



Graph mining

Karel Vaculík

FI MU

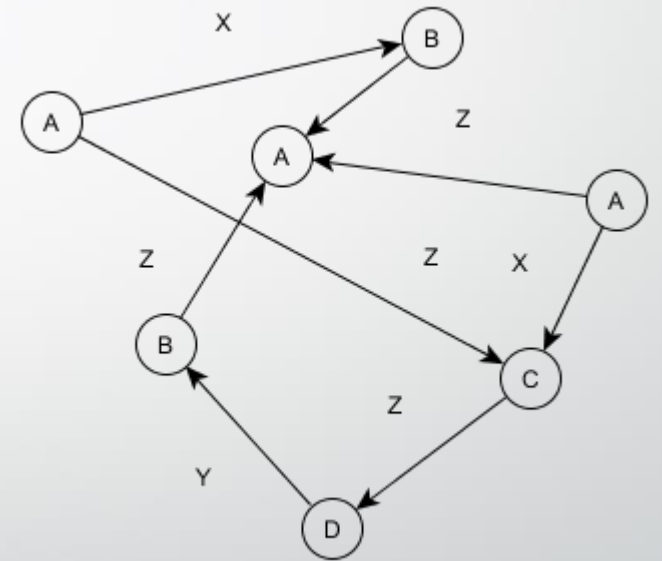
PV056 13. 5. 2014

Outline

- Graph definition revision
- Motivation examples
- Graph mining settings
- Similarity measures
- Frequent subgraph mining
- Basic mining tasks (classification, clustering, ...)

Graphs

- Directed graph $G = (V, E, \mu, \nu)$
 - V set of nodes
 - $E \subseteq V \times V$ set of edges
 - $\mu: V \rightarrow L_V$ node labeling function, L_V set of node labels
 - $\nu: E \rightarrow L_E$ edge labeling function, L_E set of edge labels
- Undirected graph similarly



Motivation – graphs in real world

- World Wide Web graph
- Social networks
- Chemical compounds (atoms and bonds)
- Biological networks (protein-protein interaction, metabolic pathways, ...)
- XML documents
- Software engineering (UML diagrams - work flows, ...)

Graph mining settings

- One (large) graph (*single-graph setting*) vs. database of graphs (*graph-transaction setting*)
- Directed vs. undirected graphs
- Dynamic vs. static graphs

Measuring similarity

Inter-graph similarity (between graphs)

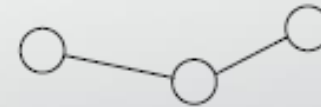
- Isomorphism – in NP (but not known whether in P or in NP-complete)



(a)



(b)



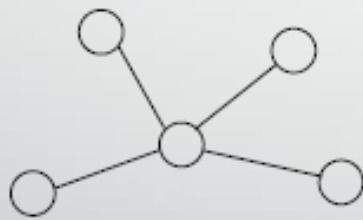
(c)

Graph (b) is isomorphic to (a) and (c) is isomorphic to a subgraph of (a)

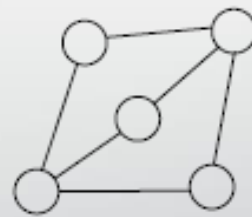
Measuring similarity

Inter-graph similarity (between graphs)

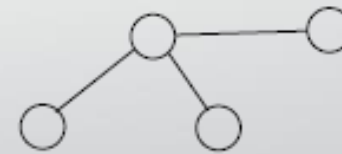
- Isomorphism – in NP (but not known whether in P or in NP-complete)
- Maximum Common Subgraph – NP-hard



(a)



(b)



(c)

Two graphs (a) and (b) and a maximum common subgraph (c)

Measuring similarity

Inter-graph similarity (between graphs)

- Isomorphism – in NP (but not known whether in P or in NP-complete)
- Maximum Common Subgraph – NP-hard
- Graph-edit distance – for given costs of operations (node addition/deletion, edge addition/deletion, ...), find the minimum total cost of operations needed to transform one graph into another one.

