Data Mining

Lecture 11
Mining Structure from Graphs and High-Dimensional Data

http://www.informatik.uni-hamburg.de/WTM/

Case Based Reasoning

- Remember k-nearest neighbours:
  - Task is to classify a new data point x_n
  - Find the k nearest points $\{x_{k'}\}$ with their class labels $\{y_{k'}\}$
  - Assign class y_n based on the majority vote of $\{y_{k'}\}$
- i.e. use existing data $\{x_{k'}, y_{k'}\}$ ("past experience") directly for future decisions
- k=1: decide as in one precedence case

CBR – A way to solve complex problems

- By remembering how we solved a similar case in the past

- This is Case Based Reasoning (CBR)
  - memory-based problem-solving
  - re-using past experiences

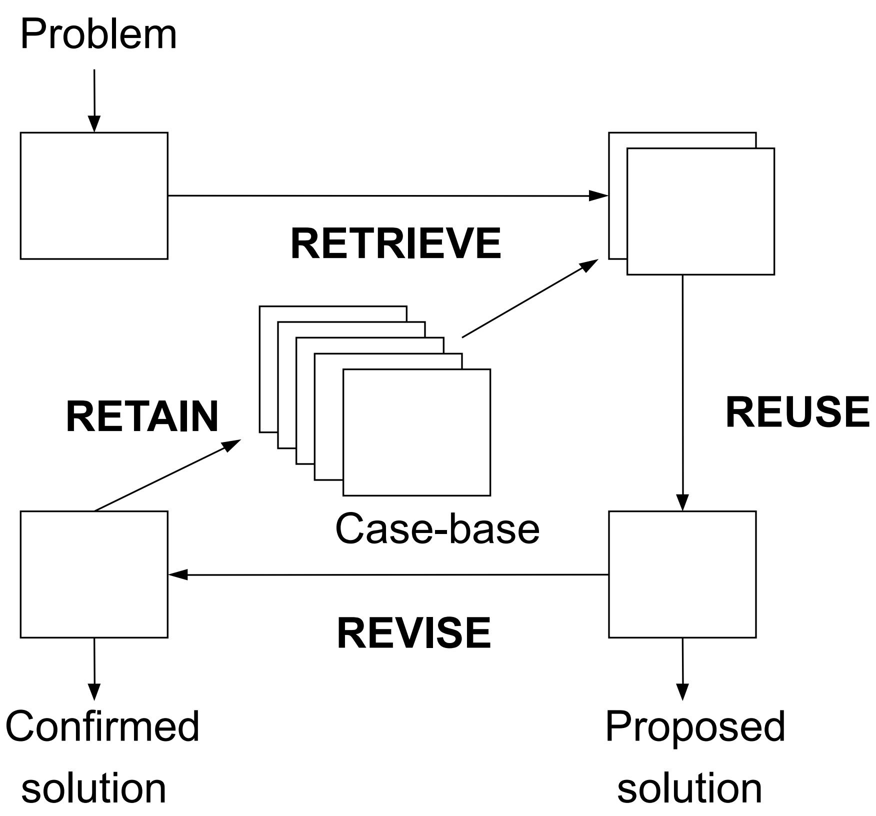
- Experts often find it easier to relate to past cases than to formulate rules about reasoning

CBR – Problems we solve this way

- Medicine
  - doctor remembers previous patients especially for rare combinations of symptoms
- Law
  - law depends on precedence
  - case histories are consulted
- Management
  - decisions are often based on past rulings
- Financial
  - performance is predicted by past results
- Robotics
  - Robot soccer – imitate good moves

CBR – Overview

- CBR provides an automated method for storing experience and reusing it to make decisions in the future



CBR Process

- Expertise is embodied in a library of past cases (experiences)
- Each case typically contains
  - a *description* of the problem
  - *goals* and subgoals that arise in reasoning
  - *successful attempts* at achieving those goals

    → to propose solutions to new problems
  - *failed attempts*

    → to warn of possible failure

CBR Process

- Basic algorithm to solve a current problem:
  - Match the problem's features against the cases in the case base, and retrieve similar cases.
  - If multiple solutions are found then resolve any ambiguities.
  - Use retrieved cases to suggest a solution to reuse and test for success.
  - If necessary, revise the solution.
  - Retain the current problem, i.e. its defining features and its final solution, as part of a new case.

CBR Evaluation

- What does the CBR process depend on?
  - Appropriate methods for ***indexing cases*** using their key attributes
  - Efficient mechanisms for ***retrieving cases*** given a set of index values
  - Existing cases – a smaller case-base can be compensated for by more creativity in retrieval and revision
  - Good ***presentation*** of the information to the user

What are good CBR Applications?

