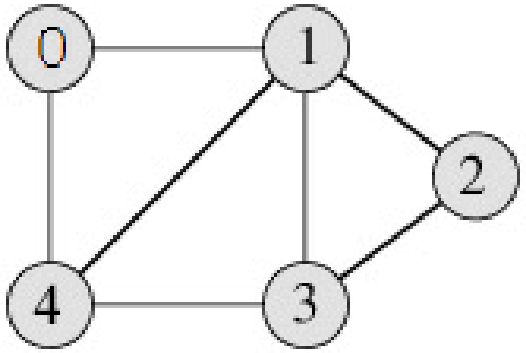
$$I = \begin{array}{c} \begin{matrix} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

- Adjacency list



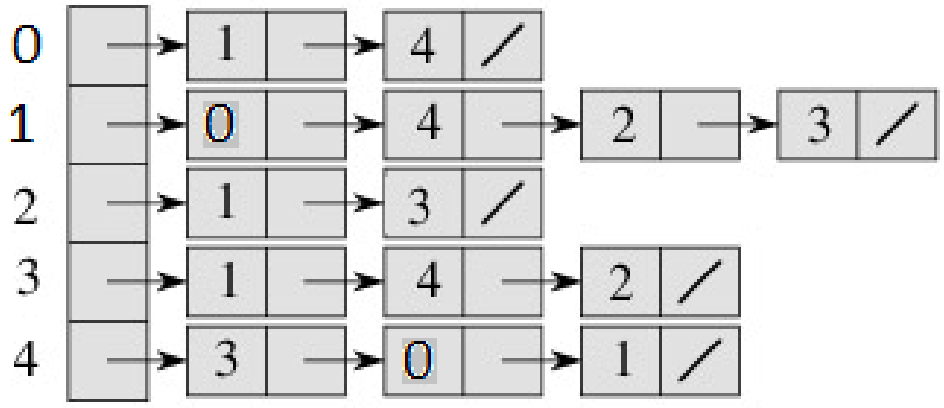
$$A = \begin{array}{l} \mathbf{1}\ 2\ 5 \\ \mathbf{2}\ 1\ 3\ 5 \\ \mathbf{3}\ 2\ 4 \\ \mathbf{4}\ 3\ 5 \\ \mathbf{5}\ 1\ 2\ 4 \end{array}$$

Questions?



- Matrix representation?
- List representations?

	0	1	2	3	4
0	0	1	0	0	1
1	1	0	1	1	1
2	0	1	0	1	0
3	0	1	1	0	1
4	1	1	0	1	0



Adjacency matrix

- ✓ Easy to implement
- ✓ Removing an edge takes O(1) time
- ✓ Check (u->v) edge existence in O(1) time

- x Memory consumption

n	$\mid U_{ij} \mid$	RAM
10,000	399,960,000	$\approx$ 400Mo
50,000	9,999,800,000	$\approx$ 10Go
100,000	39,999,600,000	$\approx$ 40Go
500,000	999,998,000,000	$\approx$ 1To

Adjacency list

- ✓ Take less space: O(|V|+|E|)
- ✓ Adding a vertex is easy

- x Check (u->v) edge existence in more complex !
  In O(V) at work case scenario

‘Text’ example : GML

```
graph [
        comment "This is a sample graph"
        directed 1
        id n42
        label "Hello, I am a graph"
        node [
                id 1
                label "node 1"
                thisIsASampleAttribute 42
        ]
        node [
                id 2
                label "node 2"
                thisIsASampleAttribute 43
        ]
        node [
                id 3
                label "node 3"
                thisIsASampleAttribute 44
        ]
        edge [
                source 1
                target 2
                label "Edge from node 1 to node 2"
        ]
        edge [
                source 2
                target 3
                label "Edge from node 2 to node 3"
        ]
        edge [
                source 3
                target 1
                label "Edge from node 3 to node 1"
        ]
]
```

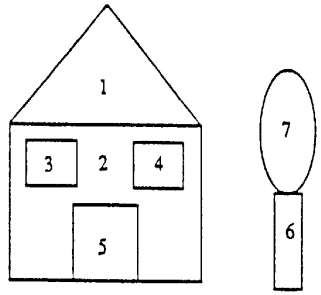
```
<?xml version="1.0"?>
<gxl>
    <graph>
        <node id="0">
            <attr name="x"><Integer>472</Integer></attr>
            <attr name="y"><Integer>19</Integer></attr>
            <attr name="type"><String>corner</String></attr>
        </node>
        <node id="1">
            <attr name="x"><Integer>41</Integer></attr>
            <attr name="y"><Integer>447</Integer></attr>
            <attr name="type"><String>corner</String></attr>
        </node>
        <node id="2">
            <attr name="x"><Integer>44</Integer></attr>
            <attr name="y"><Integer>494</Integer></attr>
            <attr name="type"><String>corner</String></attr>
        </node>
        <node id="3">
            <attr name="x"><Integer>475</Integer></attr>
            <attr name="y"><Integer>437</Integer></attr>
            <attr name="type"><String>intersection</String></attr>
        </node>
        <edge from="0" to="1">
            <attr name="frequency"><Integer>1</Integer></attr>
            <attr name="type0"><String>line</String></attr>
            <attr name="angle0"><String>-.00</String></attr>
        </edge>
        <edge from="0" to="2">
            <attr name="frequency"><Integer>1</Integer></attr>
            <attr name="type0"><String>line</String></attr>
            <attr name="angle0"><String>1.56</String></attr>
        </edge>
        <edge from="1" to="3">
            <attr name="frequency"><Integer>1</Integer></attr>
            <attr name="type0"><String>line</String></attr>
            <attr name="angle0"><String>-.79</String></attr>
        </edge>
    </graph>
</gxl>
```

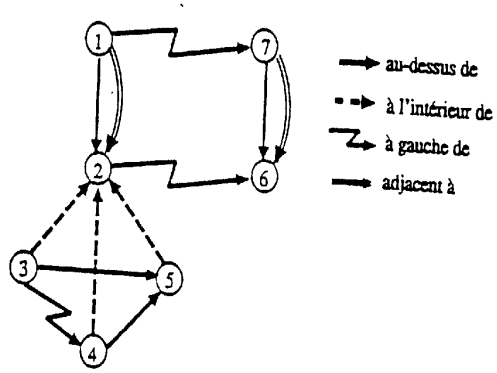
Graphs : A powerful representation tool …

- **Typical examples ?**
- **How ? Why ?**
- **Advantages ?**
- **Drawbacks ?**

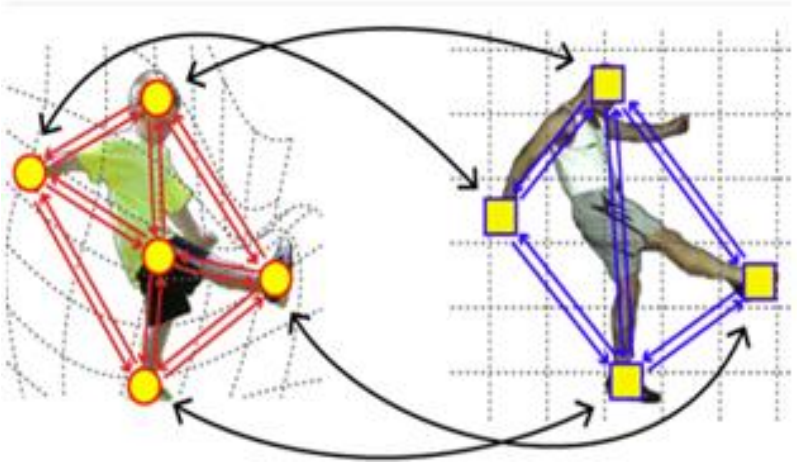
Graphs : A powerful representation tool …

- Topology
  - Nodes = primitives, elements, parts
  - Edges = relations





- Attributes
  - Statistical : observations, distributions, …
  - Geometrical : metrics (distances, angles, similarities)
  - Positions : absolutes or relatives
  - Visual features : discriminative elements

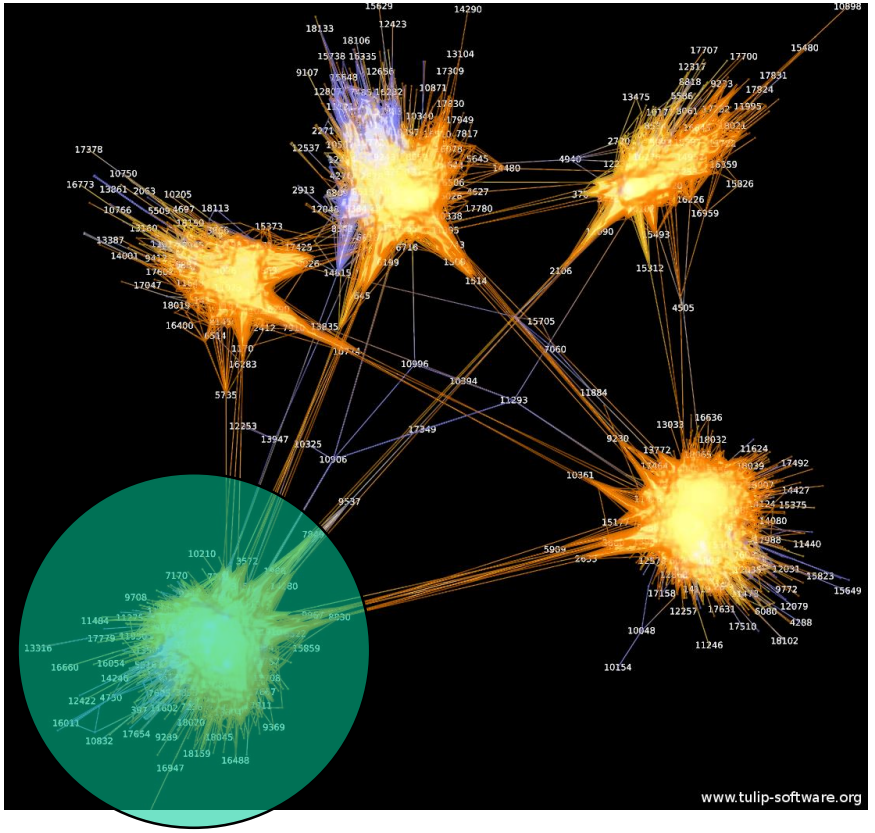


- Trying to ensure
  - Stability (invariance)
  - Tolerance : noises, variations
  - Classes discrimination
- But
  - Symbolic VS numerical
  - Discretisation

Social networks

- Facebook100 data set

”People and friendships from the Facebook networks of 100 different colleges and universities from a single snapshot from September 2005.”



UNC Chapel Hill (18163 Nodes, 766,800 Edges) displayed with Tulip by David Auber (2011)

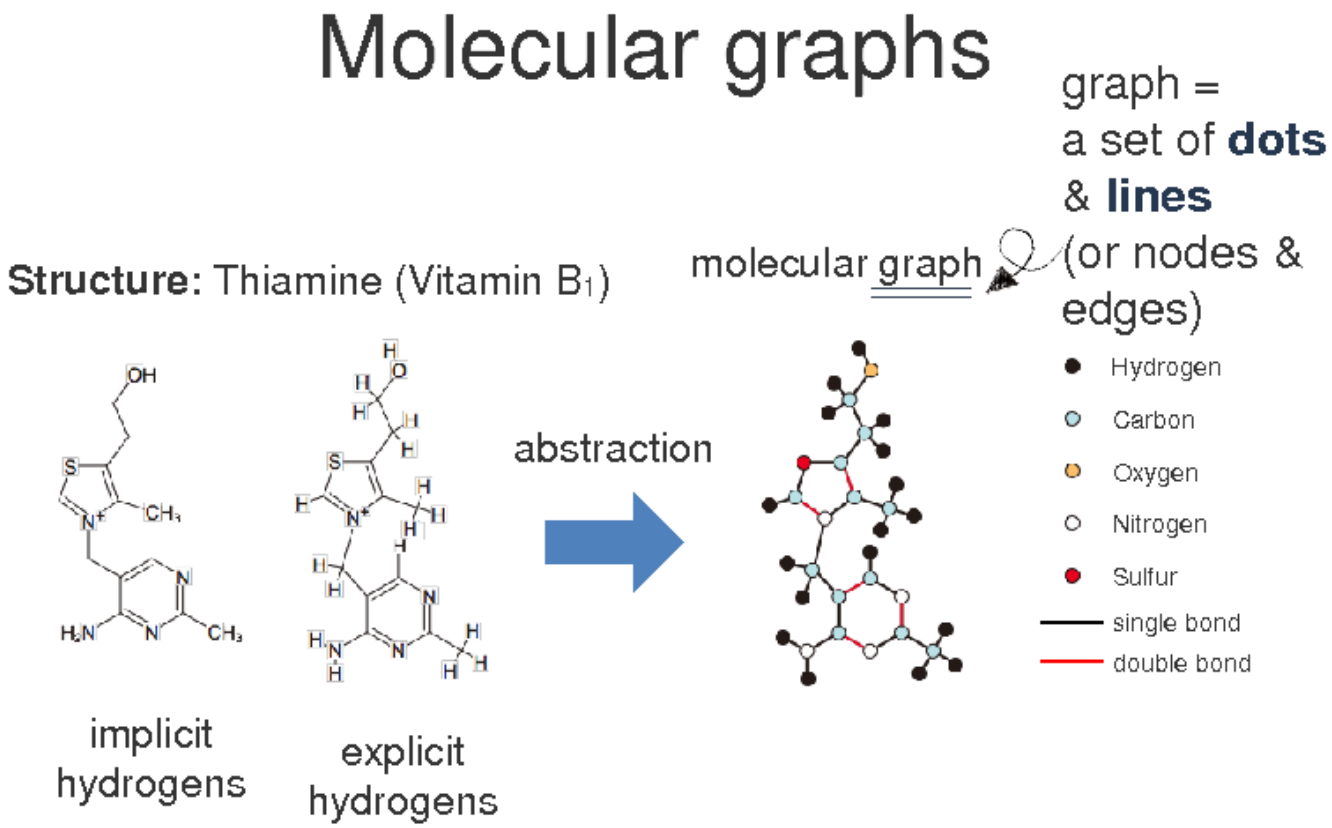
- Everyone can be a media outlet
- Disappearing of communications barrier
  - Rich User Interaction
  - User-Generated Contents
  - User Enriched Contents
  - User developed widgets
  - Collaborative environment
  - Collective Wisdom
  - Long Tail

Broadcast Media
Filter, then Publish

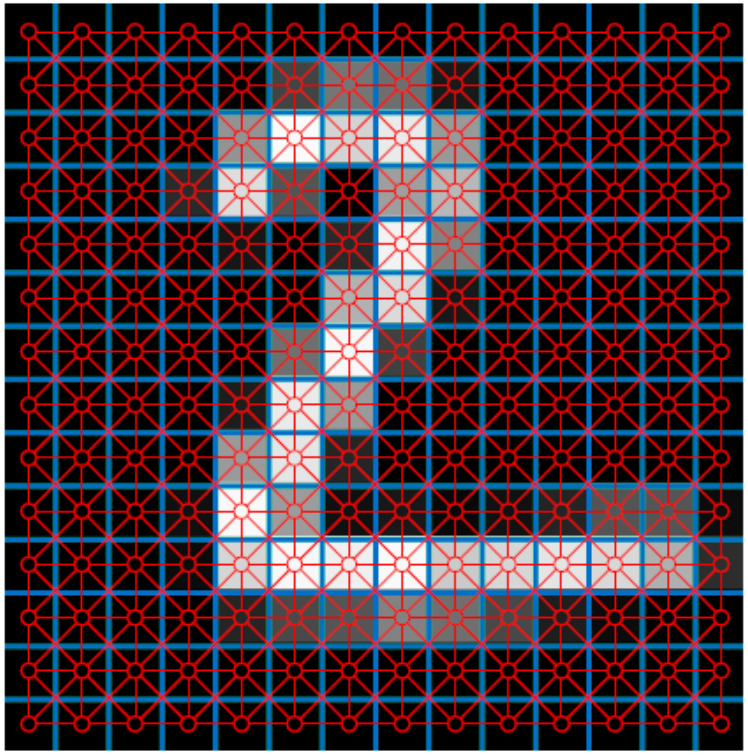
→

Social Media
Publish, then Filter

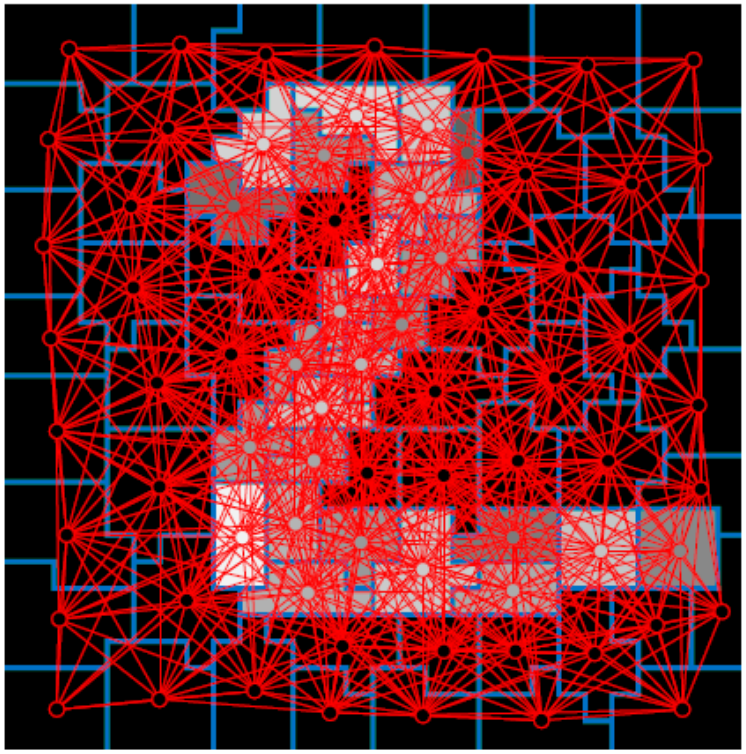
Graph in chemioinformatics



Graph of pixels

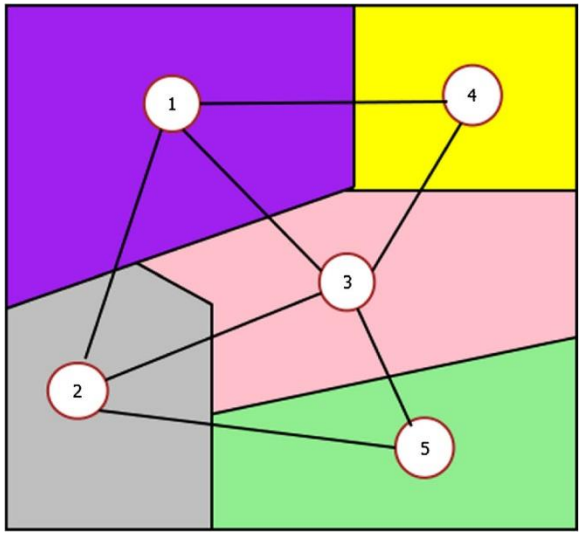


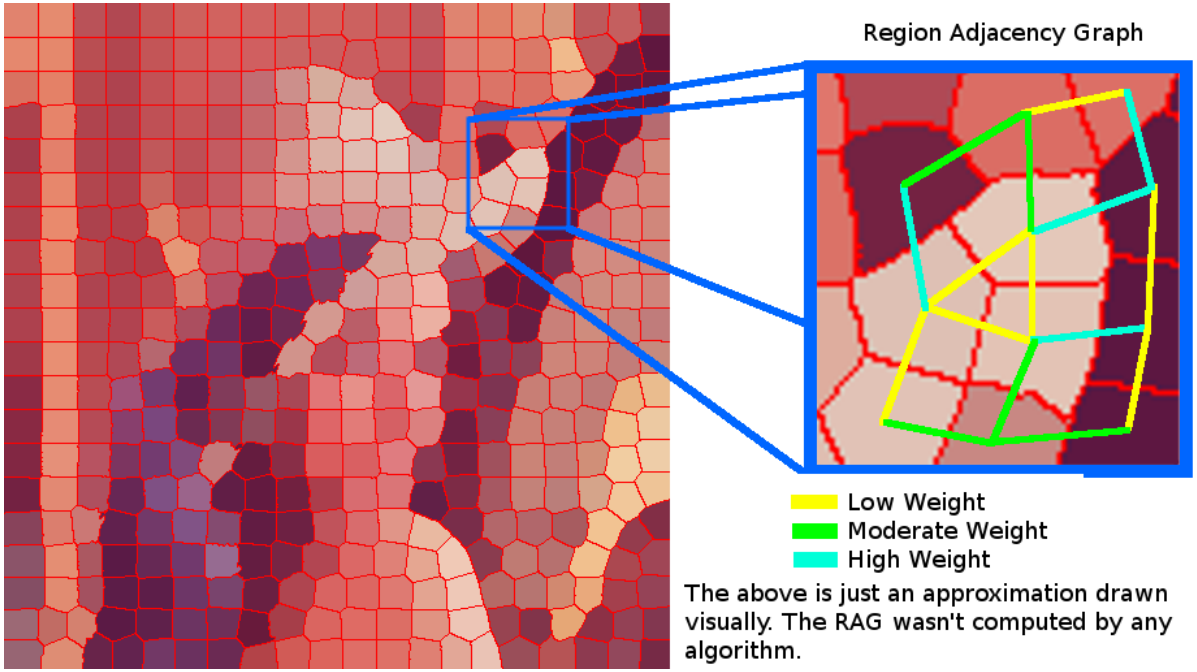
Regular grid



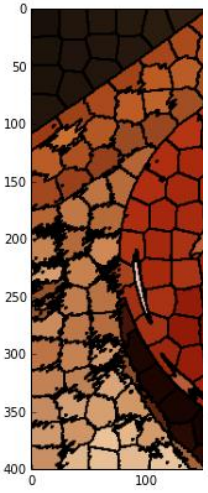
Superpixels

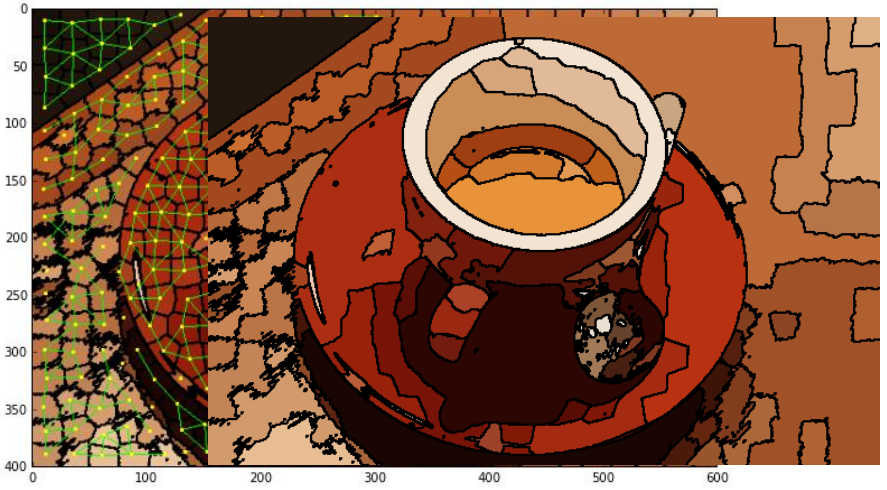
Region Adjacency Graph



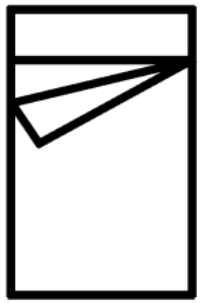


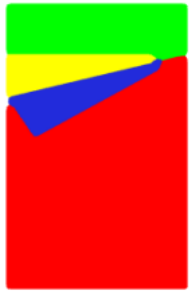
The above is just an approximation drawn visually. The RAG wasn't computed by any algorithm.



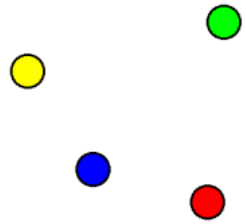


Region Adjacency Graph

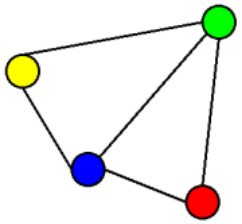




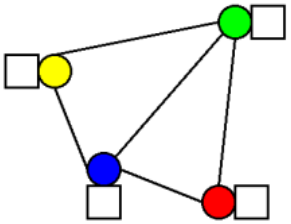
⇒ Segmentation
« région »



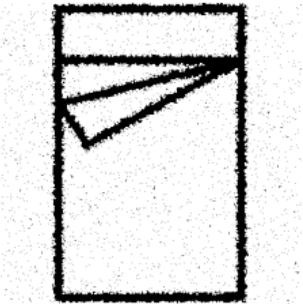
région ⇔ nœud

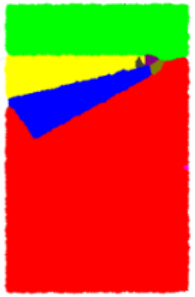


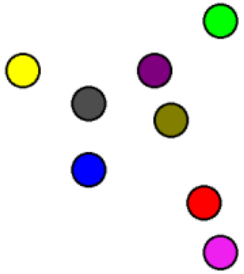
adjacence ⇔ arc

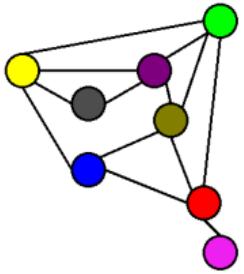


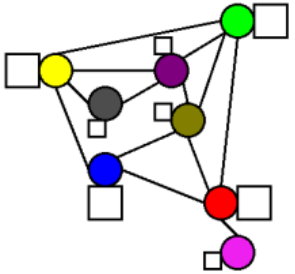
⇒ Description







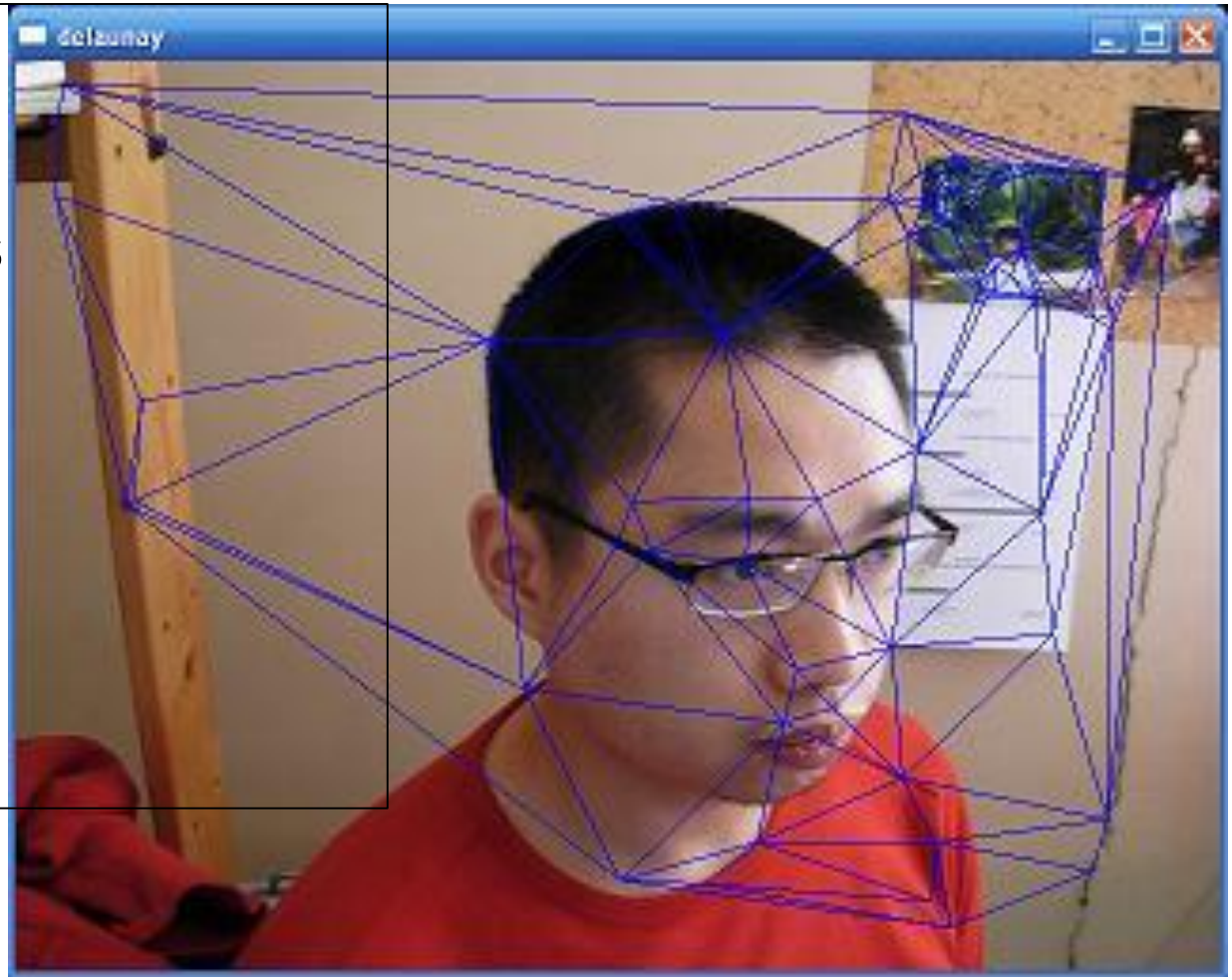




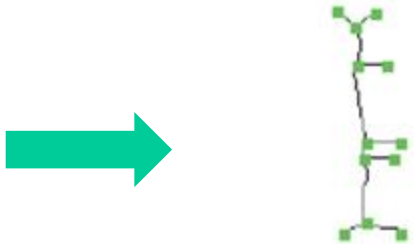
Impact of noise on Graph-Based Representation

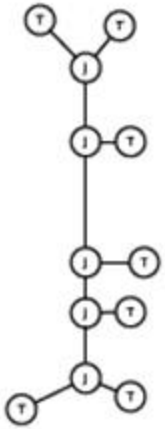
Interest Point Graph

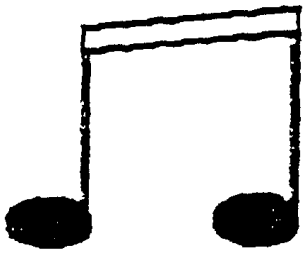
- Node ➔ Keypoints
- Edges ➔ distances between Keypoints
- Many other possibilities
  - Similarities
  - Angles
  - …



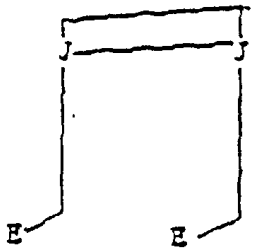
Skeleton Graph





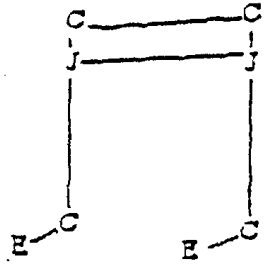


Objet : 2 doubles croches



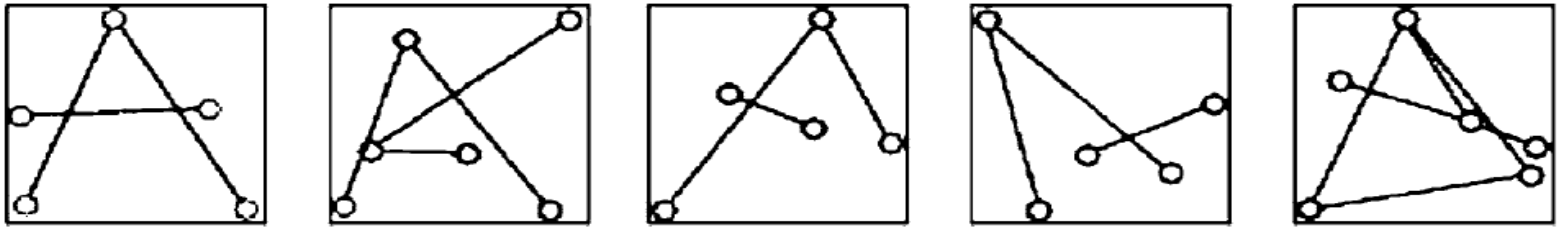
Graphe du squelette avant découpage récursif :

4 segments
2 extrémités (E)
2 jonctions (J)

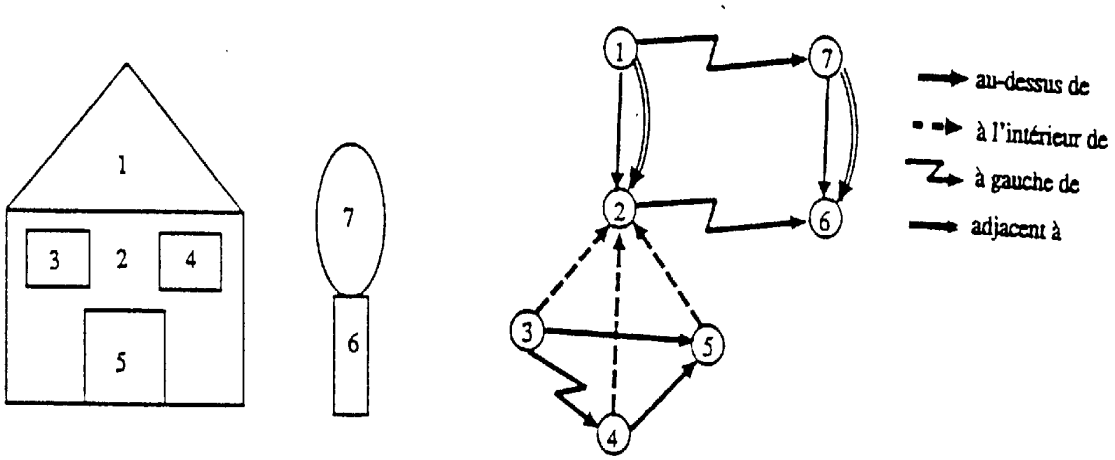


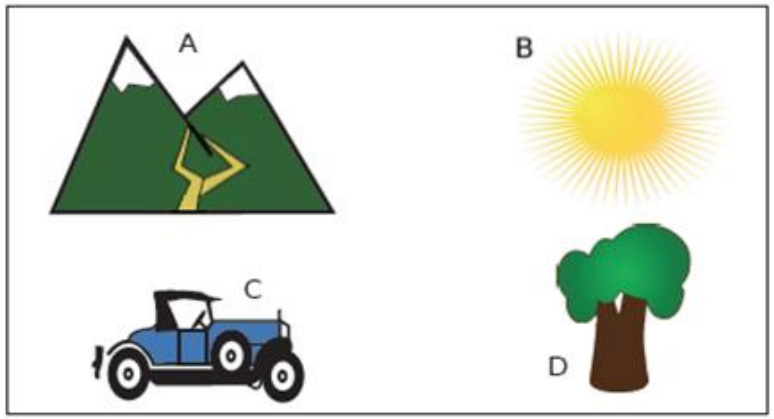
Graphe du squelette après découpage récursif :

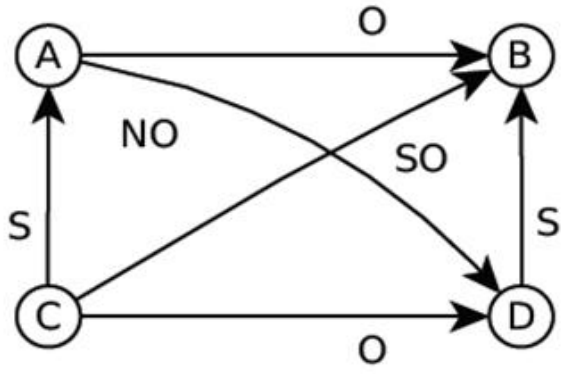
8 segments
2 extrémités (E)
2 jonctions (J)
4 coupures (C)



Spatial relationship graph







Graph Databases

- *OPEN GRAPH BENCHMARK*
  https://ogb.stanford.edu/

- TORCH_GEOMETRIC.DATASETS
- HTTPS://PYTORCH-GEOMETRIC.READTHEDOCS.IO/EN/LATEST/MODULES/DATASETS.HTML

- *IAM Graph Database Repository*

for Graph Based Pattern Recognition and Machine Learning" (2008)

https://iapr-tc15.greyc.fr/links.html#Benchmarking%20and%20data%20sets

Graphs for PR and Mining

Graph-based problems

- Graph analysis / Graph Mining
- Link prediction
- Vertex classification/clustering/regression
- Graph classification/clustering/regression
- Graph matching
- Graph distance
- Graph-based search
  - Subgraph search
  - Similarity search
- Graph prototypes - Median graphs

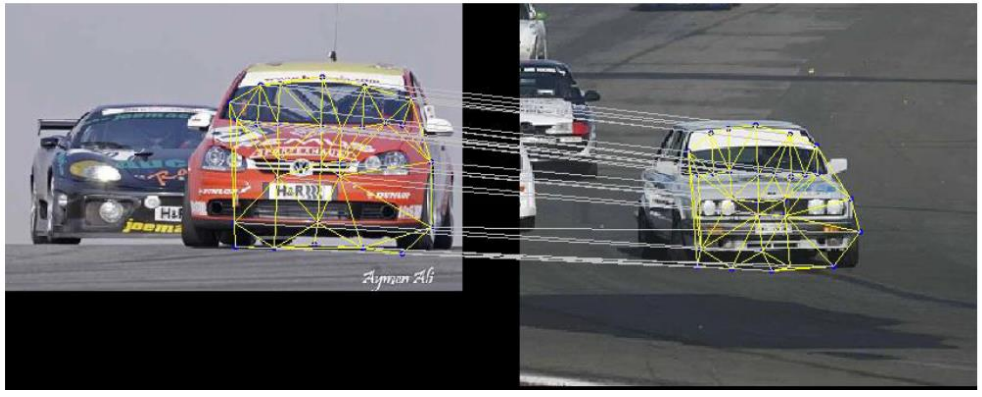
What does it mean ?