Measuring similarity

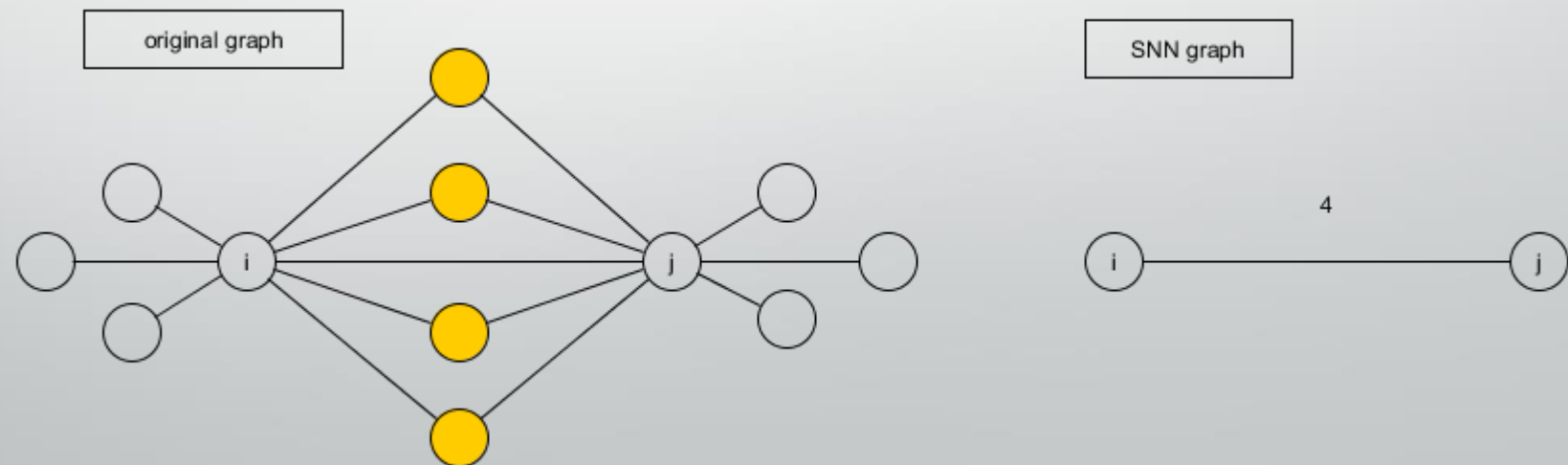
Inter-graph similarity (between graphs)

- Isomorphism – in NP (but not known whether in P or in NP-complete)
- Maximum Common Subgraph – NP-hard
- Graph-edit distance
- Kernels
 - Complete kernel (distinguishes nonisomorphic graphs) as hard as isomorphism
 - For example: Walk-based kernels (e.g. Direct Product Kernel)
- Frequent subgraphs (next section)

Measuring similarity

In-graph similarity (between vertices)

- Shared Nearest Neighbor (SNN) – vertices v_i and v_j from the original graph are neighbors in SNN graph, if they have at least k neighbors in common in the original graph



Measuring similarity

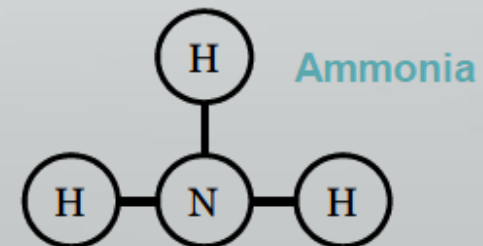
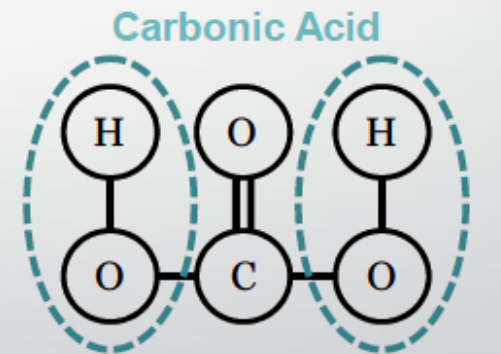
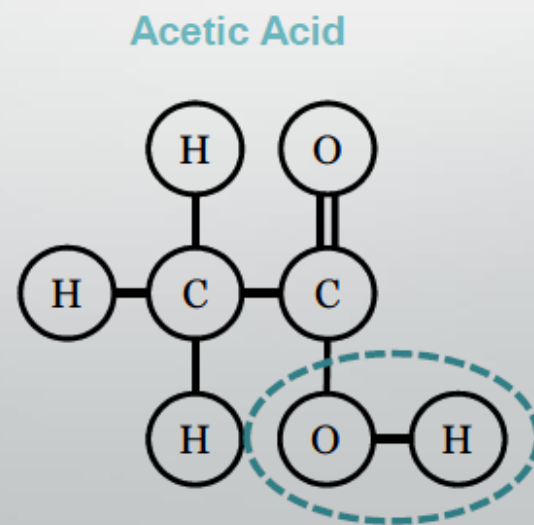
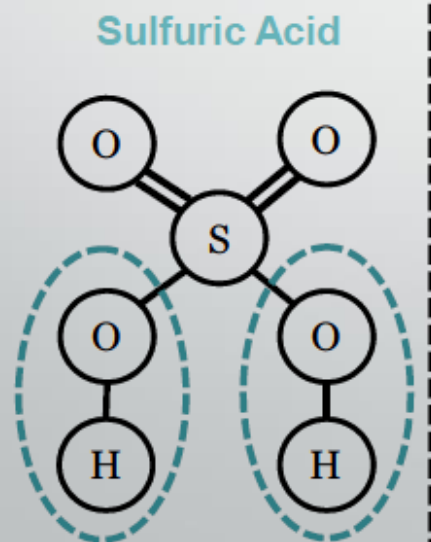
In-graph similarity (between vertices)