- Failure prediction
  - ultrasonic non destructive testing for Dutch railways
  - water in oil wells for Schlumberger
- Failure analysis
  - Mercedes cars for DaimlerChrysler
  - semiconductors at National Semiconductor
- Maintenance scheduling
  - Boeing 737 engines
  - TGV trains for SNCF
- Planning
  - mission planning for US navy
  - route planning for DaimlerChrysler cars

CBR in Business

"those who ignore history are doomed to repeat it"

Norwegian CBR consultant Verdande started in the oil business

CBR as Presented by “Verdande Technology”

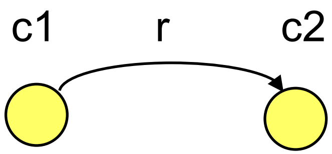
- ”Based on the principle that similar problems have similar solutions, CBR .. analyzes data patterns in real-time, using past events to proactively predict future problems.”
- ”harvests multiple, heterogeneous data types to index and search for those past experiences and provides organizations with the information they need”
- ”transforms big data into actionable insight”
- ”offering a realistic assessment as to whether a similar scenario is likely to occur in the future”
- ”can help reduce drilling NPT” (non-productive time) by:
  - ”Identify problem precursors.
  - Interpret and resolve the drilling situation.
  - Retrieve relevant solutions and lessons learned.”

CBR – Summary

- CBR does not require an explicit domain model and so elicitation becomes a task of gathering case histories
- Implementation is reduced to identifying significant features that describe a case, an easier task than creating an explicit model
- CBR systems can learn by acquiring new knowledge as cases making maintenance easier

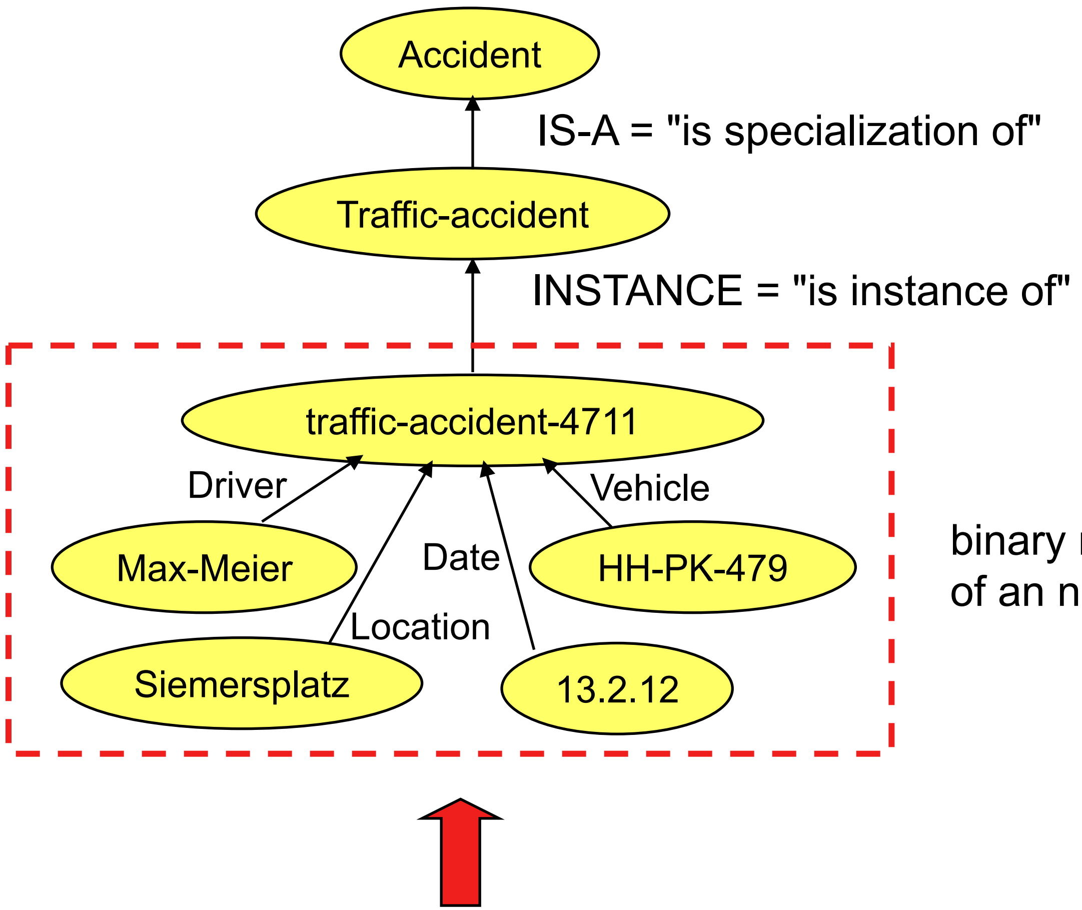
Semantic Networks

- Graphical representation of concepts and relations:
  - labeled nodes (vertices) = concepts
  - directed labeled links (edges) = binary relations



- Questions: But where is the semantics?
  - Are there any ***nodes*** or node types and ***links*** or link types which are valid in general, independent of a particular domain?
  - Is there any ***structuring rule*** which is valid in general, independent of a particular domain?
  - Are there ***generally valid inference procedures*** to derive knowledge which is not explicitly stated?

Basic Relations in Semantic Networks



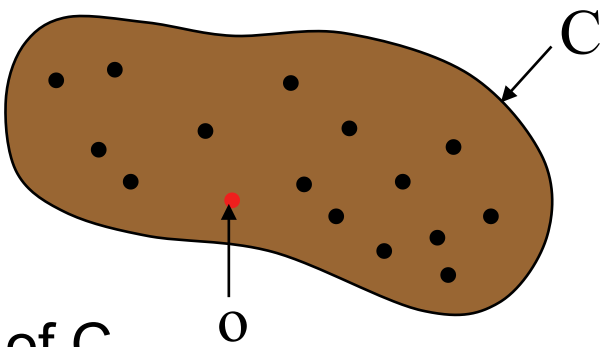
binary relations linking to parts of an n-ary relation

(Traffic-accident 4711 Max-Meier Siemersplatz 13.2.2012 HH-PK 479)

Concepts and Individuals

Nodes of a Semantic Network describe concepts and individuals.