Graphs for PR and Mining

Graph-based problems

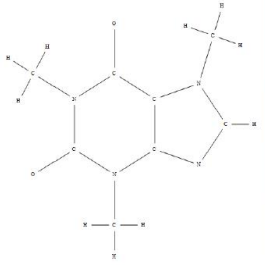
- Graph analysis / Graph Mining
- Link prediction
- Vertex classification/clustering/regression
- Graph classification/clustering/regression
- Graph matching
- Graph distance
- Graph-based search
  - Subgraph search
  - Similarity search
- Graph prototypes - Median graphs

Graph matching



Graph classification

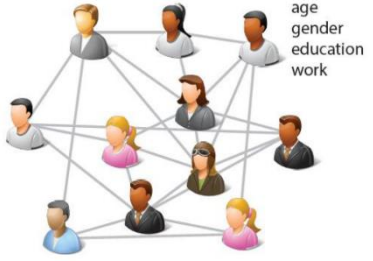
1. cancerous or not cancerous molecules
2. determination of the boiling point



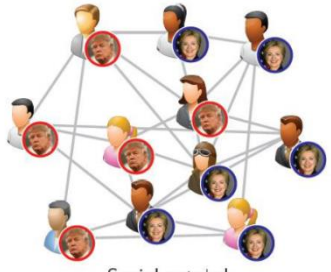
Molecular graph

Image recognition

Vertex classification

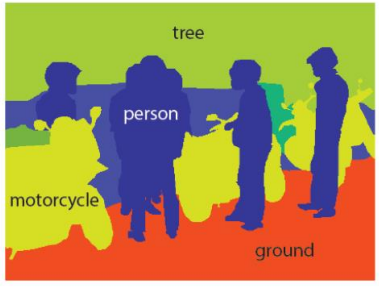


Social network



Social network

Semantic image segmentation



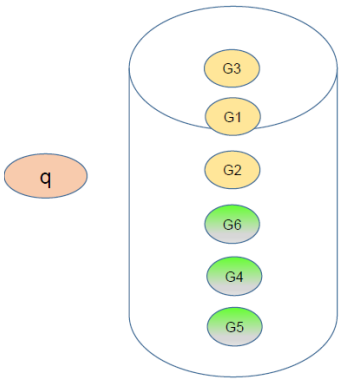
Semantic image segmentation

Graphs for PR and Mining

Graph-based problems

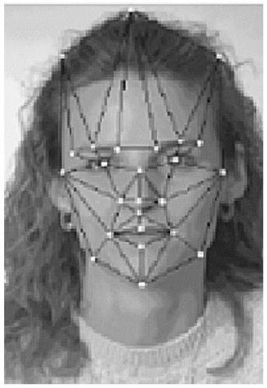
- Graph analysis / Graph Mining
- Link prediction
- Vertex classification/clustering/regression
- Graph classification/clustering/regression
- Graph matching
- Graph distance
- Graph-based search
  - Subgraph search
  - Similarity search
- Graph prototypes - Median graphs

Graph-based search

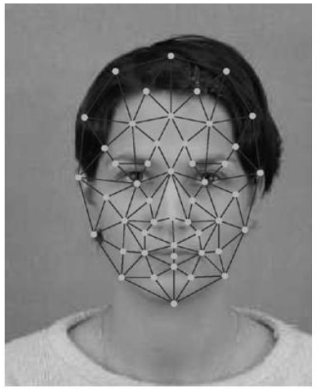


1. Given a graph database consisting of n graphs, $D = g1, g2, ..., gn$, and a query graph q, almost all existing algorithms of processing graph search can be classified into the following four categories: Full graph search, Subgraph search, Similarity search and Graph mining.
2. **Full graph search.** Find all graphs gi in D s.t. gi is the same as q.
3. **Subgraph search**. Find all graphs gi in D containing q or contained by q.
4. **Similarity search**. Find all graphs gi in D s.t. gi is similar to q within a user-specified threshold based on some similarity measures.
5. **Graph mining** Graph mining problem gathers similar graph or subgraph of D in order to find clusters or prototypes. No query is provided by the user.

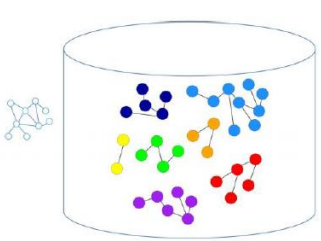
Graph distance

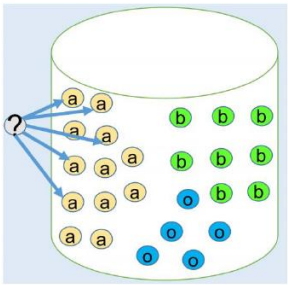


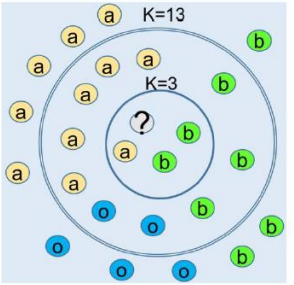
How similar are theses graphs ?



Graph similarity search







How to solve these problems?

Staying in the Graph space

- Graph matching
  - Combinatorial problems (NP Hard)

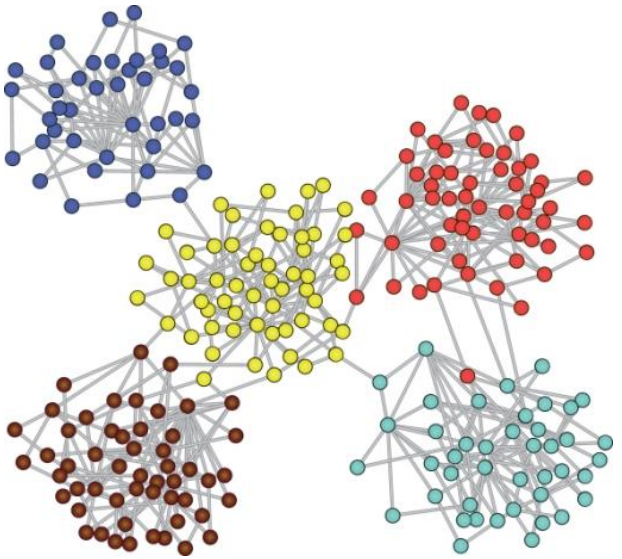
Graph embedding $G \rightarrow \mathbb{R}^n$

- Embedded of graphs/ nodes edges into a vector space
  - Explicit embedding
    - Through feature extraction
      (handcrafted or end to end learning) or dissimilarities
  - Implicit embedding
    - Through graph kernel

➔ Modelisation, PR, Machine learning, Optimization

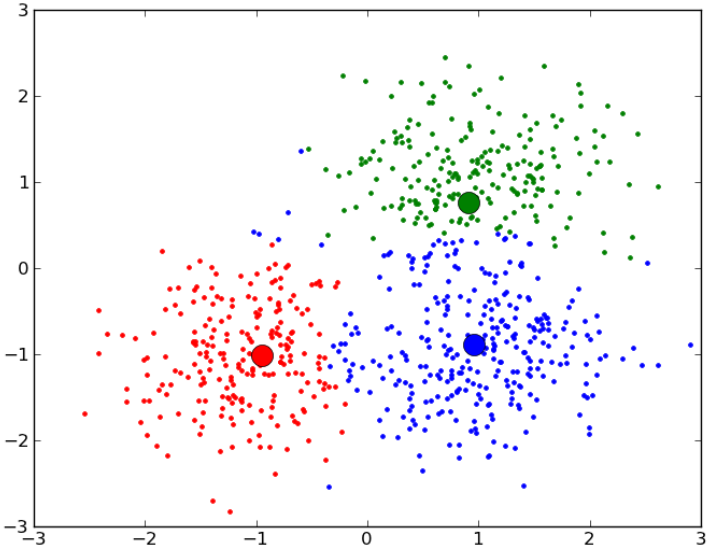
Graph Analysis (only one or several) ?

- Graph clustering
  - finding sets of “related” vertices in graphs
  - graph partitioning



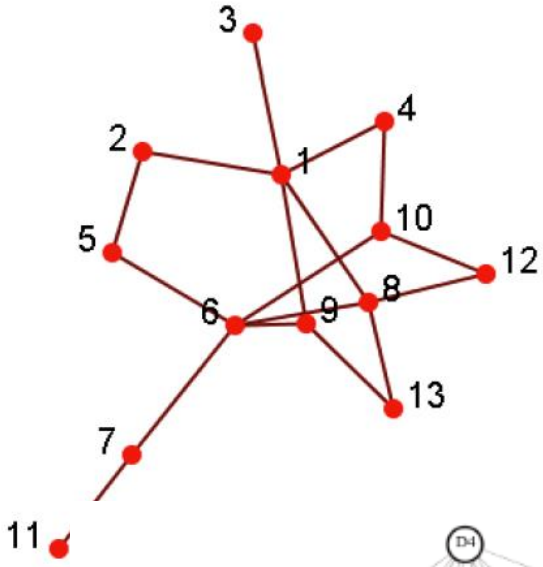
≠

- Graph clustering
  - clustering of sets of graphs based on structural similarity

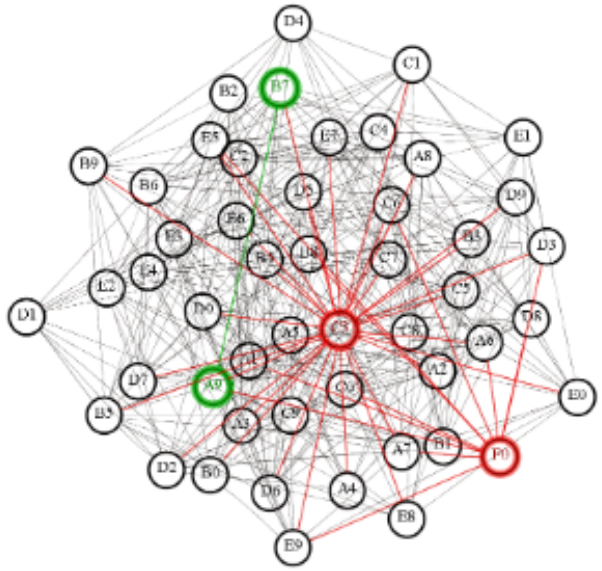


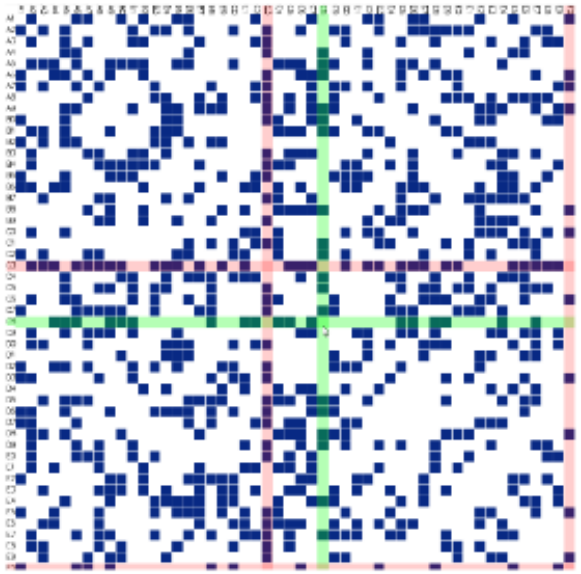
2. Graph characterization (1 big graph)

Graph depictions



	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	1	1	1	0	0	0	1	1	0	0	0	0
2	1	0	0	0	1	0	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0	0	0	0	0	0
...													





a

Figure 1 Two visualizations of the same undirected graph containing 50 vertices and 400 edges.

Graph depictions

- The order of the graph G, $n = |V|$
- The size of the graph G, $m = |E|$
- A graph is planar if it can be drawn in a plane without any of the edges crossing
- The density of the graph G, $\frac{m}{\binom{n}{2}}$
- A graph of density 1 is called complete

- Uniform random graphs (Gilbert model)
  - With n vertices, each of the $\binom{n}{2}$ possible edges is included in the graph with probability p

$\binom{n}{2}$ → ***n.(n-1)/2***

Graph drawing: Node-diagram

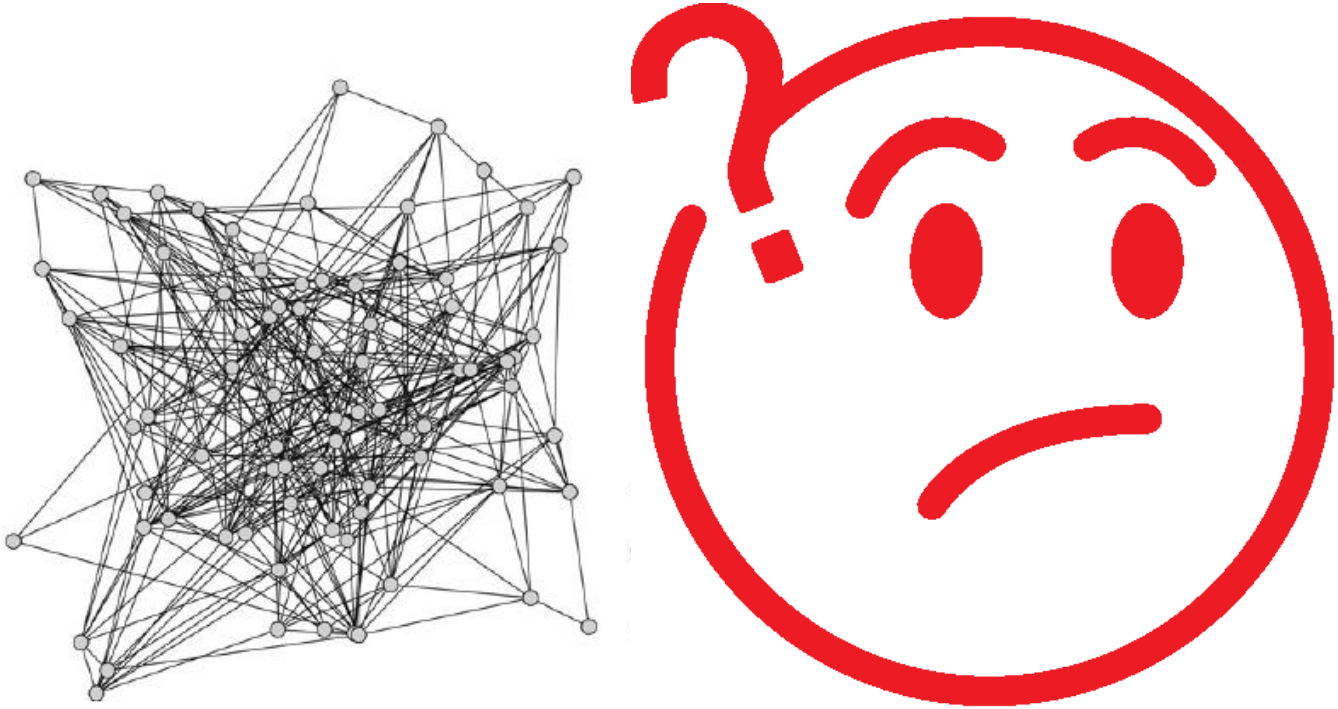


Fig. 2 – Two graphs both of which have 84 vertices and 358 edges. The graph on the left is a uniform random graph of the $\mathcal{G}_{n,m}$ model [84,85] and the graph on the right has a relaxed caveman structure [228]. Both graphs were drawn with spring-force visualization [203].

Node-diagram

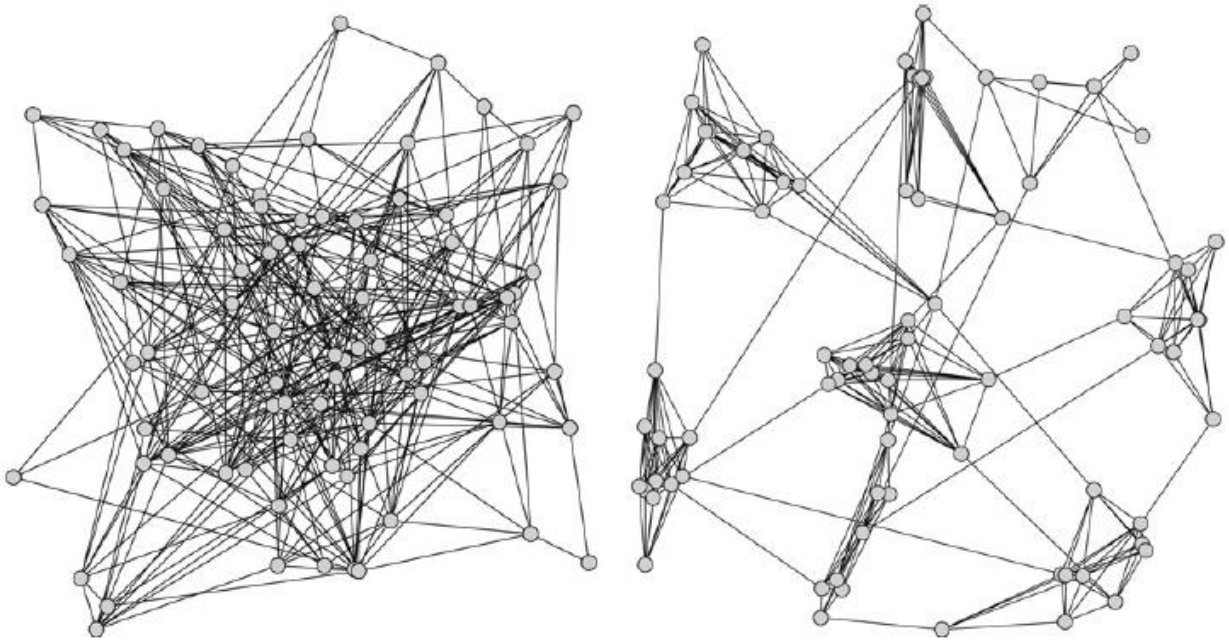
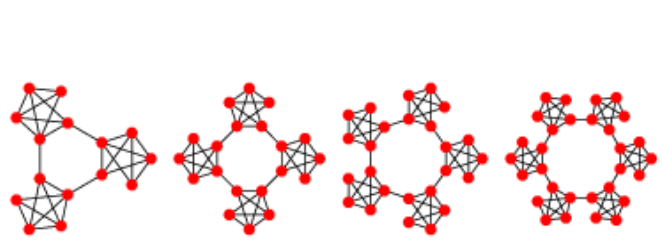


Fig. 2 – Two graphs both of which have 84 vertices and 358 edges. The graph on the left is a uniform random graph of the $\mathcal{G}_{n,m}$ model [84,85] and the graph on the right has a relaxed caveman structure [228]. Both graphs were drawn with spring-force visualization [203].



Matrix representation

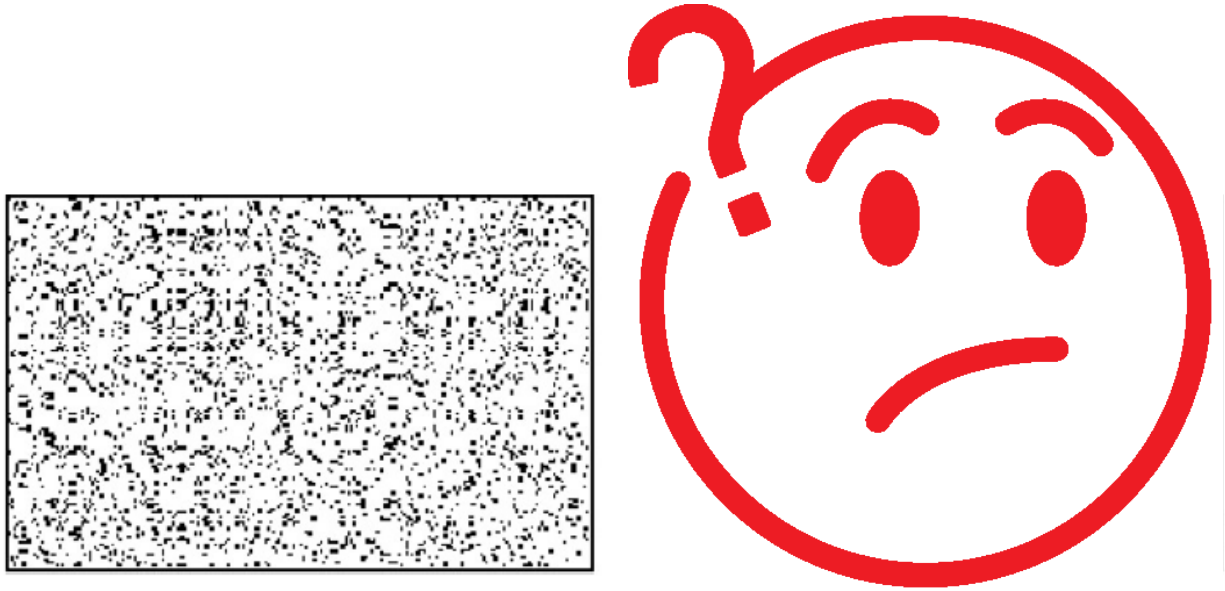


Fig. 1 – The adjacency matrix of a 210-vertex graph with 1505 edges composed of 17 dense clusters. On the left, the vertices are ordered randomly and the graph structure can hardly be observed. On the right, the vertex ordering is by cluster and the 17-cluster structure is evident. Each black dot corresponds to an element of the adjacency matrix that has the value one, the white areas correspond to elements with the value zero.

Matrix representation

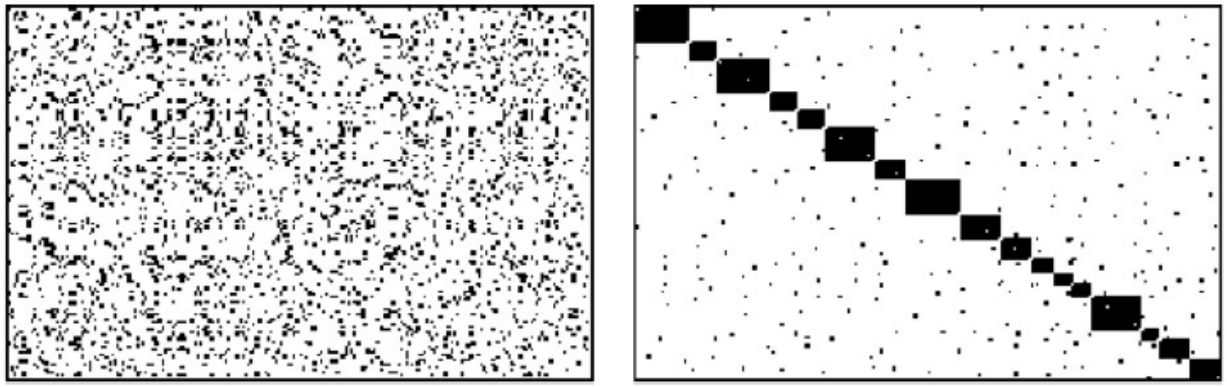
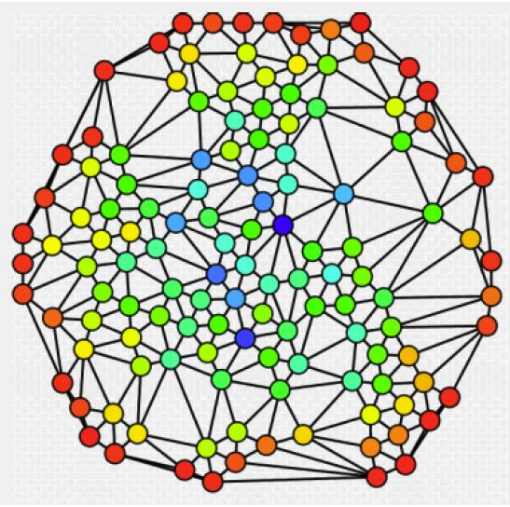


Fig. 1 – The adjacency matrix of a 210-vertex graph with 1505 edges composed of 17 dense clusters. On the left, the vertices are ordered randomly and the graph structure can hardly be observed. On the right, the vertex ordering is by cluster and the 17-cluster structure is evident. Each black dot corresponds to an element of the adjacency matrix that has the value one, the white areas correspond to elements with the value zero.

Influence Study

Centrality Analysis

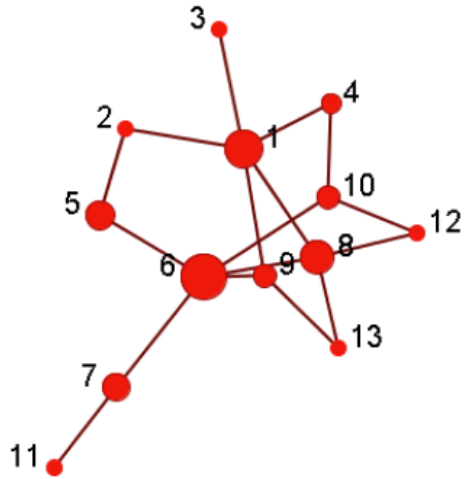


- **Node centrality**: number of shortest paths including this node
- **Edge-Betweenness**: Number of shortest paths between any pair of nodes that pass through the edge
- To identify the most important "actors" in a social network → Given a graph, output a list of top-ranking nodes or edges
- Ratio formulation:

$$bc(v) = \sum_{\substack{s \neq t \neq v \\ s,t \in V}} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}}$$

$\sigma_{s,t}(v)$: number of paths from node ***s*** to ***t*** that include node ***v***

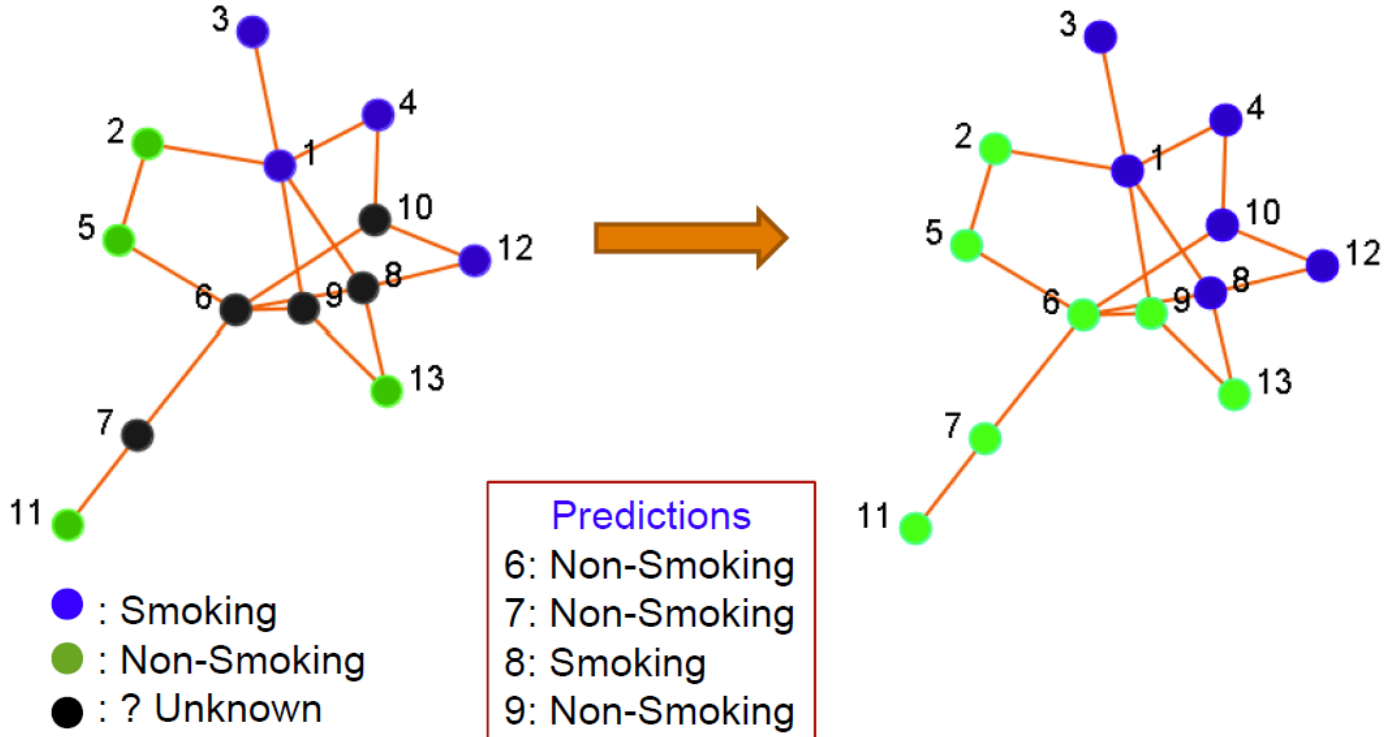
$\sigma_{s,t}$: total number of paths from ***s*** to ***t***



(Nodes resized by Importance)

Node Prediction

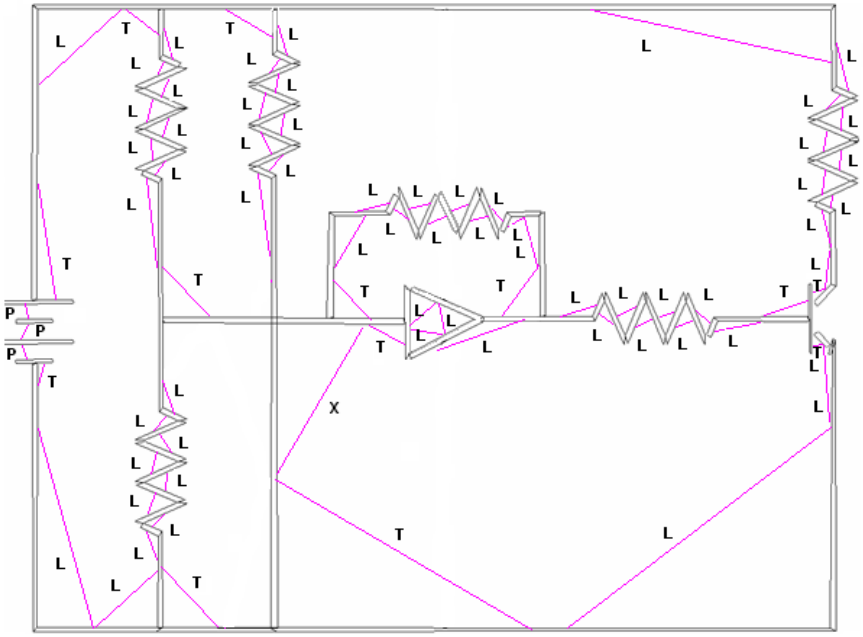
Prediction ➔ Node classification



Notion of node signature ?

