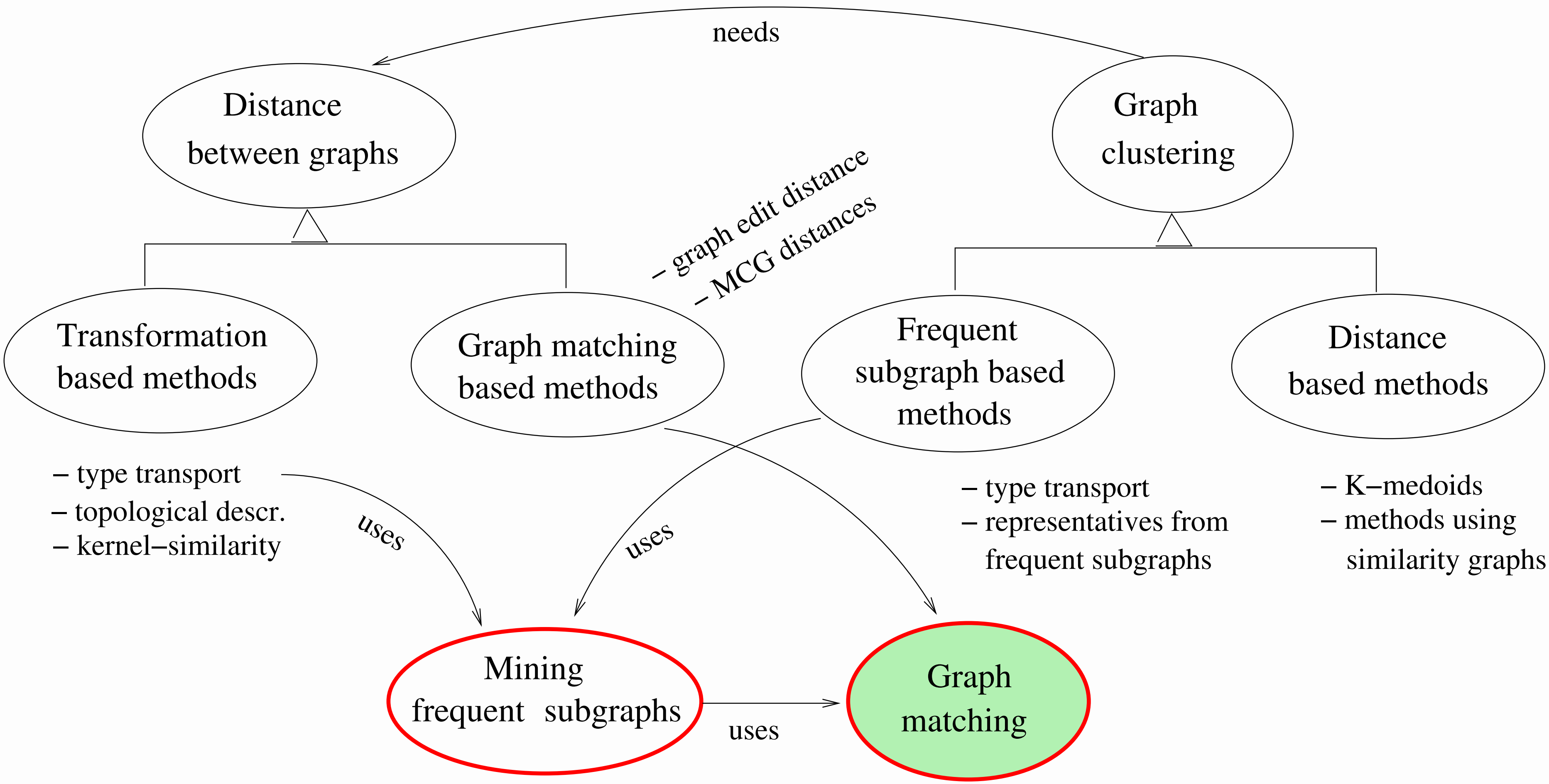
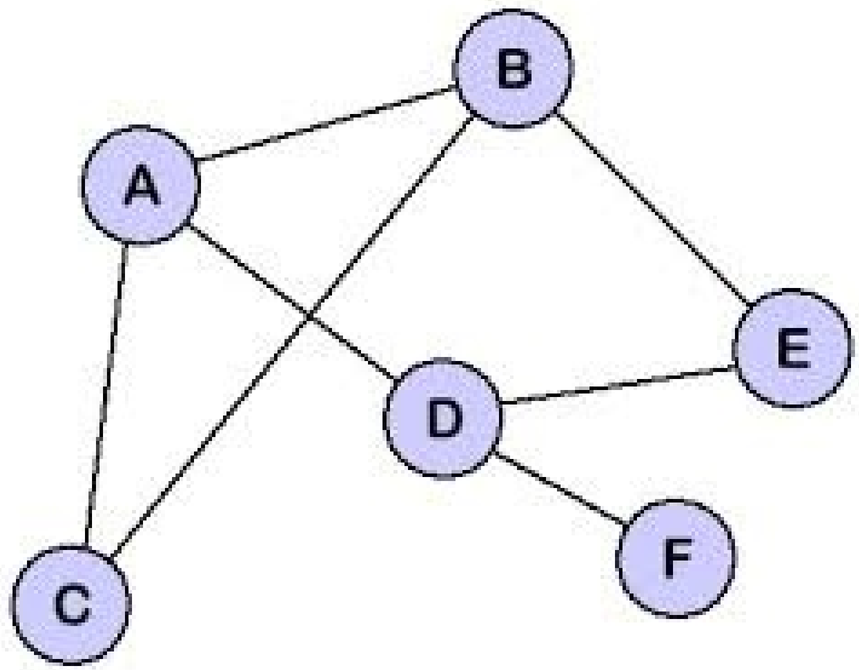
Mining database of multiple graphs



Graph notations

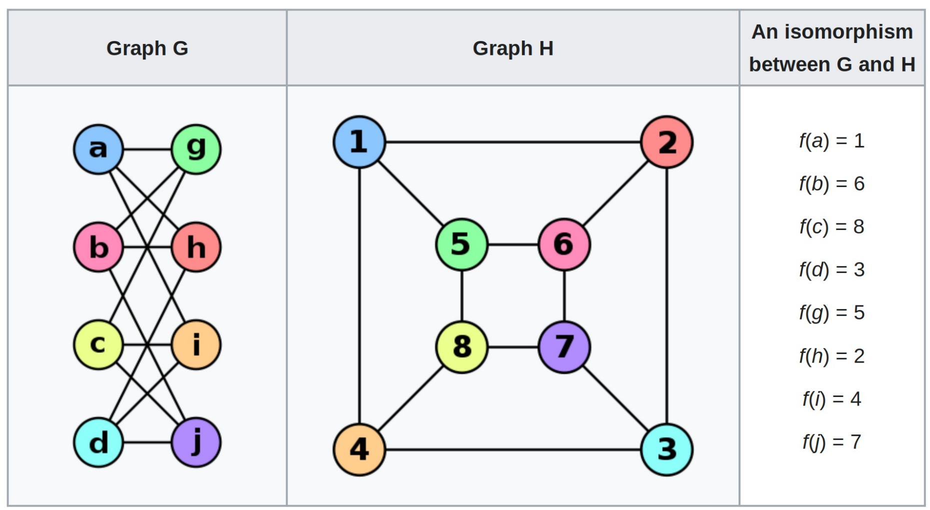


- $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ graph
- $\mathbf{V} = \{v_1, \ldots, v_n\}$ = set of vertices or nodes
- $|\mathbf{V}|$ = number of nodes
- node label $l(v_i)$
- $\mathbf{E} = \{e_1, \ldots, e_m\}$ = set of edges, $e_i = (v, u)$, $v, u \in V$
- $|\mathbf{E}|$ = number of edges

Now we assume that edges undirected and don't have labels

Graph isomorphism of unlabelled graphs

Two unlabelled graphs $\mathbf{G}_1 = (\mathbf{V}, \mathbf{E})$ and $\mathbf{G}_2 = (\mathbf{U}, \mathbf{F})$ are **isomorphic** or **matching** if there is an edge-preserving bijection $f : \mathbf{V} \to \mathbf{U}$ such that for any $v_1, v_2 \in \mathbf{V}$:
$(v_1, v_2) \in \mathbf{E} \Leftrightarrow (f(v_1), f(v_2)) \in \mathbf{F}$.



Matching can be presented as $\mathcal{M} = \{(v, u) \mid v \in V, u \in U, u = f(v)\}$

Image source `https://en.wikipedia.org/wiki/Graph_isomorphism`

Graph isomorphism of labelled graphs

Two labelled graphs $\mathbf{G}_1 = (\mathbf{V}, \mathbf{E})$ and $\mathbf{G}_2 = (\mathbf{U}, \mathbf{F})$ are **isomorphic**, if there is an edge- and label-preserving bijection $f : \mathbf{V} \rightarrow \mathbf{U}$ such that

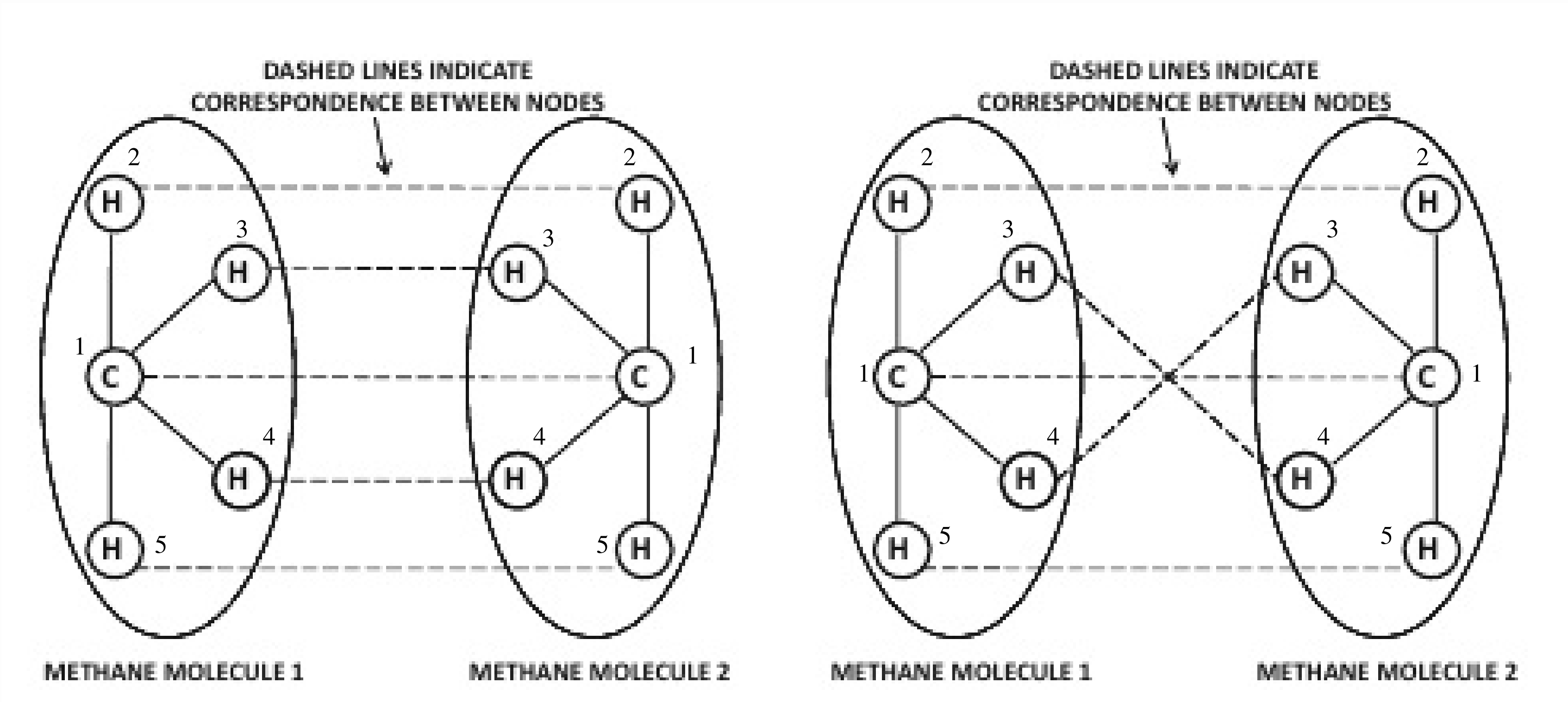
(i) Corresponding nodes have same labels: $\forall v \in \mathbf{V}$ and $f(v) \in \mathbf{U}$ $l(v) = l(f(v))$.