- Shared Nearest Neighbor (SNN)
- Kernels
 - For example: Neumann Kernel, Laplacian Kernel

Frequent subgraph mining (FSM)

- Frequent subgraph: A subgraph which *occurs* in at least n graphs (or n times in one graph) on input for a given minimum support n .

O-H present in $\frac{3}{4}$ inputs \rightarrow frequent if support ≤ 3



Frequent subgraph mining (FSM)

- Frequent subgraph: A subgraph which *occurs* in at least n graphs (or n times in one graph) on input for a given minimum support n .
- Possible to use *relative* support (%) for graph-transaction setting
- Applications: characterization of graphs, classifying, clustering, association rules, indexing
 - Specific example: Analyzing web server logs (each user \rightarrow one tree of visited pages; mine frequent subgraphs to find a better way of organizing structure of hyperlinks)
- Problem: not possible to use subgraph isomorphism (NP-complete!)

Frequent subgraph mining (FSM)

- General process:
 - *Candidate generation*: which patterns will be considered
 - *Candidate pruning*: if a candidate is not a viable frequent pattern, can we exploit the pattern to prevent unnecessary work?
 - Subgraphs and subsets exponentiate as size increases!
 - *Support counting*: how many of a given pattern exist?
- Apriori principle: if an itemset is frequent, then all of its subsets are also frequent.
 - Example: if itemset {A, B, C, D} is frequent, then {A, B} is frequent

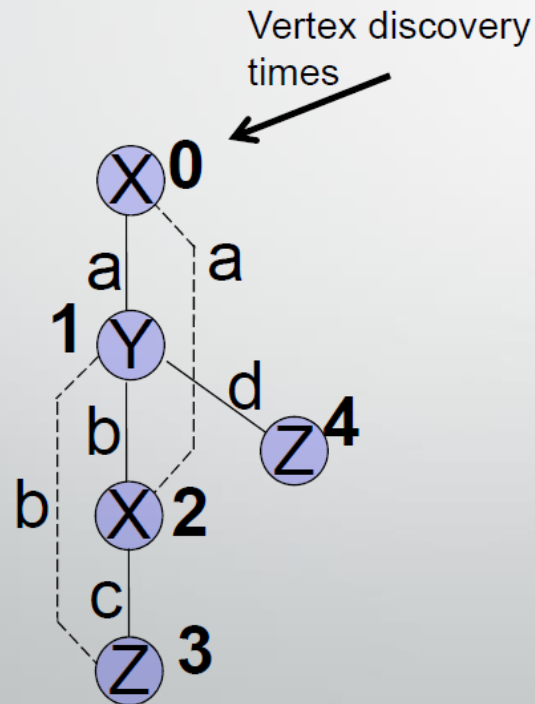
Frequent subgraph mining (FSM)

gSpan

- Complete (finds all frequent subgraphs) FSM algorithm on labeled undirected graphs
- Generates candidates by adding edges (one at a time) to patterns already discovered
- Encodes patterns (graphs) according to DFS traversal order (DFS code)

Frequent subgraph mining (FSM)

DFS Code: sequence of edges traversed during DFS



Edge #	Code
0	(0,1,X,a,Y)
1	(1,2,Y,b,X)
2	(2,0,X,a,X)
3	(2,3,X,c,Z)
4	(3,1,Z,b,Y)
5	(1,4,Y,d,Z)