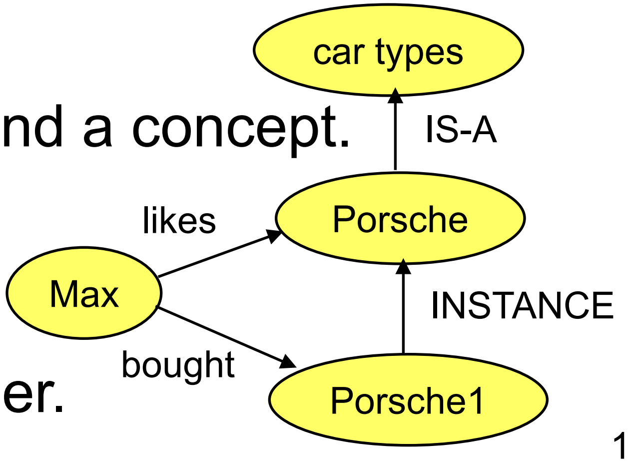
A concept denotes a ***set of objects***.
An individual denotes a ***single object***.



C_1 *IS-A* C_2 specifies that C_1 is a subset of C_2
o INSTANCE C specifies that o is a member of C

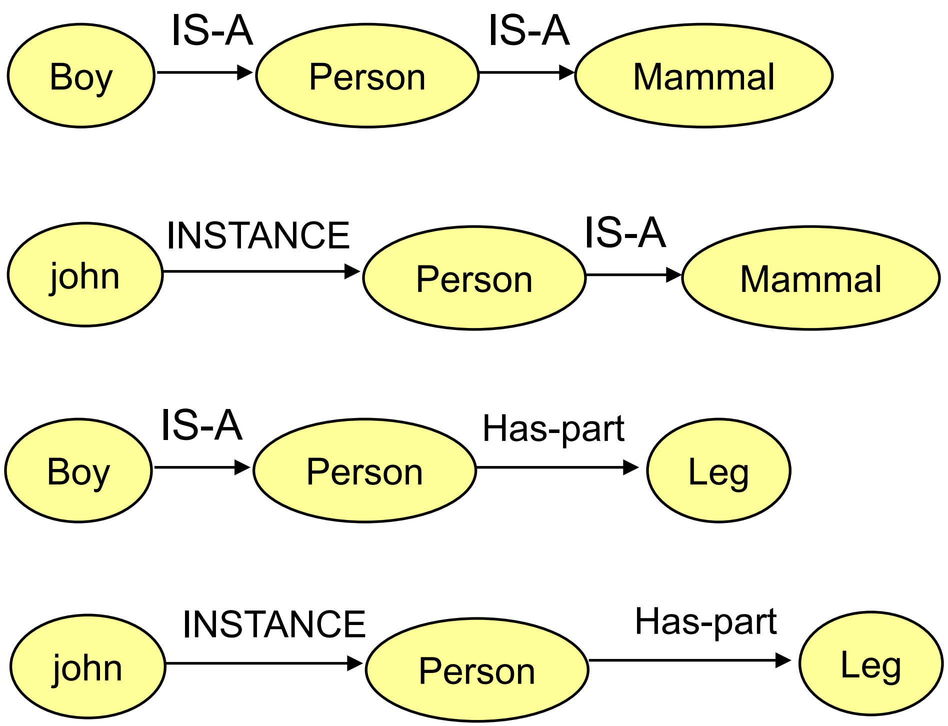
A node may represent both, an individual and a concept.
Example:

Max likes a Porsche.
Max bought a Porsche at the car dealer.



Inferences in Semantic Networks

Examples



Rules

$$C_1 \text{ IS-A } C_2, \quad C_2 \text{ IS-A } C_3 \quad \Rightarrow \quad C_1 \text{ IS-A } C_3$$

$$c \text{ INSTANCE } C_1, \quad C_1 \text{ IS-A } C_2 \quad \Rightarrow \quad c \text{ INSTANCE } C_2$$

$$C_1 \text{ IS-A } C_2, \quad C_2 \text{ Rel } C_3 \quad \Rightarrow \quad C_1 \text{ Rel } C_3$$

$$c \text{ INSTANCE } C_2, \quad C_2 \text{ Rel } C_3 \quad \Rightarrow \quad c \text{ Rel } C_3$$

Special Semantics for Special Relations

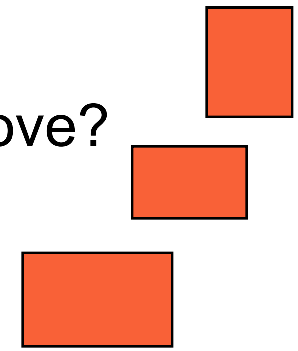
- Special relations may support special inferences.
- **Examples**:

  Above(a, b) $\wedge$ Above(b, c) $\Rightarrow$ Above(a, c)