(ii) An edge between matched nodes exists in G_1 iff the corresponding edge exists in G_2: $\forall v_1, v_2 \in \mathbf{V}$: $(v_1, v_2) \in \mathbf{E} \Leftrightarrow (f(v_1), f(v_2)) \in \mathbf{F}$.

Note: no polynomial time algorithms are known (except special cases)

There can be many matchings!

Two matchings for molecules 1 and 2. Totally 4!=24 matchings!



$\mathcal{M} = \{(1,1),(2,2),(3,3),(4,4),(5,5)\}$ $\mathcal{M} = \{(1,1),(2,2),(3,4),(4,3),(5,5)\}$

Image source: Aggarwal Fig. 17.2

Subgraph isomorphism

Does a certain **query graph** $\mathbf{G}_q$ match a part of another graph $\mathbf{G}$?

Query graph $\mathbf{G}_q = (\mathbf{V}, \mathbf{E})$ is a **subgraph isomporphism** of $\mathbf{G} = (\mathbf{U}, \mathbf{F})$, if there is an injection $f : \mathbf{V} \to \mathbf{U}$ such that

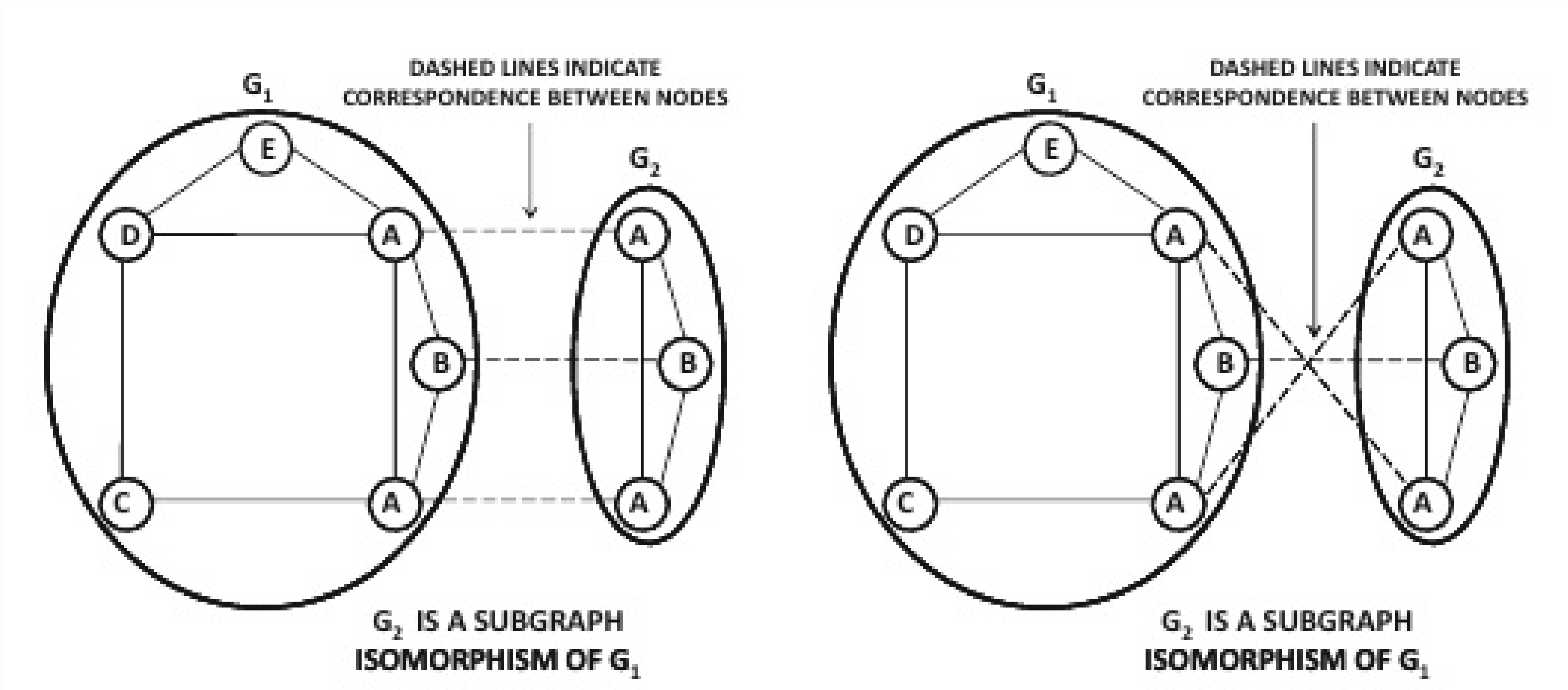
(i) For all $v \in \mathbf{V}$ there is $f(v) \in \mathbf{U}$ such that $l(v) = l(f(v))$; and

(ii) For any $v_1, v_2 \in \mathbf{V}$: $(v_1, v_2) \in \mathbf{E} \Leftrightarrow (f(v_1), f(v_2)) \in \mathbf{F}$.

Notes: 1) Usually it is required that the graphs are connected.

2) Sometimes a weaker condition suffices for (ii):
if $(v_1, v_2) \in \mathbf{E} \Rightarrow (f(v_1), f(v_2)) \in \mathbf{F}$

Subgraph isomorphism: example



- Algorithm: see Aggarwal Ch 17.2.1

Image source: Aggarwal Fig. 17.3

Maximum common subgraph (MCG)

Problem: Given $\mathbf{G}_1$ and $\mathbf{G}_2$, find $\mathbf{G}_0 = (\mathbf{V}_0, \mathbf{E}_0)$ such that

(i) $\mathbf{G}_0$ is a subgraph isomorphism of both $\mathbf{G}_1$ and $\mathbf{G}_2$ and

(ii) $|\mathbf{V}_0|$ is as large as possible.

+ useful for comparing graphs
  - distances between graphs
  - frequent subgraph discovery

– NP-hard problem (like subgraph isomorphism)

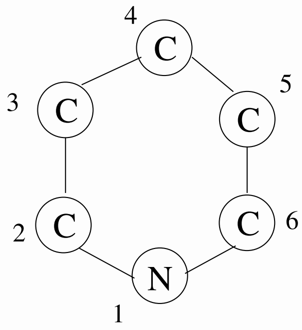
Algorithm for $MCG(G_1, G_2)$

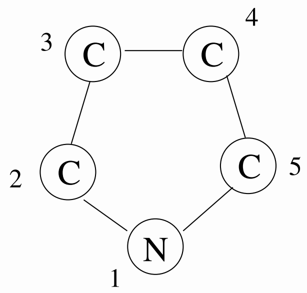
function $MCG(\mathbf{G}_1, \mathbf{G}_2, \mathcal{M}, \mathcal{M}_{best})$
/ Create candidates for matching node pairs */*
$C = \{(v, u) \mid v \in \mathbf{V}, u \in \mathbf{U}, l(v) = l(u), (v, u) \notin \mathcal{M}\}$
Prune C
/ Recursion: */*
for all $(v, u) \in C$
  if valid($\mathcal{M}, (v, u)$)   *// is (u, v) a valid extension?*
    $\mathcal{M}_{best} = MCG(\mathbf{G}1, \mathbf{G}2, \mathcal{M} \cup (v, u), \mathcal{M}_{best})$
if ($|\mathcal{M}| > |\mathcal{M}_{best}|$)
  return $\mathcal{M}$
else return $\mathcal{M}_{best}$

$\mathbf{G}_1 = (\mathbf{V}, \mathbf{E}), \mathbf{G}_2 = (\mathbf{U}, \mathbf{F})$
Call: $MCG(\mathbf{G}_1, \mathbf{G}_2, \emptyset, \emptyset)$

Algorithm: valid extensions

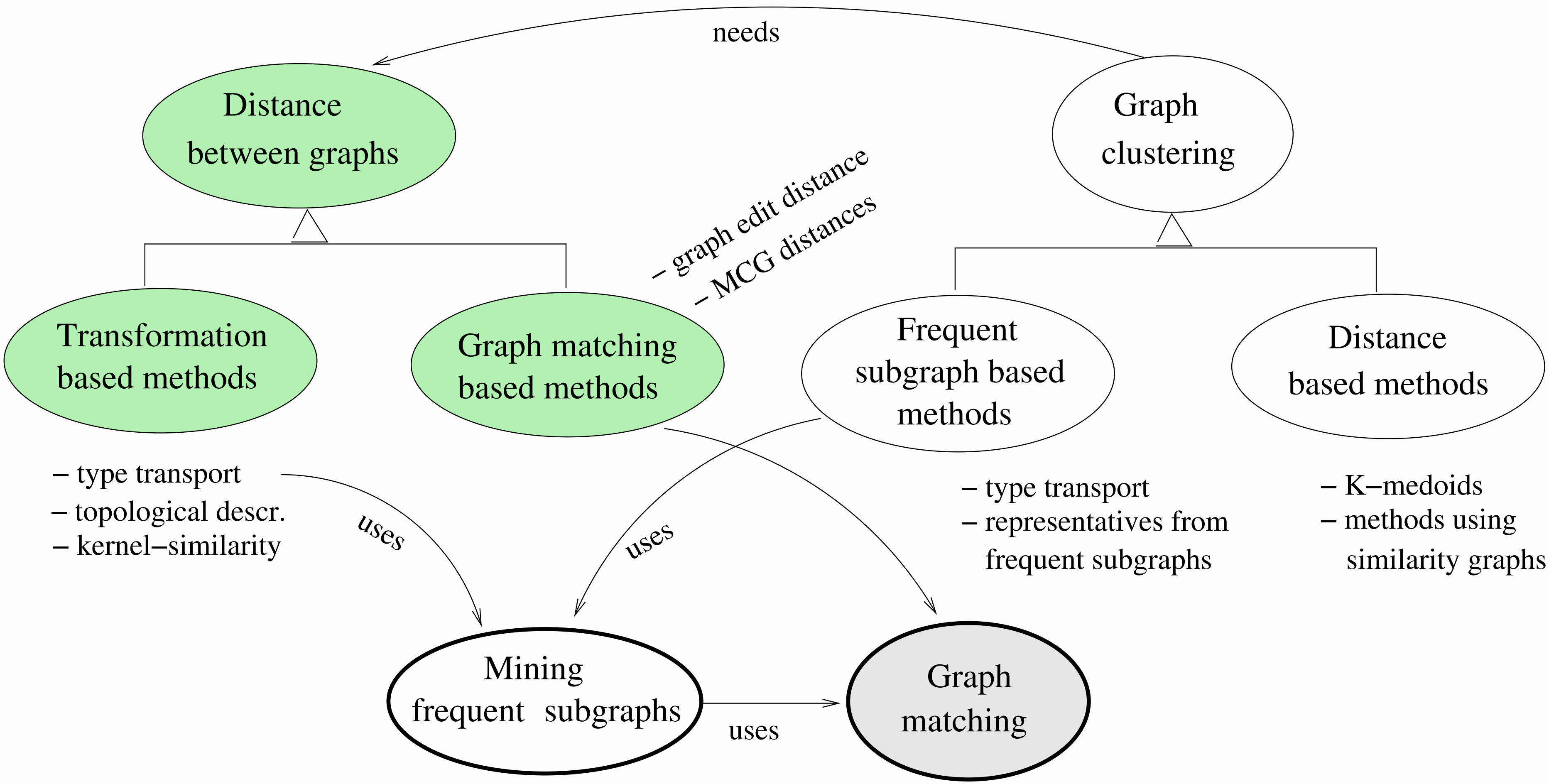
valid$(\mathcal{M}, (v, u))$
if $\big(\exists u_2 \in \mathbf{U} : ((u, u_2) \in F)\&\&((v_2, u_2) \in \mathcal{M})\&\&((v, v_2) \notin E)\big)$ **or**
$\big(\exists v_2 \in \mathbf{V} : ((v, v_2) \in E)\&\&((v_2, u_2) \in \mathcal{M})\&\&((u, u_2) \notin F)\big)$
return 0
else return 1





E.g., $\mathcal{M} = \{(1,1),(2,2),(3,3),(6,5)\}$.
$(4,4)$ is invalid extension – why?
Is there any valid extension?