Format: $(i, j, L_i, L_{(i,j)}, L_j)$

i, j – vertices by time of discovery

L_i, L_j - vertex labels of v_i, v_j

$L_{(i,j)}$ – edge label between v_i, v_j

$i < j$: forward edge

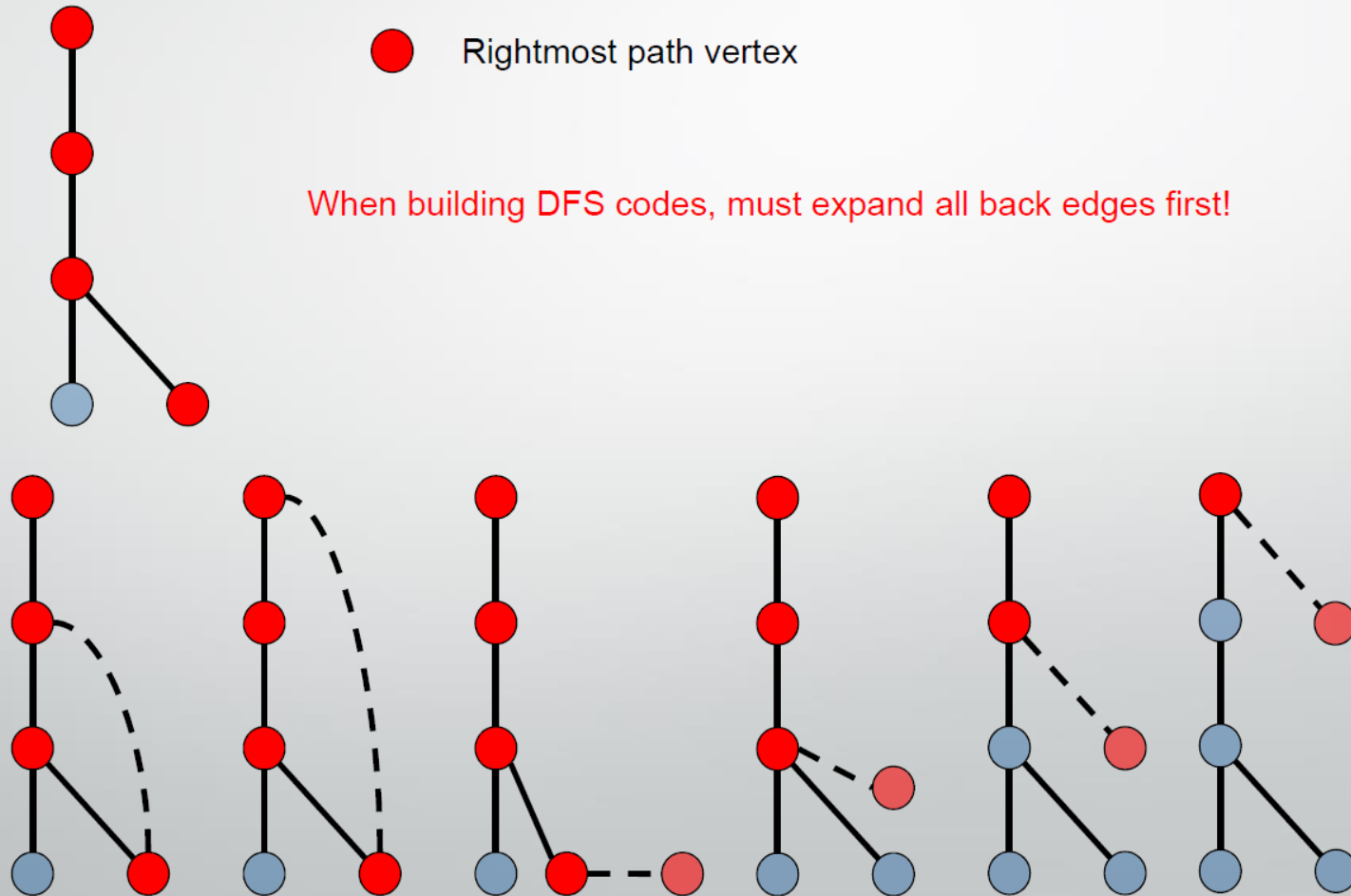
$i > j$: back edge

Frequent subgraph mining (FSM)

gSpan

- Complete (finds all frequent subgraphs) FSM algorithm on labeled undirected graphs
- Generates candidates by adding edges (one at a time) to patterns already discovered
- Encodes patterns (graphs) according to DFS traversal order (DFS code)
 - Lexicographically minimal code for each graph
- Building candidates by rightmost expansion – adding of an edge to a vertex on the rightmost path