From Node prediction to Sub-graph Spotting

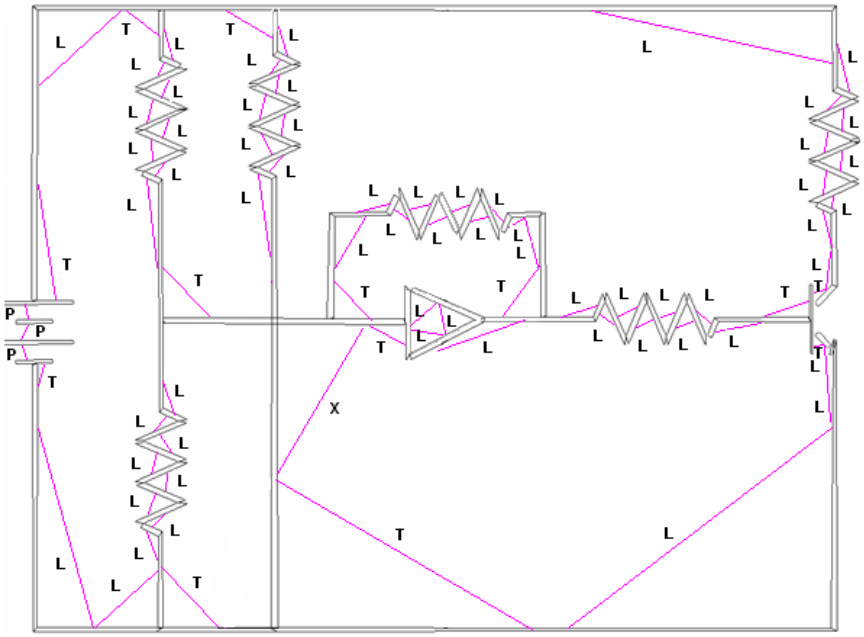
Drawing ➔ Graph ➔ Symbol detection ?



Can you imagine some features / signatures ?

From Node prediction to Sub-graph Spotting

Drawing ➔ Graph ➔ Symbol detection ?



A first draft ➔ Using heuristics

To associate score to the nodes and edges

- H1 – Symbols are composed of small segments compared to the other parts
- H2 – Segments inside a symbol are of similar length
- H3 – Symbols can correspond to loops
- H4 – Symbols can correspond to parallel segments
- H5 – Segments inside symbols are connected to maximum 3 other segment
- H6 – Two segments with 90° usually correspond to a symbol

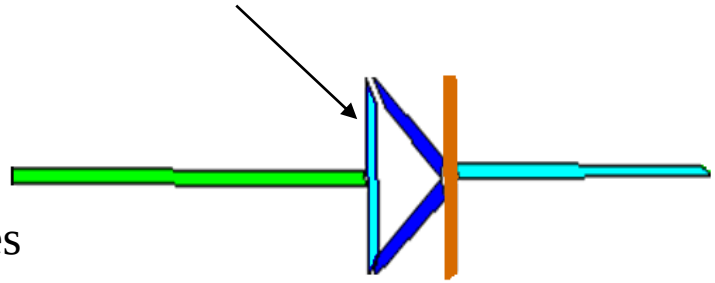
Using Machine Learning

➔ the new “heuristic free” approach…

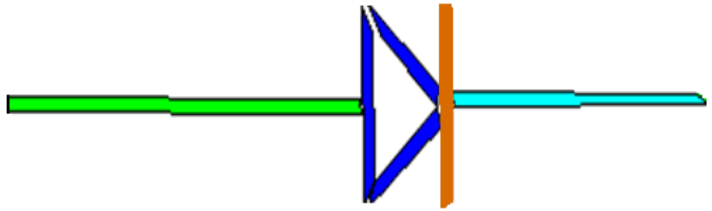
➔ Node / Graph embedding

Region Spotting with graphs

Score Propagation



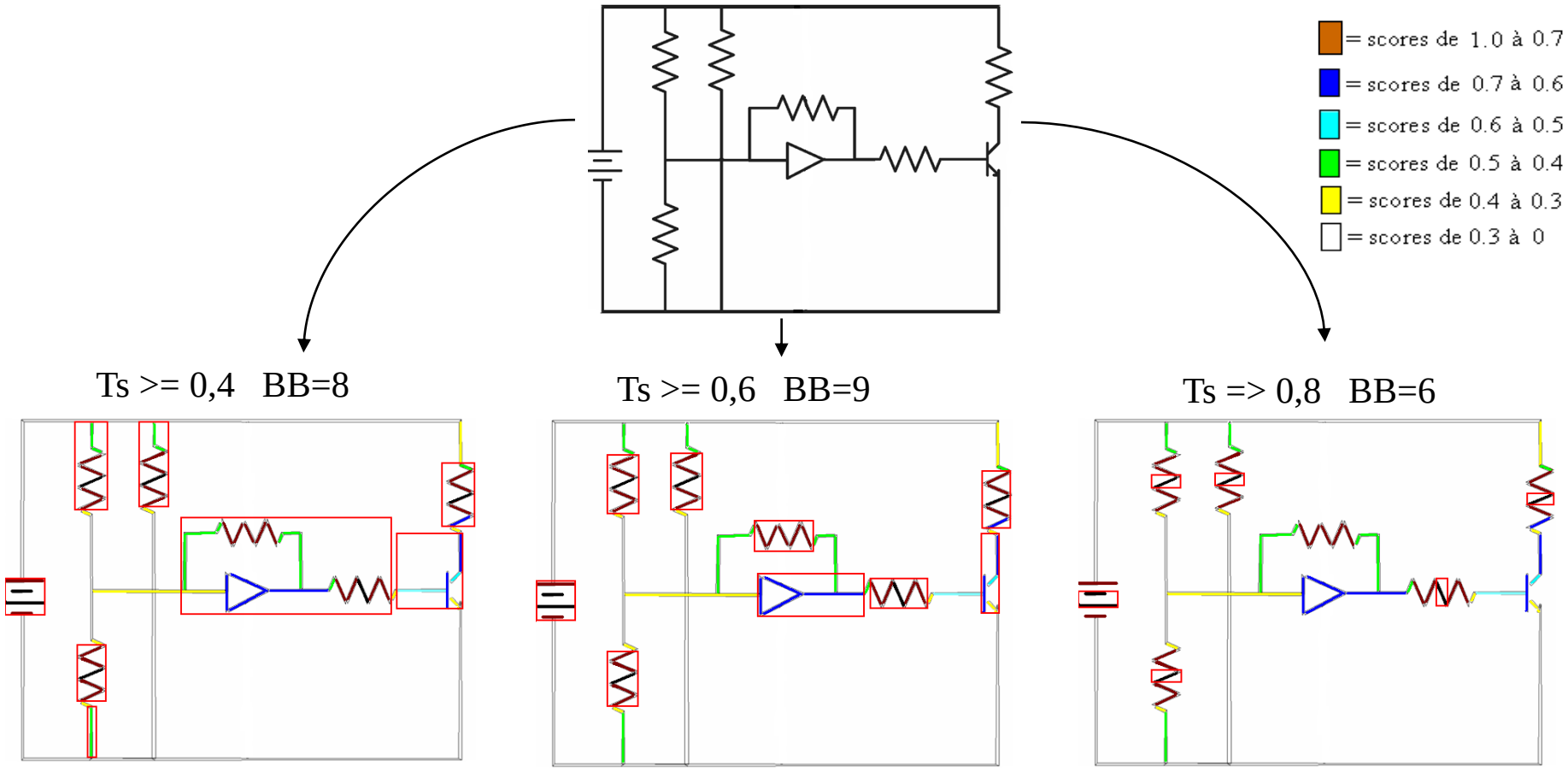
(a) Initial scores



(b) score propagation inside loops ➔ 0.7 has been propagated

Region Spotting with graphs

Extraction of RoI / sub-graphs using the scores



Region Spotting with graphs

Experimentations on different types of documents

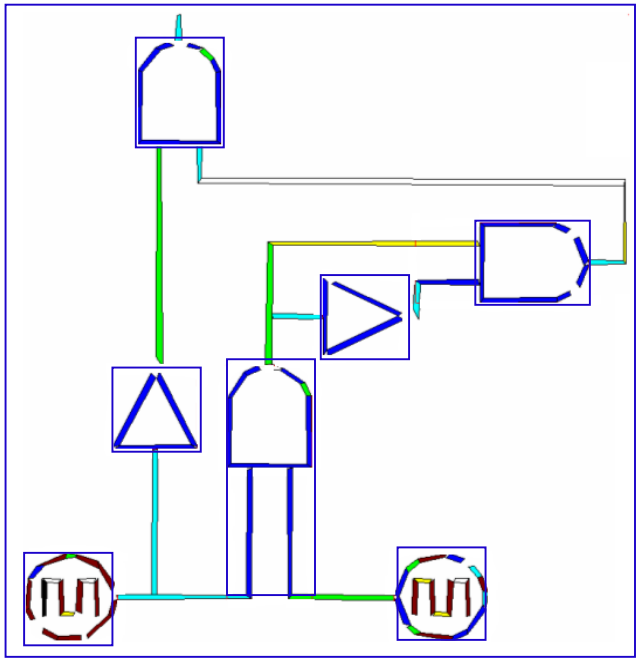
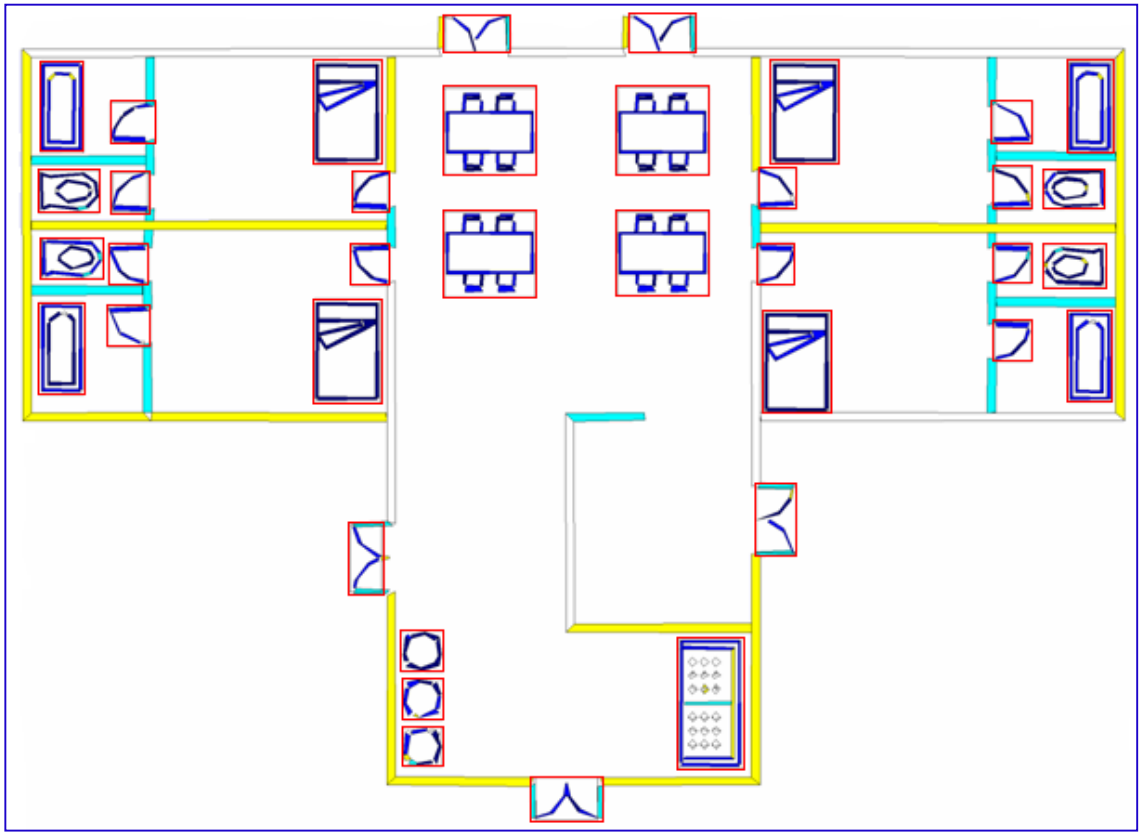


Diagramme logique Ts >= 0,6



Plan d'architecture Ts >= 0,5

3. Graph partitioning (community detection)

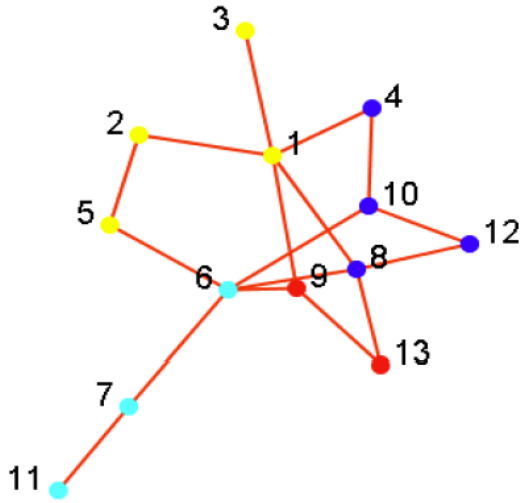
Community detection

No universally accepted definition for “community”

• Members within a community are more similar among each other

• Communities correspond to densely connected nodes: Set of nodes with more connections between its members, than to the rest of the network

• A community corresponds to a group of nodes with more intra-cluster edges than inter-clusters edges



(Nodes colored by Community Membership)

So, detection methods can be divided into several categories:

- **Node-Centric** Community → Each node in a group satisfies certain properties
- **Group-Centric** Community → Partition the whole network into several disjoint sets → Consider the connections within a group as a whole. The group has to satisfy certain properties without zooming into node-level
- **Hierarchy-Centric** Community → Construct a hierarchical structure of communities

Evaluation metrics

- Focus on :
  - Intra-cluster edge density (number of edges within community)
  - Inter-cluster edge density (number of edges across communities)
  - Both two criteria

Intra Cluster Density

$$\delta_{int}(\mathcal{C}) = \frac{\#\text{ internal edges of } \mathcal{C}}{n_c(n_c-1)/2}.$$

Inter Cluster Density

$$\delta_{ext}(\mathcal{C}) = \frac{\#\text{ inter-cluster edges of } \mathcal{C}}{n_c(n-n_c)}.$$

Notations	
V	set of vertices
E	set of edges
n	|V|
m	|E|
C	A subset of V
n_c	|C|

maximum number of inter-cluster edges possible

Max number of edge inside C

Etienne Cuvelier, Marie-Aude Aufaure. Graph Mining and Communities Detection. Aufaure, MarieAude; Zimanyi, Esteban. First European Summer School, eBISS 2011, Paris, France, July 3-8, 2011, Tutorial Lectures, Springer, pp.117-138, 2012, Lecture Notes in Business Information Processing. ffhal-00704356f

Node-Centric Community Detection

- Node similarity is defined by how similar their interaction patterns are.
  - Two nodes are structurally equivalent if they connect to the same set of actors
    → e.g., nodes 8 and 9 are structurally equivalent

	1	2	3	4	5	6	7	8	9	10	11	12	13
a vector → 5		1				1							
structurally equivalent 8	1					1							1
structurally equivalent 9	1					1							1

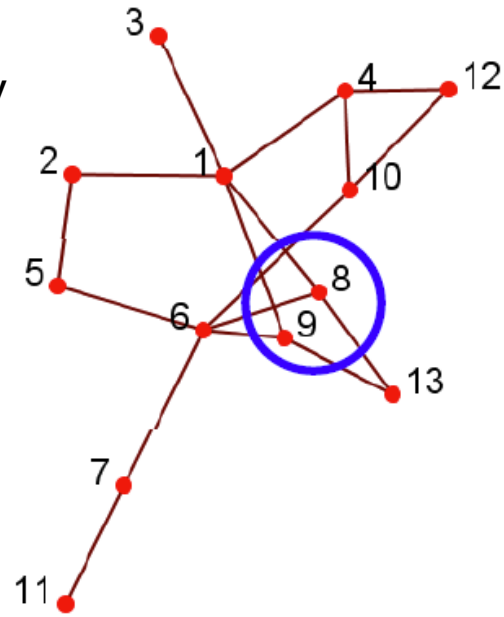
- For practical use with huge graphs:
  - Consider the connections as features
  - Use Cosine or Jaccard similarity to compute vertex similarity
  - Apply classical k-means clustering Algorithm

Cosine Similarity: $\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}.$

$$sim(5,8) = \frac{1}{\sqrt{2} \times \sqrt{3}} = \frac{1}{\sqrt{6}}$$

Jaccard Similarity: $J(A,B) = \frac{|A \cap B|}{|A \cup B|}.$

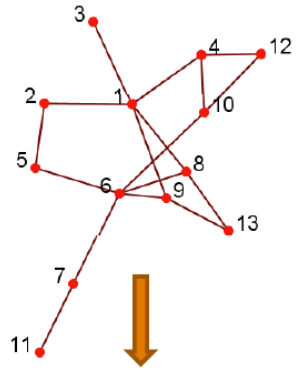
$$J(5,8) = \frac{|\{6\}|}{|\{1,2,6,13\}|} = 1/4$$



Node-Centric Community Detection

Multi-Dimensional Scaling (MDS) – Spectral methods *("ACP-like method")*

- Given a Graph, construct a proximity matrix to denote the distance between nodes
- Δ(*D*) denotes the *square distance matrix* between nodes
- Λ = diag(λ_1, λ_2,…,λ_k) = the top-k eigenvalues of Δ(*D*) and V = the top-k eigenvectors of Δ(*D*)
- $S \in R^{nxk}$ denotes the coordinates of nodes in the lower-dimensional space
- MDS objective → minimize the difference min || Δ(*D*) − $S.S^T$ ||
- MDS solution → $S = V\Lambda^{1/2}$
- Apply k-means to S to obtain clusters



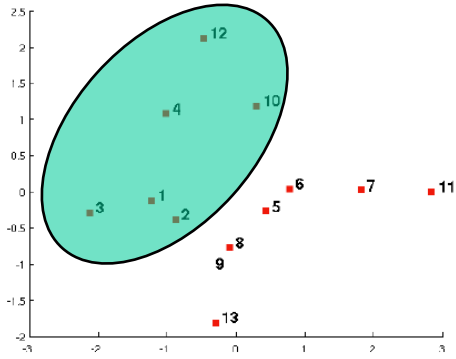
Geodesic Distance Matrix

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	1	1	1	2	2	3	1	1	2	4	2	2
2	1	0	2	2	1	2	3	2	2	3	4	3	3
3	1	2	0	2	3	3	4	2	2	3	5	3	3
4	1	2	2	0	3	2	3	2	2	1	4	1	3
5	2	1	3	3	0	1	2	2	2	2	3	3	3
6	2	2	3	2	1	0	1	1	1	1	2	2	2
7	3	3	4	3	2	1	0	2	2	2	1	3	3
8	1	2	2	2	2	1	2	0	2	2	3	3	1
9	1	2	2	2	2	1	2	2	0	2	3	3	1
10	2	3	3	1	2	1	2	2	2	0	3	1	3
11	4	4	5	4	3	2	1	3	3	3	0	4	4
12	2	3	3	1	3	2	3	3	3	1	4	0	4
13	2	3	3	3	3	2	3	1	1	3	4	4	0

MDS

S

-1.22	-0.12
-0.88	-0.39
-2.12	-0.29
-1.01	1.07
0.43	-0.28
0.78	0.04
1.81	0.02
-0.09	-0.77
-0.09	-0.77
0.30	1.18
2.85	0.00
-0.47	2.13
-0.29	-1.81



k-means

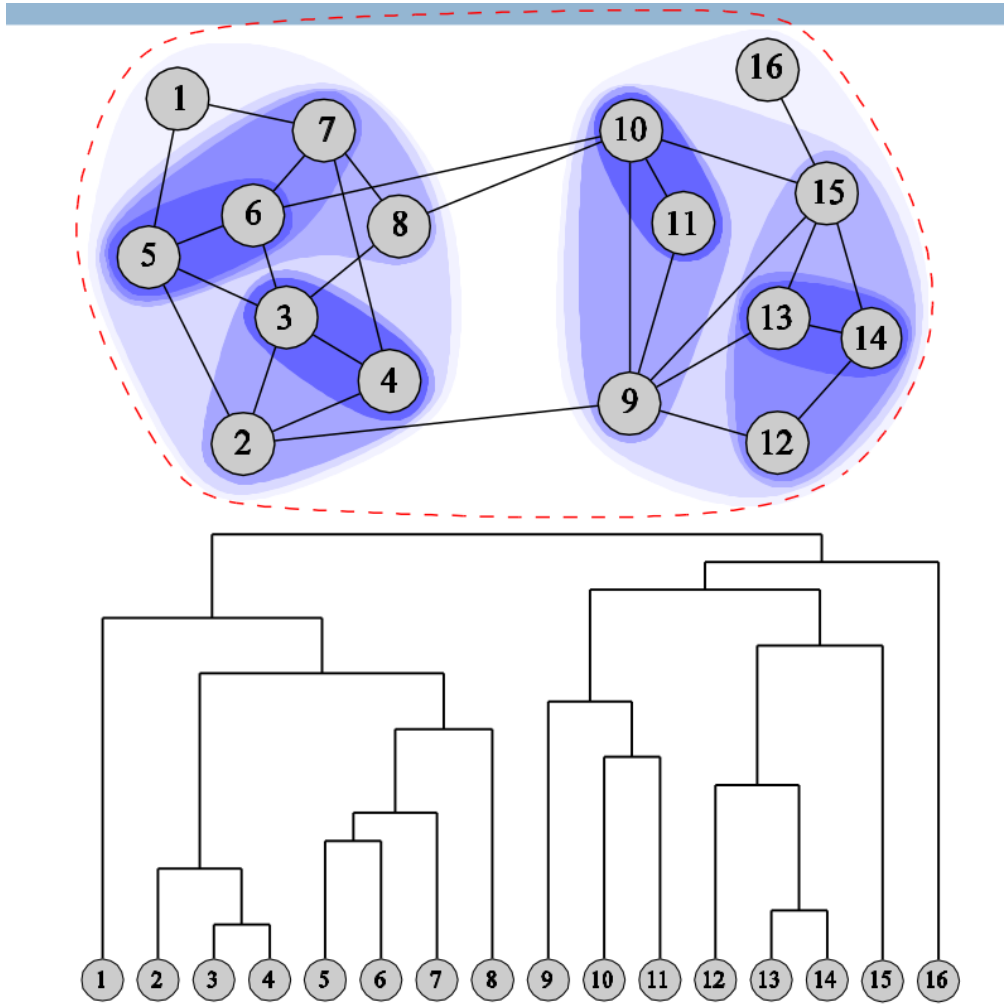
1, 2, 3, 4, 10, 12

5, 6, 7, 8, 9, 11, 13

Node-Centric Community Detection

Ascendant Hierarchical Classification of the nodes

- Each node = 1 community
- Compute distances between communities
- Merge most similar.
- Go to Point 2.



Hierarchy-Centric Community Detection

Goal: Build a hierarchical structure of communities based on graph topology

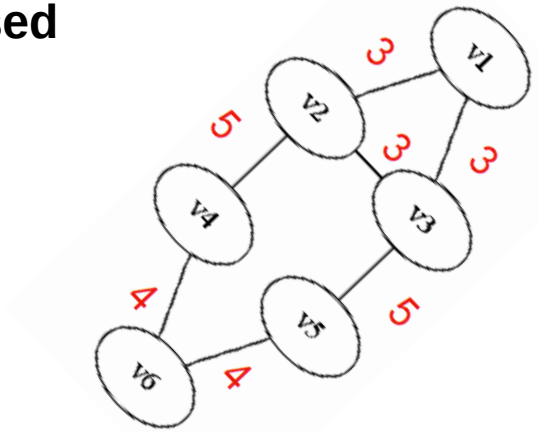
- Facilitate the analysis at different resolutions
- Representative Approaches:
  - Divisive Hierarchical Clustering
  - Agglomerative Hierarchical Clustering

Divisive Hierarchical Clustering

- Partition the nodes into several sets
- Each set is further partitioned into smaller sets
- Between-group edges tend to **have larger edge-betweenness**

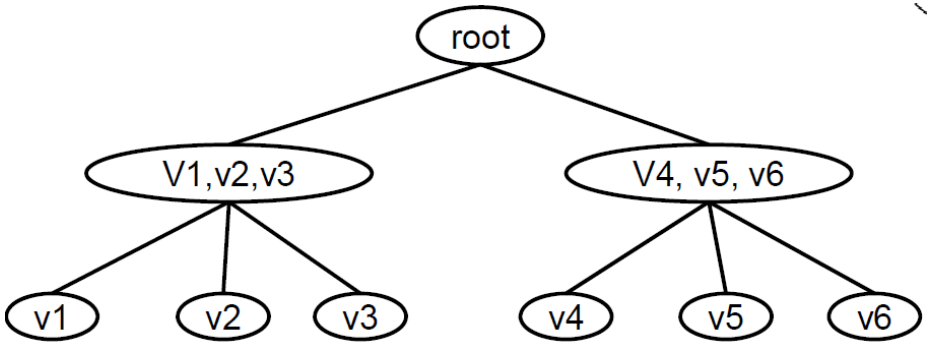
Agglomerative Hierarchical Clustering

- Similar to node-centric methods



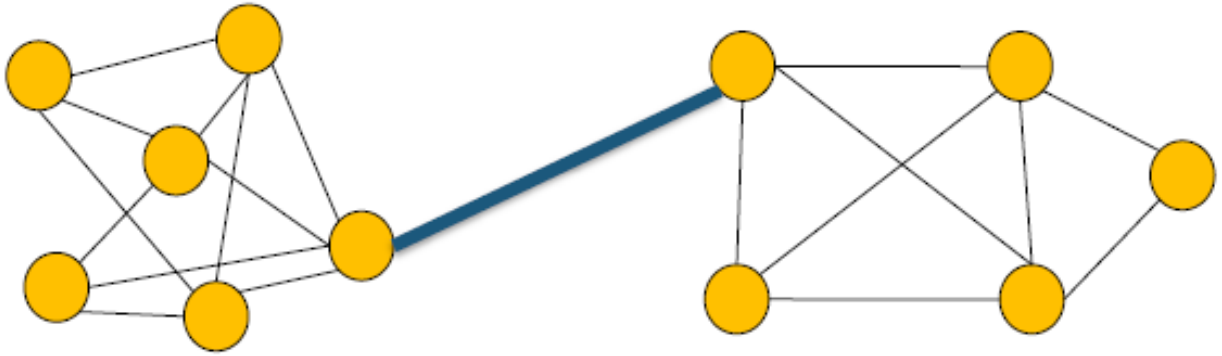
Progressively remove edges with the highest betweenness

- Remove e(2 4) e(3 5)
- Remove e(4,6), e(5,6)
- Remove e(1,2), e(2,3), e(3,1)



Hierarchical method: Newman-Girvan algorithm

- Newman-Girvan algorithm [Newman and Girvan '04]
  - A **divisive** algorithm (detect and remove edges that connect vertices of different communities)
  - **Idea:** try to identify the edges of the graph that are most between other vertices → responsible for connecting many node pairs
  - Select and remove edges based to the value of **betweenness**
  - **Betweennes** (edges) : number of **shortest paths** between every pair of nodes, that pass through an edge



Edge betweenness is higher for edges that connect different communities

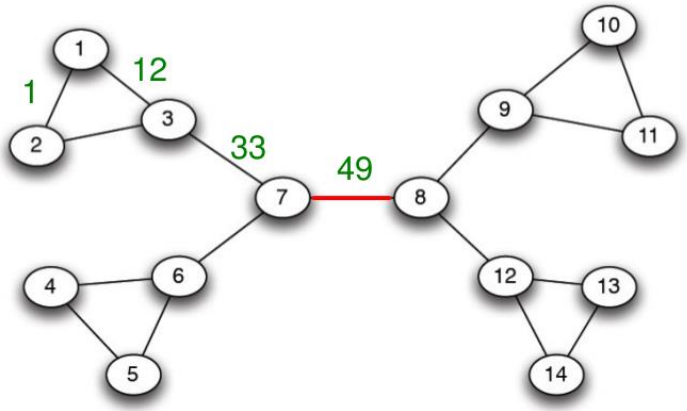
Hierarchical Method: Newman-Girvan algorithm

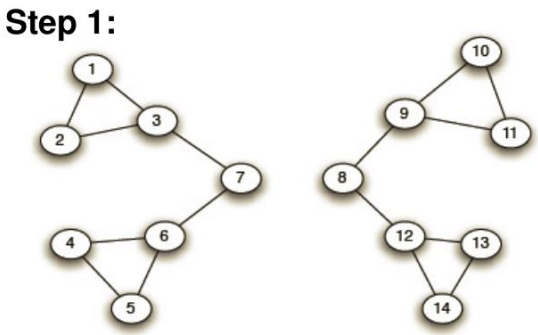
Algorithm

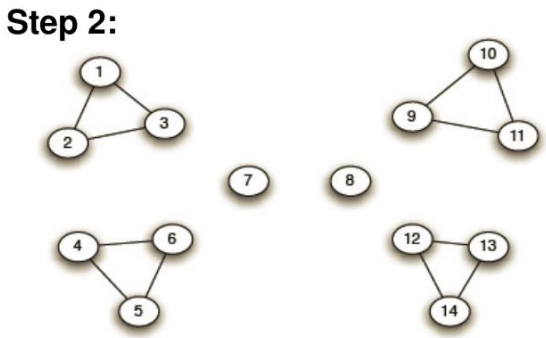
1. Compute betweenness (edge centrality) for all edges in the graph
2. Find and remove the edge with the highest score
3. Recalculate betweenness centrality score for the remaining edges
4. Go to step 2

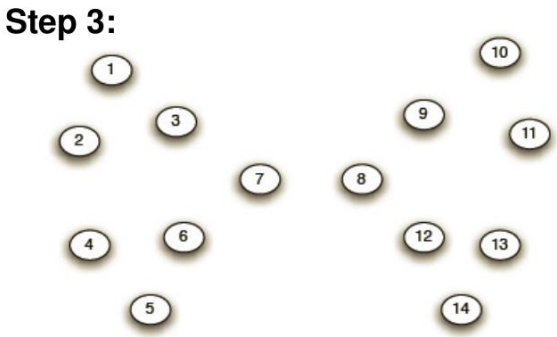
Stopping criteria → modularity

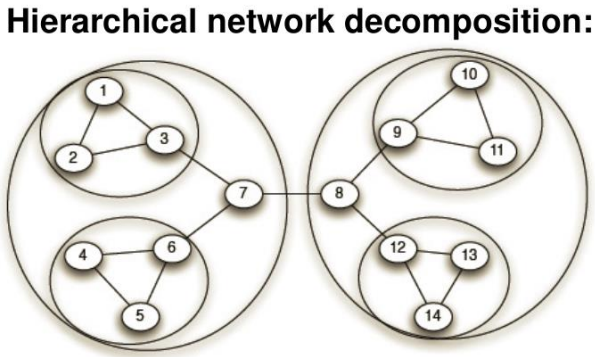
Complexity → $O(m^2n)$











Group-centric: Modularity Maximization

Modularity measures the group interactions compared with the expected random connections in the group *(the community have to be detected beforehand)*

$$Q = \frac{1}{2m}\sum_{i,j}(A_{ij} - \frac{k_i k_j}{2m})\delta(c_i, c_j)$$

Observed number of intra-community edges.

Expected number of edges between ***i*** and ***j***, if edges are placed randomly.

A_{ij}: adjacency matrix (value(i,j))

k_i: degree of node ***i***

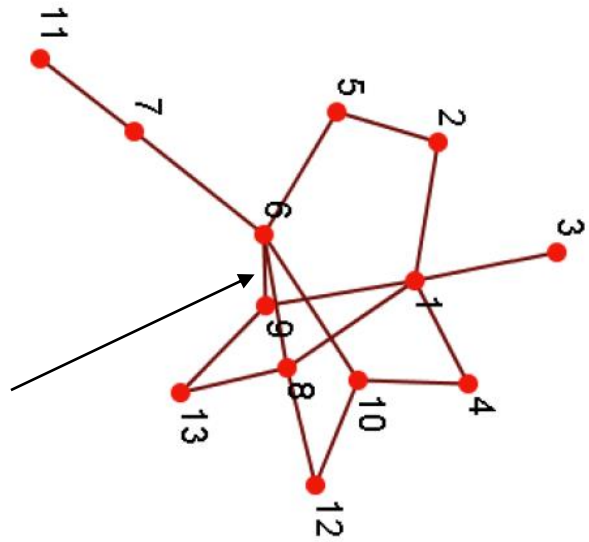
c_i: community of node ***i***

$\delta(c_i,c_j) = 1$ if ***i***, ***j*** belong to the same community

m: number of edges on the graph

In a Graph with *m* edges, for 2 nodes with degree *ki* and *kj* , the expected random connections probability between them are :

ki.kj / 2.m

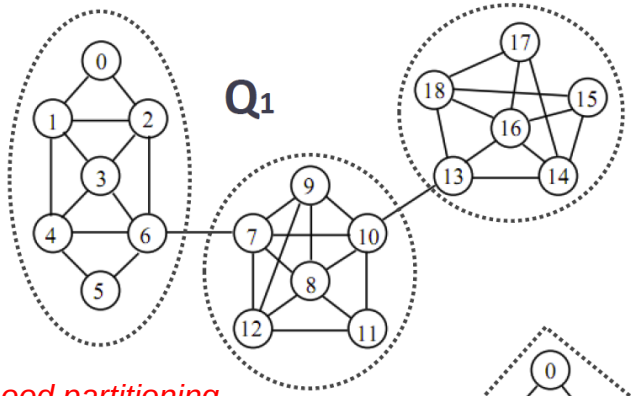


Expected probability of edge presence between 6 and 9 is 5x3/(2x17)

Modularity Maximization

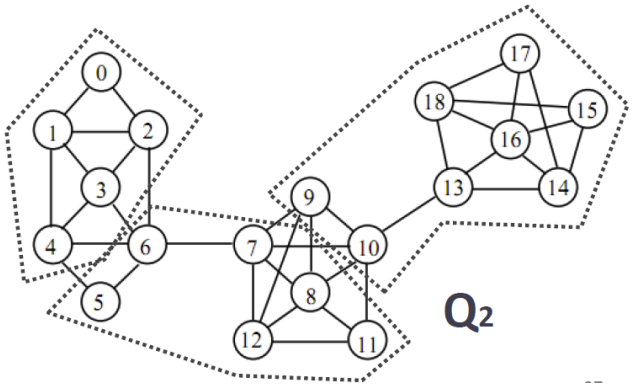
In a random graph (ER model), we expect that any possible partition would lead to ***Q = 0***.

Typically, in non-random graphs modularity takes values between 0.3 and 0.7.