Next to distances



Distances based on maximum common subgraphs

- Let's assume graph size = number of nodes, i.e., for $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ notate $|\mathbf{G}| = |\mathbf{V}|$
- Let $MCS(\mathbf{G}_1, \mathbf{G}_2)$=maximum common subgraph of $\mathbf{G}_1$ and $\mathbf{G}_2$ and $|MCS(\mathbf{G}_1, \mathbf{G}_2)|$=its size

1. **Unnormalized non-matching measure**:

$$U(\mathbf{G}_1, \mathbf{G}_2) = |\mathbf{G}_1| + |\mathbf{G}_2| - 2 \cdot |MCS(\mathbf{G}_1, \mathbf{G}_2)|$$

- = number on non-matching nodes
- Problem: what if graphs have very different sizes?

Normalized MCS distances

2. **Union-normalized distance** $Udist \in [0, 1]$

$$Udist(\mathbf{G}_1, \mathbf{G}_2) = 1 - \frac{|MCS(\mathbf{G}_1, \mathbf{G}_2)|}{|\mathbf{G}_1| + |\mathbf{G}_2| - |MCS(\mathbf{G}_1, \mathbf{G}_2)|}$$

= number of non-matching nodes normalized by union size

3. **Max-normalized distance** $Mdist \in [0, 1]$

$$Mdist(\mathbf{G}_1, \mathbf{G}_2) = 1 - \frac{|MCS(\mathbf{G}_1, \mathbf{G}_2)|}{\max\{|\mathbf{G}_1|, |\mathbf{G}_2|\}}$$

- metric

MCS distances can be computed efficiently **only for small graphs!**

Graph edit distance

What is the minimum cost of edit operations to transform $\mathbf{G}_1$ to $\mathbf{G}_2$?

(i) node insertion

(ii) node deletion (deletes also incident edges)

(iii) edge insertion

(iv) edge deletion

(v) label substitution of nodes

- application-specific costs
- may be exponentially many possible edit paths!
- NP-hard

Graph edit distance: example

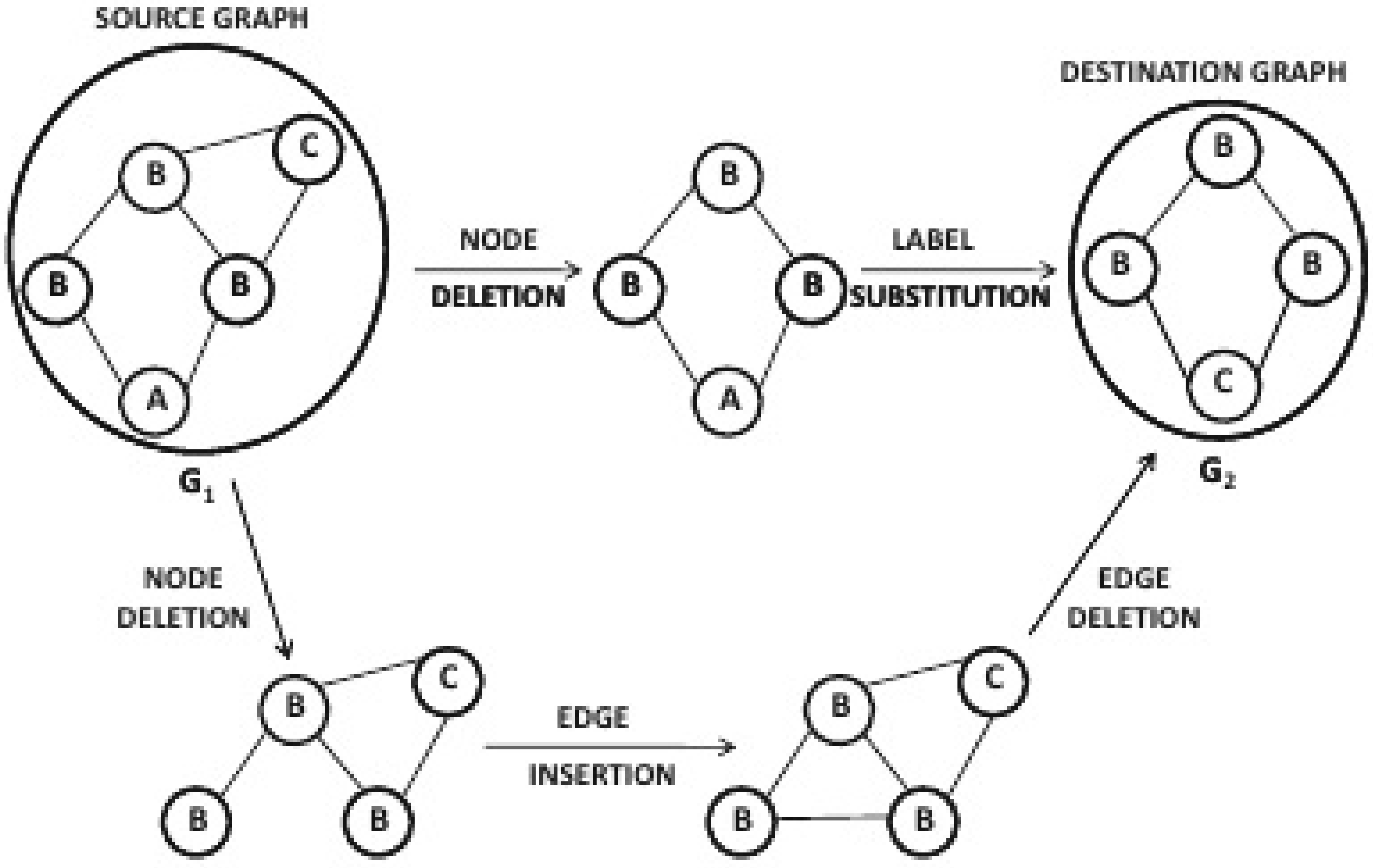


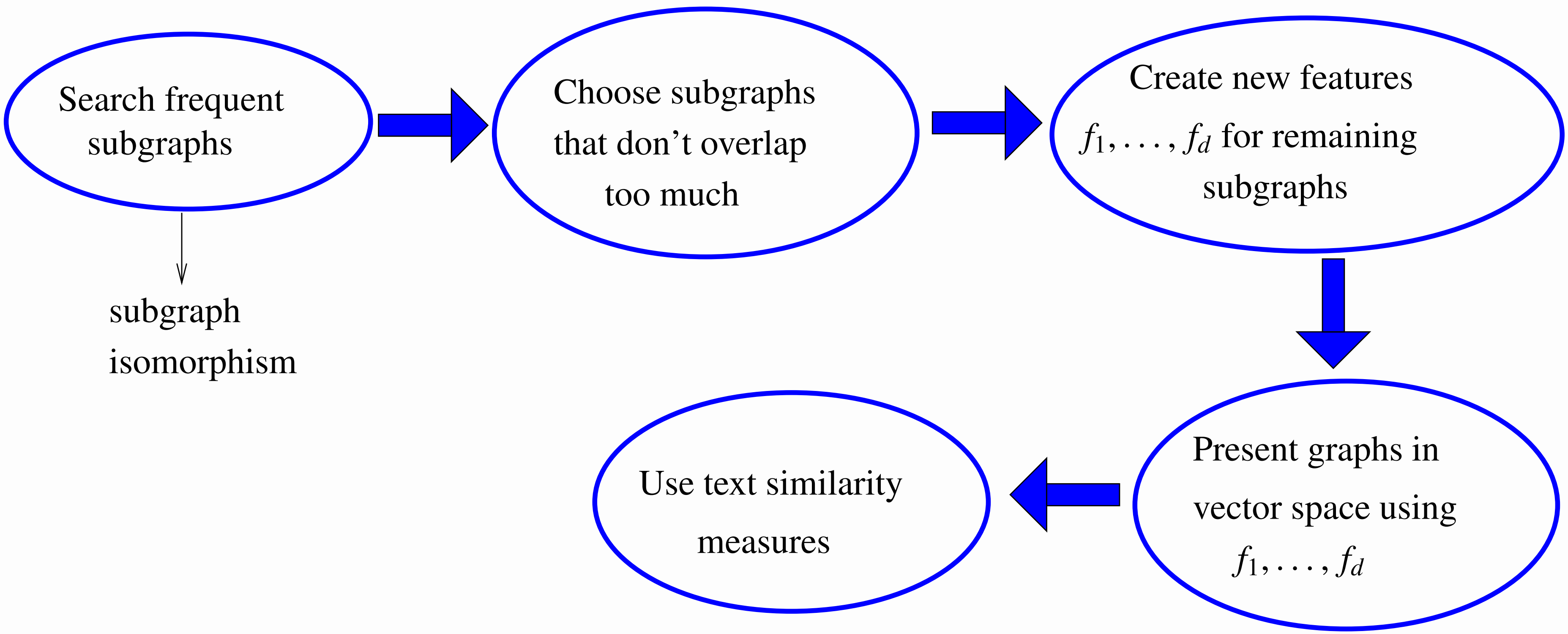
Image source: Aggarwal Fig. 17.6

Transformation-based distances

Idea: Transform graphs into a new space where distances are easier to calculate

a) Type transport using frequent subgraphs
b) Topological descriptors
c) Kernel similarity

Type transport using frequent subgraphs



f_i = number of times
ith subgraph occurs in G
or binary or tf-idf presentation

involves an NP-hard subproblem

Topological descriptors

Idea: calculate different kinds of indices from graphs $\Rightarrow$ new numerical features $\Rightarrow$ Use distances for numerical data