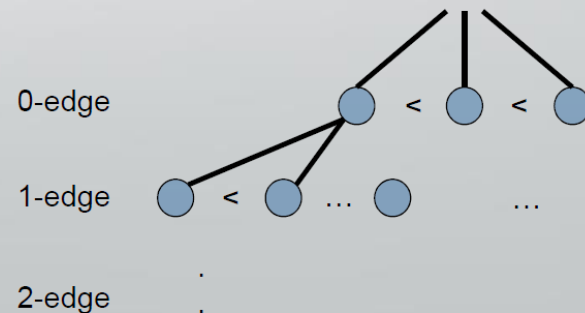
Frequent subgraph mining (FSM)



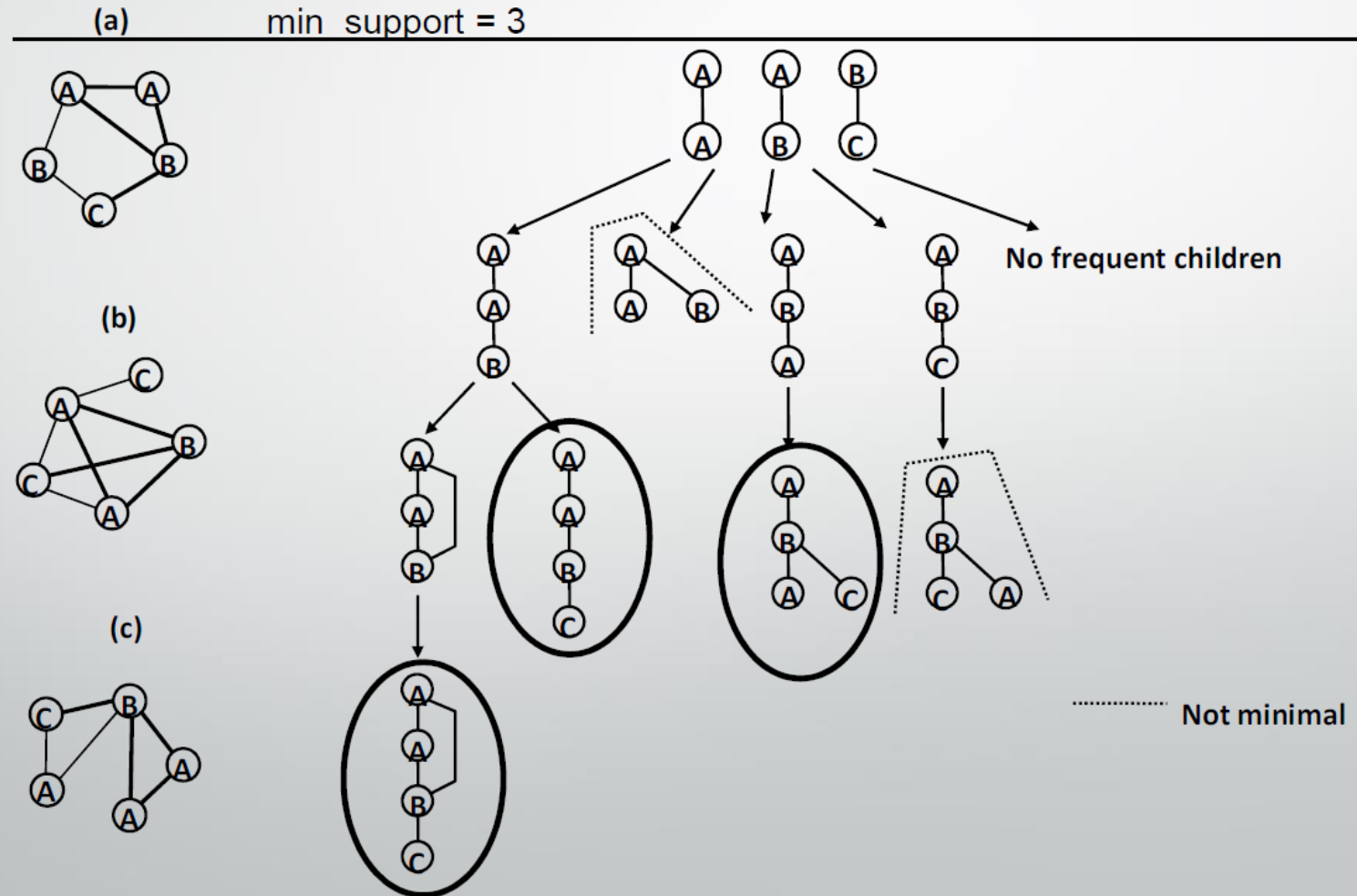
Frequent subgraph mining (FSM)