  Left(a, b) $\Rightarrow$ Right(b, a)

  Has-part(a, b) $\wedge$ Has-Part(b, c) $\Rightarrow$ Has-Part(a, c)

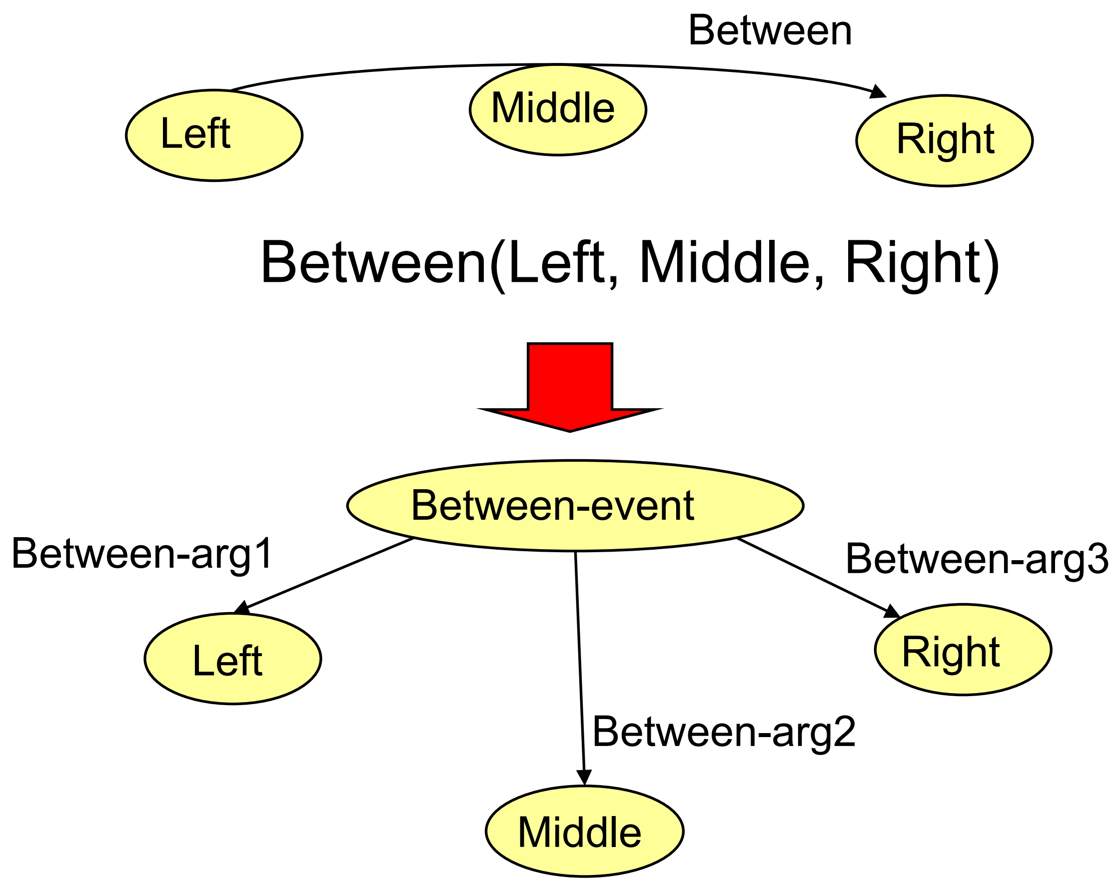
Above?



- The rules for inferences may change from domain to domain, hence they must be explicitely stated.
  - ⇨ "axiomatizing a domain"
- Spatial reasoning, temporal reasoning are disciplines dealing with axiomatizations of spatial, part-of- and temporal relationships.

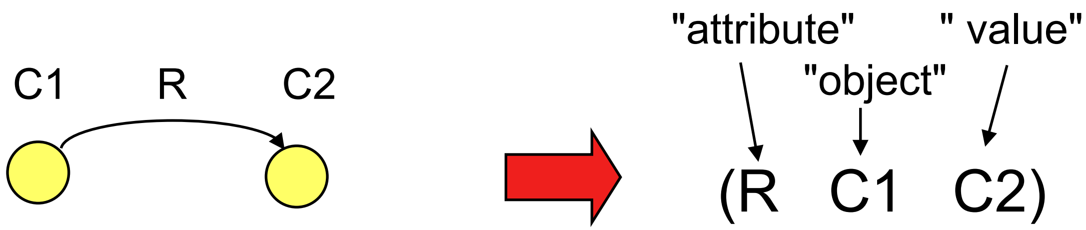
N-ary Relations in Semantic Networks

- Semantic Networks allow the representation of binary relations.
- Any N-ary relation can be represented by ***multiple binary relations***
- **Example**:



Attribute-Object-Value Triplets

- In knowledge representation- and programming languages, a Semantic Network can be represented by a set of triplets:



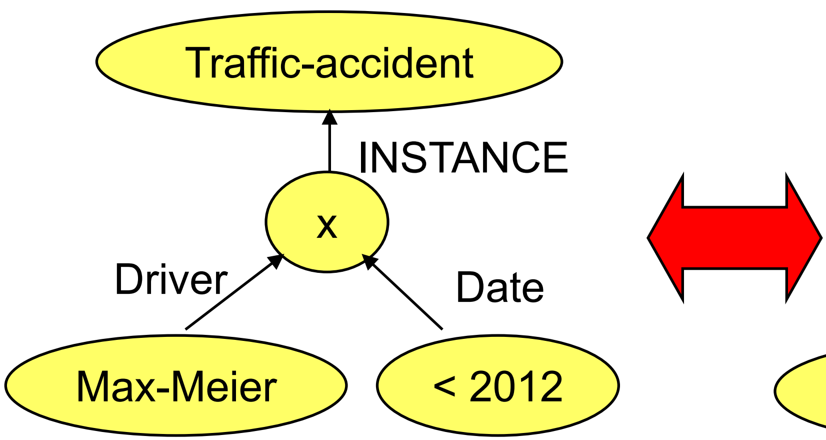
- The accident example:

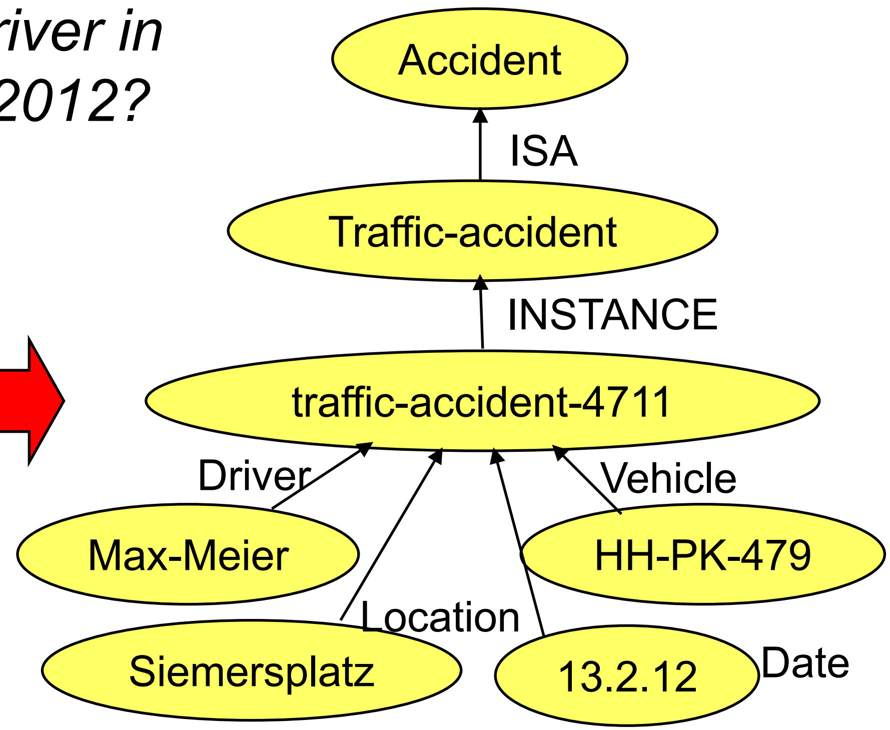
  (is-a traffic-accident accident)
  (instance traffic-accident-4711 traffic-accident)
  (driver traffic-accident-4711 Max-Meier)
  (location traffic-accident-4711 Siemersplatz)
  (date traffic-accident-4711 13.2.12)
  (vehicle traffic-accident-4711 HH-PK-479)

Matching Relational Structures

- Semantic Networks applications often involve matching one network against another
- **Example:**

  Has Max Meier been the driver in any traffic accident before 2012?





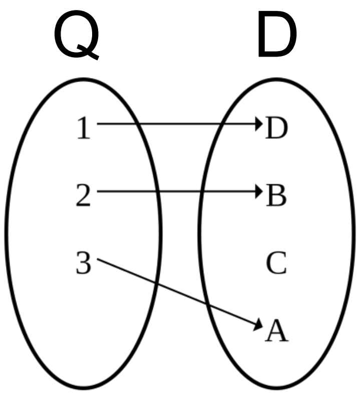
What services are required?

What are the matching rules?

Semantic Network (SN) Queries

- A SN query is a description of desired query responses in terms of a SN using an extended concept language.
- Typical concept language extensions:

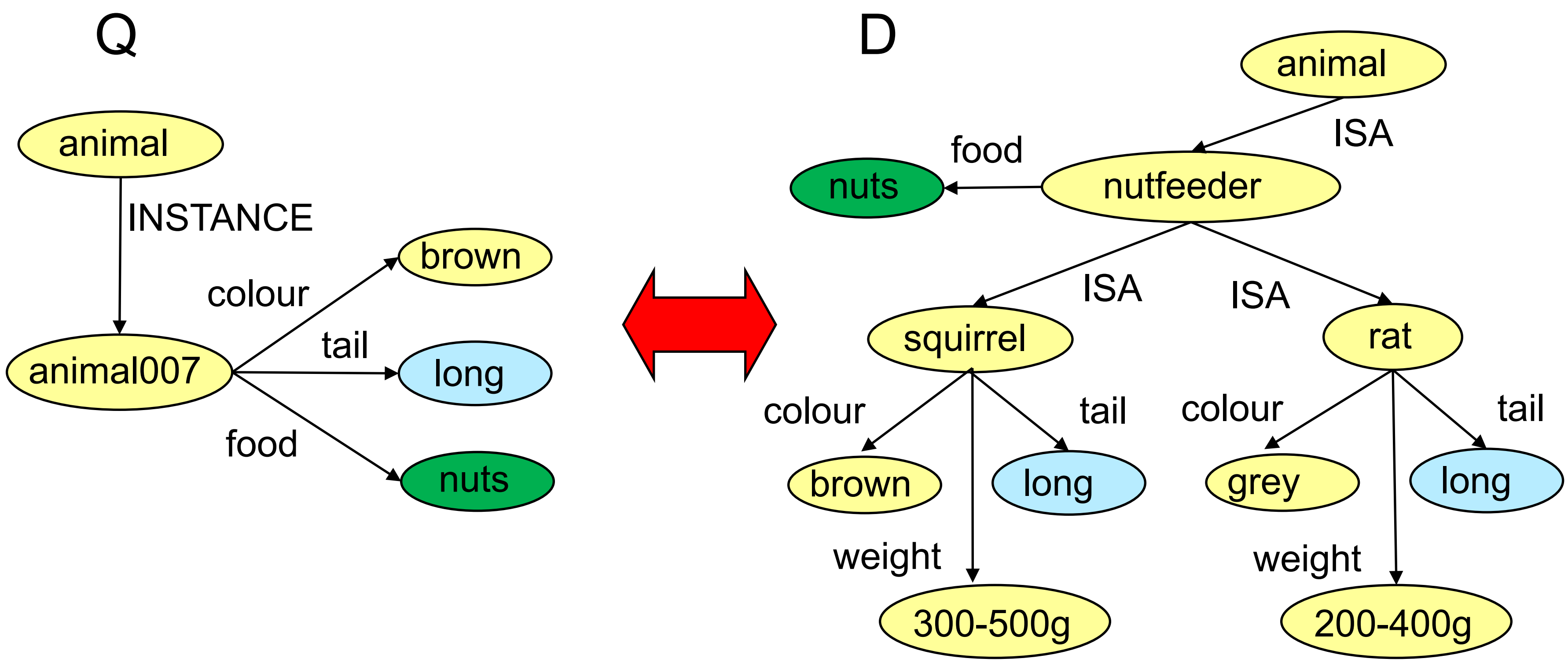
x	individual variable
X	concept variable
{a, b, c}	set of individuals
< 2012	predicate over a concrete domain individual



Matching rules:

A query Q matches a database D, if there is an *injective* mapping of all nodes and links in Q to nodes and links in D such that the corresponding nodes and links are compatible.

Object Classification by Relational Matching



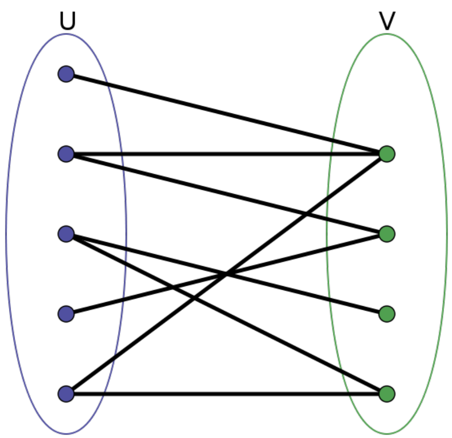
- INSTANCE and ISA inheritance must be exploited for matching
- Class decriptions must be given in terms of sufficient conditions

→ graphs are classified by query matching

Semantic Networks – Summary

- Intuitive graphical knowledge representation
- Complex problems can be expressed by graphs and basic information retrieval and classification done by query matching
- Semantics of relations is well-defined for ISA and INSTANCE, but not clearly defined in general
- Relations between relations cannot be expressed
- Need for domain-specific inference rules (“axiomatizing”)
- Generally useful services require additional formalisms such as rule-based inferences and new techniques from machine learning and automatic access, tagging, retrieval, pattern matching