- structural information lost
- utility domain-specific (e.g., good in chemical domain)
- e.g., Wiener index:

$$W(\mathbf{G}) = \sum_{v,u \in \mathbf{V}} d(v,u)$$

  $d(v,u)$=length of shortest path from v to u

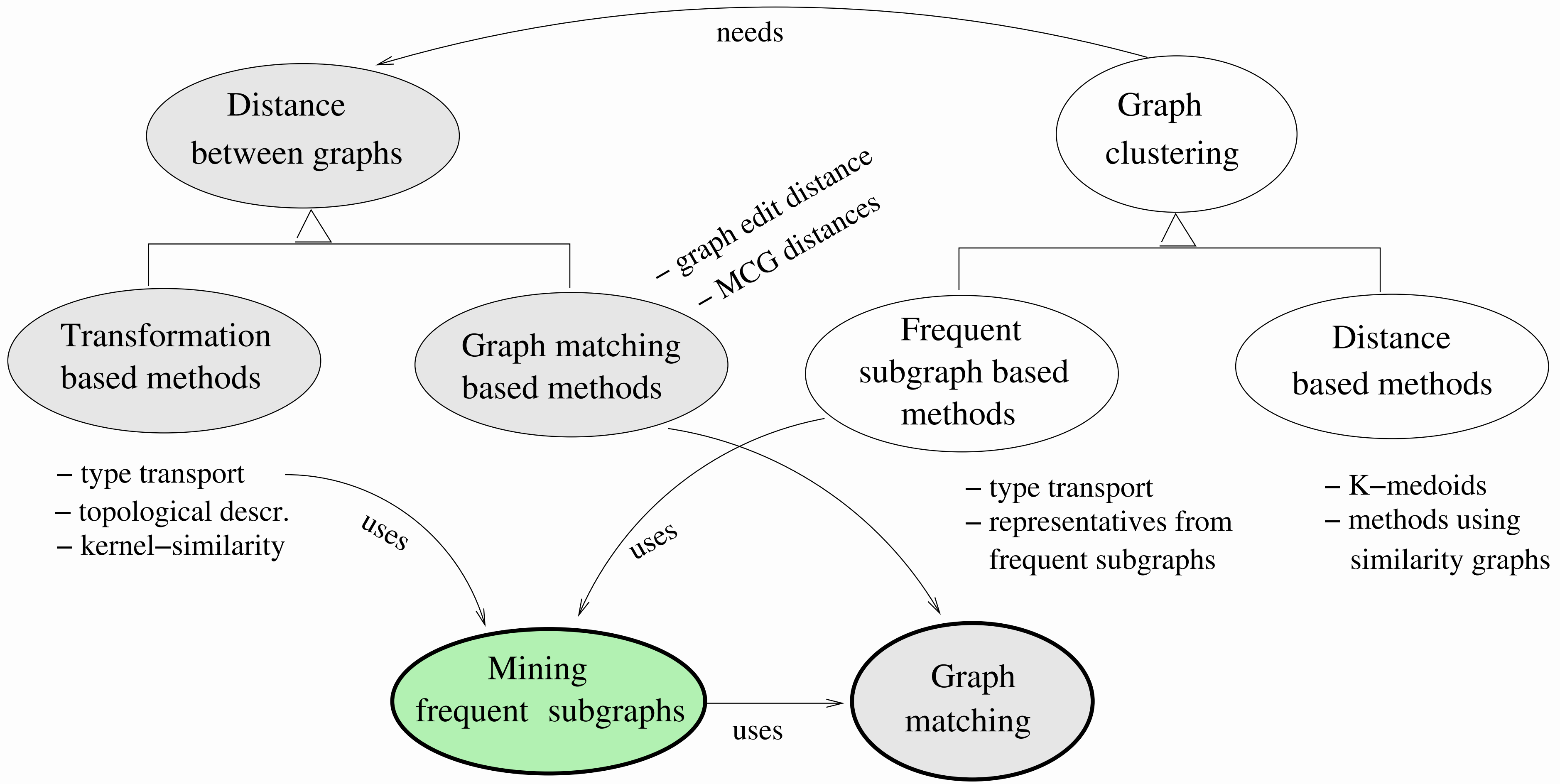
- more in Aggarwal Ch 17.3.2

Kernel similarity

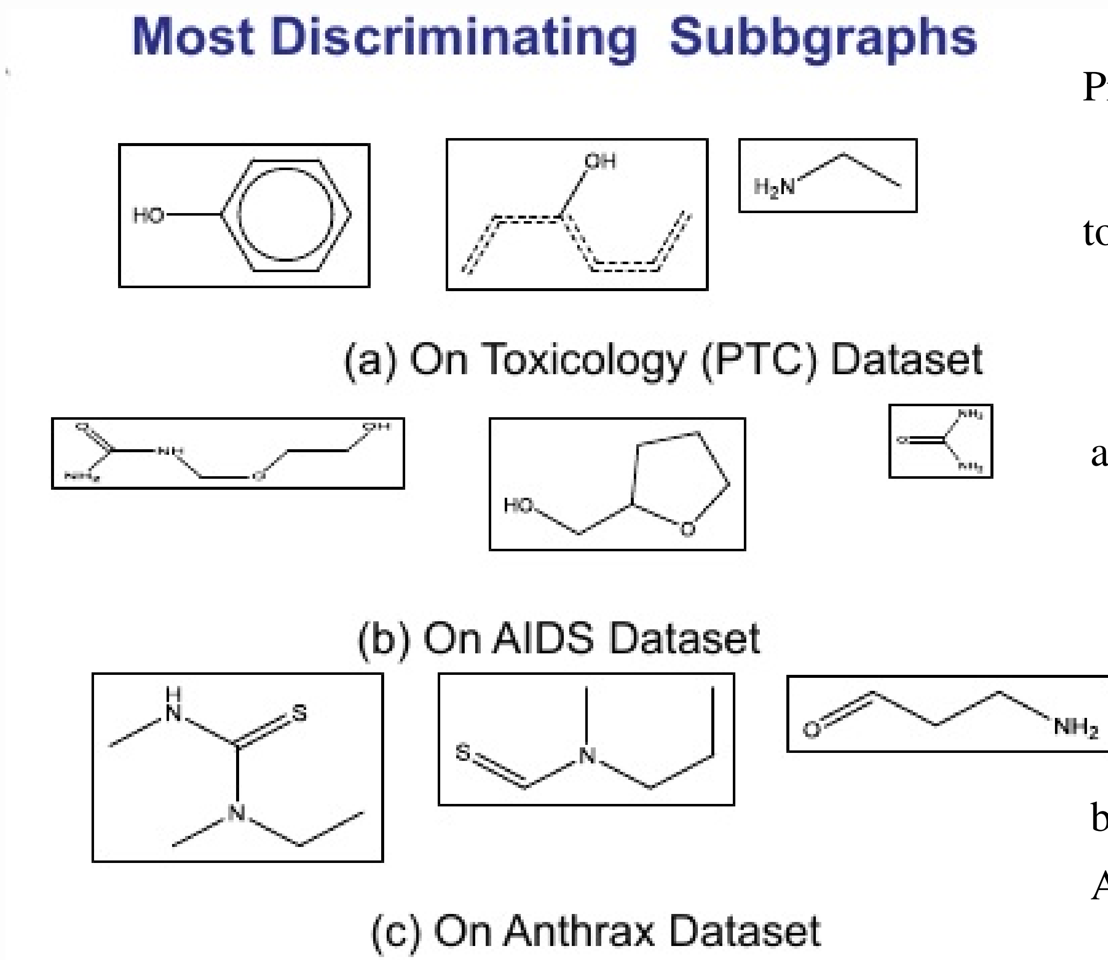
Idea:

- Assume transformation Φ such that similarity of $\mathbf{G}_1$ and $\mathbf{G}_2$ can be measured by $\Phi(\mathbf{G}_1) \cdot \Phi(\mathbf{G}_2)$
- Design **kernel function** K such that $K(\mathbf{G}_1, \mathbf{G}_2) = \Phi(\mathbf{G}_1) \cdot \Phi(\mathbf{G}_2)$ and use it as a similarity measure (without transformation)
- e.g. shortest path kernel ($O(n^4)$) and random walk kernel ($O(n^6)$)
- practical for small graphs
- more in Aggarwal Ch 17.3.3

Next to frequent subgraph discovery



Frequent subgraph discovery: Motivation



Predict:

toxicity of compounds

anti–HIV activity

binding ability with

Anthrax toxin

Image source: https://slideplayer.com/slide/5894097/

Frequent subgraph discovery

Task: Given graph database, search frequent subgraphs given threshold $\min_{fr}$.

- Search idea: utilize **monotonicity of frequency**!
- If $\mathbf{G}_1$ is a subgraph of $\mathbf{G}_2$, then $fr(\mathbf{G}_1) \geq fr(\mathbf{G}_2)$
- similar algorithms than for frequent itemsets, but more complex
- two variants: size of graph may refer to a) number of nodes b) number of edges
  $\Rightarrow$ how new candidates are generated

GraphApriori algorithm

$\mathcal{F}_i$ = frequent subgraphs of size i, $\mathcal{C}_i$ = candidates

- $\mathcal{F}_1 = \{\mathbf{G} \mid \text{where } |\mathbf{G}| = 1, P(\mathbf{G}) \geq \min_{fr}\}$; $i = 1$
- while $\mathcal{F}_i \neq \emptyset$
  - generate candidates $\mathcal{C}_{i+1}$ from $\mathcal{F}_i$
  - prune $\mathbf{G} \in \mathcal{C}_{i+1}$ if $\mathbf{G}$ has a subgraph $\mathbf{G}'$ such that $|\mathbf{G}'| = i$ and $\mathbf{G}' \notin \mathcal{F}_i$ (=**monotonicity criterion**)
  - count frequencies $fr(\mathbf{G})$, $\mathbf{G} \in \mathcal{C}_{i+1}$
  - set $\mathcal{F}_{i+1} = \{\mathbf{G} \in \mathcal{C}_{i+1} \mid P(\mathbf{G}) \geq \min_{fr}\}$
  - $i = i + 1$
- return $\cup_i \mathcal{F}_i$

GraphApriori: Candidate generation

For all $\mathbf{G}_1, \mathbf{G}_2 \in \mathcal{F}_i$, $|\mathbf{G}_1| = |\mathbf{G}_2| = i$

1. determine if $\mathbf{G}_1$ and $\mathbf{G}_2$ have a common subgraph $\mathbf{G}_0$ of size $i - 1$
   - may be many isomorphic matchings $\Rightarrow$ **many alternative $\mathbf{G}_0$s!**
2. for each $\mathbf{G}_0$ create candidate graphs of size $i + 1$
   - **node-based**: include all common + 2 non-matching nodes (with extra edge or not)
   - **edge-based**: include all $i - 1$ common edges and 2 unique edges (with extra node or not)

- same subgraphs may be generated multiple times $\Rightarrow$ redundancy checking