gSpan

- Complete (finds all frequent subgraphs) FSM algorithm on labeled undirected graphs
- Generates candidates by adding edges (one at a time) to patterns already discovered
- Encodes patterns (graphs) according to DFS traversal order (DFS code)
 - Lexicographically minimal code for each graph
- Building candidates by rightmost expansion – adding of an edge to a vertex on the rightmost path
- Uses DFS Code Tree to keep all possible DFS codes



Frequent subgraph mining (FSM)



Frequent subgraph mining (FSM)

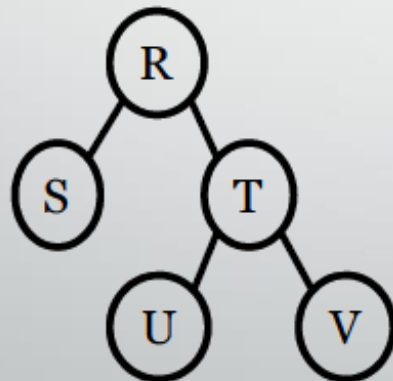
Subdue

- Approximate FSM algorithm on labeled graphs or a single graph
- Directed or undirected graphs
- Not based on support
- Outputs structures which compress the input graph well

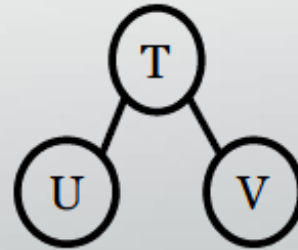
Frequent subgraph mining (FSM)

Sleuth

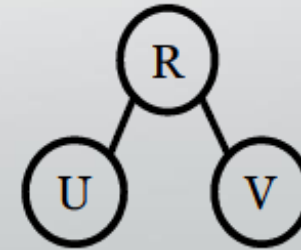
- FSM algorithm on rooted trees – both ordered and unordered
- Mines *induced* (can only contain edges from the original tree) or *embedded* (can have edges between ancestors and descendants) subtrees



Original



Induced



Embedded

Classification, regression

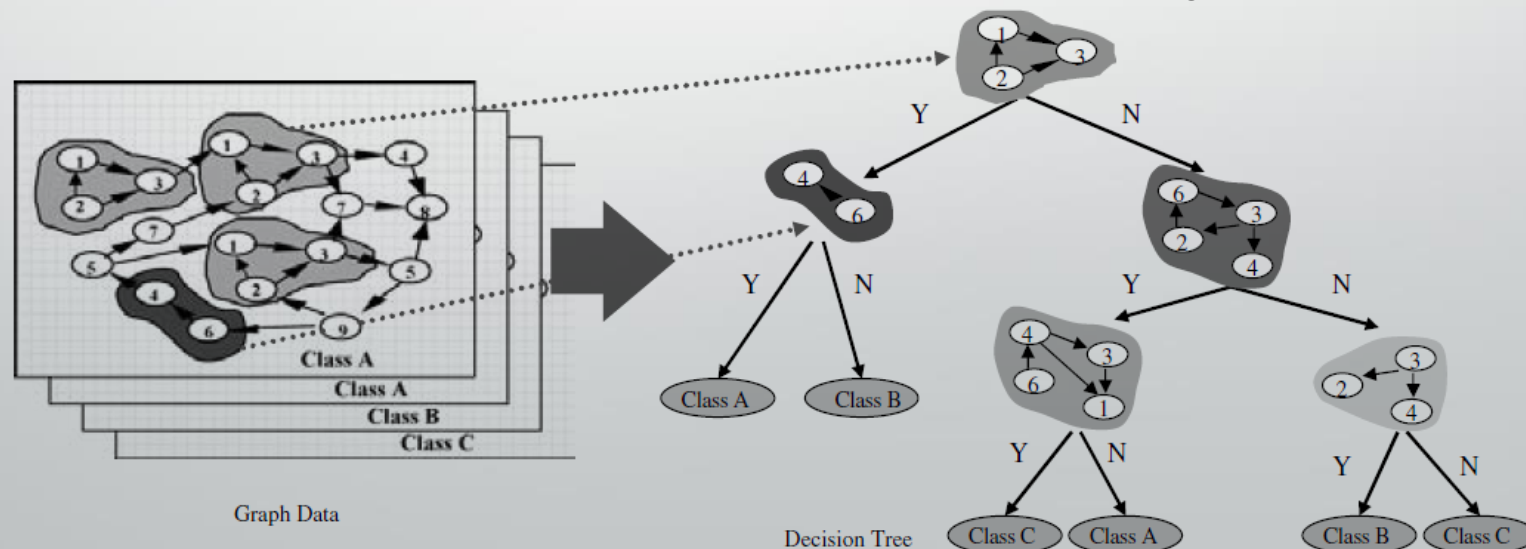
- Link-based Object Classification
 - assign class labels to nodes according to their link characteristics (e.g. node degree, average path length, ...)
 - Applications: From a graph with persons as nodes and preferences as edges, select set of individuals (e.g., as team leaders) and then assign groups to everyone else