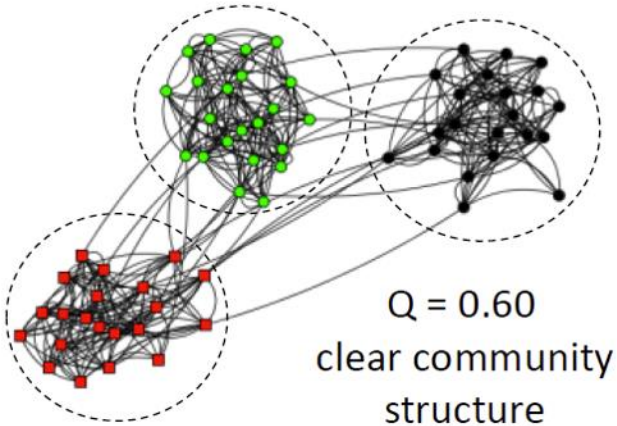


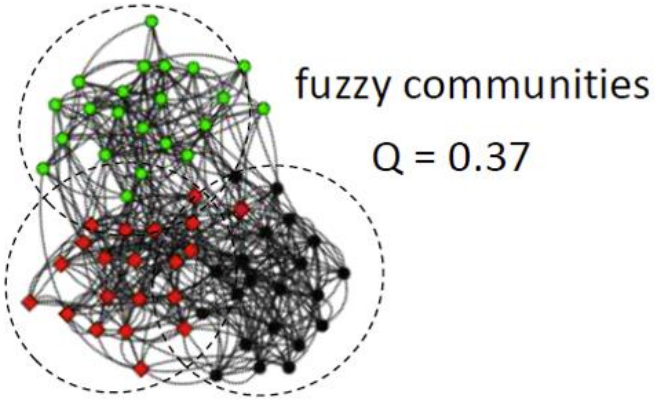
Good partitioning

Q1 > Q2



Not Good partitioning





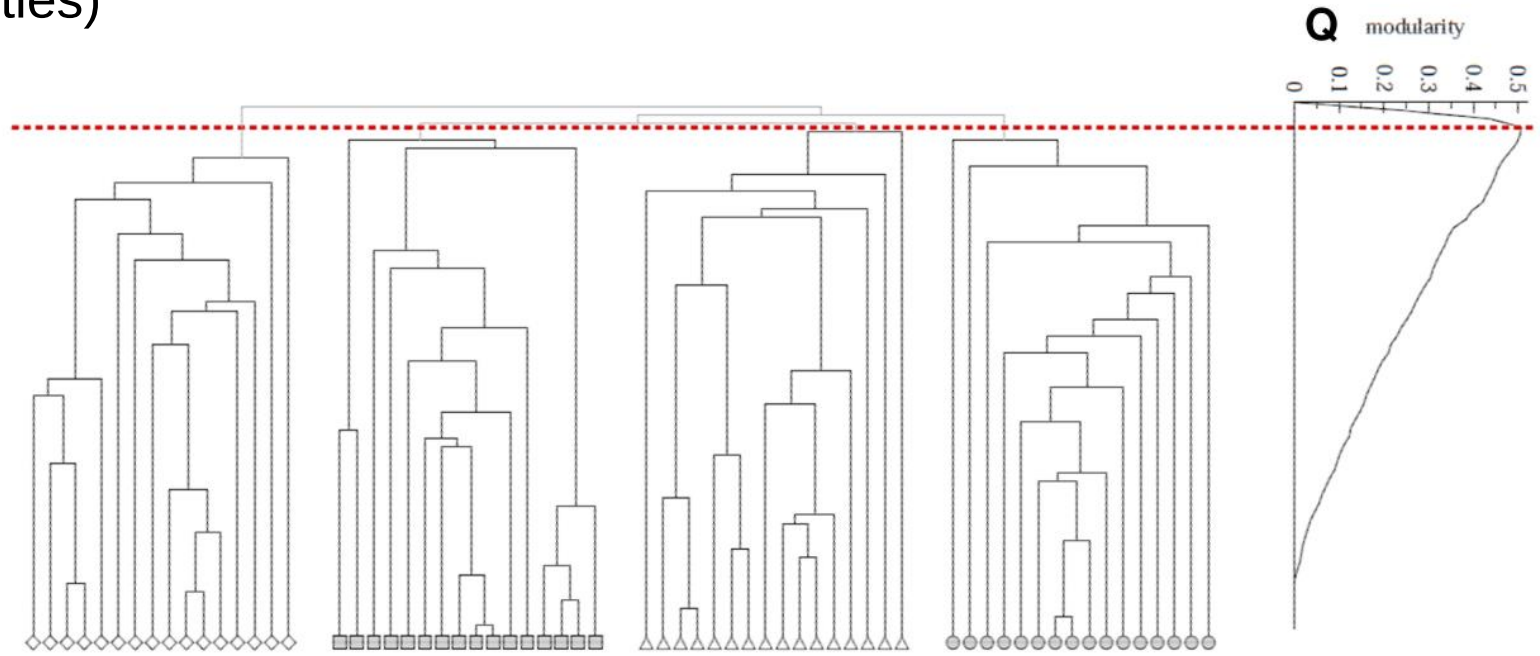
Modularity Maximization

Modularity measures the group interactions compared with the expected random connections in the group

To partition the graph into optimal communities, we should maximize the modularity

$$\boldsymbol{Max}\ \frac{1}{2m}\sum_{i,j}(A_{ij}-\frac{k_i k_j}{2m})\delta(c_i,c_j)$$

➔ The problem is the time complexity to find this maximum (without testing all the possibilities)



Louvain modularity algorithm

Before, the modularity is just used as a threshold (stop criteria)

- Issue(s) → Empirical value/threshold !

Better idea(s)?

→ Find the partition that corresponds to the maximum value of modularity

→ Modularity **optimisation** problem

But:

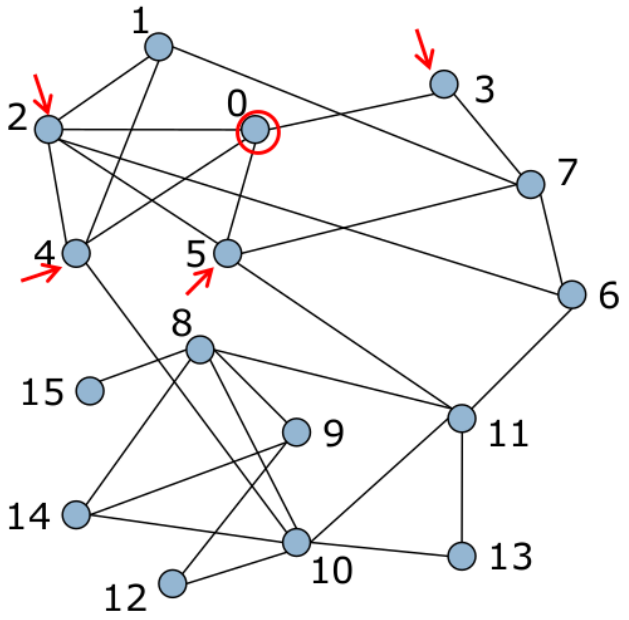
Modularity optimisation problem → NP-complex → Approximation techniques & heuristics

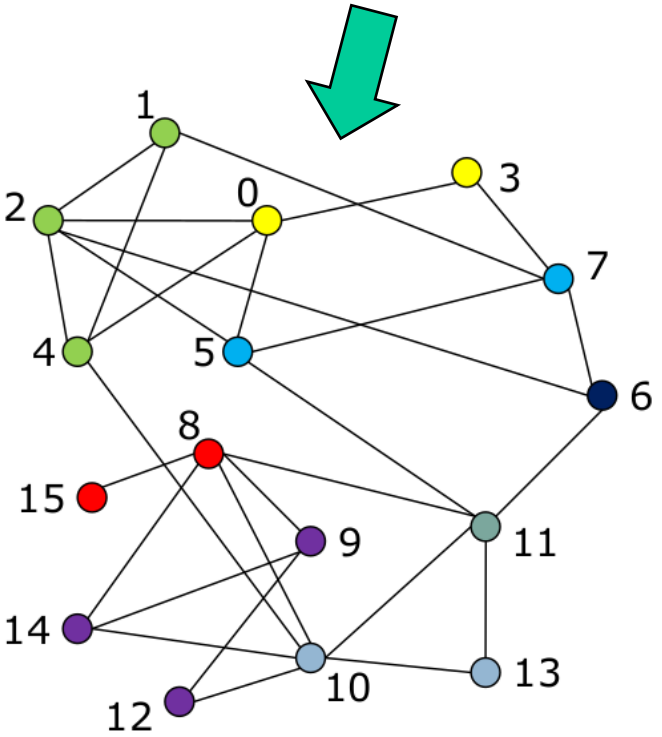
Louvain modularity algorithm

Louvain (2008) run in O(n.log n)

Algo:

1. Each node i is a cluster
2. For each node i, move it into the community of each neighbour j and compute the modularity change
3. Assign i to the neighbour j that yields the greatest modularity increase
4. Repeat until modularity local maximum is achieved ➔ *Level i*
5. Build new network by merging nodes from the same communities
6. Go to step 1





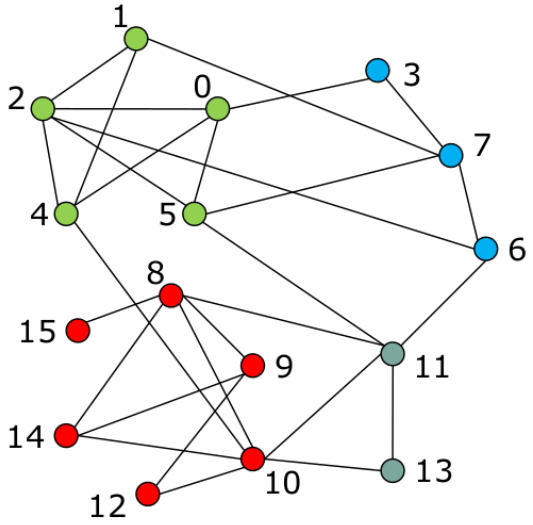
0 -> c[3]
1 -> c[4]
2 -> c[1,4]
3 -> c[0]
4 -> c[1]
5 -> c[7]
6 -> c[11]
7 -> c[5]
8 -> c[15]
9 -> c[12]
10 -> c[13]
11 -> c[10,13]
12 -> c[9]
13 -> c[10,11]
14 -> c[9,12]
15 -> c[8]

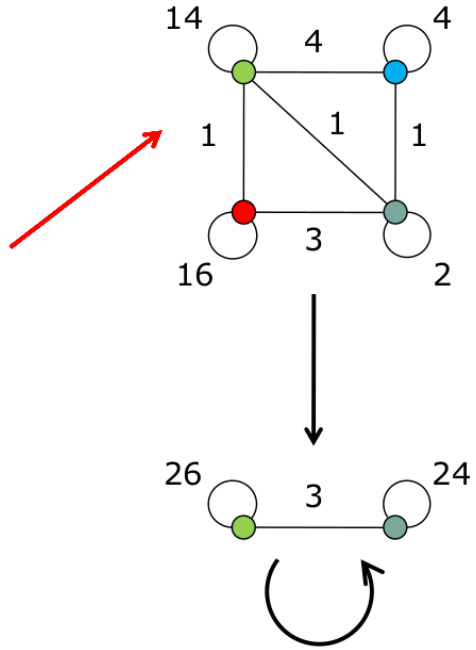
Louvain modularity algorithm

Louvain (2008) run in O(n log n)

Algo:

1. Each node i is a cluster
2. For each node i, move it into the community of each neighbour j and compute the modularity change
3. Assign i to the neighbour j that yields the greatest modularity increase
4. Repeat until modularity local maximum is achieved ➔ *Level i*
5. Build new network by merging nodes from the same communities
6. Go to step 1





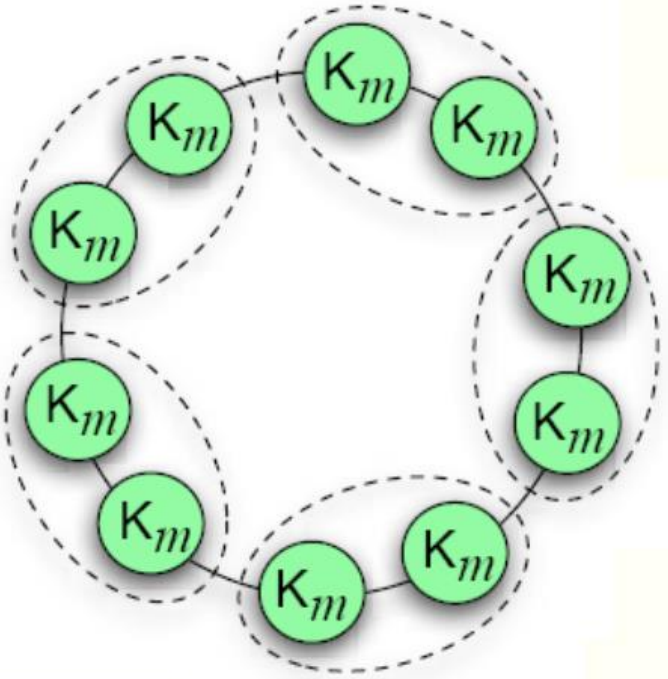
Louvain modularity algorithm

The "limit of resolution" of the modularity

Shown on a ring of n cliques with m nodes in each

For m=5 and n=30, the modularity is higher for the partition with 2 cliques merged together (Q=0.888) than the one with n groups (Q=0.876).

It shows that with this method some interesting communities could be missed

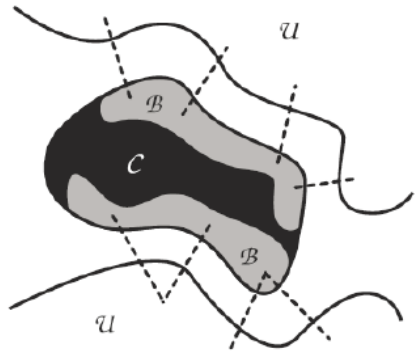


Local methods

More often, we cannot have access to the whole graph
➔ local methods are needed

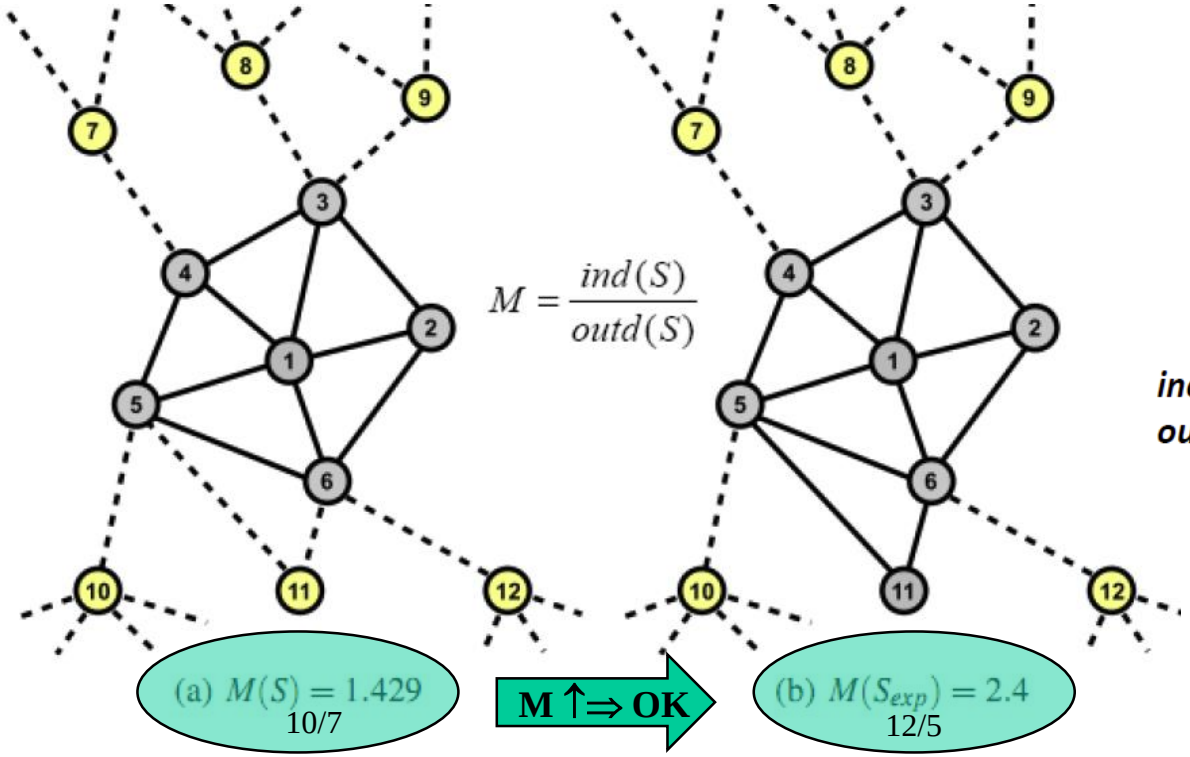
- Starts from a set of nodes (seeds)
- Expands the community boundaries using some criterion to stop (impossible to increase modularity)
- Local modularity or Subgraph modularity (M)

U: unexplored portion of graph



B: community boundary

C: "inside" part of community



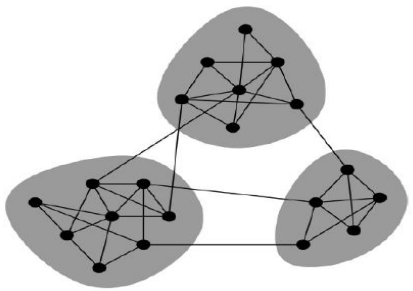
ind(S): in-degree of subgraph ***S***
outd(S): out-degree of subgraph ***S***

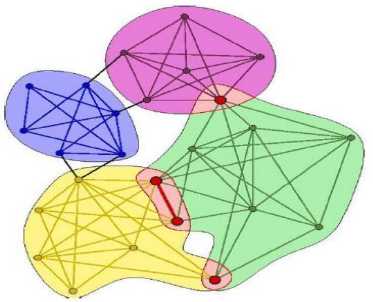
Challenges?

Challenges?

- Large graphs : complexity
- Known number of clusters?
- Directed graphs

- Overlapping clusters





Yang and Leskovec, SIAM AN '12

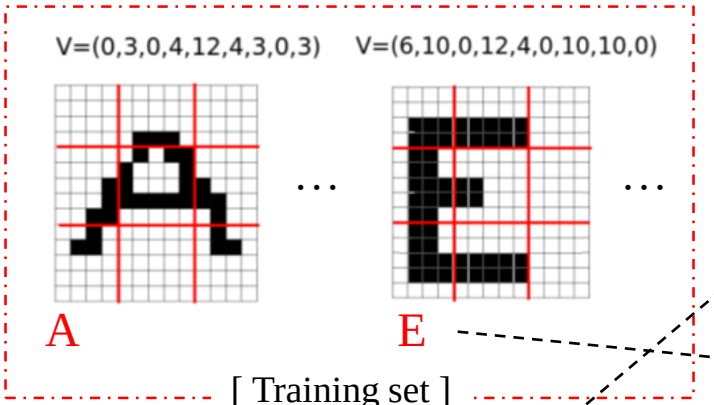
4. Graphs for Pattern recognition (several graphs)

A recall about PR mechanisms ?

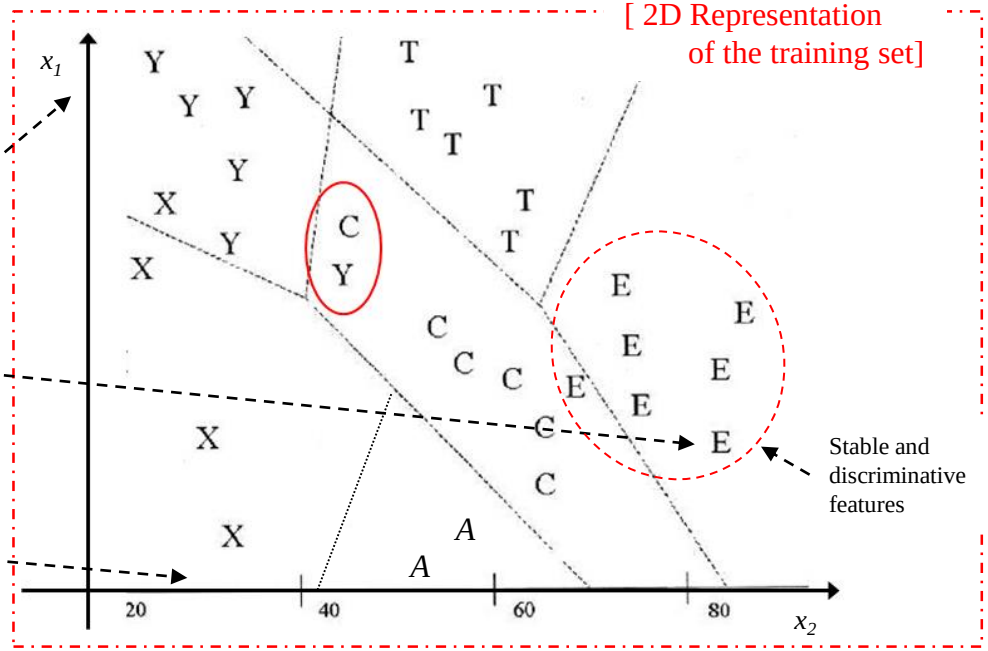
Pattern / Object recognition (toward Machine Learning)

How computers can recognize objects?

- We need a large set of (labelled) examples similar to the patterns to be recognized ➔ a training set
- We need a list of stable and discriminative features (shape, color, size,…) used to describe the patterns (labelled ones and unknown one)



1 EoC ➔ 1 Vecteur $\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ .. \\ x_n \end{bmatrix} \in IR^n$

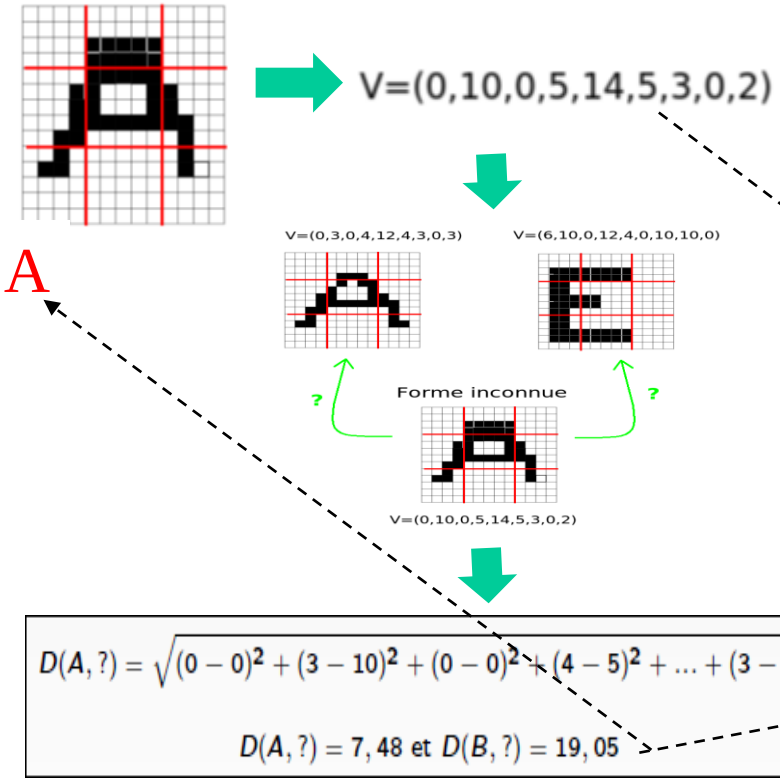


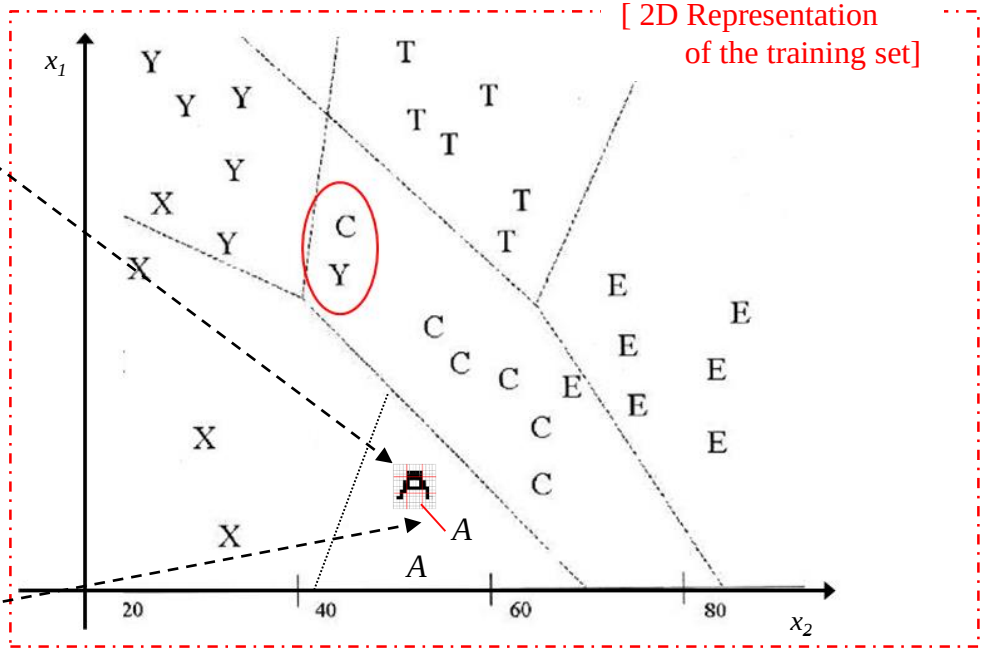
A recall about PR mechanisms

Pattern / Object recognition (toward Machine Learning)

How computers can recognize objects?

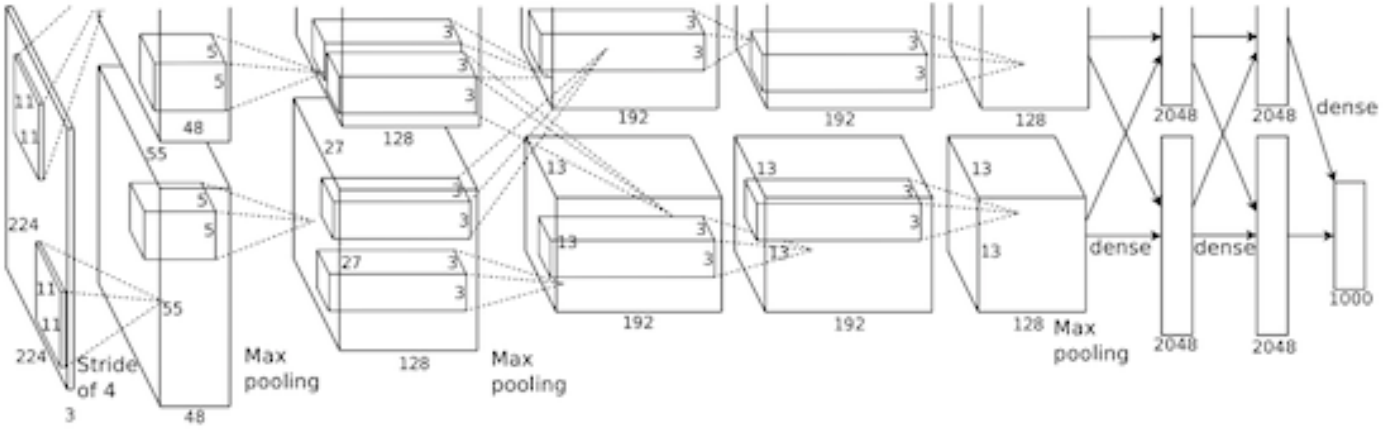
- When an unknown Object arrives, we compute its features and compare it with the content of the training set (associated built models)

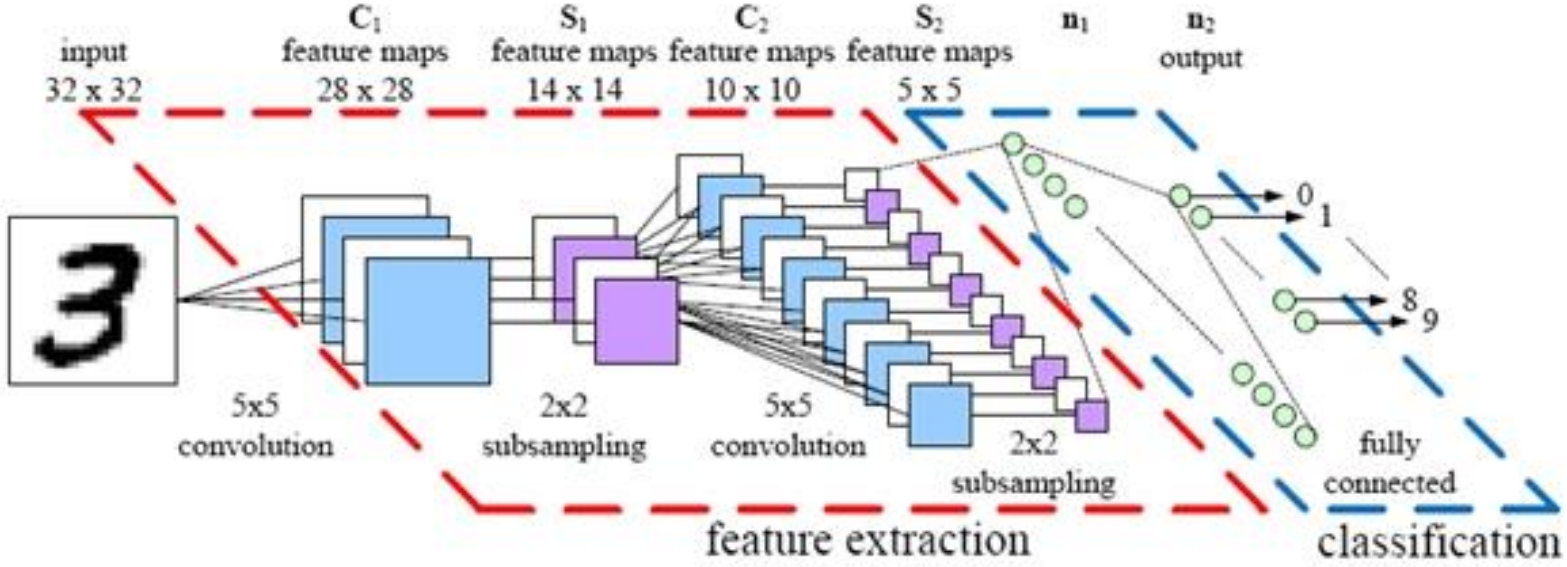




A recall about PR mechanisms

Deep Learning (Conv. Neural Net)





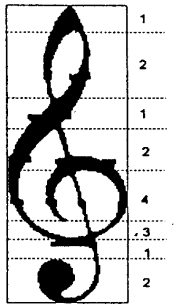
A recall about PR mechanisms

Many possible choices and techniques

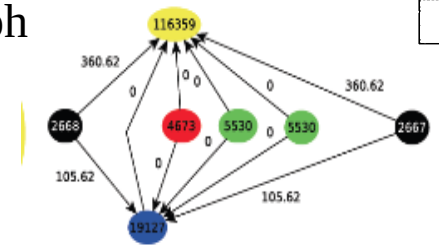
- For selection of discriminative features

1 Object ➔ 1 Vector

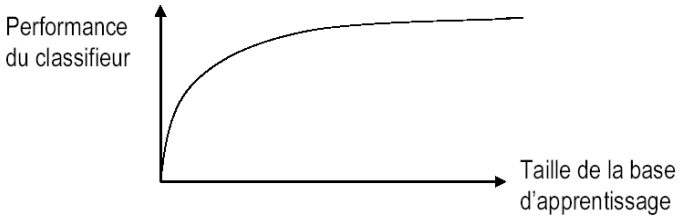
$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ .. \\ x_n \end{bmatrix} \in IR^n$$



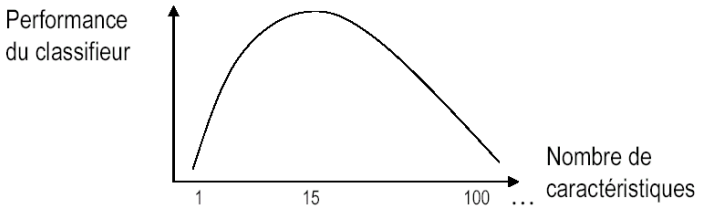
1 Object ➔ 1 Graph



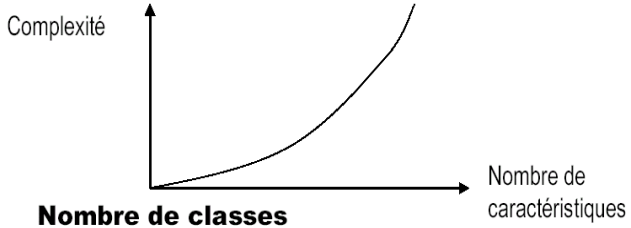
Statistique suffisante



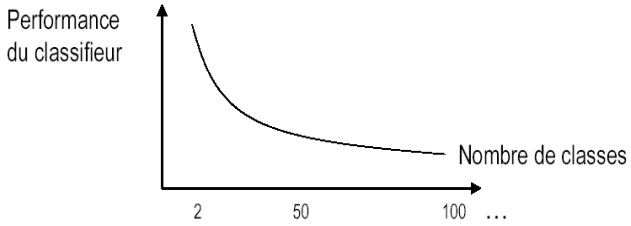
Malédiction de la dimensionalité



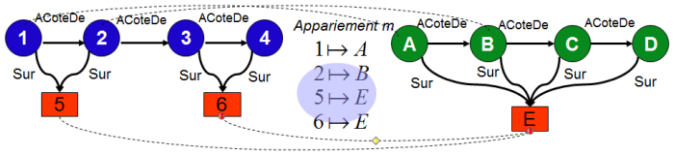
Complexité vs nombre de caractéristiques

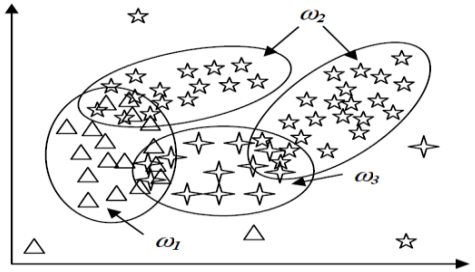


Nombre de classes



- Many Machine Learning models and tools



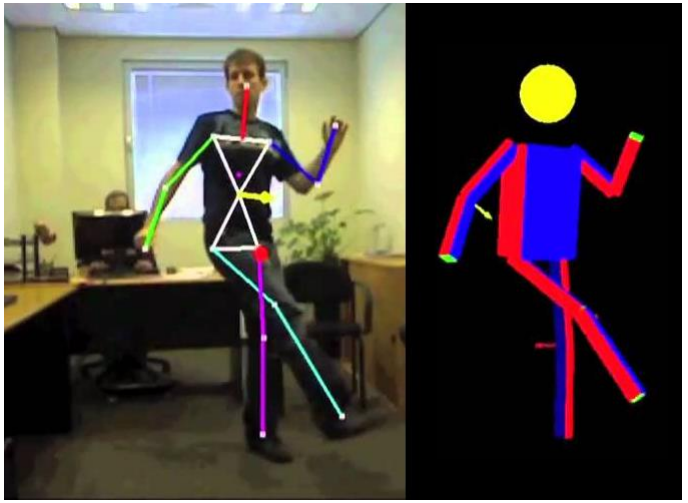


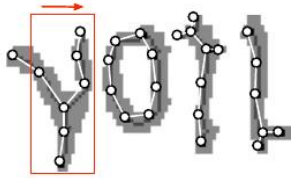
Why using graphs?