Example of node-based join

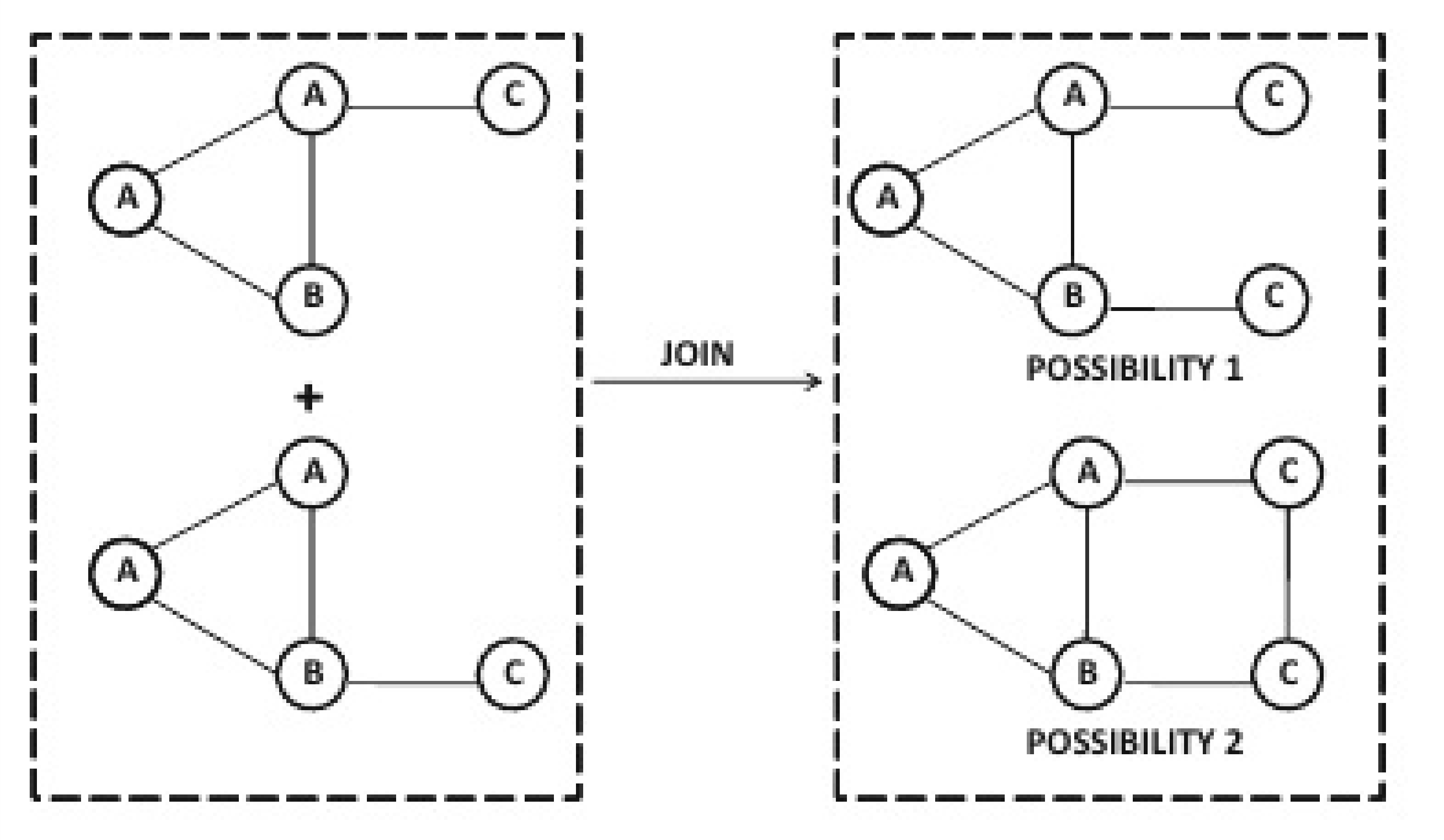


Image source: Aggarwal Fig. 17.12

Example of edge-based join

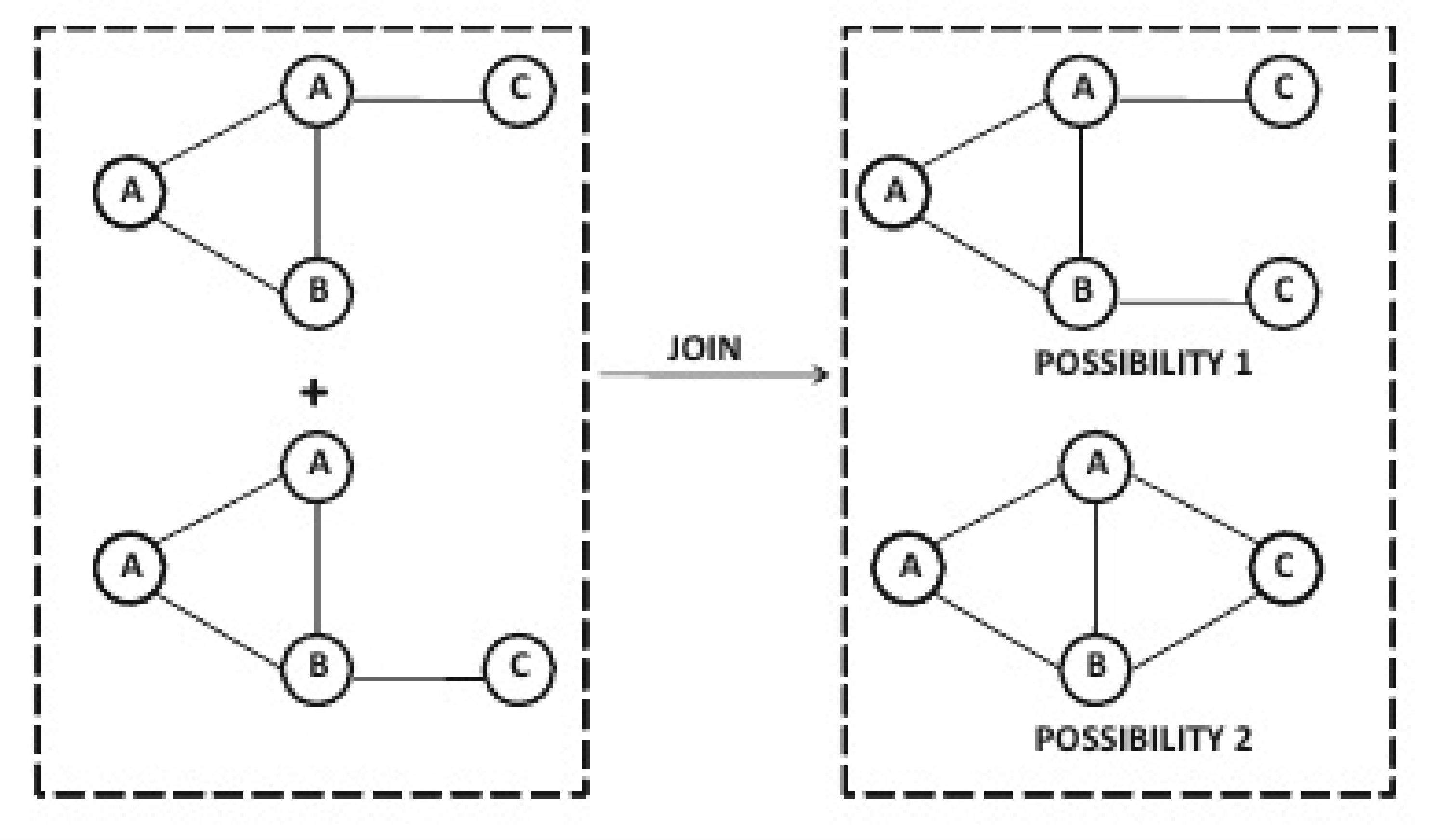


Image source: Aggarwal Fig. 17.13

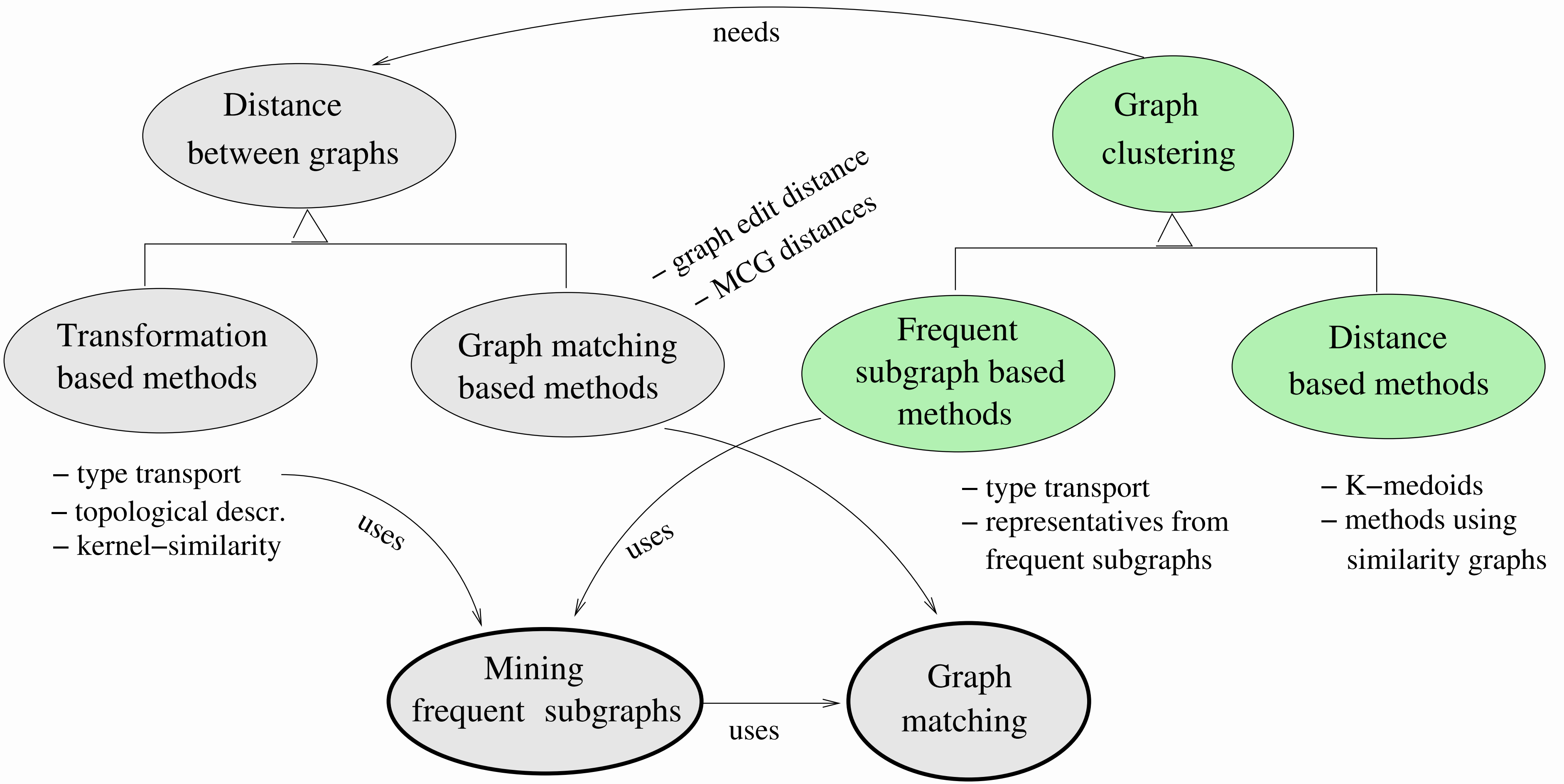
Why this is heavy?