Classification, regression

- Link-based Object Classification
 - assign class labels to nodes according to their link characteristics (e.g. node degree, average path length, ...)
 - Applications: From a graph with persons as nodes and preferences as edges, select set of individuals (e.g., as team leaders) and then assign groups to everyone else
- Link-based Object Ranking
 - all nodes are usually understood to be of the same type, the goal is to associate a relative quantitative assessment with each node
 - Applications: ranking web pages by a search engine (PageRank)

Classification, regression

- Between-graph classification – each graph is assigned to a class
 - Methods based on substructures – for example: gBoost, DT-CLGBI (figure)
 - Attribute: a pattern/subgraph in graph graph-structured data
 - Value of an attribute: existence/nonexistence of the pattern in each graph



Classification, regression

- Between-graph classification – each graph is assigned to a class
 - Methods based on substructures – for example: gBoost, DT-CLGBI
 - Attribute: a pattern/subgraph in graph graph-structured data
 - Value of an attribute: existence/nonexistence of the pattern in each graph
 - It is also possible to use a FSM algorithm and then an arbitrary classification algorithm for attribute-value data

Classification, regression

- Between-graph classification – each graph is assigned to a class
 - Methods based on substructures – for example: gBoost, DT-CLGBI
 - Attribute: a pattern/subgraph in graph graph-structured data
 - Value of an attribute: existence/nonexistence of the pattern in each graph
 - It is also possible to use a FSM algorithm and then an arbitrary classification algorithm for attribute-value data
 - Kernel methods (e.g. Direct Product Kernel) -> kernel-based classifier (e.g. SVM)
 - Applications: classifying molecules as toxic or non-toxic

Classification, regression

- Within-graph classification – each node in a graph is assigned to a class
 - Kernel methods (e.g. Regularized Laplacian Kernel) -> kernel-based classifier (e.g. SVM)
 - Applications: classification of web pages

Clustering

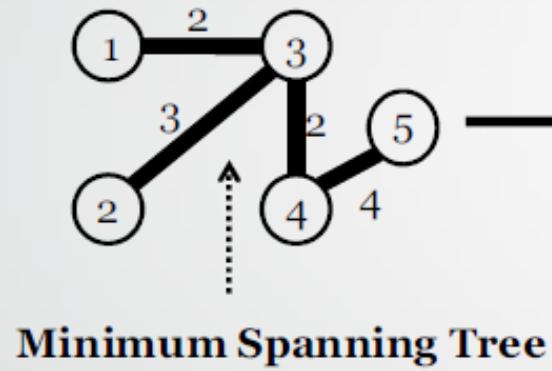
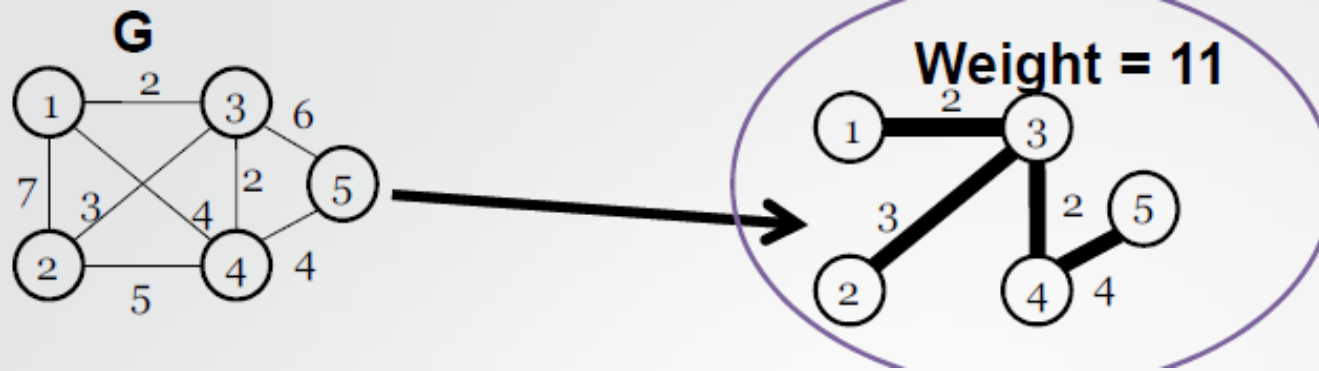
Clustering graphs