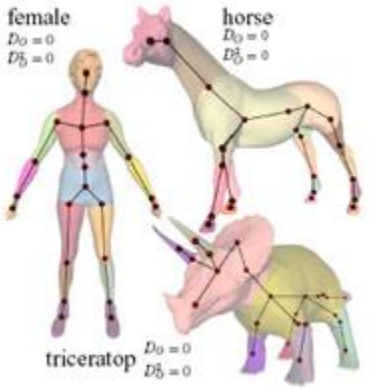
- **Statistical Methods**
  - Classes and frontiers
  - Existing statistical tools for evaluation of the quality of the chosen feature space
  - So many models and toolbox

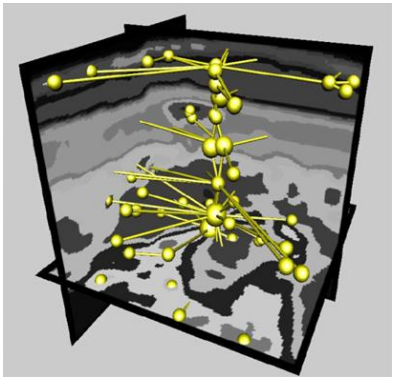
- **Structural Methods**
  - **Taking into account the context**
  - **A matching between sub-parts as results in addition to the decision**
  - **Partial or incremental recognition**
  - **Adaptive dimensionality of the models**
  - **Multimodal Features**

  - **Computational limitations?**
  - **Learning?**









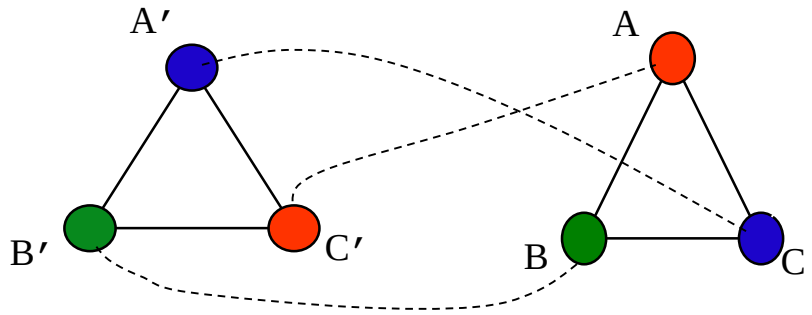
Structural PR → Graph Matching

- Definition (Matching)
  - A matching between G1 = (V1;E1) and G2 = (V2;E2)
  - = a relation $m \subseteq V1 \times V2$ $(u1; u2) \in m$
  - $\Rightarrow$ The vertex u1 is matched with the vertex u2

- Different types of matching
  - Bijective matching : cardinality = (1; 1)
  - Injective matching : cardinality = (1; 0..1)
  - Univoque matching : cardinality = (0..1; 0..1)
  - ...
  - Multivoque matching : cardinality = (0..|V2|; 0..|V1|) ← What does it mean?

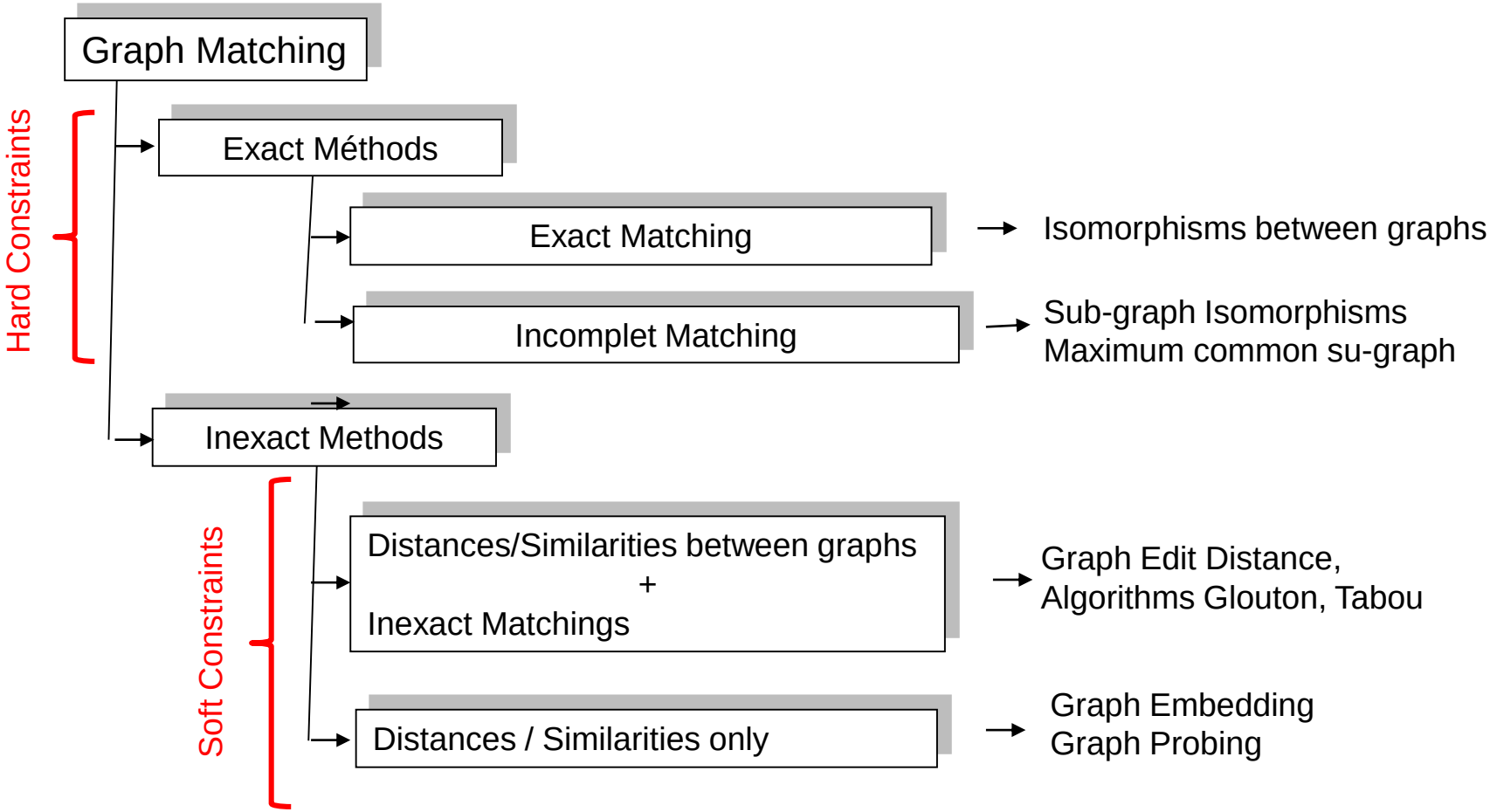
- High Complexity → Toward approximative methods !

[Solnon, 2007]



Structural PR ➔ Graph matching

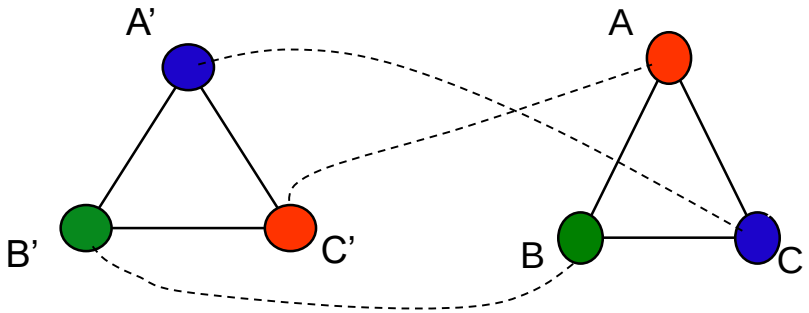
Taking care of the attributes in addition to the graph topology



Structural PR ➔ Graph Matching

Univoques Matching – Hard Constraints

Graph isomorphism problem



Model Graph G_M Test Graph G_D

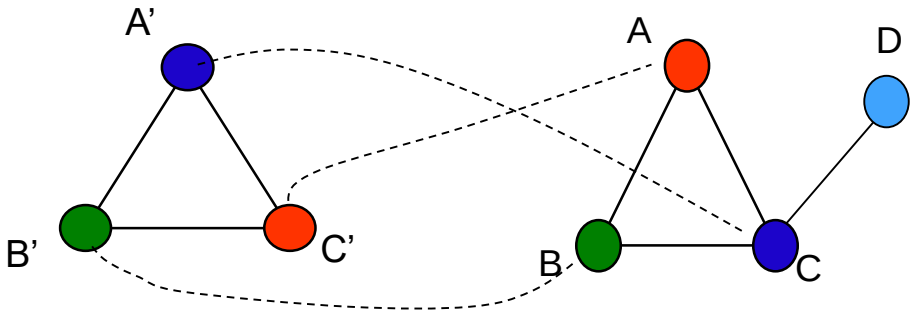
Objective

Bijective Matching
Hard Constraints
Possible on huge graphs

Problem

Not robust to noise and distorsions

Sub-gtaph isomorphism



Test Graph G_D Model Graph G_M

Objective

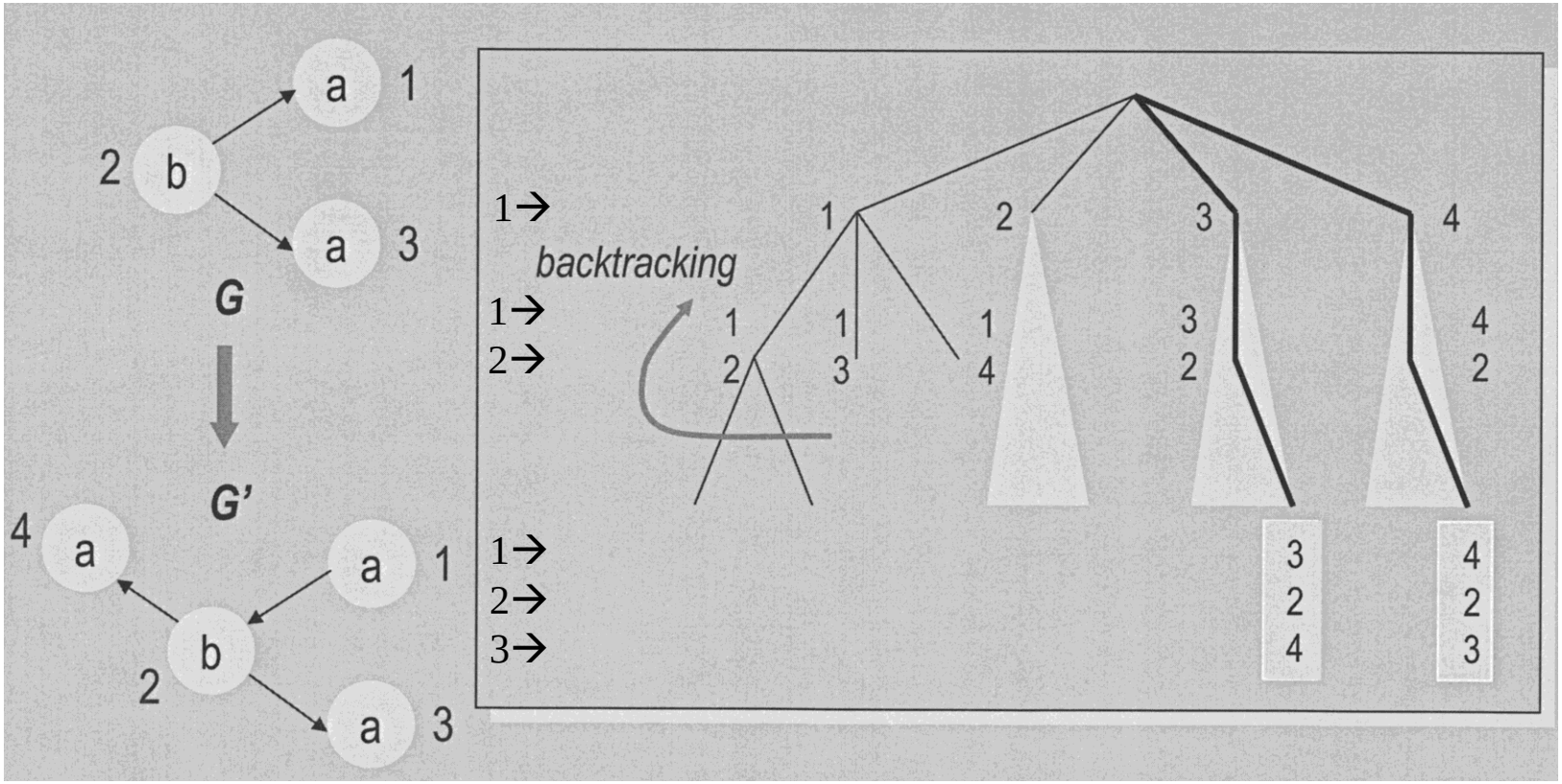
Injective Matching
Hard Constraints
NP-complete

Problem

Possible on medium size graphs

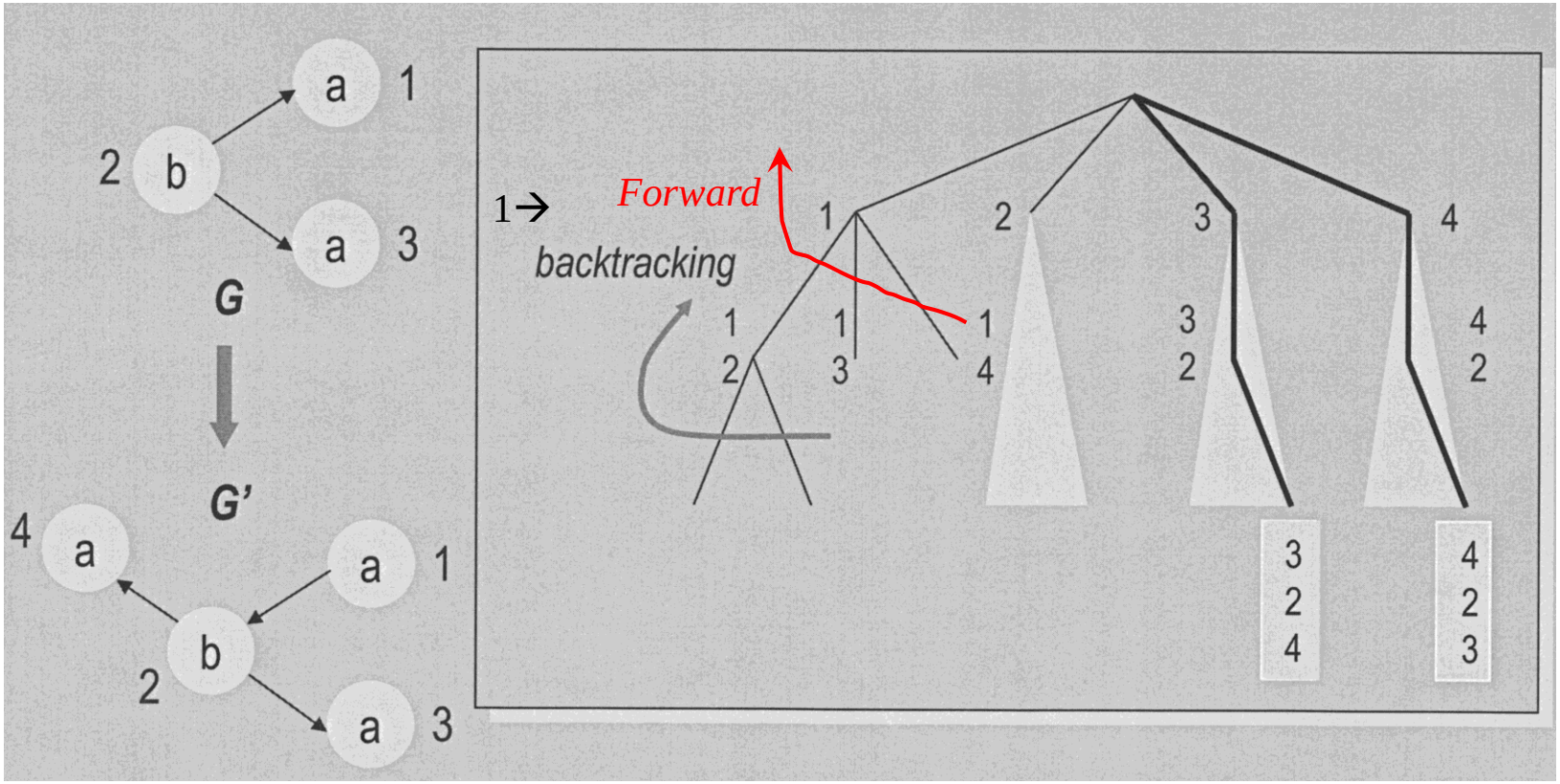
Structural PR ➔ Graph matching

Tree search Algorithms (with backtrack)



Structural PR ➔ Graph matching

Tree search Algorithms (with forward checking)

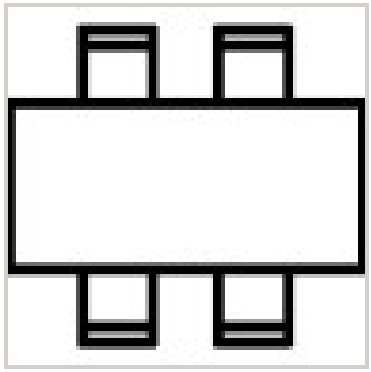


A look-ahead() checks before each association, the existence of a possible matching at the next step (using edge information for instance)

Structural PR ➔ Graph matching

Univoque Matching – Hard Constraints

TOO HARD…



ArchitecturalH.BMP

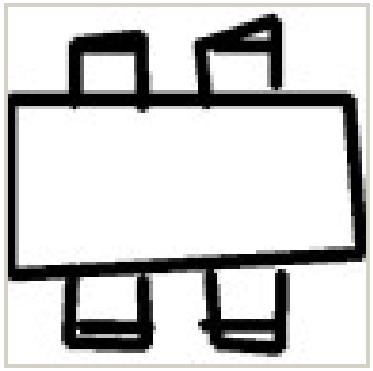
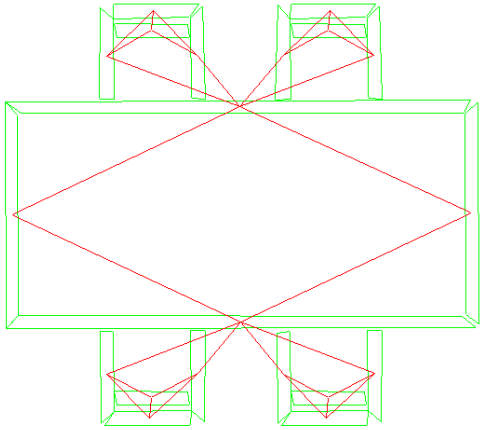
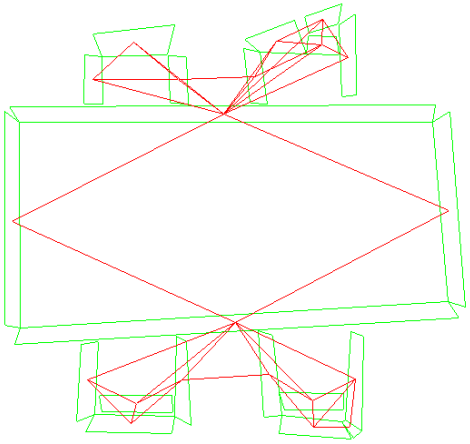


Image98.bmp

?





b. Inexact matching

(aka. Error-tolerant matching)

b. Inexact matching

- Optimal vs. suboptimal
  - Optimal
    - if it exists, the global minimum of the matching cost is given as the solution
  - Suboptimal (approximate)
    - Find a local minimum of the matching cost.
    - Might be not very far from the global one, but there are no guarantees.
    - Shorter, usually polynomial, matching time.

Structural PR ➔ Inexact Graph Matching

Univoque Matching – SOFT Constraints

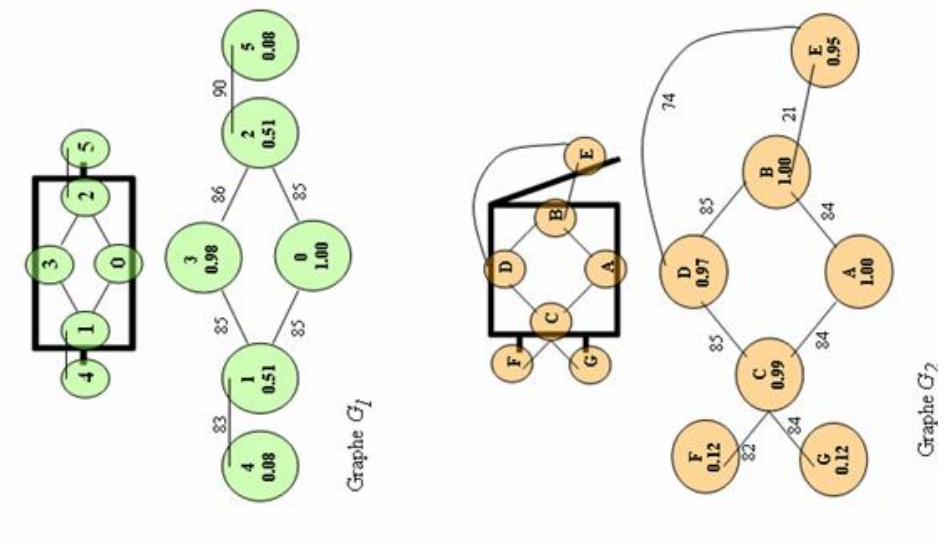
- Soft Constraints
  - Notion of similarity ≠ Exact matching
  - Similarity Matrix between nodes & edges
  - **Exploration of possibilities…**

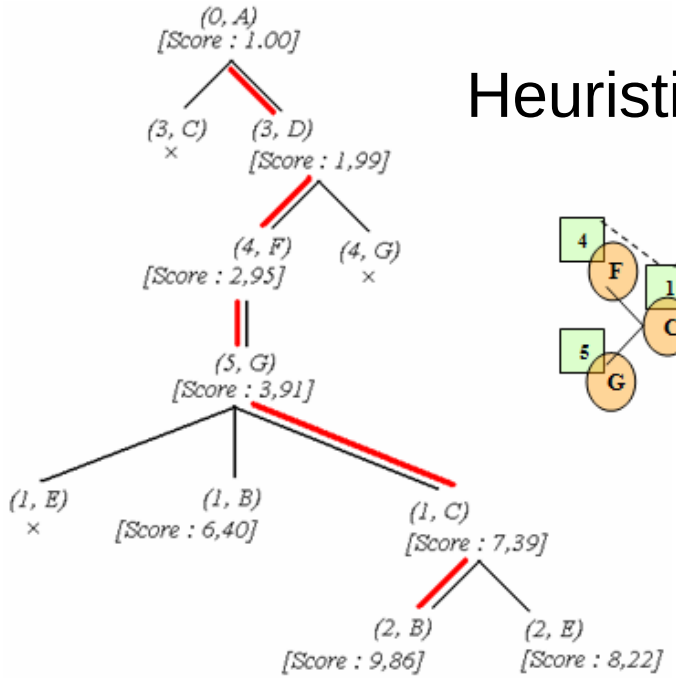
    Very time consuming…

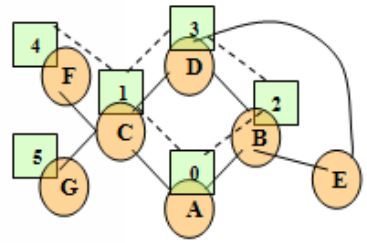
Etape-1 Etape-6 Etape-5 Etape-2 Etape-3 Etape-4

		A	B	C	D	E	F	G
		1,00	1,00	0,99	0,97	0,95	0,12	0,12
0	1,00	1,00	1,00	0,99	0,97	0,95	0,12	0,12
1	0,51	0,51	0,51	0,52	0,54	0,56	0,61	0,61
2	0,51	0,51	0,51	0,52	0,54	0,56	0,61	0,61
3	0,98	0,98	0,98	0,99	0,99	0,97	0,14	0,14
4	0,08	0,08	0,08	0,09	0,11	0,13	0,96	0,96
5	0,08	0,08	0,08	0,09	0,11	0,13	0,96	0,96

Heuristics ☹







b. Inexact matching

- Graph Edit Distance (GED) [Bunke,1999]

The minimum amount of distortion that is needed to transform G_1 into G_2

- Distortions s_i: deletions, insertions, substitutions of nodes and edges.
- Edit path $S = s_1, ..., s_n$: A sequence of edit operations that transforms G_1 into G_2.
- Cost functions: Measuring the strength of a given distortion.
- Edit distance $d(G_1, G_2)$: Minimum cost edit path between two graphs.

Problem of Edit Distance: NP complete

- Explore the space of all possible mappings of the nodes and edges of G_1 to the nodes and edges of G_2.
- Edit Distance computation also has a worst case exponential complexity which prevents its use in large datasets.

Structural PR ➔ Inexact Graph matching

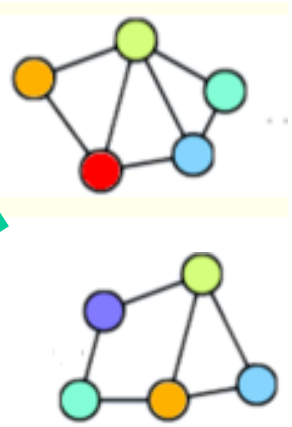
Univoque Matching – Soft constrainsts

Cost : associated to transformations (Insertion, suppression, substitution of edges and nodes)

Edit Path : set of needed transformations to obtain G2 from G1

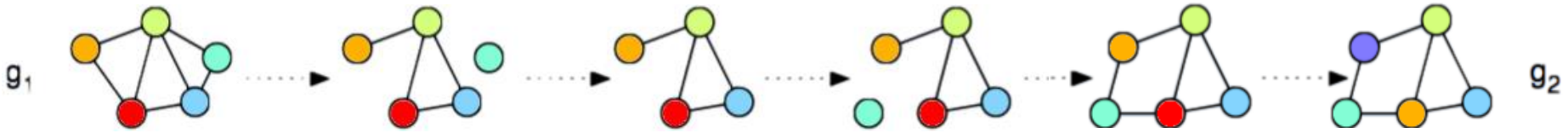
Global Error : Sum of all the elementary costs

Objective : Search for the minimal cost edit path



Let $G_1 = (V_1, E_1, L_{V_1}, L_{E_1}, \mu_1, \zeta_1)$ and $G_2 = (V_2, E_2, L_{V_2}, L_{E_2}, \mu_2, \zeta_2)$ be two graphs, the graph edit distance between G_1 and G_2 is defined as:

$$d_{plain}(g_m, g_t) = \min_{e_1,\cdots,e_k \in \gamma(g_1,g_2)} \sum_{i=1}^{k} c(e_i)$$



GM/GED as an optimization problem

- Assignment problem → Mathematical Programming → Hungarian algorithm, LSAP, QAP, …

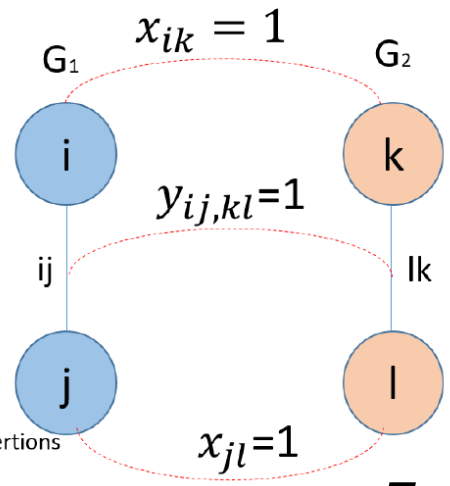
Edit operation	Variable	Cost
Substitution of vertex i by vertex k	$x_{i,k}$	$c_{i,k}$
Deletion of vertex i	u_i	$c_{i,\epsilon}$
Insertion of vetex k	v_k	$c_{\epsilon,k}$
Substitution of edge ij by edge kl	$y_{ij,kl}$	$c_{ij,kl}$
Deletion of edge ij	e_{ij}	$c_{ij,\epsilon}$
Insertion of edge kl	f_{kl}	$c_{\epsilon,kl}$

$$\mathbf{C} = \left[\begin{array}{cccc|cccc} c_{1,1} & c_{1,2} & \cdots & c_{1,m} & c_{1,\varepsilon} & \infty & \cdots & \infty \\ c_{2,1} & c_{2,2} & \cdots & c_{2,m} & \infty & c_{2,\varepsilon} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \ddots & \infty \\ c_{n,1} & c_{n,2} & \cdots & c_{n,m} & \infty & \cdots & \infty & c_{n,\varepsilon} \\ \hline c_{\varepsilon,1} & \infty & \cdots & \infty & 0 & 0 & \cdots & 0 \\ \infty & c_{\varepsilon,2} & \ddots & \vdots & 0 & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \infty & \vdots & \ddots & \ddots & 0 \\ \infty & \cdots & \infty & c_{\varepsilon,m} & 0 & \cdots & 0 & 0 \end{array}\right]$$

Figure credit: Riesen and Bunke IVC 2009

Objective function:

$$d(\mathbf{x},\mathbf{y},\mathbf{u},\mathbf{v},\mathbf{e},\mathbf{f}) = \Bigg(\sum_{i\in V_1}\sum_{k\in V_2} c(i,k)\cdot x_{i,k} \rightarrow \text{Vertex substitutions}$$
$$+ \sum_{ij\in E_1}\sum_{kl\in E_2} c(ij,kl)\cdot y_{ij,kl} \rightarrow \text{Edge substitutions}$$
$$+ \sum_{i\in V_1} c(i,\epsilon)\cdot u_i + \sum_{k\in V_2} c(\epsilon,k)\cdot v_k \rightarrow \text{Vertex insertions/deletions}$$
$$+ \sum_{ij\in E_1} c(ij,\epsilon)\cdot e_{ij} + \sum_{kl\in E_2} c(\epsilon,kl)\cdot f_{kl} \Bigg) \rightarrow \text{Edge deletions/insertions}$$



From [HDR Raveaux2019]

GM → ILP (F1 formulation)

Vertices mapping constraints

deletion substitutions

$$u_i + \sum_{k \in V_2} x_{i,k} = 1 \quad \forall i \in V_1$$

Insertion

$$v_k + \sum_{i \in V_1} x_{i,k} = 1 \quad \forall k \in V_2$$

Edges mapping constraints

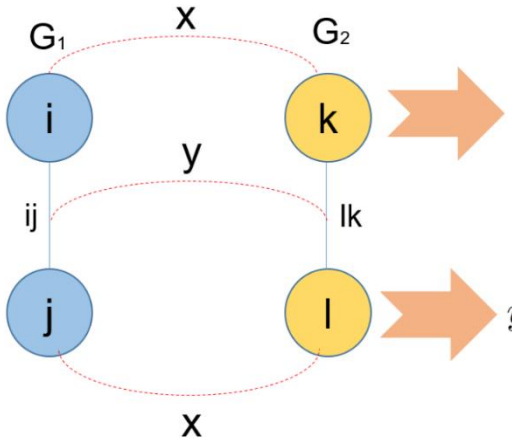
deletion substitutions

$$e_{ij} + \sum_{kl \in E_2} y_{ij,kl} = 1 \quad \forall ij \in E_1$$

$$f_{kl} + \sum_{ij \in E_1} y_{ij,kl} = 1 \quad \forall kl \in E_2$$

Insertion

If $y_{ij,kl} = 1$ then $x_{i,k}$ must be set to 1



$$y_{ij,kl} \leq x_{i,k} \quad \forall (ij,kl) \in E_1 \times E_2$$

$$y_{ij,kl} \leq x_{j,l} \quad \forall (ij,kl) \in E_1 \times E_2$$

F1

$$\min_{\mathbf{x},\mathbf{y},\mathbf{u},\mathbf{v},\mathbf{e},\mathbf{f}} \quad d(\mathbf{x},\mathbf{y},\mathbf{u},\mathbf{v},\mathbf{e},\mathbf{f})$$

subject to

$$u_i + \sum_{k \in V_2} x_{i,k} = 1 \quad \forall i \in V_1$$

$$v_k + \sum_{i \in V_1} x_{i,k} = 1 \quad \forall k \in V_2$$

$$e_{ij} + \sum_{kl \in E_2} y_{ij,kl} = 1 \quad \forall ij \in E_1$$

$$f_{kl} + \sum_{ij \in E_1} y_{ij,kl} = 1 \quad \forall kl \in E_2$$

$$y_{ij,kl} \leq x_{i,k} \quad \forall (ij,kl) \in E_1 \times E_2$$

$$y_{ij,kl} \leq x_{j,l} \quad \forall (ij,kl) \in E_1 \times E_2$$

with

$$x_{i,k} \in \{0,1\} \quad \forall (i,k) \in V_1 \times V_2$$

$$y_{ij,kl} \in \{0,1\} \quad \forall (ij,kl) \in E_1 \times E_2$$

$$u_i \in \{0,1\} \quad \forall i \in V_1$$

$$v_k \in \{0,1\} \quad \forall k \in V_2$$

$$e_{ij} \in \{0,1\} \quad \forall ij \in E_1$$

$$f_{kl} \in \{0,1\} \quad \forall kl \in E_2$$

From [HDR Raveaux2019]

Structural PR ➔ Inexact Graph Matching

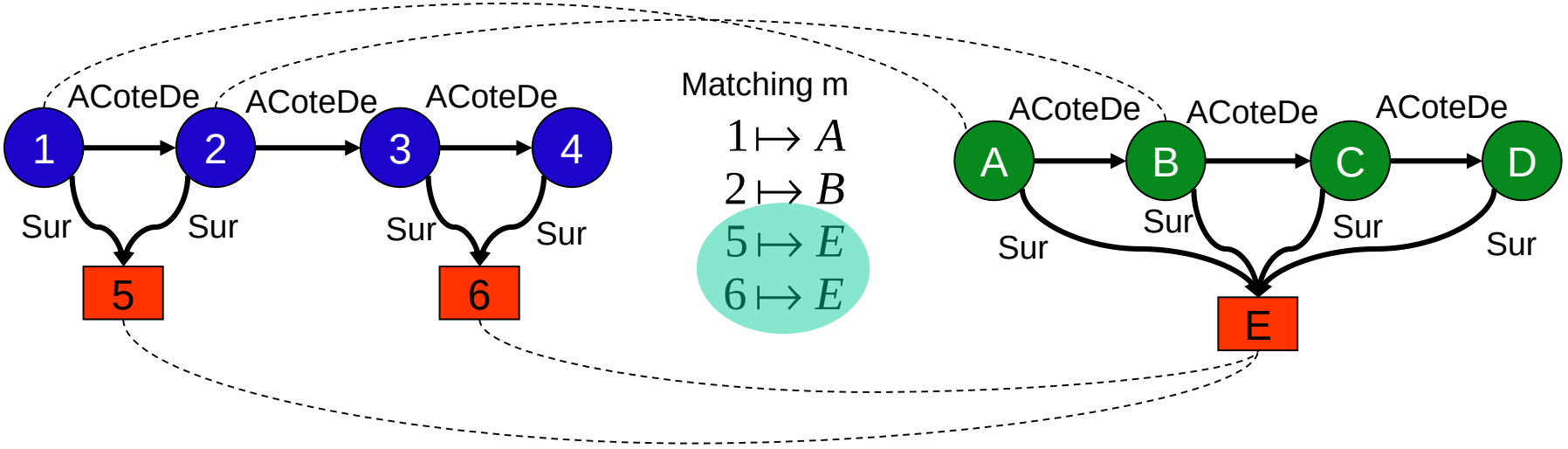
Multivoque Matching – Soft constrainsts

Evoluted version of the Graph Edit distance

Univoque Matching ➔ each node of G1 can be matched with only one node of G2

New version of GED with additional possible transformations

MERGE & SPLIT → Multivoque Matching …



[Champin / Solnon] (2003-2005)

Structural PR ➔ Inexact Graph Matching

Multivoque Matching – Soft constrainsts

Problem : Definition of the similarity measure and edit costs [Qureshi03]

$$Sc_{Mp} = \left[\sum_{i=1}^{m} (1 - \Delta V_i) + \sum_{j=1}^{n} (1 - \Delta E_j) - \left(\sum_{i=1}^{k} \omega_i + \sum_{j=1}^{\ell} \omega'_j \right) \right]$$

Node to Node similarity

Edge to Edge similarity

Penalties for multiple matchings (splits)

Matching Exploration → a very combinatory problem !