Clustering Graphs and Network Data

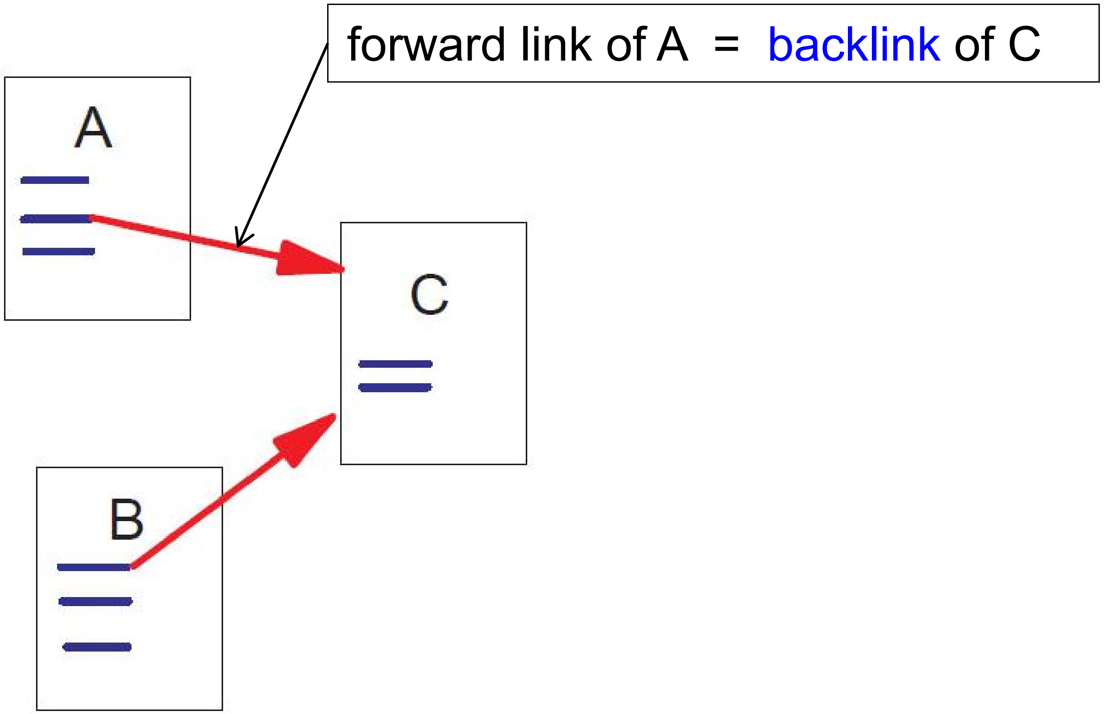


- Applications
  - Bipartite graphs, e.g.:
    - customers and products,
    - authors and conferences, …
  - Web search engines, e.g.:
    - click-through graphs, webgraph, …
  - Social networks, friendship/coauthor graphs
- Similarity measures
  - Geodesic distances
  - SimRank distance
- Graph clustering methods
  - Minimum cuts: FastModularity (Clauset, Newman & Moore, 2004)
  - Density-based clustering: SCAN (Xu et al., KDD’2007)

Applications: Google (1)

Webgraph: web page = vertex, weblink = edge

A web page is important if many pages refer to it   *(vote)*



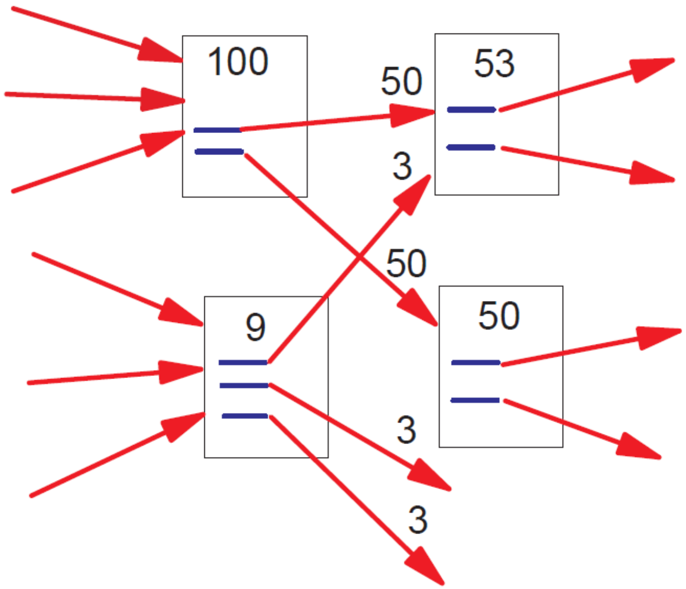
Brin, Page, et al. (1998) The PageRank Citation Ranking: Bringing Order to the Web. Tech Rep Stanford Uni

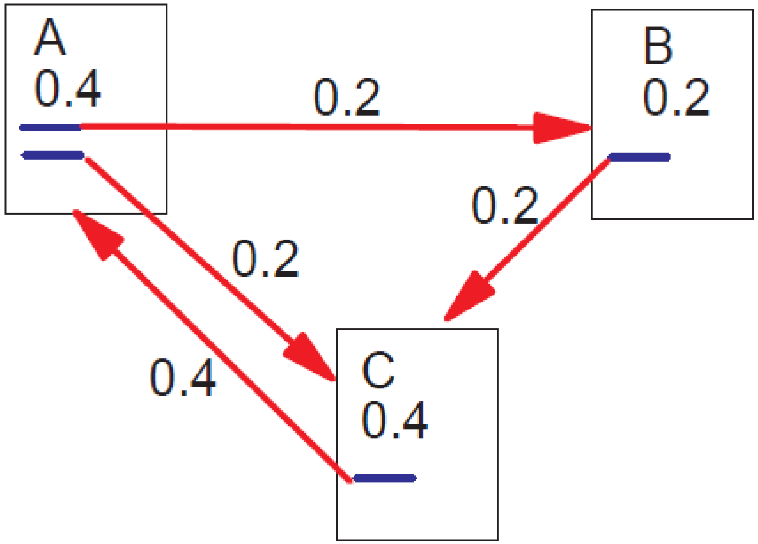
Google (2)

Ranking Function for web page *u*:

$$R(u) = c \sum_{v \in B_u} \frac{R(v)}{N_v}$$

v: web page that links to u
B_u: backlinks
$N_v = |F_v|$: # forward links from v
c: normalization factor