- number of candidate patterns may be huge!
- subgraph isomorphism to identify pairs of subgraphs for joining
- graph isomorphism for redundancy checking
- subgraph isomorphism for monotonicity pruning
- subgraph isomorphism for frequency counting

Easier if

- many unique node labels
- only small subgraphs are searched
- edge-based join is used (usually less candidates)

Next to graph clustering



Distance-based clustering methods

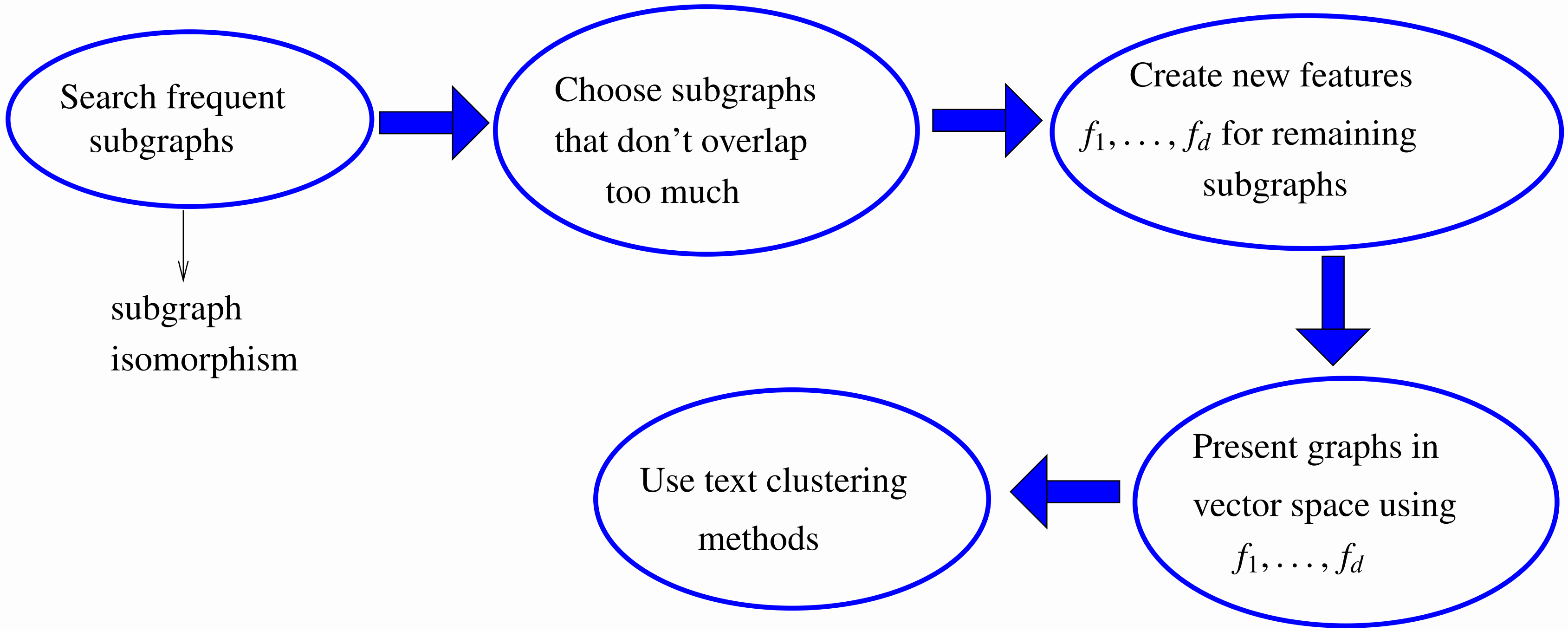
Common approaches:

1. K-medoids (needs just a distance function)
2. Spectral and other graph-based methods
   - construct a nearest neighbour/similarity graph of graph objects
   - cluster nodes of the new graph

Remember: graph distance measures very expensive to compute! $\rightarrow$ suitable for smaller graphs

Methods based on frequent subgraphs

Approach 1. Type transport: graphs → multidimensional



f_i = number of times ith subgraph occurs in G
or binary or tf-idf representation

involves an NP-hard subproblem

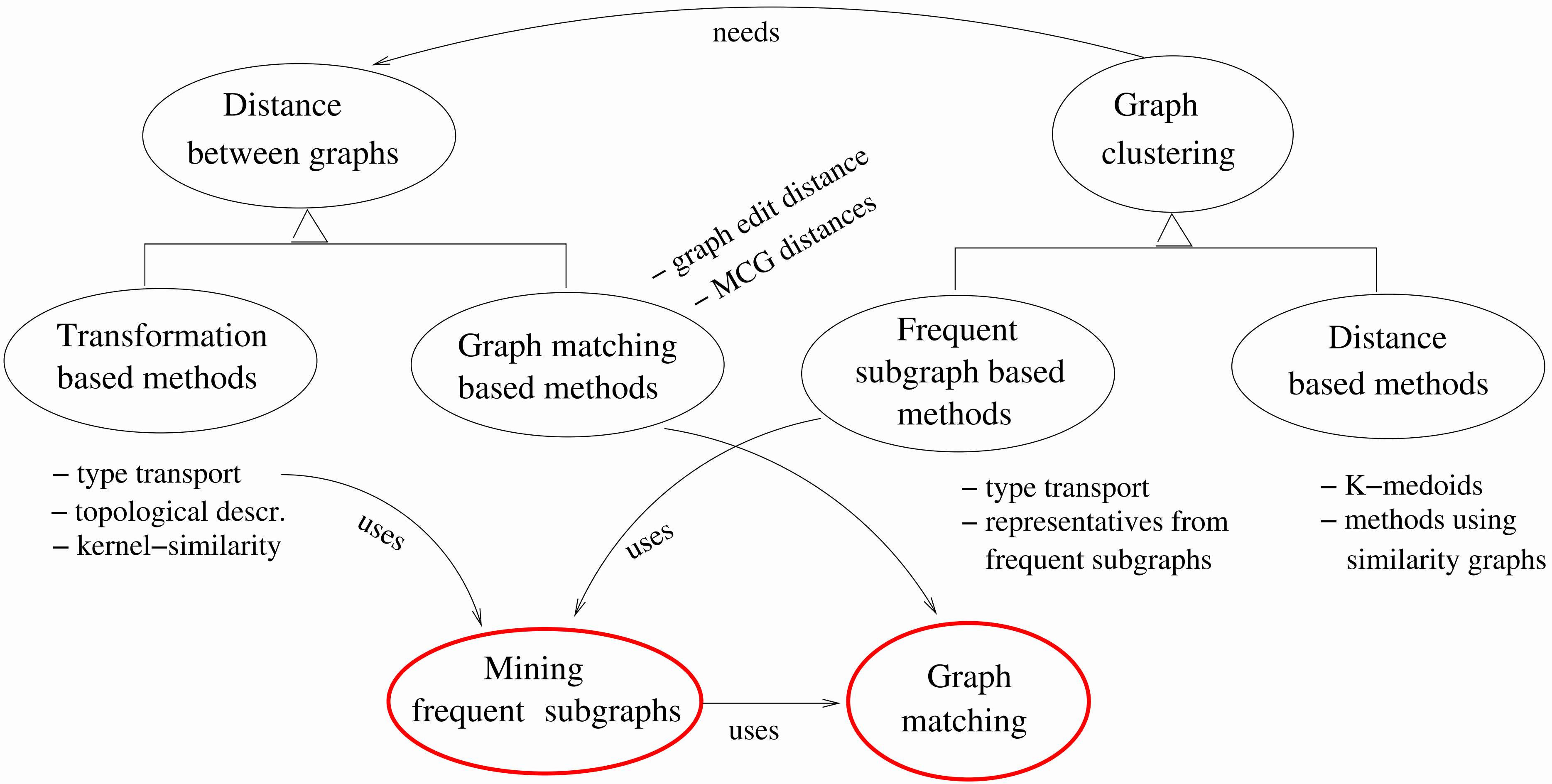
Methods based on frequent subgraphs

Approach 2. XProj: cluster representatives = **sets** of frequent subgraphs

- Initialization: Create K random clusters $C_1, \ldots, C_K$
- for all C_i: $\mathcal{F}_i$ = set of frequent subgraphs (of a given size) from C_i
- repeat until convergence:
  - assign each $\mathbf{G}_j$ to C_i where $sim(G_j, \mathcal{F}_i)$ largest
  - for all C_i determine new $\mathcal{F}_i$

$sim(G_j, \mathcal{F}_i)$ = fraction of frequent graphs in $\mathcal{F}_i$ that occur in $\mathbf{G}_j$

Summary


needs
Distance between graphs
Graph clustering
– graph edit distance
– MCG distances
Transformation based methods
Graph matching based methods
Frequent subgraph based methods
Distance based methods
– type transport
– topological descr.
– kernel–similarity
uses
uses
– type transport
– representatives from frequent subgraphs
– K–medoids
– methods using similarity graphs
Mining frequent subgraphs
Graph matching
uses

Overview of social network analysis

Emphasis:

- Properties of social networks
- Important analysis tasks
- Useful measures and solution principles

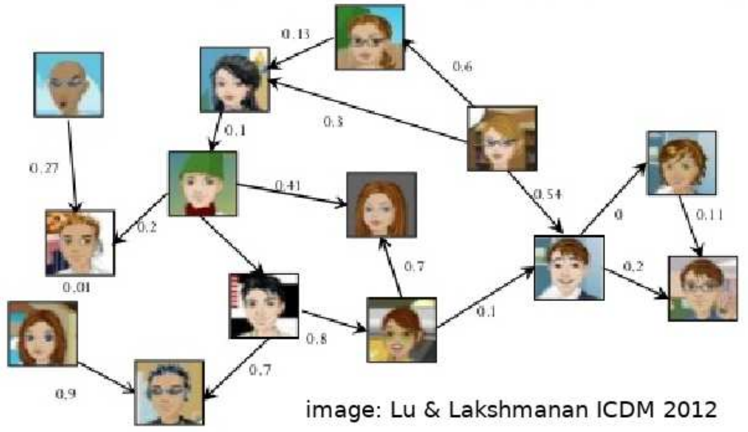


image: Lu & Lakshmanan ICDM 2012

More on course CS-E5740 **Complex Networks**

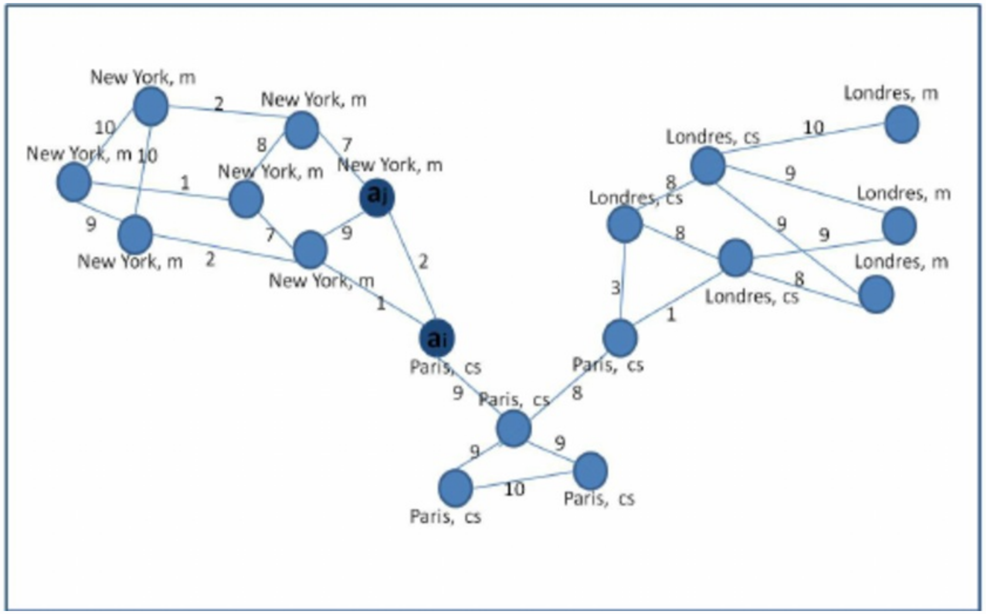
I Introduction: Types of social networks

- online networks (Twitter, LinkedIn, Facebook)
- indirect communication networks (telecommunications, email, chat messages)
- media sharing sites (Youtube, Instagram, Tiktok)
- interaction networks in professional communities (e.g., citation networks between researchers)
- networks recorded in observational studies (e.g., interactions in a class room, between animals)