- Divides a set of graphs into different clusters
- Applications: grouping of chemical compounds into clusters based on their structural similarity
- Kernels -> kernel k -means clustering
- Regular clustering methods on graphs represented by frequent substructures

Clustering

Clustering nodes

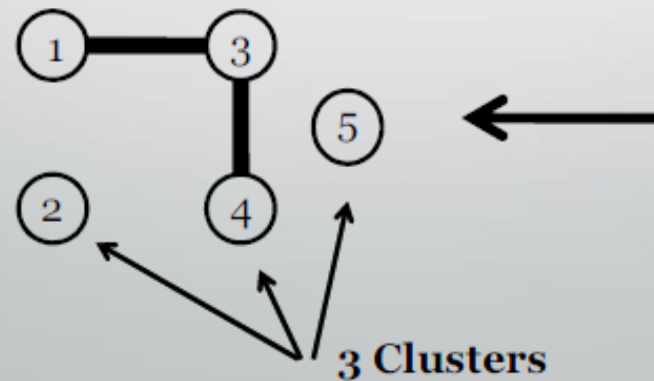
- Divides the nodes of a graph into clusters
- Applications: clustering of people in social networks; clusters group people with similar interests
- Minimum spanning tree clustering
 1. Find minimum spanning tree of a graph
 2. Remove $k-1$ edges with the highest weight \rightarrow k clusters



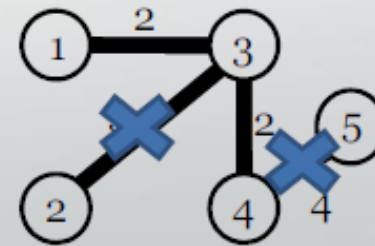
**Remove $k-1$ edges
with highest weight**

Note: k – is the
number of
clusters

E.g., $k=3$



E.g., $k=3$



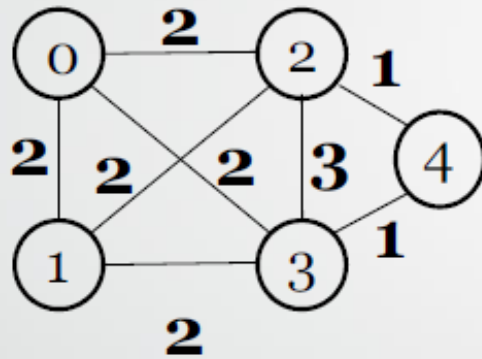
Clustering

Clustering nodes

- Shared Nearest Neighbor (SNN) Clustering
 - Place a pair of objects into the same cluster, if the number of common neighbors they share is more than some *threshold* τ .

Clustering

SNN graph of input graph G



**If u and v share more than τ neighbors
Place them in the same cluster**

E.g., $\tau = 3$




Clustering

Clustering nodes

- Shared Nearest Neighbor (SNN) Clustering
 - Place a pair of objects into the same cluster, if the number of common neighbors they share is more than some *threshold* τ .
- Maximal Clique Enumeration

Examples of other tasks

- Outlier detection
 - Searching for outlying nodes or whole graphs
 - Methods for measuring similarity and clustering can be used
- Association rules
 - Frequent subgraphs may be used as frequent patterns in classic algorithms
- Link prediction
 - In dynamic graphs; task is to predict new edges that will be probably added to a graph



Thank you.

References and resources

- Samatova, N., et al.: *Practical Graph Mining with R*. CRC Press (2013)
 - Slides from the book webpage:
<http://www.csc.ncsu.edu/faculty/samatova/practical-graph-mining-with-R/PracticalGraphMiningWithR.html>
- Cook, D. J., Holder, L. B.: *Mining Graph Data*. John Wiley & Sons (2007)