Goal = Finding $m \subseteq V_1 \times V_2$ maximising

$score(m) = f(G_1 \cap_m G_2) - g(splits(m))$

Problem NP-difficile → $2^{|V1|.|V2|}$ combinaisons

Résolution by a complete search ?

Structuring the search space with lattices...

...but the score function is not monotonous….

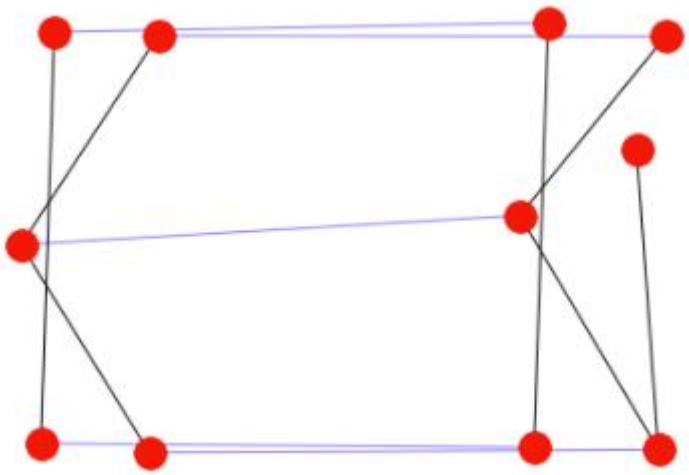
Limited to very small graphs (10 nodes)

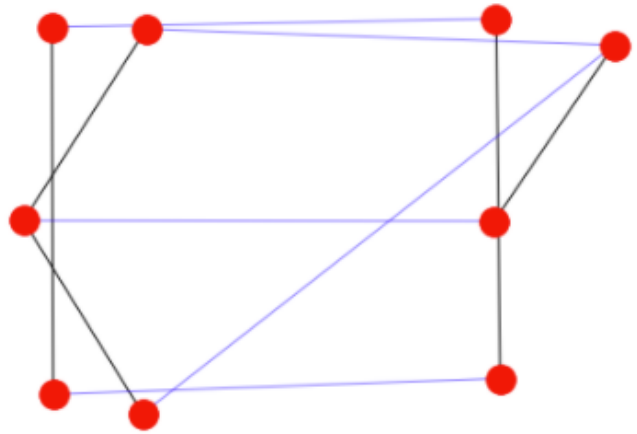
Using heuristics approaches (not exact)

Exemple on Letters

Graph Matching with GED ?

Cost Function ?





Structural PR ➔ ML ➔ Graph Comparison

Graph Probing and Embedding

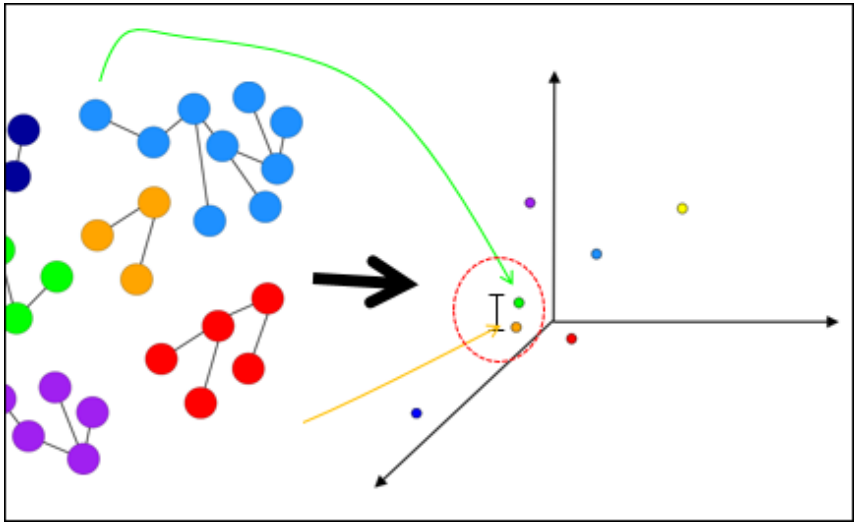
From graph space back to Vector space…

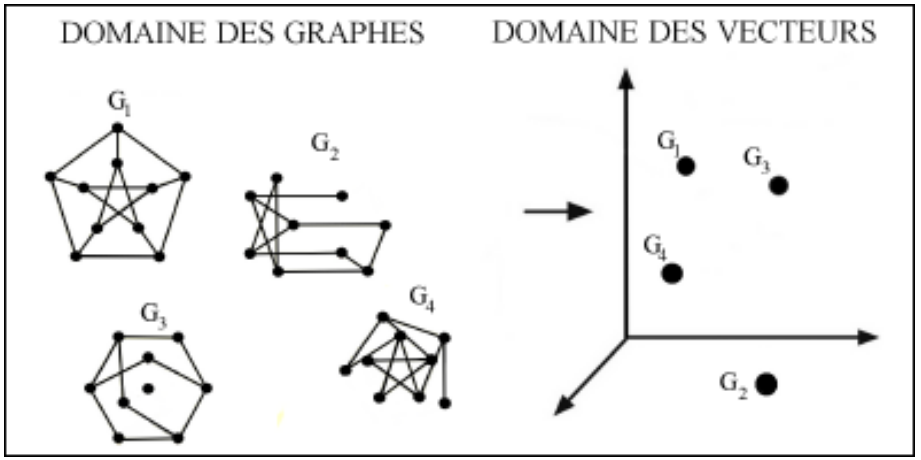
➔The node to node matching is lost !

Information extraction by feature selection → Construction of a feature vector:

$$\varphi : G \rightarrow R^n \quad => \quad \varphi(g) = (x_1, \ldots, x_n)$$

Combination of structural and statistical approaches



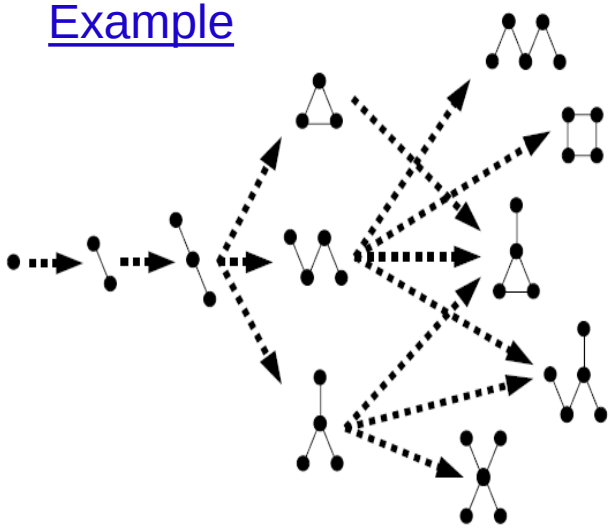


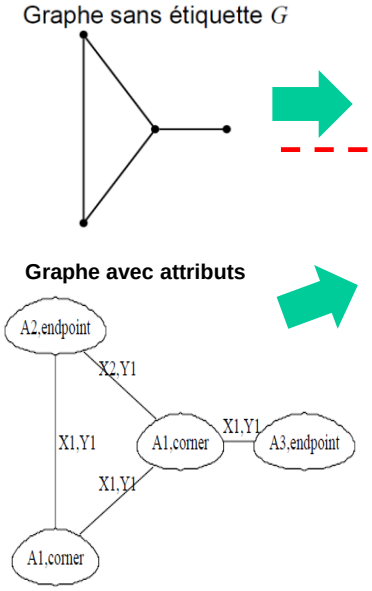
What does it mean?

Structural PR ➔ Graph Comparison

Embedding topological information [Sidère09]

Example





Pattern	•	•–•	•–•–•	•–•–•–•		
Freq.	4	4	5	2	1	1
A1, corner	2	4	9	4	2	2
A2, corner	0	0	0	0	0	0
A3, corner	0	0	0	0	0	0
A1, endpoint	0	0	0	0	0	0
A2, endpoint	1	2	4	2	1	1
A3, endpoint	1	1	2	2	0	1
X1, Y1	0	3	7	5	2	2
X1, Y2	0	0	0	0	0	0
X2, Y1	0	1	3	1	1	1
X2, Y2	0	0	0	0	0	0

Patterns enumeration

$\varphi(g) = (x_1, \ldots, x_n) = (4,4,5,2,1,1)$

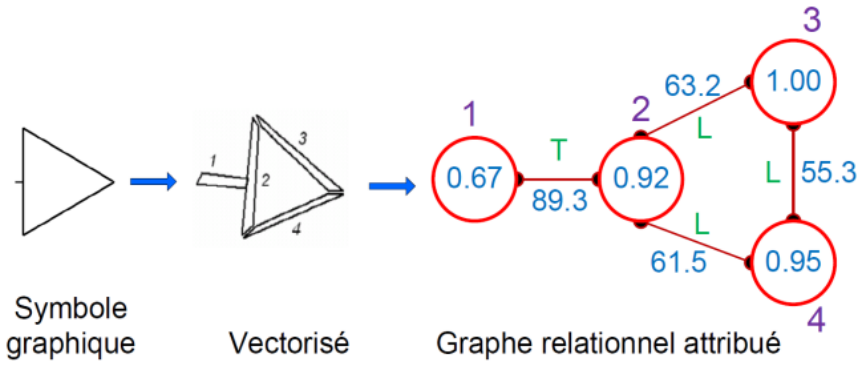
- Lexicon of Topological patterns
- Frequency of the patterns ➔ Construction of a vector
- Many possible extensions : Graphlet, Treelet, …

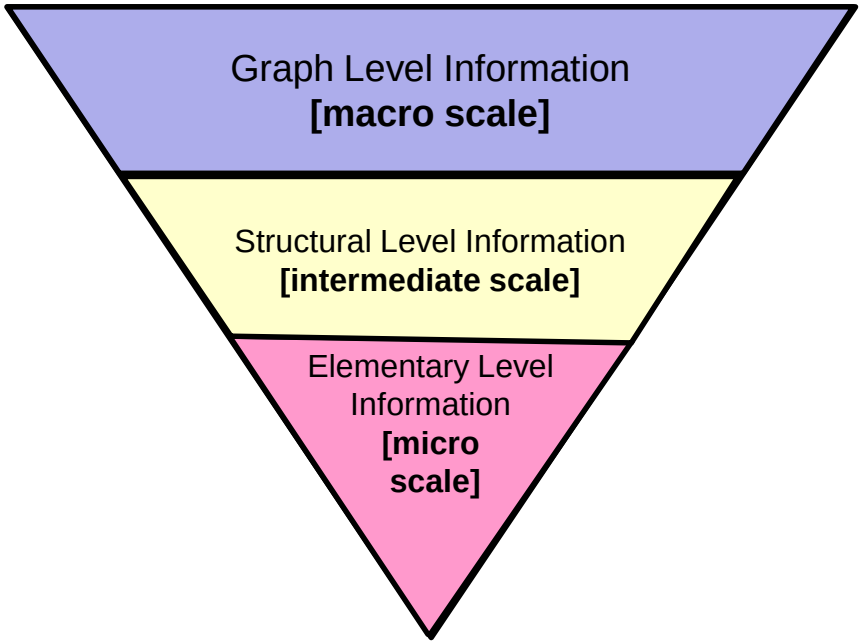
Trying to take care of attributes ➔ Construction of a Matrix ➔ discrétisation

Structural PR ➔ Graph Comparison

Fuzzy multi-level Graph Embedding [Luqman13]

Trying to embbed topological and statistical information





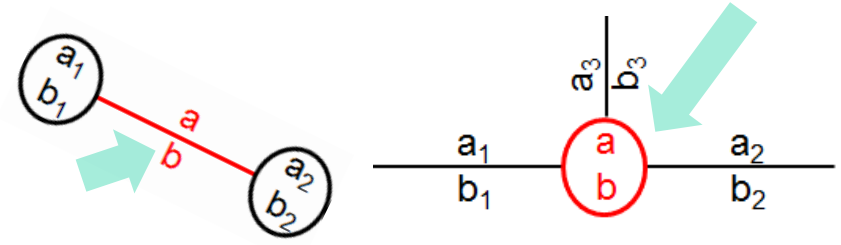
$$\varphi(g) = (x_1, \ldots, x_n)$$

Graph order	Graph size	Fuzzy histogram of node degrees	Fuzzy histograms of numeric resemblance attributes	Crisp histograms of symbolic resemblance attributes	Fuzzy histograms of numeric node attributes	Crisp histograms of symbolic node attributes	Fuzzy histograms of numeric edge attributes	Crisp histograms of symbolic edge attributes

Structural PR ➔ Graph Comparison

Fuzzy multi-level Graph Embedding [Luqman13]

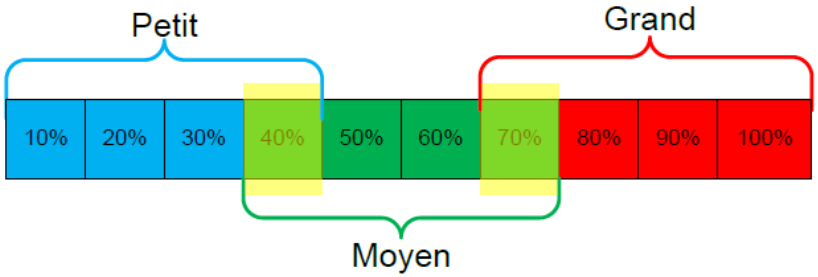
Adding local topology information
Ressemblance Degree on node attributes
Ressemblance Degree on edge attributes



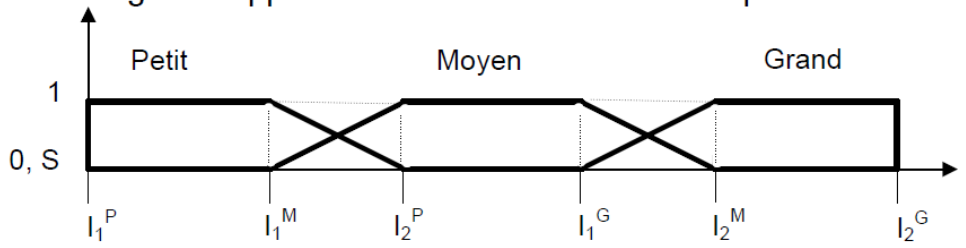
$$numeric\ resemblance = \frac{\min(|a_1|,|a_2|)}{\max(|a_1|,|a_2|)}$$

$$symbolic\ resemblance = \begin{cases} 1 & if\ b_1 = b_2 \\ 0 & otherwise \end{cases}$$

Fuzzy embedding
Frequency Histogram
Fuzzy transformation



Dégree d'appartenance = contribution d'une primitive



$\varphi(g) = (x_1, \ldots, x_n)$

Graph order	Graph size	Fuzzy histogram of node degrees	Fuzzy histograms of numeric resemblance attributes	Crisp histograms of symbolic resemblance attributes	Fuzzy histograms of numeric node attributes	Crisp histograms of symbolic node attributes	Fuzzy histograms of numeric edge attributes	Crisp histograms of symbolic edge attributes

➔ **Notion of node signature**

Graph Comparison ➔ Graph Kernel

A kernel

Let define a kernel $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ between two objects x and x' corresponds to a scalar product between two projections $\phi(x)$ and $\phi(x')$ in a Hilbert space $\mathcal{H}$.

$$\forall (x, x') \in \mathcal{X} \times \mathcal{X}, \quad k(x, x') =< \phi(x), \phi(x') >$$

Scalar product

In order to define a valid kernel, it is not necessary to explicitly define the projection function $\phi : \mathcal{X} \rightarrow \mathcal{H}$. However, the kernel k must verify certain properties:

Definition 27. *(Positive-definite kernel)*
A positive-definite kernel on $\mathcal{X} \times \mathcal{X}$ is a function $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$:

$$k(x, x') = k(x', x)$$

and semi-definite positive:

$$\{x_1, \cdots, x_M\} \in \mathcal{X}^M, c \in \mathbb{R}^M, \sum_{i=1}^{M} \sum_{j=1}^{M} c_i k(x_i, x_j) c_j \geq 0$$

Element i,j of K (the Gramm Matrix of the kernel)

Definition 28. *(Gram matrix)*
A Gram matrix $K \in \mathbb{R}^{M \times M}$ associated to a kernel k on a finite set $X = \{x_1, \cdots, x_M\}$

$$K_{i,j} = k(x_i, x_j), (i, j) \in \{1, ..., M\}^2$$

Matrix corresponding to the Scalar product after projection

If k is a positive-definite kernel then the Gram matrix K is semi-definite positive. The reverse is also true.

Graph Comparison ➔ Graph Kernel

Kernel trick

Definition

Kernel Trick

- Let $\vec{x}$ and $\vec{y}$ be two vectors in $\mathbb{R}^n$
- Let $\varphi(\vec{x})$ and $\varphi(\vec{y})$ be two functions projecting $\vec{x}$ and $\vec{y}$ into $\mathbb{R}^m$.
- with $n \leq m$
- $k(\vec{x}, \vec{y}) = \langle \varphi(\vec{x}), \varphi(\vec{y}) \rangle$

Scalar product here

φ can be defined as we want

- An explicit representation for φ is not required.
- It suffices to know that $\mathbb{R}^m$ is an inner product space

Graph Comparison ➔ Graph Kernel

Kernel trick

Example:

- Let $\vec{x}$ and $\vec{y}$ be two vectors in $\mathbb{R}^2$
- Let $\vec{x} = (x_1, x_2)$ and $\vec{y} = (y_1, y_2)$
- Let $\varphi(\vec{x})$ and $\varphi(\vec{y})$ be two functions projecting $\vec{x}$ and $\vec{y}$ into $\mathbb{R}^3$.

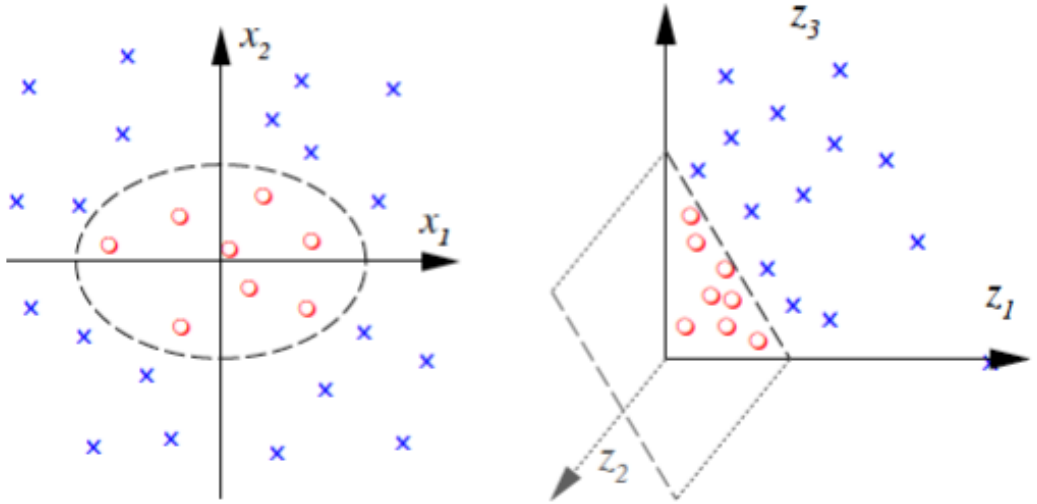
(in R^2)
- $k(\vec{x}, \vec{y}) = \langle \vec{x}, \vec{y} \rangle^2$ (Scalar product in R^2)^2
- $k(\vec{x}, \vec{y}) = x_1^2 y_1^2 + 2x_1 y_1 x_2 y_2 + y_2^2 y_2^2$

(in R^3)
- $k(\vec{x}, \vec{y}) = \langle (x_1^2, \sqrt{2}x_1 x_2, x_2^2), (y_1^2, \sqrt{2}y_1 y_2, y_2^2) \rangle$
- $k(\vec{x}, \vec{y}) = \langle \varphi(\vec{x}), \varphi(\vec{y}) \rangle$ (Scalar product in R^3)
- $\varphi(\vec{x}) = (x_1^2, \sqrt{2}x_1 x_2, x_2^2)$

φ can be defined as we want

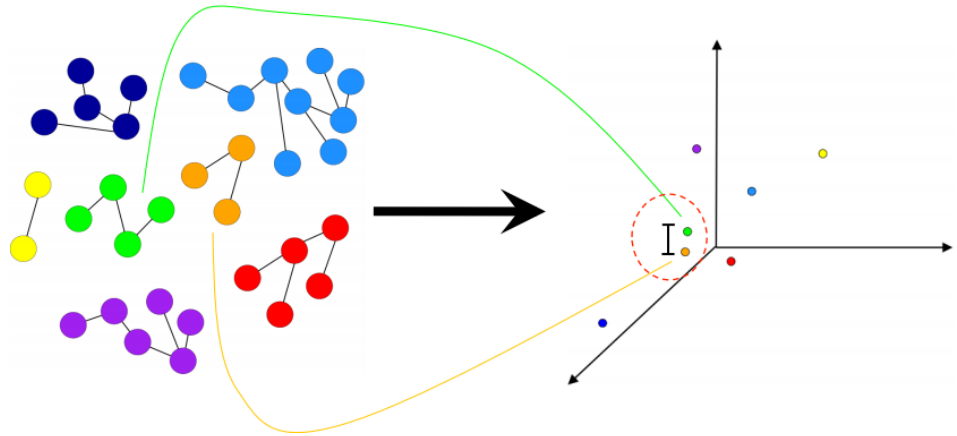
(Scalar product in R^3) = (Scalar product in R^2)^2

$$\Phi : R^2 \to R^3$$
$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{(2)}x_1 x_2, x_2^2)$$

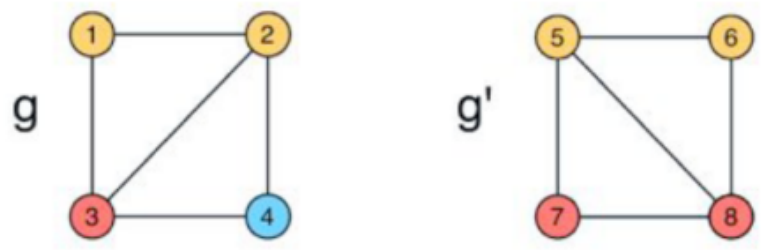


Graph Embedding with Graph Kernel

A very simple graph kernel [Bunke2009]



Example



- $\varphi(g) = \frac{(2,1,1)}{4}$
- $\varphi(g') = \frac{(2,2,0)}{4}$
- $\kappa(g, g') = \frac{4+2}{16} = \frac{6}{16}$

Node color enumeration

Scalar product

Graph Embedding

$\varphi : G \rightarrow R^n \quad \varphi(g) = (x_1, \ldots, x_n)'$

Many information can be extracted :

→nodes, cliques, paths, walk, …

An other question for ML with graphs…

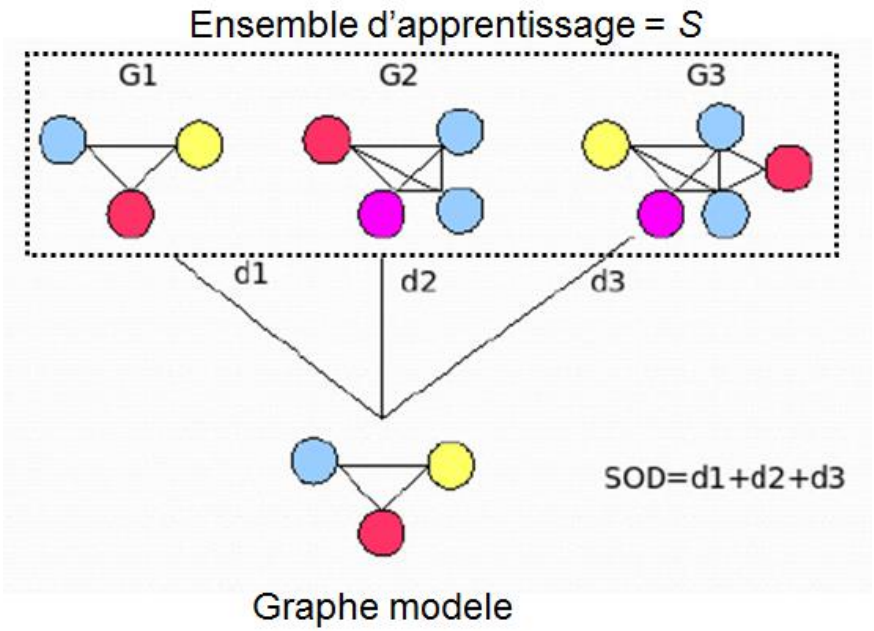
Class model definition and prototypes?

Median Graph

$$\overline{g} = arg_{g \in C} \, min \, SOD\,(g, S)$$

$$SOD\,(g, S) = \sum_{i=1}^{|S|} d\,(g, g_i)$$

S = a set of graphs
C = set of possible graphs derived from S
d = an edit distance



Remaining Problems…

How to define GED Costs?

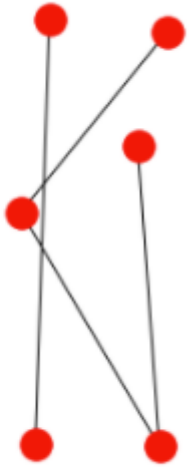
How to define good embedding functions?

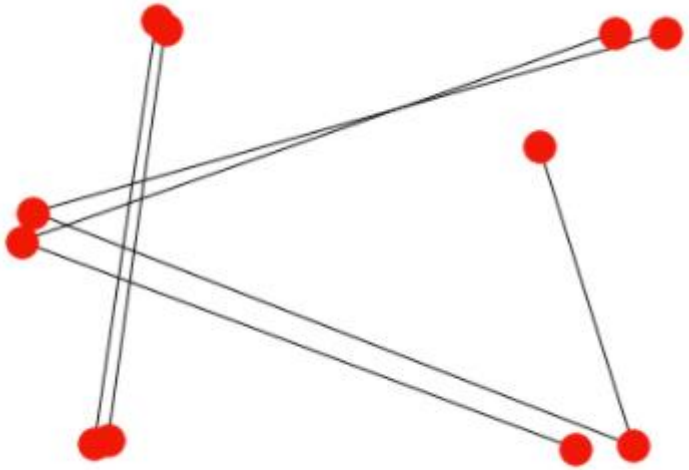
How to get the Graph Matching at the end (not only the decision)

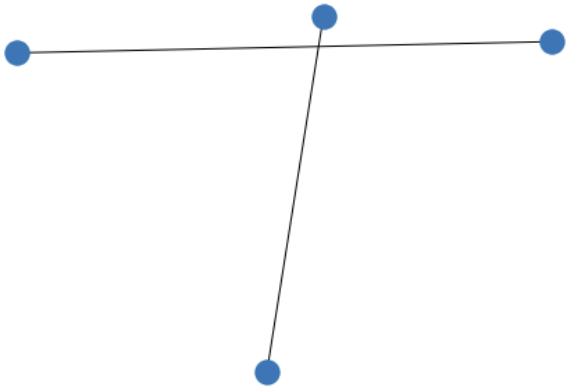
➔ **Learning to match Graphs is the actual crutial question…**

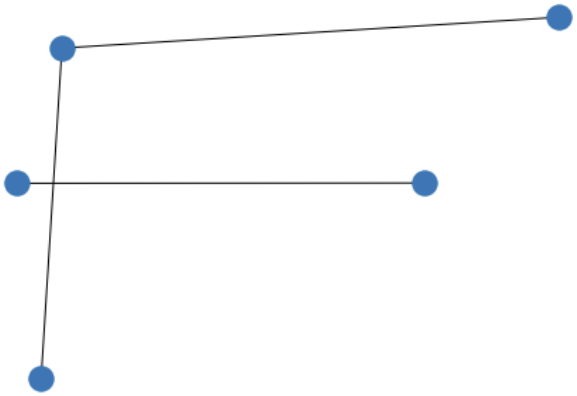
Exemple on Letters

Graph Embeddings









Graph Neural Networks

How to do Deep Learning on Graphs ?

Taxonomy:

- Graph/node embedding
  - Explicit embedding
    - Through feature extraction
      - End-to-end learning : **Here are the GNN**
  - Implicit embedding
- Graph space

- Input: A graph
- Output: Node embeddings
- Assumptions: stationarity and compositionality
- The goal:
  - Graph Neural Networks (GNN) perform an end-to-end learning including feature extraction and classification.

Graph Neural Networks

Classical NN

Let $x \in \mathbb{R}^{1 \times m}$ be a vector considered as an input data.

$$H^{(l+1)} = f(H^{(l)})$$

$$H^{(l+1)} = \sigma(H^{(l)} W^{(l)}) \quad \forall l > 0$$

$W^{(l)} \in \mathbb{R}^{m_l \times m_{l+1}}$ is a matrix of trainable parameters. m_l is the number of neurons of the layer l. For the layer 0, $H^{(0)} = x$. Layer $l+1$ produces a vector $H^{(l+1)} \in \mathbb{R}^{1 \times m_{l+1}}$. Finally, σ is a non linear function. This neural network is considered as a model where parameters can be learned. This model is also denoted as a *"dense" layer* or *"Fully Connected (FC)" layer* or a *"MuLtilayer Perceptron"* (MLP). The question is how to generalize this artificial neural networks to graphs? What to do when the input is a graph?