+ many more! but not always data

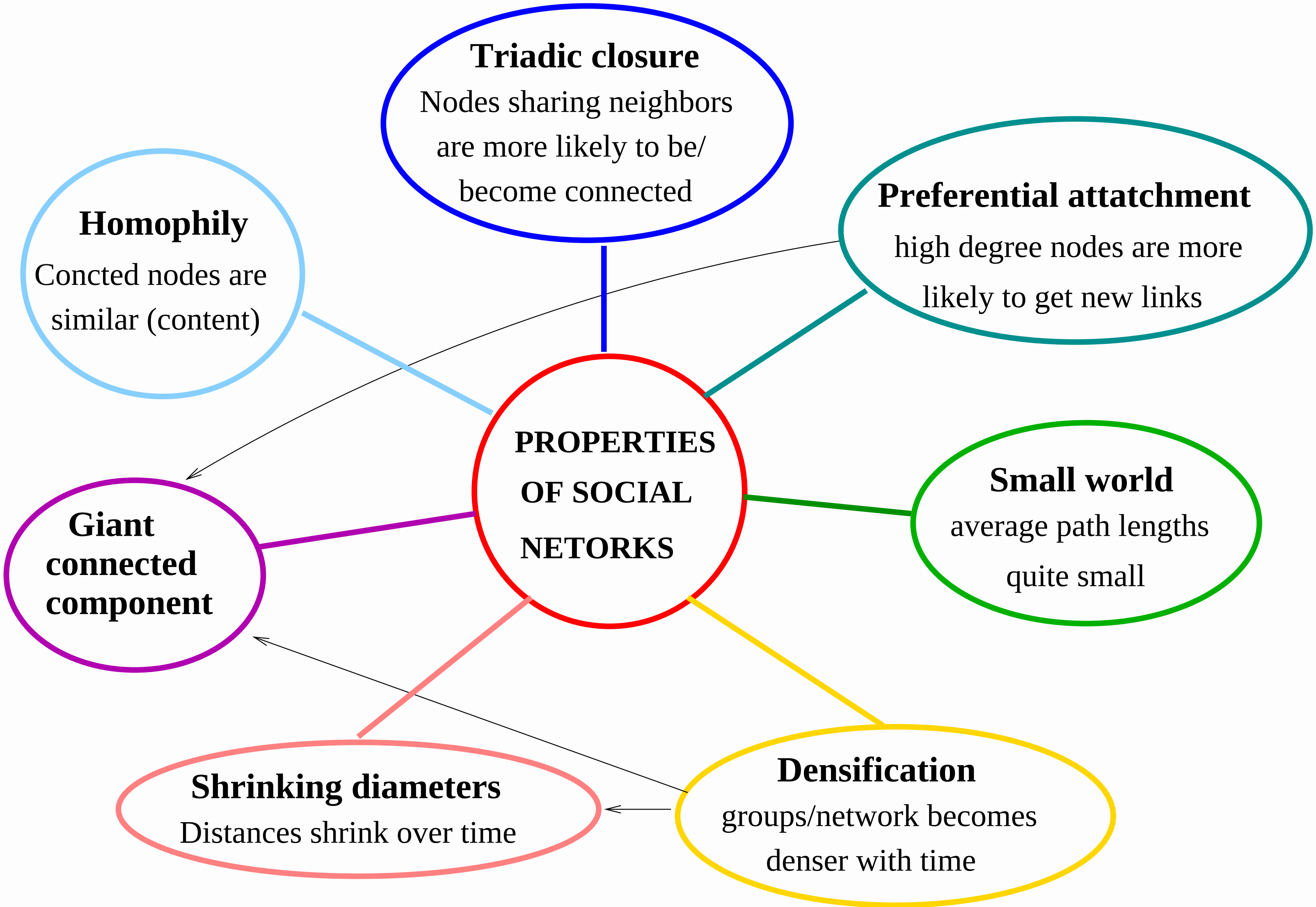
Presentation as a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$

- $\mathbf{V}$ set of nodes corresponding to **actors**
  - may have labels or content (attributes, documents)
- $\mathbf{E}$ set of edges corresponding to links
  - undirected (friendship) or directed (“following”)
  - may have weights w_{ij}



Example by Zardi et al. (2014) node attributes: city and education edge weight = number of exchanged messages

Basic properties



Analysis tasks

- Social influence analysis (influential nodes and influence spread)
- Community detection (graph clustering)
- Link prediction (predict future links between nodes)
- Collective classification (predict missing node labels)

II Social influence analysis

Which nodes have most influence? How influence (information, ideas, opinions) spreads?

A valuable advertising channel!

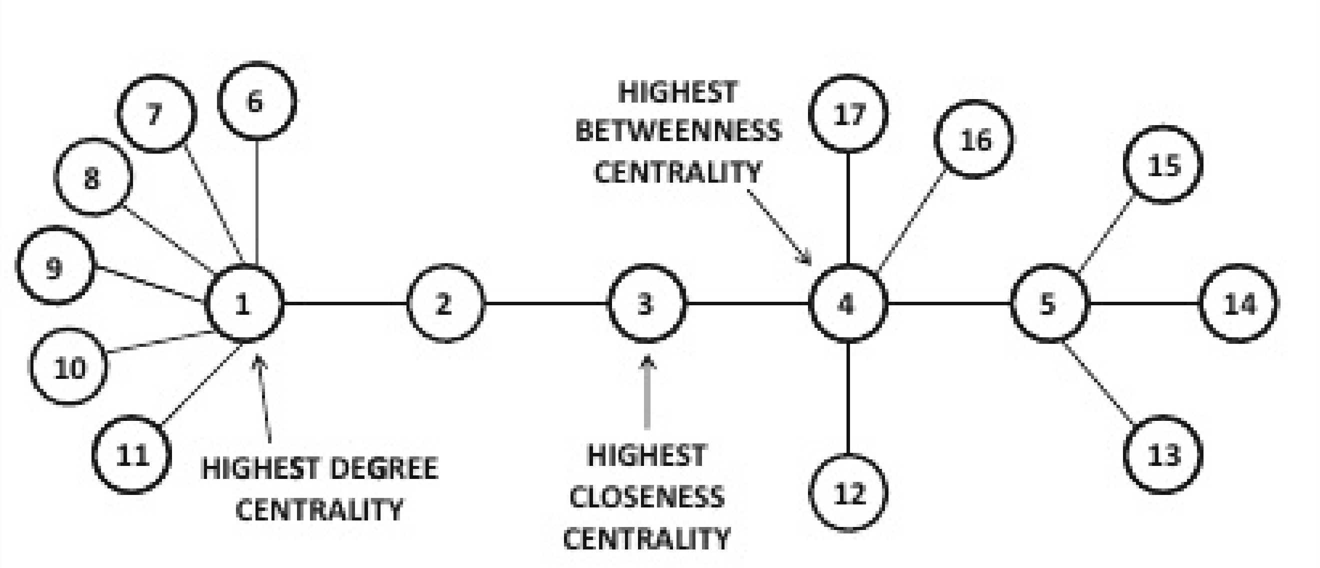
1. Measures for evaluating which nodes are influential:
   - **centrality** of a node in an undirected graph
   - **prestige** of a node in a directed graph
2. **Influence propagation** or **diffusion models**
   - given influence weights on edges and a model to evaluate total influence of a set of nodes
   - determine a set of **seed nodes** such that spread of influence is maximal

Measures for the centrality of node v

Degree centrality: $C_D(v) = \frac{Degree(v)}{n-1}$

Closeness centrality: $C_C(v) = \frac{1}{avg_{u \in V, u \neq v}\{Dist(v,u)\}} = \frac{n-1}{\sum_{u \in V, u \neq v} Dist(v,u)}$

Betweenness centrality: $C_B(v) = \frac{\sum_{u,w \in V, u \neq w} \frac{\#\{\text{shortest-paths(u,w) through v}\}}{\#\{\text{shortest-paths(u,w)}\}}}{\binom{n}{2}}$



Note: $C_c(v)$ may be calculated such $v \neq u$, $v \neq w$. Image: Aggarwal Fig. 19.1

III Community detection: cluster the graph

Given $\mathbf{G} = (\mathbf{V}, \mathbf{E})$. Each edge (v_i, v_j) has weight w_{ij}

- if cost c_{ij}, transform, e.g. by $w_{ij} = \frac{1}{c_{ij}}$ $(c_{ij} \neq 0)$

Common objective: Cluster $\mathbf{V}$ into groups $\mathbf{V}_1, \ldots, \mathbf{V}_K$ such that the edge-cut cost

$$cost(\mathbf{V}_1, \ldots, \mathbf{V}_k) = \sum_{(v_i,v_j) \in E, v_i \in \mathbf{V}_p, v_j \in \mathbf{V}_q, p \neq q} w_{ij}$$

is minimal.

- **many variants and extra constraints!**
- in general NP-hard problem, but polynomially solvable, if $\forall i, j : w_{ij} = 1$, $K = 2$ and no balancing requirements

Example

Clustering based on both structural and content-based features

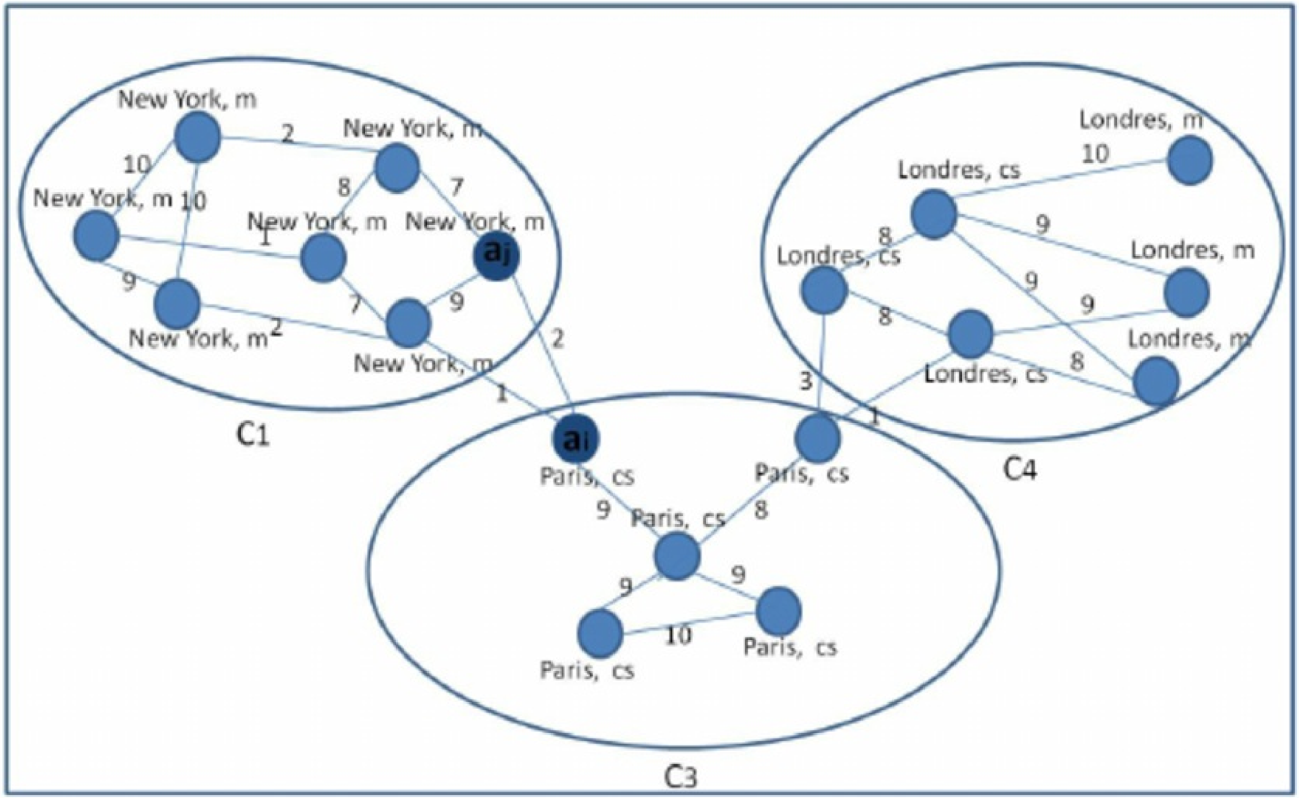
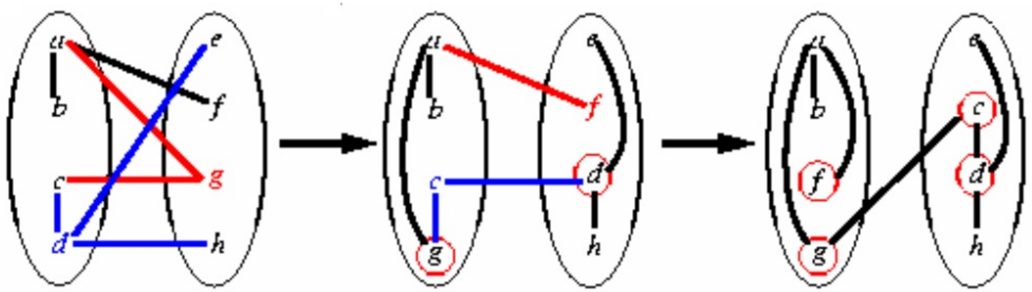