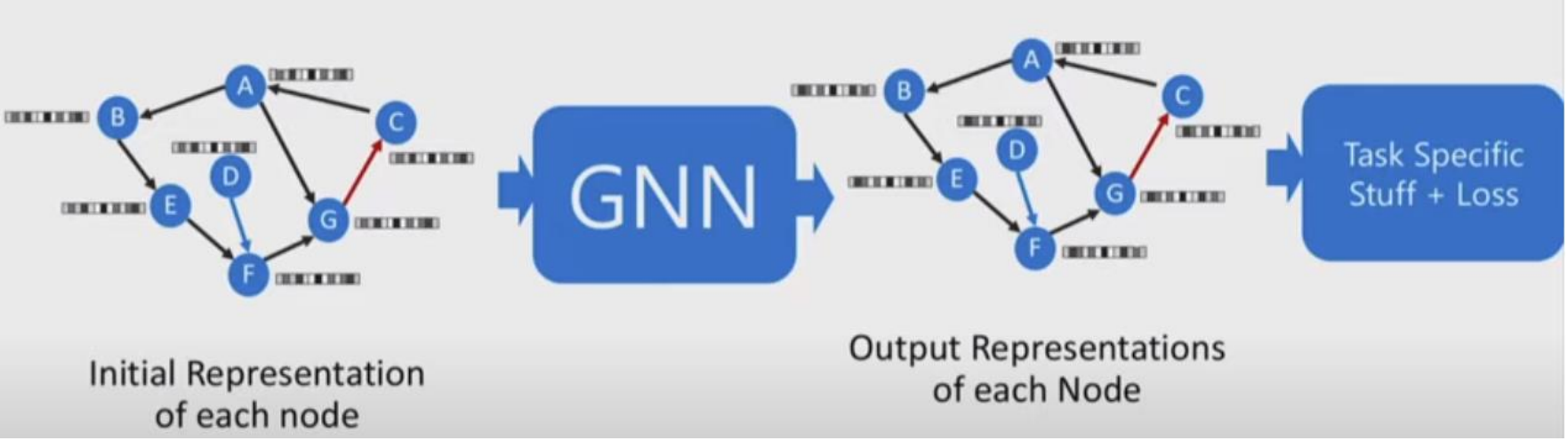
Graph NN

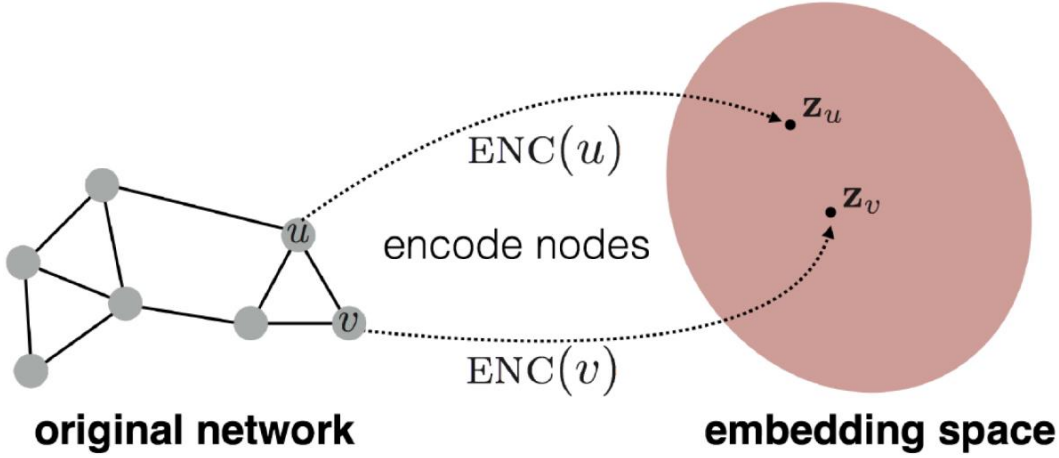
Instead of x $\in$ $\mathbb{R}^{1xm}$, we have G with

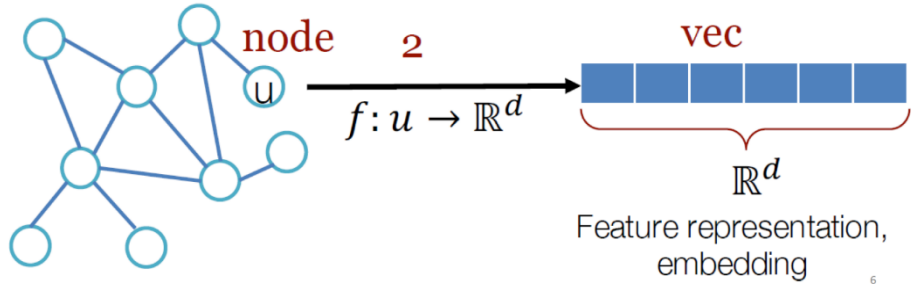
- V is the vertex set.
- E is the edge set.
- A is the adjacency matrix (assume binary). $A \in \{0,1\}^{|V| \times |V|}$
- F $\in \mathbb{R}^{|V| \times m}$ is a matrix of node features.
  - Categorical attributes, text, image data
  - Node degrees, clustering coefficients, etc.
  - Indicator vectors (i.e., one-hot encoding of each node)

Graph Neural Networks

- GNN as node encoder / decoder







Figures coming from : An Introduction to Graph Neural Networks: Models and Applications - Microsoft https://www.youtube.com/watch?v=zCEYiCxrL_0

Two Key Components

Encoder maps each node to a low-dimensional vector.

$$\text{ENC}(v) = \mathbf{z}_v$$

d-dimensional embedding

node in the input graph

Similarity function specifies how relationships in vector space map to relationships in the original network.

$$\text{similarity}(u, v) \approx \mathbf{z}_v^\top \mathbf{z}_u$$

Similarity of u and v in the original network

dot product between node embeddings

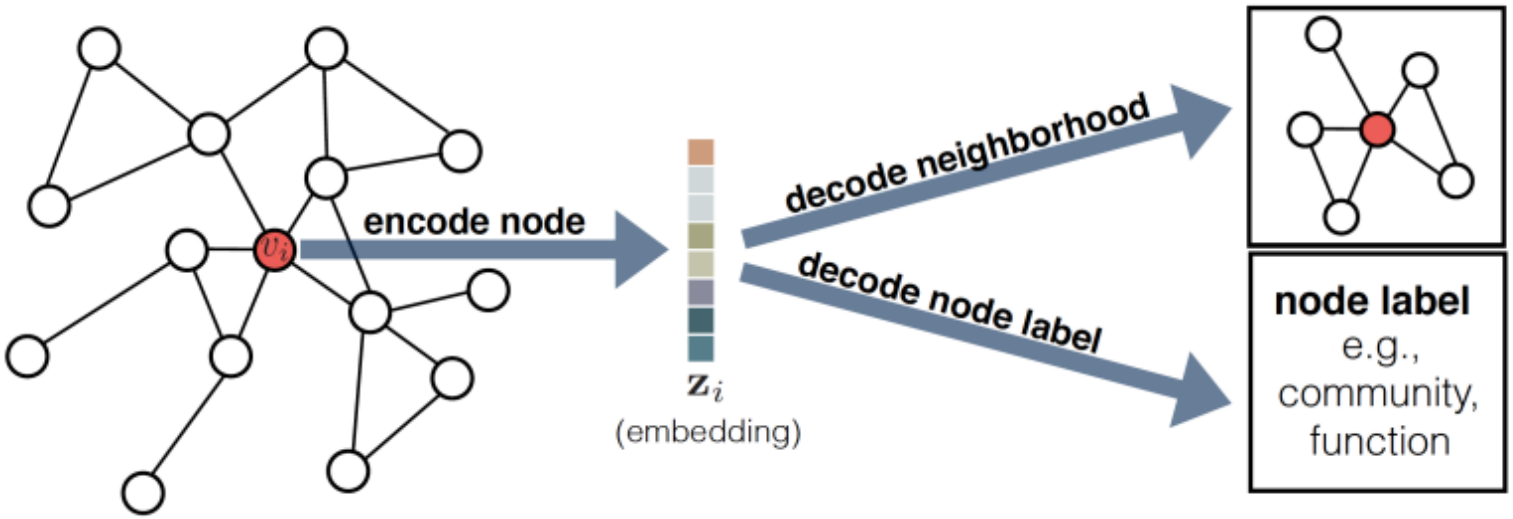
Optimize the encoder such that 2 nodes have similar embeddings if they….

1. are connected?
2. Share neighbors?
3. Similarity estimate the probability of visiting v from u ?

Graph Neural Networks

- Decoder

What we would like…



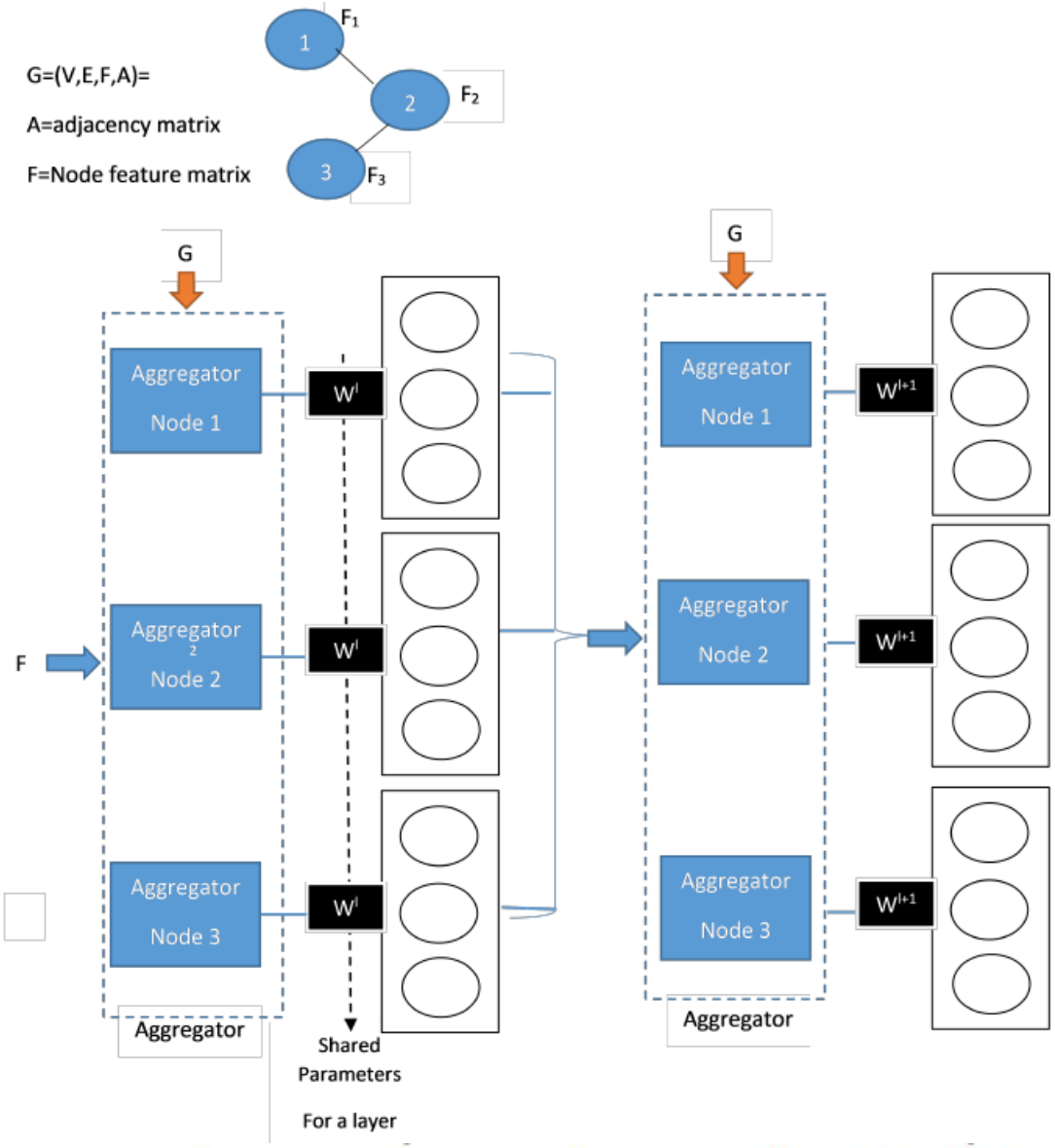
Graph Neural Networks

Key idea and Intuition
[Kipf and Welling, 2016]

The key idea is to generate node embeddings based on local neighborhoods.
The intuition is to aggregate node information from **their neighbors** using neural networks ➔ **done by including A**.
Nodes have embeddings at each layer and the neural network can be arbitrary depth. “layer-0” embedding of node u is its input feature

$$H^{(l+1)} = f(H^{(l)}, A)$$

$$f(H^{(l)}, A) = \sigma\left(AH^{(l)}W^{(l)}\right)$$



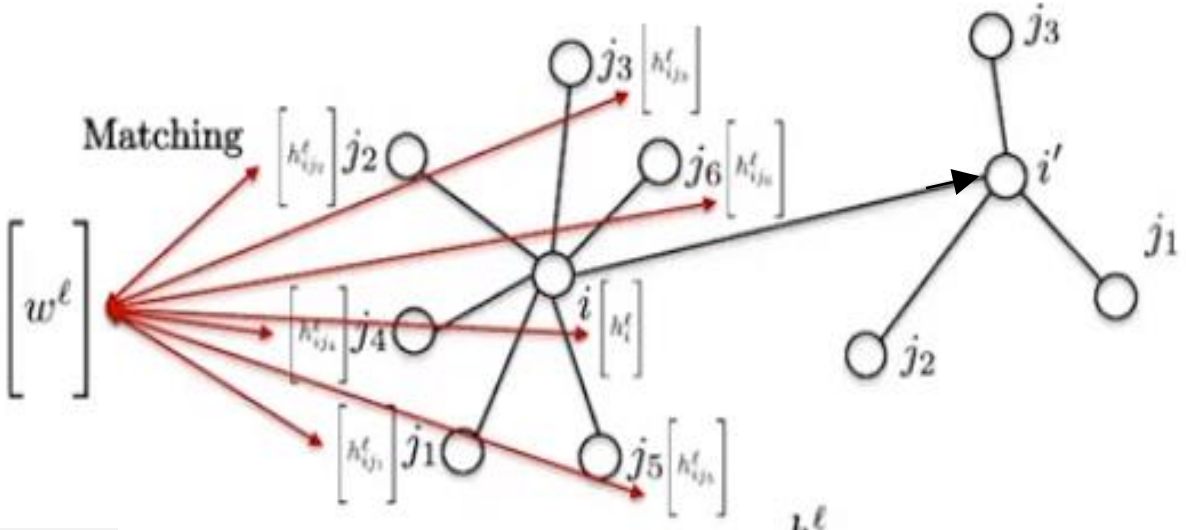
Hyperparameters: $W^{(l)} \in \mathbb{R}^{m_l \times m_{l+1}}$ $\sigma(.)$ is a non-linear activation function like the ReLU.

Graph Neural Networks

From [Bresson2020]

$$H^{(l+1)} = f(H^{(l)}, A)$$

$$f(H^{(l)}, A) = \sigma\left(AH^{(l)}W^{(l)}\right)$$



$$h_i^{\ell+1} = \eta\Big(\sum_{j \in \mathcal{N}_i} \underbrace{\langle w^\ell, h_{ij}^\ell \rangle}_{(h_{ij}^\ell)^T w^\ell}\Big)$$

scalar — One feature

$1 \times m_l$, $m_l \times 1$

$$h_i^{\ell+1} = \eta\Big(\sum_{j \in \mathcal{N}_i} W^\ell h_{ij}^\ell\Big)$$

$m_{l+1} \times 1$ — m_{l+1} features

$m_{l+1} \times m_l$, $m_l \times l$

$$h^{\ell+1} = \eta\big(Ah^\ell W^\ell\big)$$

$n \times m_{l+1}$ — Vectorial representation

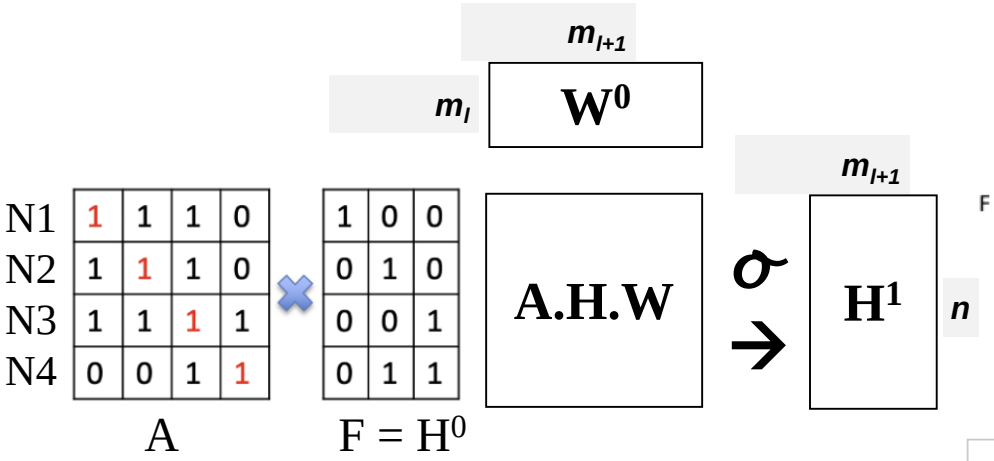
$n \times n$, $n \times m_l$, $m_l \times m_{l+1}$

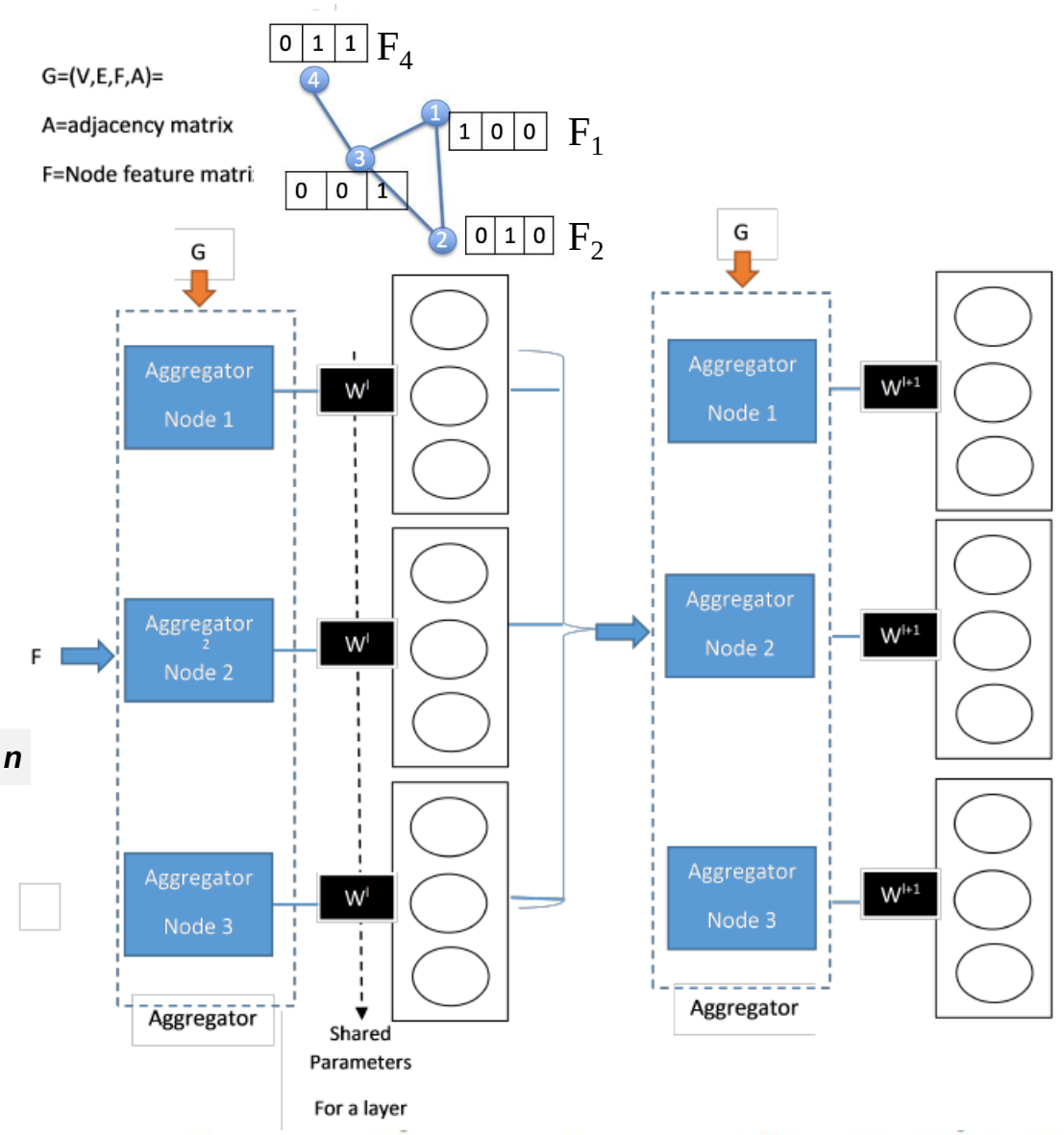
Hyperparameters: $W^{(l)} \in \mathbb{R}^{m_l \times m_{l+1}}$ $\sigma(.)$ is a non-linear activation function like the ReLU.

Graph Neural Networks

$$H^{(l+1)} = f(H^{(l)}, A)$$

$$f(H^{(l)}, A) = \sigma\left(AH^{(l)}W^{(l)}\right)$$

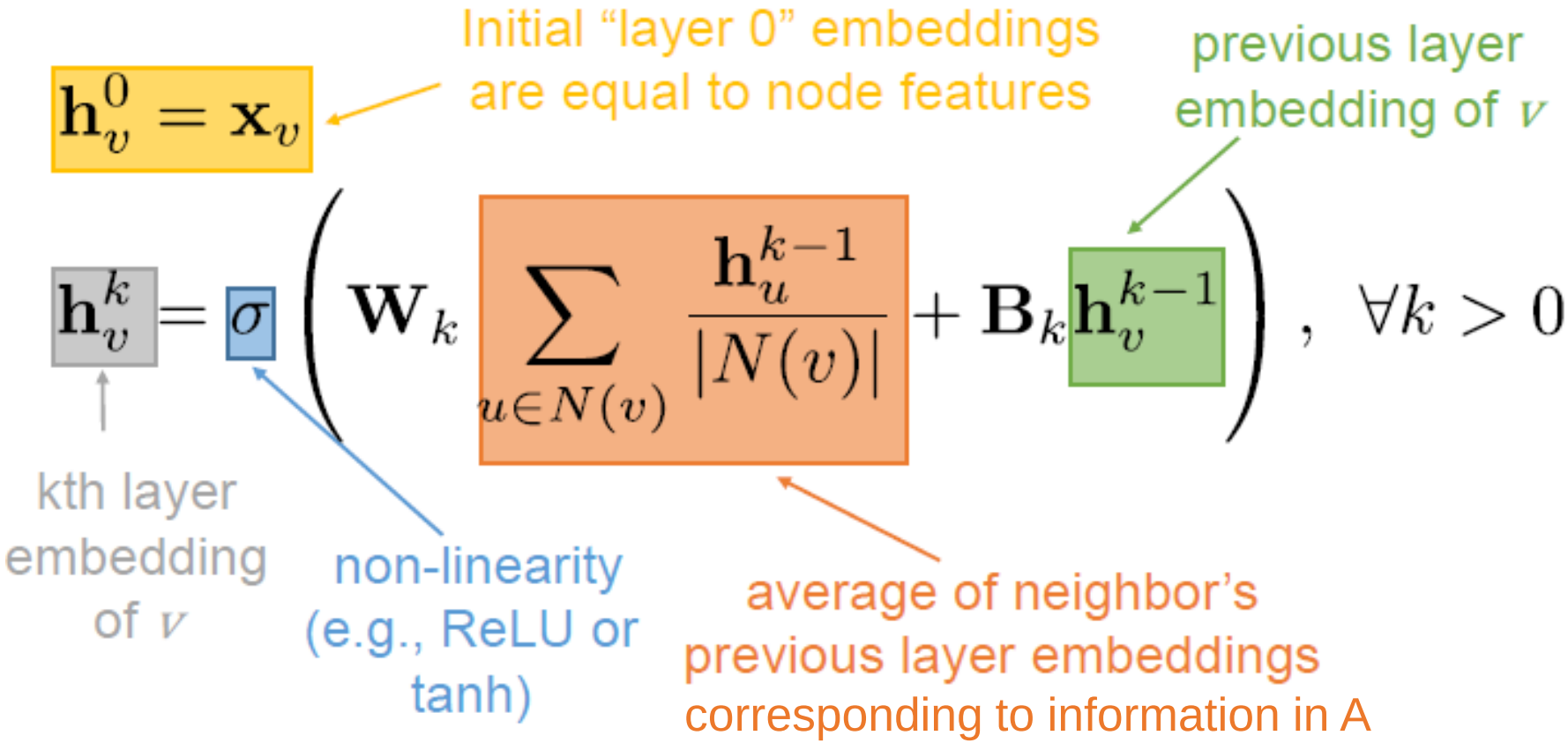




Hyperparameters: $W^{(l)} \in \mathbb{R}^{m_l \times m_{l+1}}$ $\sigma(.)$ is a non-linear activation function like the ReLU.

Graph Neural Networks

Basic approach: Average neighbor messages and apply a neural network.



Training the Model

trainable matrices
(i.e., what we learn)

$$\mathbf{h}_v^0 = \mathbf{x}_v$$

$$\mathbf{h}_v^k = \sigma\left(\mathbf{W}_k \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|} + \mathbf{B}_k \mathbf{h}_v^{k-1}\right), \quad \forall k \in \{1, ..., K\}$$

$$\mathbf{z}_v = \mathbf{h}_v^K$$

- After K-layers of neighborhood aggregation, we get output embeddings for each node.
- **We can feed these embeddings into any loss function** and run stochastic gradient descent to train the aggregation parameters.

2 issues of this simple example

Issue 1

- for every node, f sums up all the feature vectors of all neighboring nodes but not the node itself.
- Fix: simply add the identity matrix to $A \rightarrow \hat{A} = A + Id$

Issue 2

- A is typically not normalized and therefore the multiplication with A will completely change the scale of the feature vectors.
- Fix: Normalizing A such that all rows sum to one → $D^{-1}.A$

$$f(H^{(l)}, A) = \sigma\left(D^{-1}AH^{(l)}W^{(l)}\right)$$

The issues Altogether [Kipf et Welling, 2016]

- The two patched mentioned before +
- A better (symmetric) normalization of the adjacency matrix Ď

$$f(H^{(l)}, A) = \sigma\left(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right)$$

Final formulation of GCNs

- It is a slight variation on the neighborhood aggregation idea [Kipf et al]

$$\mathbf{h}_v^k = \sigma\left(\mathbf{W}_k \sum_{u \in N(v) \cup v} \frac{\mathbf{h}_u^{k-1}}{\sqrt{|N(u)||N(v)|}}\right)$$

Final formulation of GCNs

Empirically, they found this configuration to give the best results.

- More parameter sharing.
- Down-weights high degree neighbors.

$$\mathbf{h}_v^k = \sigma\left(\mathbf{W}_k \sum_{u \in N(v) \cup v} \frac{\mathbf{h}_u^{k-1}}{\sqrt{|N(u)||N(v)|}}\right)$$

use the same transformation matrix for self and neighbor embeddings

instead of simple average, normalization varies across neighbors

Final formulation of GCNs

Basic idea: Neighborhood Aggregation

$$\mathbf{h}_v^k = \sigma\left(\mathbf{W}_k \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|} + \mathbf{B}_k \mathbf{h}_v^{k-1}\right)$$

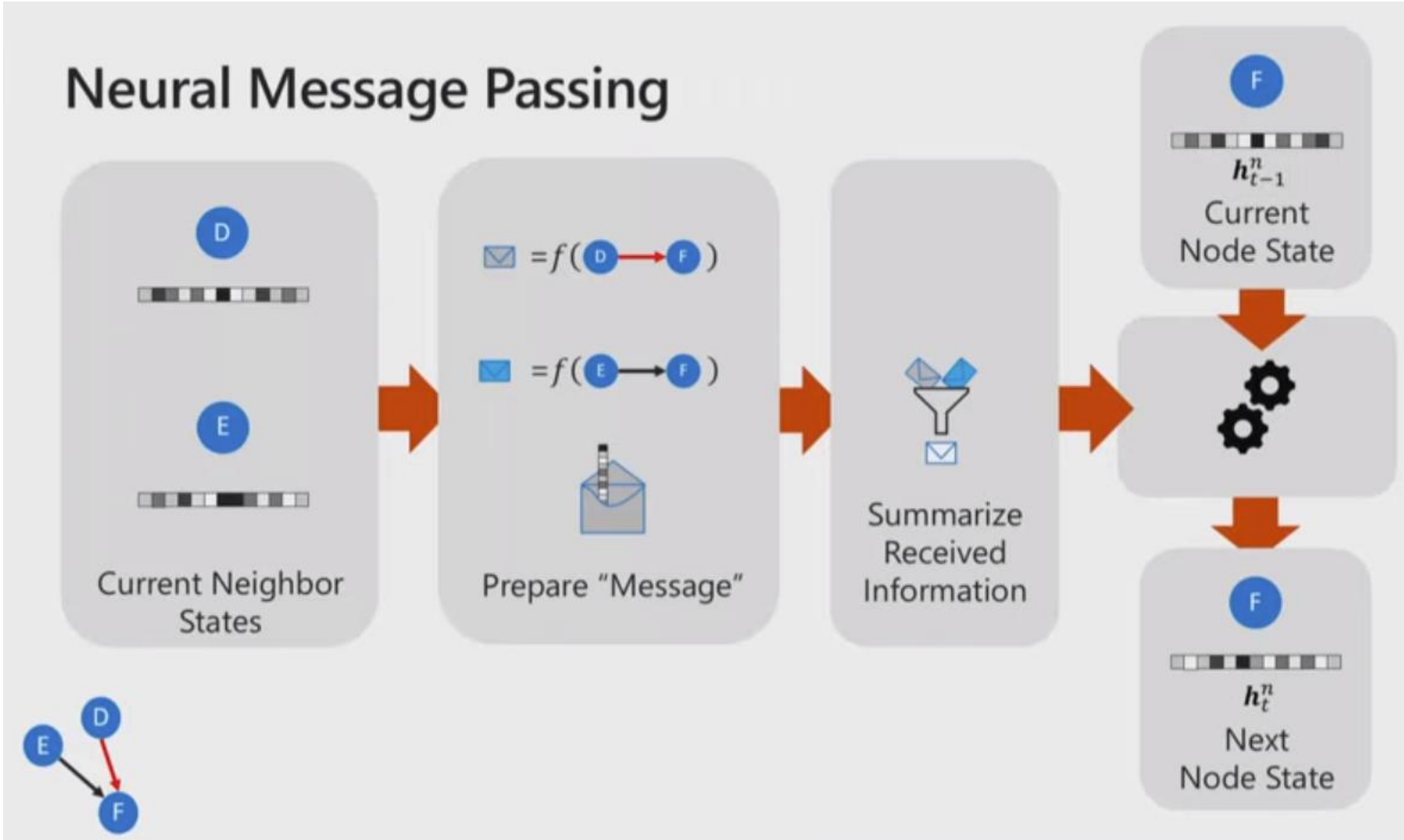
GCN Neighborhood Aggregation

$$\mathbf{h}_v^k = \sigma\left(\mathbf{W}_k \sum_{u \in N(v) \cup v} \frac{\mathbf{h}_u^{k-1}}{\sqrt{|N(u)||N(v)|}}\right)$$

same matrix for self and neighbor embeddings

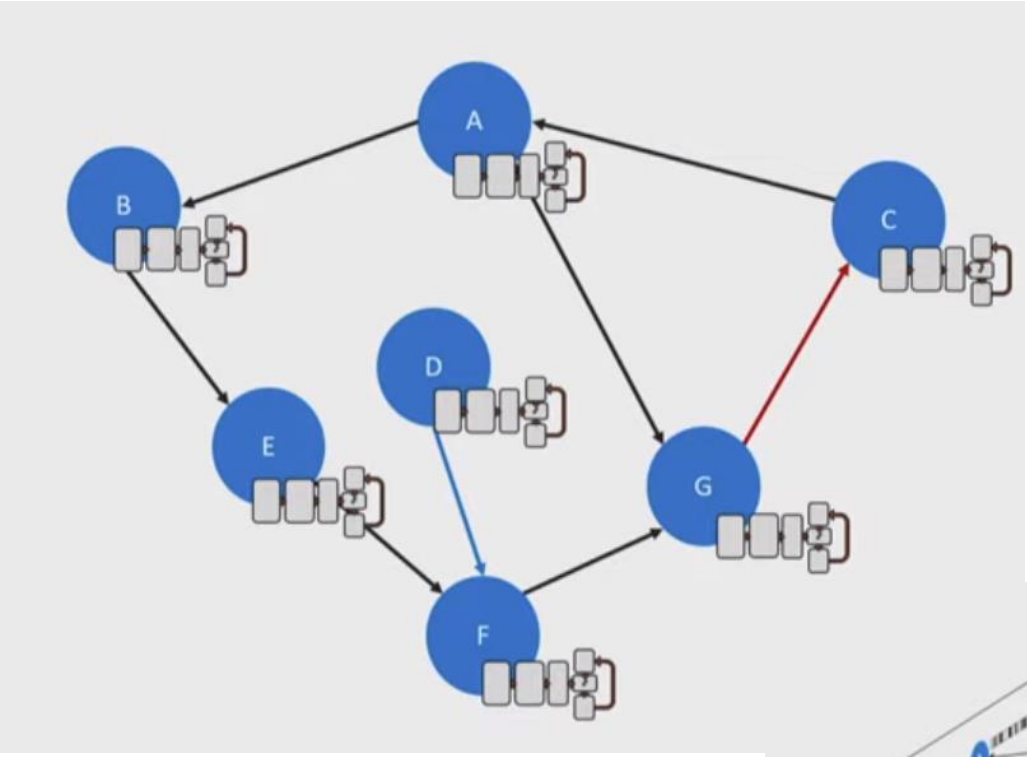
per-neighbor normalization

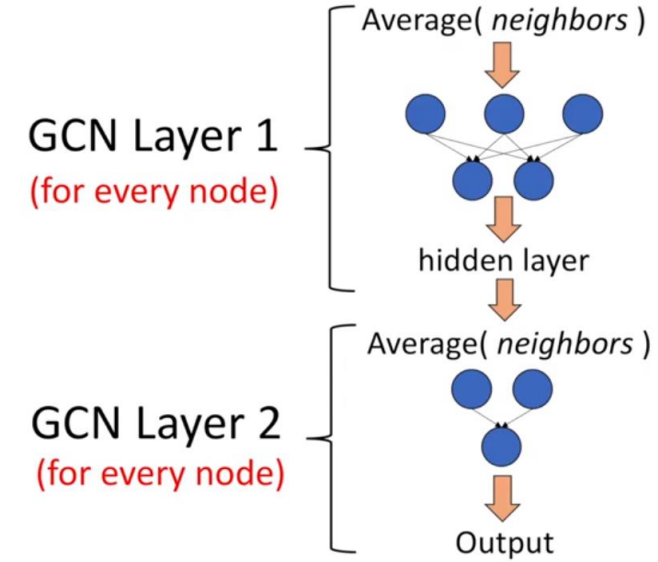
Graph Neural Networks

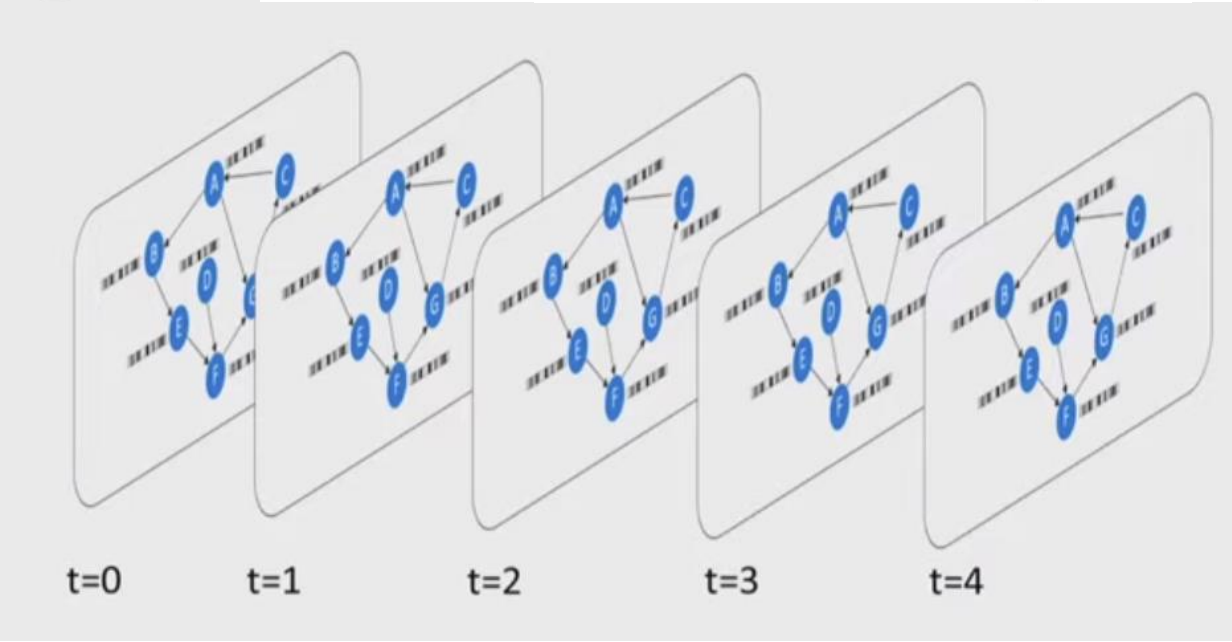


An Introduction to Graph Neural Networks: Models and Applications
Microsoft https://www.youtube.com/watch?v=zCEYiCxrL_0

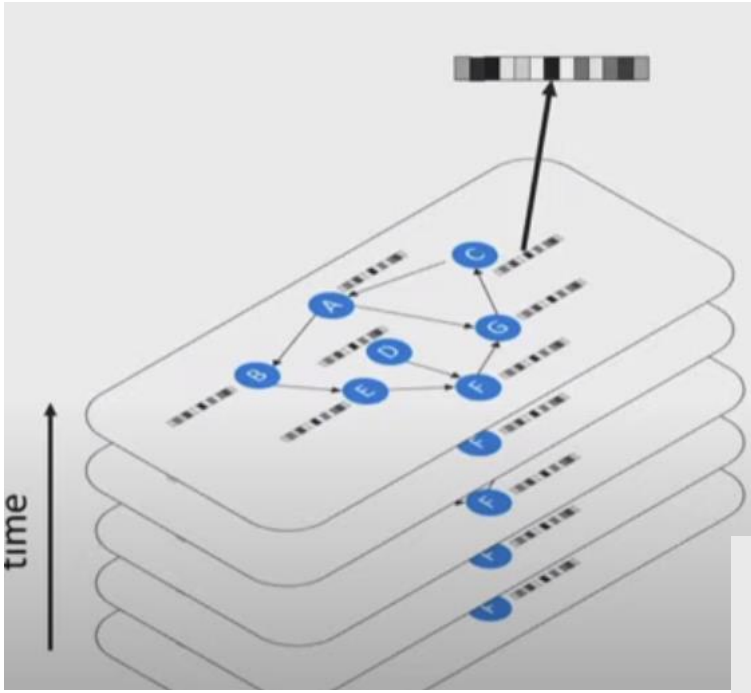
Graph Neural Networks





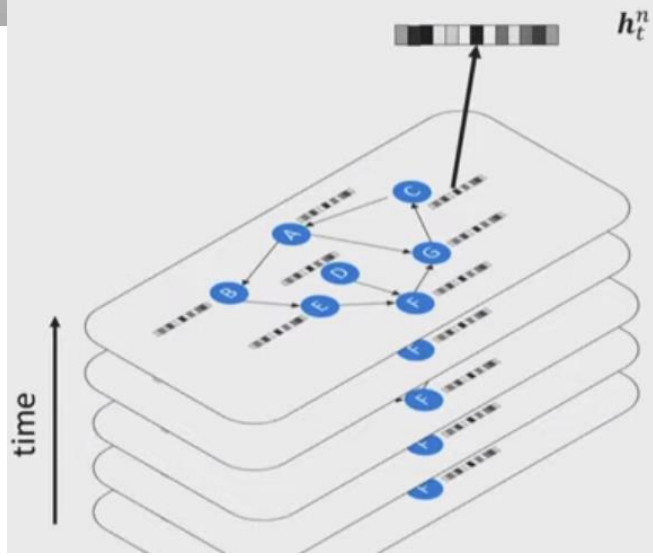


Loss function / Graph Neural Networks



- node selection
- node classification
- graph classification

Example: Node [Binary] Classification

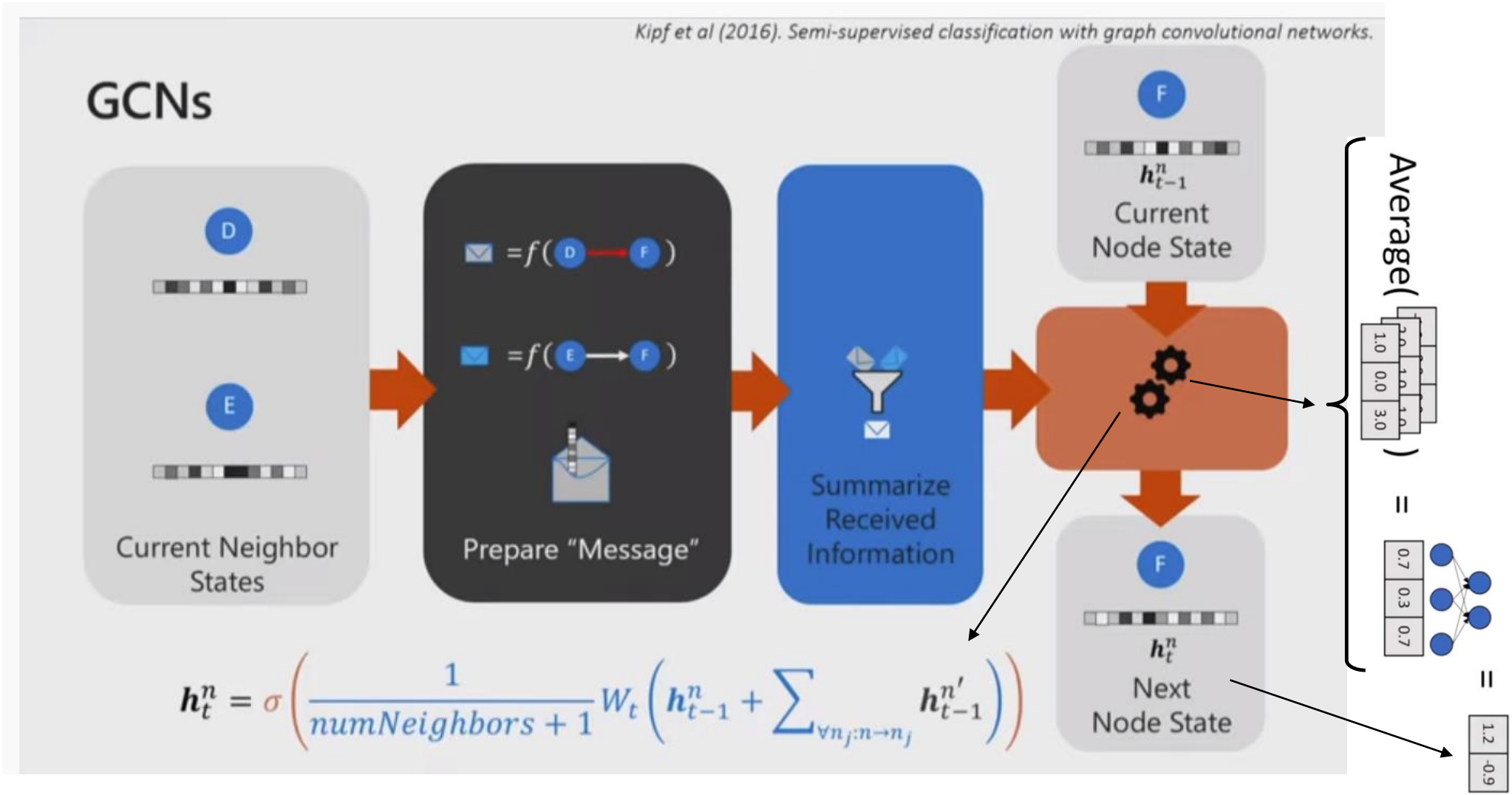


$$x_n = \sigma(\boldsymbol{w}^T \boldsymbol{h}_t^n + b)$$

Binary cross entropy

$$\mathcal{L} = y_n \cdot \log x_n + (1 - y_n) \log(1 - x_n)$$

Graph Neural Networks – Vanilla Model



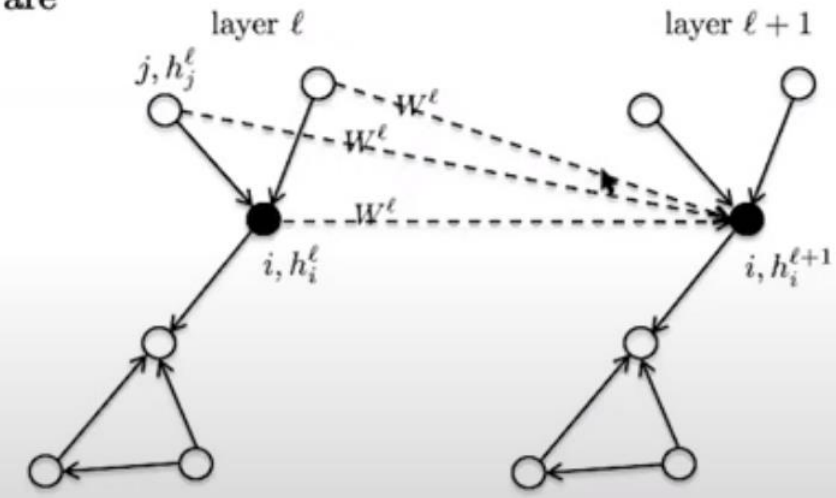
Graph Neural Networks – Vanilla Model

Vanilla GCNs[1,2,3]

- Simplest formulation of spatial GCNs
  - Handle the absence of node ordering
    - Invariant by node re-parametrization
  - Deal with different neighborhood sizes
  - Local reception field by design (only neighbors are considered)
  - Weight sharing (convolution property)
  - Independent of graph size
  - Limited to isotropic capability

$$\underset{n\times d}{h^{\ell+1}} = \eta(\underset{n\times n}{D^{-1}}\underset{n\times n}{A}\underset{n\times d}{h^{\ell}}\underset{d\times d}{W^{\ell}}) \qquad \text{Matrix representation}$$

$$\underset{d\times 1}{h_i^{\ell+1}} = \eta(\underbrace{\frac{1}{d_i}\sum_{j\in\mathcal{N}_i}}_{\text{Mean}}\underset{\text{scalar}}{A_{ij}}\underset{d\times d}{W^{\ell}}\underset{d\times 1}{h_j^{\ell}}) \qquad \text{Vectorial representation}$$



$$h_i^{\ell+1} = f_{\text{GCN}}(h_i^\ell, \{h_j^\ell : j \to i\})$$

[1] Scarselli, Gori, Tsoi, Hagenbuchner, Monfardini, The Graph Neural Network Model, 2009
[2] TN Kipf, M Welling, Semi-supervised classification with graph convolutional networks, 2016
[3] S Sukhbaatar, A Szlam, R Fergus, Learning multiagent communication with backpropagation, 2016

From [Bresson2020]

Graph Neural Networks – GraphSage Model

- Vanilla GCNs (supposing $A_{ij} = 1$) : $h_i^{\ell+1} = \eta\left(\frac{1}{d_i} \sum_{j \in \mathcal{N}_i} W^\ell h_j^\ell \right)$
- GraphSage :
  - Differentiate template weights W^l between neighbors h_j and central node h_i.
  - Isotropic GCNs

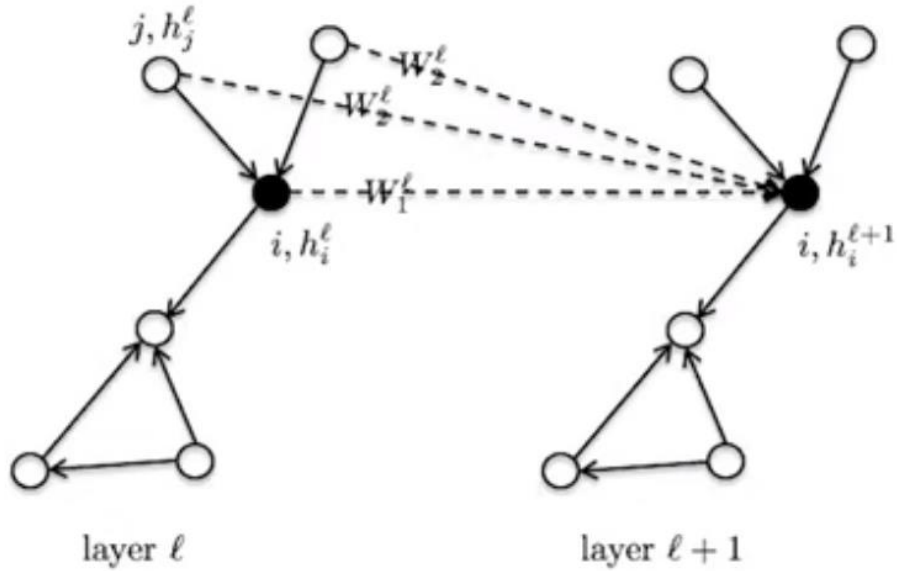
$$\underset{d \times 1}{h_i^{\ell+1}} = \eta\Big(\underset{d \times d}{W_1^\ell} \underset{d \times 1}{h_i^\ell} + \underbrace{\frac{1}{d_i} \sum_{j \in \mathcal{N}_i} \underset{d \times d}{W_2^\ell} \underset{d \times 1}{h_j^\ell}}_{} \Big)$$

$\text{Mean}_{j \in \mathcal{N}_i} W_2^\ell h_j^\ell$

Or alternatively,

$\text{Max}_{j \in \mathcal{N}_i} W_2^\ell h_j^\ell$

$\text{LSTM}^\ell(h_j^\ell)$



[1] W Hamilton, Z Ying, J Leskovec, Inductive representation learning on large graphs, 2017

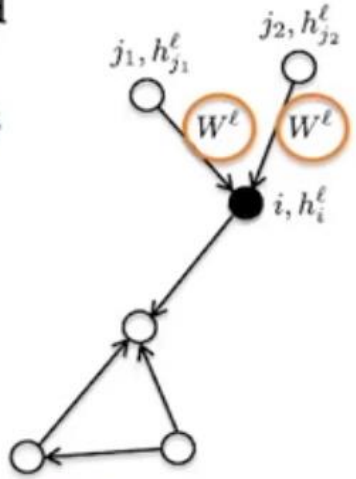
From [Bresson2020]

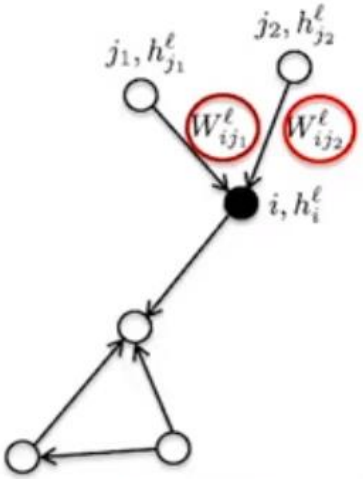
Graph Neural Networks – Anisotropic Models

In general…

- Reminder :
  - Standard ConvNets produce anisotropic filters because Euclidean grids have directional structures (up, down, left, right).
  - GCNs such as ChebNets, CayleyNets, Vanilla GCNs, GraphSage, GIN compute isotropic filters as there is no notion of directions on arbitrary graphs.

- How to get anisotropy back in GNNs ?
  - Natural edge features[1,2] if available (e.g. different bond connections between atoms).
  - We need an anisotropic mechanism that is independent of the node parametrization.
  - Edge degrees[3]/Edge gates[4]/Attention mechanism[5] : MoNets[3], GAT[5], GatedGCNs[4] can treat neighbors differently.





[1] Gilmer, Schoenholz, Riley, Vinyals, Dahl, Neural message passing for quantum chemistry, 2017
[2] X Bresson, T Laurent, A Two-Step Graph Convolutional Decoder for Molecule Generation, 2019
[3] F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, M. Bronstein, Geometric deep learning on graphs and manifolds using mixture model CNNs, 2016
[4] X Bresson, T Laurent, Residual gated graph convnets, 2017
[5] Velickovic, Cucurull, Casanova, Romero, Lio, Bengio, Graph Attention Networks, 2018

From [Bresson2020]

Graph Neural Networks – Anisotropic Models - GAT

Graph Attention Networks

- GAT uses the attention mechanism[2] to introduce anisotropy in the neighborhood aggregation function.
- The network employs a multi-headed architecture to increase the learning capacity, similar to the Transformer[3].

1 edge = 1 learned weight

$$h_i^{\ell+1}_{d\times 1} = \mathrm{Concat}_{k=1}^{K}\Big(\mathrm{ELU}\Big(\sum_{j\in\mathcal{N}_i} \underbrace{e_{ij}^{k,\ell}}_{\text{scalar}}\ W_1^{k,\ell}\ h_j^{\ell}\Big)\Big)$$

$W_1^{k,\ell}$: $\frac{d}{K}\times d$, h_j^{ℓ}: $d\times 1$

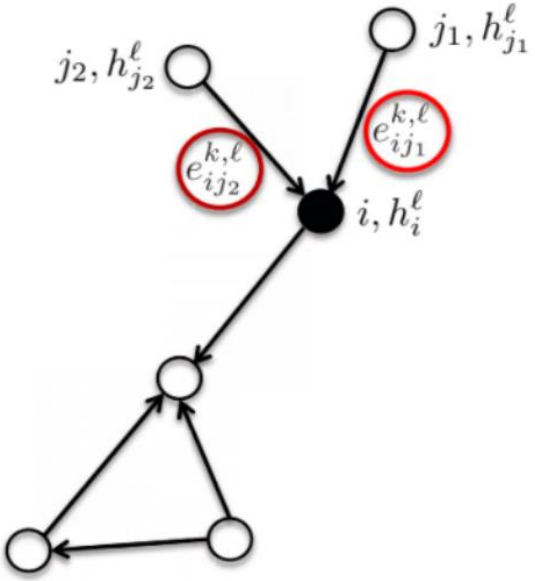
How to learn weights associated to edge….

$$e_{ij}^{k,\ell}_{\text{scalar}} = \mathrm{Softmax}_{\mathcal{N}_i}(\hat{e}_{ij}^{k,\ell}) = \frac{\exp(\hat{e}_{ij}^{k,\ell})}{\sum_{j'\in\mathcal{N}_i}\exp(\hat{e}_{ij'}^{k,\ell})}$$

$$\hat{e}_{ij}^{k,\ell} = \mathrm{LeakyReLU}\Big(W_2^{k,\ell}\ \mathrm{Concat}\big(W_1^{k,\ell}h_i^{\ell},\ W_1^{k,\ell}h_j^{\ell}\big)\Big)$$

$W_2^{k,\ell}$: $1\times\frac{2d}{K}$, $\mathrm{Concat}(\cdots)$: $\frac{2d}{K}\times 1$

Learned weights can be dependant of connected node labels….



[1] Velickovic, Cucurull, Casanova, Romero, Lio, Bengio, Graph Attention Networks, 2018
[2] D Bahdanau, K Cho, Y Bengio, Neural machine translation by jointly learning to align and translate, 2014
[3] A Vaswani, N Shazeer, N Parmar, J Uszkoreit, L Jones, A. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, 2017

From [Bresson2020]

Graph Neural Networks – Anisotropic Models - GAT

Graph Transformers

- Graph version of Transformer[1] :

$$h_i^{\ell+1} = W^{\ell} \, \mathrm{Concat}_{k=1}^{K} \Big(\sum_{j \in \mathcal{N}_i} e_{ij}^{k,\ell} \, V^{k,\ell} \, h_j^{\ell} \Big)$$

$h_i^{\ell+1}$: $d \times 1$; W^{ℓ}: $d \times d$; $V^{k,\ell} \, h_j^{\ell}$: $\frac{d}{K} \times d$

Value

scalar

$$e_{ij}^{k,\ell} = \mathrm{Softmax}_{\mathcal{N}_i}(\hat{e}_{ij}^{k,\ell}) = \frac{\exp(\hat{e}_{ij}^{k,\ell})}{\sum_{j' \in \mathcal{N}_i} \exp(\hat{e}_{ij'}^{k,\ell})}$$

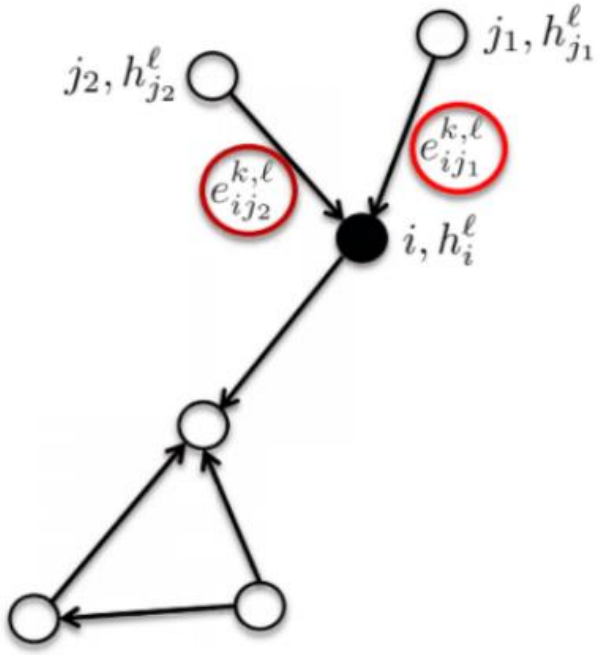
$$\hat{e}_{ij}^{k,\ell} = \Big(Q^{\ell,k} h_i^{\ell} \Big)^T \Big(K^{\ell,k} h_j^{\ell} \Big)$$

$Q^{\ell,k} h_i^{\ell}$: $1 \times d$; $K^{\ell,k} h_j^{\ell}$: $d \times 1$

Query

Key

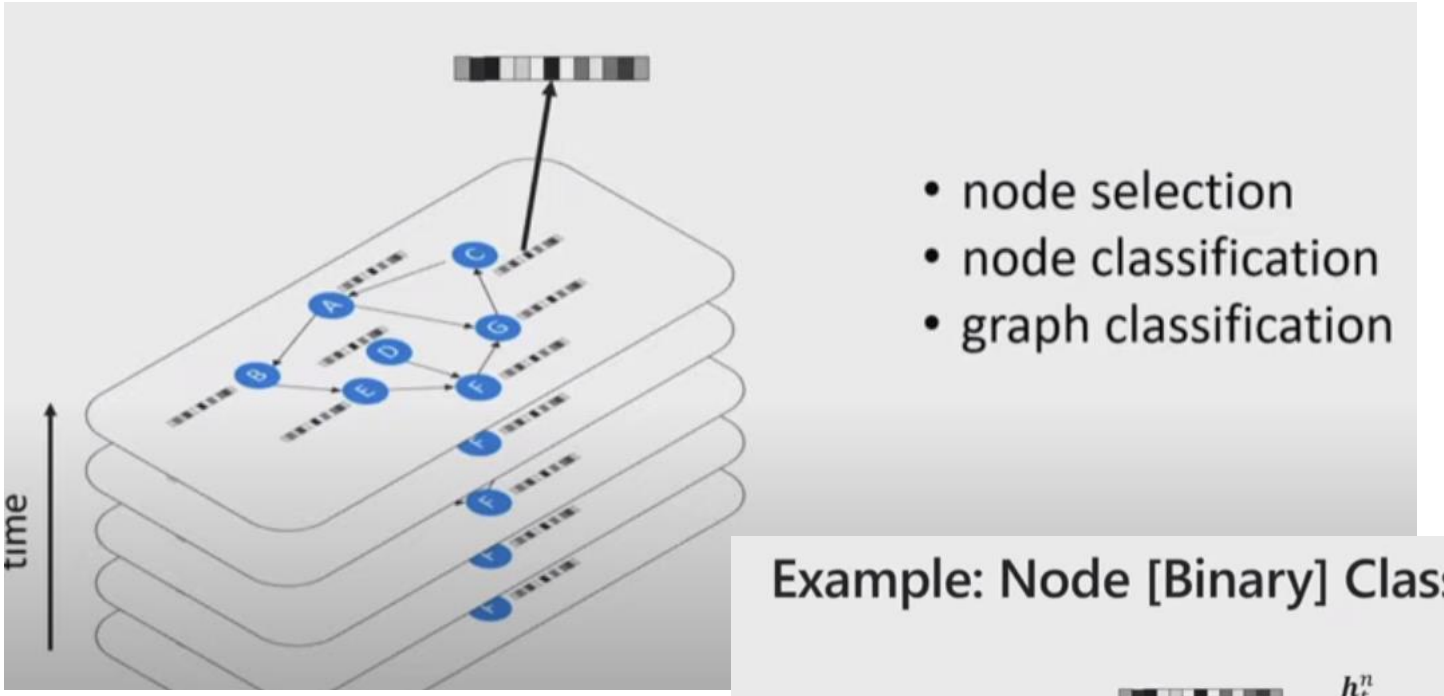
Attention mechanism in 1-hop neighborhood



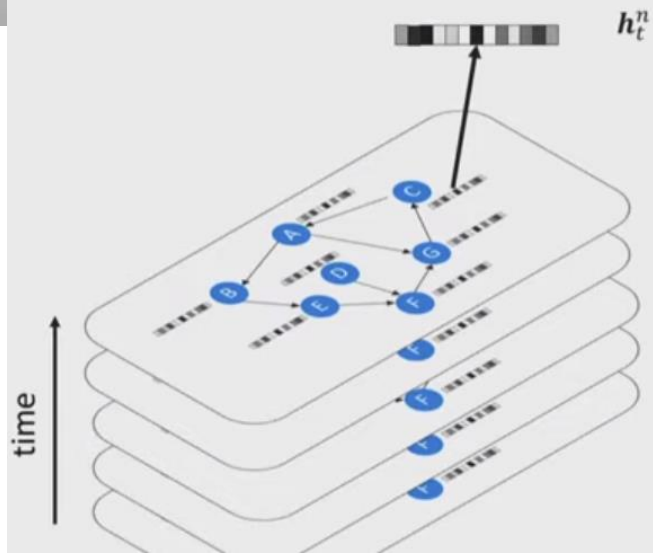
[1] A Vaswani, N Shazeer, N Parmar, J Uszkoreit, L Jones, A. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, 2017

From [Bresson2020]

Loss function for GM ???



Example: Node [Binary] Classification



$$x_n = \sigma(\boldsymbol{w}^T \boldsymbol{h}_t^n + b)$$

Binary cross entropy

$$\mathcal{L} = y_n \cdot \log x_n + (1 - y_n) \log(1 - x_n)$$

GM with GNN

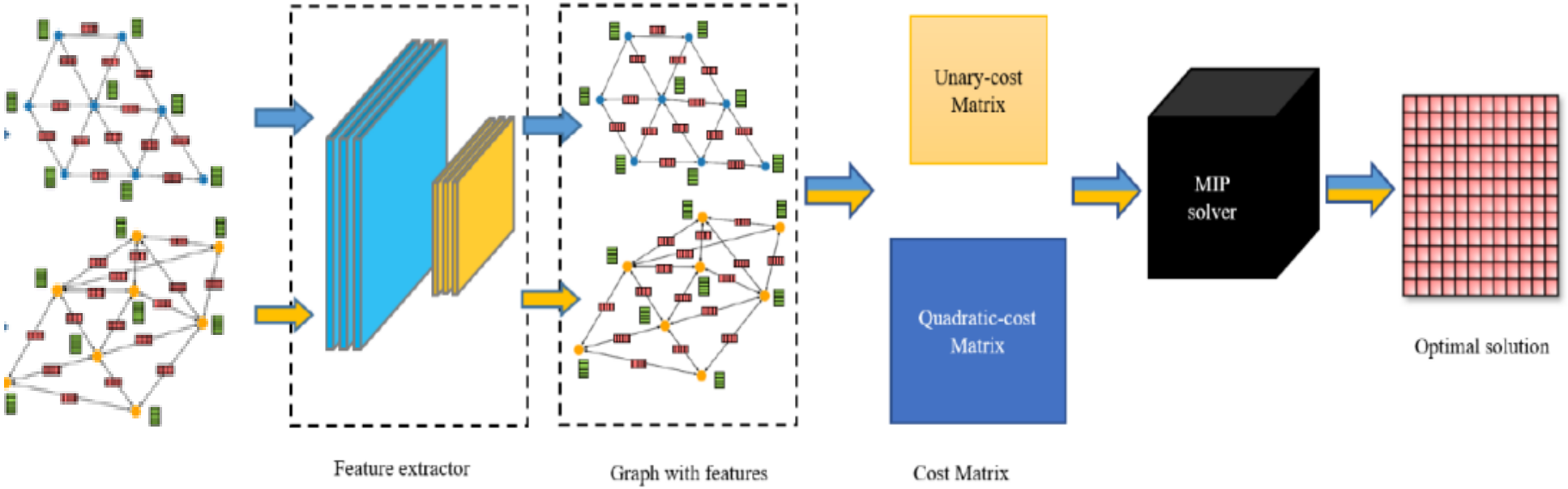
- 2 steps pipeline
  - GNN ➔ Features extractor
  - LP Solver ➔ Optimal Matching
  - Global gradient descent

Problem 2. Learning graph matching problem

$$\hat{\theta} = arg\min_{\theta\in\Theta} \sum_{((G_1,G_2),\mathbf{v}^{gt})\in TrS} L(\mathbf{v}^{gt}, \Phi(\theta; G_1, G_2))$$

$$TrS = \{((G_1, G_2)_k, \mathbf{v}_k^{gt})\}_{k=1}^{M}$$

$$L = \|\mathbf{v}^{gt} - \hat{\mathbf{v}}\|_1$$ ➔ Hamming loss



References (graph Partitioning)

Satu Elisa Schaeffer - “Graph clustering” (2007)

Santo Fortunato – “*Community detection in graphs*” (2010)

Giatsidis et al. – “*Graph Mining Tools for Community Detection & Evaluation in Social Networks & the Web*” (2013)

Newman and Girvan – “*Finding and evaluating community structure in networks*” (2004)

Blondel et al. – “*Fast unfolding of communities in large networks*” (2008)

Ulrike van Luxburg – “*A Tutorial on Spectral Clustering*” (2007)

References (Graph Matching)

- Zeina Abu-Aisheh, Romain Raveaux, P Martineau, Jean-Yves Ramel. Distributed Graph Matching and Graph Indexing Approaches: Applications to Pattern Recognition, ICPRAM 2015, Doctoral consortium
- Muhammad Muzzamil Luqman, Jean-Yves Ramel, Josep Lladós, Thierry Brouard: Fuzzy multilevel graph embedding. Pattern Recognition 46(2): 551-565 (2013)
- Muhammad Muzzamil Luqman, Jean-Yves Ramel, Josep Lladós, Thierry Brouard: Subgraph Spotting through Explicit Graph Embedding: An Application to Content Spotting in Graphic Document Images. ICDAR 2011: 870-874
- Romain Raveaux, Sébastien Adam, Pierre Héroux, Éric Trupin: Learning graph prototypes for shape recognition. Computer Vision and Image Understanding 115(7): 905-918 (2011)
- Romain Raveaux, Jean-Christophe Burie, Jean-Marc Ogier: A graph matching method and a graph matching distance based on subgraph assignments. Pattern Recognition Letters 31(5): 394-406 (2010)
- Nicolas Sidere, Pierre Héroux, Jean-Yves Ramel: Vector Representation of Graphs: Application to the Classification of Symbols and Letters. ICDAR 2009: 681-685
- Rashid Jalal Qureshi, Jean-Yves Ramel, Hubert Cardot: Graph Based Shapes Representation and Recognition. GbRPR 2007: 49-60
- Rashid Jalal Qureshi, Jean-Yves Ramel, Didier Barret, Hubert Cardot: Spotting Symbols in Line Drawing Images Using Graph Representations. GREC 2007: 91-103

References (Graph Matching)

- Gauzere, B., Brun, L., and Villemin, D. (2011).Two new graph kernels and applications to chemoinformatics. Pattern Recognition Letters.
- Riesen, K., Neuhaus, M., and Bunke, H. (2007). Graph embedding in vector spaces by means of prototype selection. In Escolano, F. and Vento, M., editors, 6th IAPR-TC15 International Workshop GbRPR 2007, pages 383{393. IAPR TC15, Springer-Verlag.
- H. Bunke et X. Jiang : Graph matching and similarity, chapitre de “Intelligent systems and interfaces”, Kluwer, 2000
- Jaume Gibert, Ernest Valveny, Horst Bunke: Graph embedding in vector spaces by node attribute statistics. Pattern Recognition 45(9): 3072-3083 (2012)
- S. Sorlin, C. Solnon et J.-M. Jolion : A Generic Graph Distance Measure Based on Multivalent Matchings, chapitre de “Applied Graph Theory in PR”, Vol 52:151-182, Springer, 2007
- S. Sorlin et C. Solnon : Reactive Tabu Search for Measuring Graph Similarity, GbR, LNCS 3434:172-182, 2005
- Zeina Abu-Aisheh. Anytime and Distributed Approaches for Graph Matching. Université François Rabelais de Tours – France. May 18th, 2016.
- D. Conte, J.-Y. Ramel, N. Sidère, M.M. Luqman, B. Gauzère, J. Gibert,L. Brun, et M. Vento. “A Comparison of Explicit and Implicit GraphEmbedding Methods for Pattern Recognition”. In : Proceedings ofthe 9th IAPR-TC15 workshop on Graph-based Representation in PatternRecognition (GbR 2013). 2013, p. 81-90.

References (Graph Matching)

K. Riesen and H. Bunke – “IAM Graph Database Repository for Graph Based Pattern Recognition and Machine Learning” (2008)

S. Sakr and G. Al-Naymat - “Graph indexing and querying: a review” (2010)

R. Guigno and D. Shasha - “GraphGrep: A Fast and Universal Method for Querying Graphs” (2002)

X. Yan et al. - “Graph Indexing: A Frequent Structurebased Approach” (2004)

H. He and A. K. Singh – “Closure-Tree: An Index Structure for Graph Queries” (2006)

D. Conte et al. – “Thirty years of Graph Matching in Pattern Recognition” (2004)

References (GNN)

Tutorials:

• Geometric Deep Learning, Tutorial, CVPR, 2017. http://geometricdeeplearning.com/

• Deep Learning on Graphs with Graph Convolutional Networks. http://deeploria.gforge.inria.fr/thomasTalk.pdf

• Graph-based Methods in Pattern Recognition & Document Image Analysis. http://gmprdia.univ-lr.fr/

List of papers:

• Gilmer et al., Neural Message Passing for Quantum Chemistry, 2017. https://arxiv.org/abs/1704.01212

• Kipfet al., Semi-Supervised Classification with Graph Convolutional Networks, ICLR 2017. https://arxiv.org/abs/1609.02907

• Defferrardet al., Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering, NIPS 2016. https://arxiv.org/abs/1606.09375

• Brunaet al., Spectral Networks and Locally Connected Networks on Graphs, ICLR 2014. https://arxiv.org/abs/1312.6203

• Duvenaudet al., Convolutional Networks on Graphs for Learning Molecular Fingerprints, NIPS 2015. https://arxiv.org/abs/1509.09292

• Li et al., Gated Graph Sequence Neural Networks, ICLR 2016. https://arxiv.org/abs/1511.05493

• Battagliaet al., Interaction Networks for Learning about Objects, Relations and Physics, NIPS 2016. https://arxiv.org/abs/1612.00222

• Kearneset al., Molecular Graph Convolutions: Moving Beyond Fingerprints, 2016. https://arxiv.org/abs/1603.00856 91

Licence

- Cette présentation est distribuée sous licence Creative Commons
- Attribution-ShareAlike 4.0 International