Image source: Zardi et al.: A Multi-agent homophily-based approach for community detection in social networks, ICTAI 2014

Some community detection methods

1. **Spectral clustering**
2. **Kerninghan-Lin**: balanced 2-way partitioning

- at each iteration, test a set of possible swap sequences and choose the one with greatest improvement



Step #	Vertex pair	Cost reduction	Cut cost
0	-	0	5
1	{d, g}	3	2
2	{c, f}	1	1
3	{b, h}	-2	3
4	{a, e}	-2	5

Image source: Chang 2004

3. Girwan-Newman algorithm

- remove “bridge edges” until K connected components remain
- edges with high **betweenness**: large proportion of shortest paths go through them

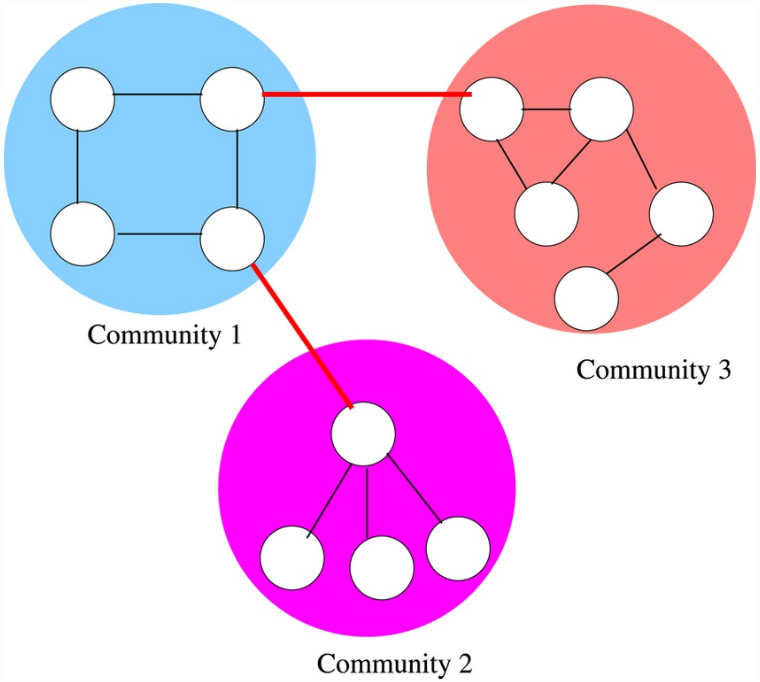


Image source: Namtirtha et al. 2023

4. METIS algorithm

1. Coarsen the graph by combining tightly interconnected nodes and parallel edges
2. Partition the coarsened representation (easier)
3. Refine partitioning when expanding graphs back

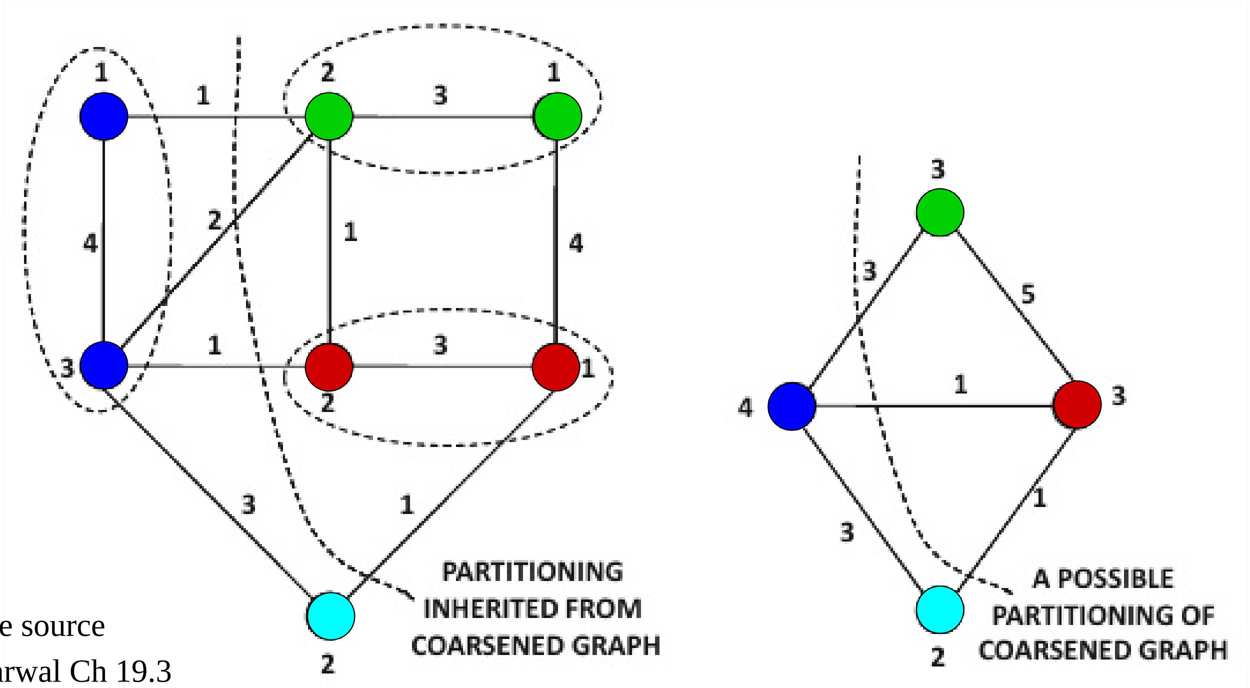


image source
Aggarwal Ch 19.3

Image source: Aggarwal Fig. 19.6

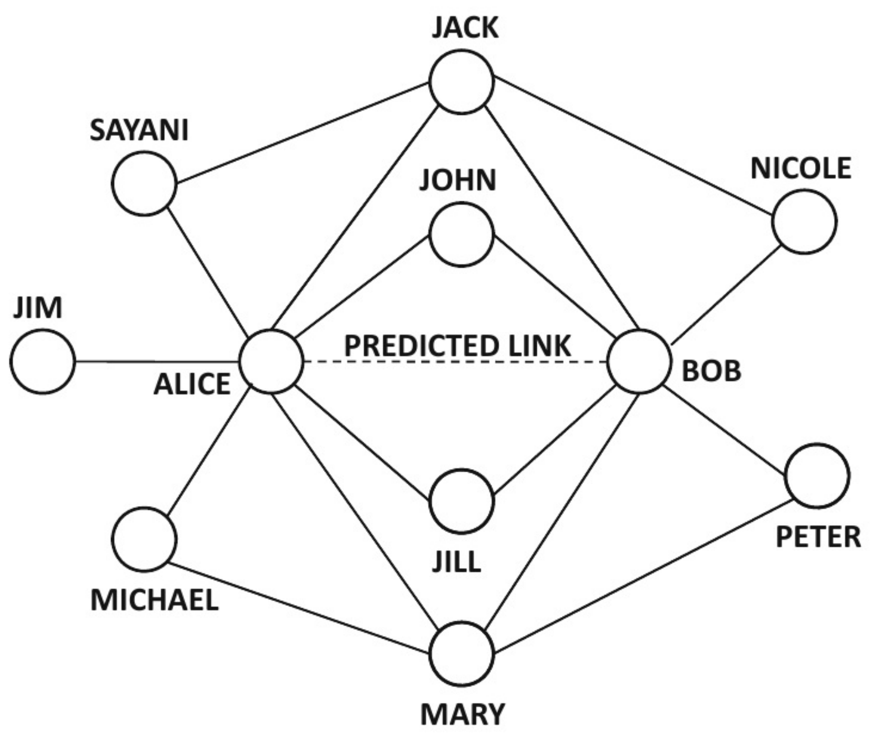
IV Link prediction and node similarity

Utilize especially **structural** features!

Approaches:

1. Evaluate potential connections with **node similarity measures**
   - \+ easy and fast to compute
2. Learn a classifier for predicting links or their absence
   - \+ more accurate
   - – computationally more expensive
3. Use missing value estimation methods (like matrix factorization)

Neighbourhood-based node similarity measures



(a) Many common neighbors between Alice and Bob

(normalized) number of common neighbours

- not good, if number of common neighbours small

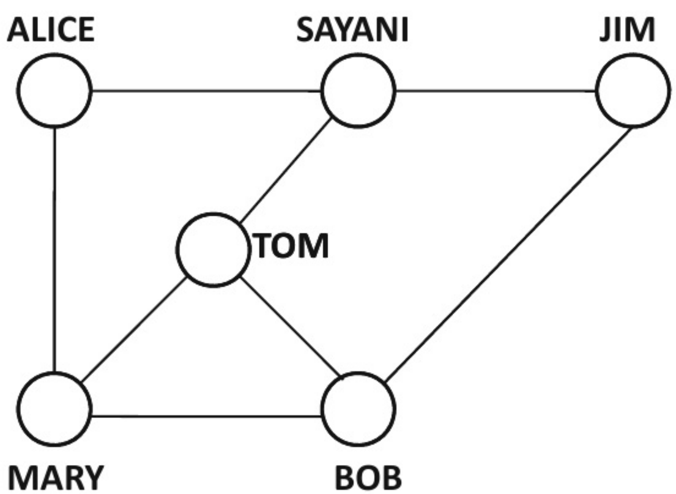
- $Jaccard(v_i, v_j) = \frac{|S_i \cap S_j|}{|S_i \cup S_j|}$
- $AdamicAdar(v_i, v_j) = \sum_{v_k \in S_i \cap S_j} \frac{1}{\log(|S_k|)}$

$S_i = \{v_k \mid v_k \text{ neighbour of } v_i\}$

Image source: Aggarwal Fig. 19.12

Walk-based node similarity measures

Is Alice more similar to Bob or Jim?



(b) Many indirect connections between Alice and Bob

- Personalized PageRank with teleportation to v_i
- SimRank
- Katz measure

$$Katz(v_i, v_j) = \sum_{t=0}^{\infty} \beta^t \cdot n_{ij}^{(t)}$$

$n_{ij}^{(t)}$ = number of walks of length t between v_i and v_j

$\beta < 1$ discount factor (punishes long walks)

Image source: Aggarwal Fig. 19.12

Image sources

- Chang (2004): Unit 4: Circuit partitioning (lecture slides). EDA course, National Taiwan University. `http://cc.ee.ntu.edu.tw/~ywchang/Courses/EDA04/lec4.pdf`
- Namtirtha et al. (2023): Placement Strategies for Water Quality Sensors Using Complex Network Theory for Continuous and Intermittent Water Distribution Systems. Water Resources Research 59(7), doi:10.1029/2022WR033112.
- Zardi et al. (2014): A Multi-agent homophily-based approach for community detection in social networks, IEEE 26th Int. Conf. Tools with Artificial Intelligence, doi: 10.1109/ICTAI.2014.81.