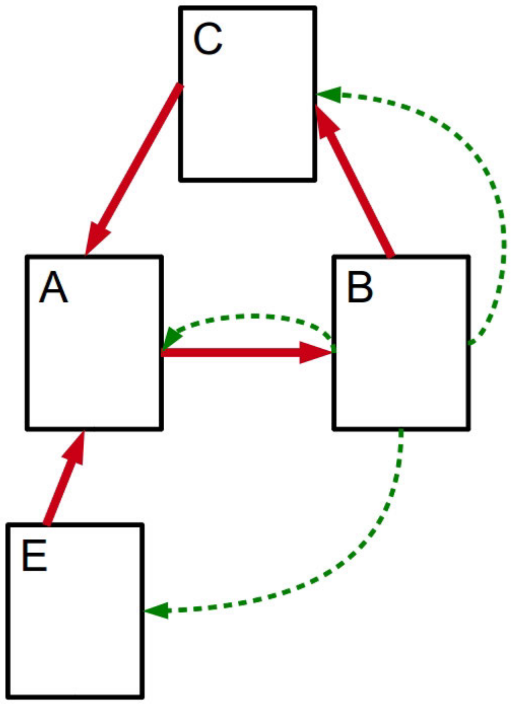




PageRanks form a probability distribution over web pages, so the sum of all web pages' PageRanks will be one

Google (3)

Problem: ***Rank Sink***
Some pages form a loop that accumulates rank (rank sink) to the infinity.

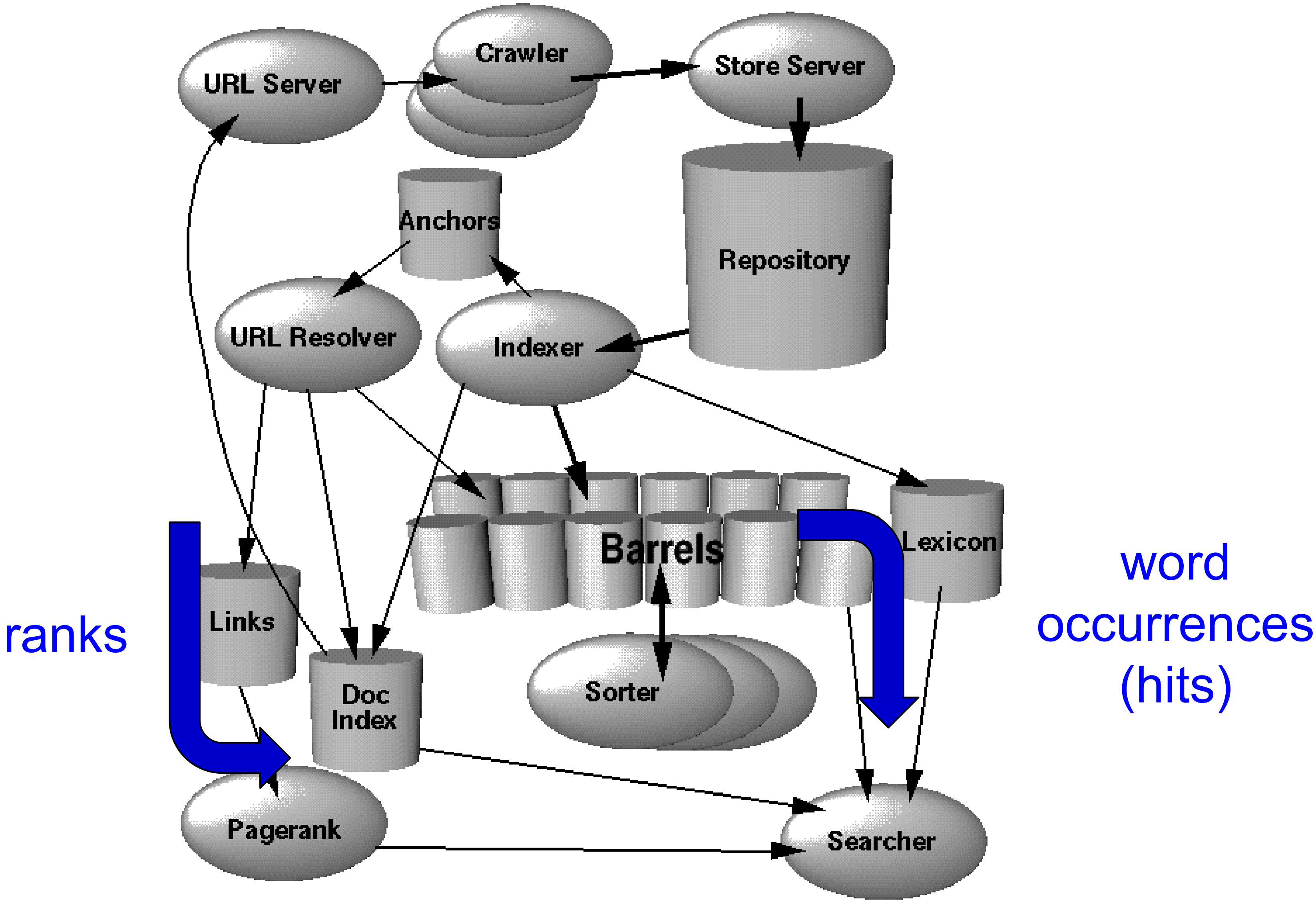


Solution: ***Random Surfer***
Jump to a random page based on some distribution *E*

$$R'(u) = c \sum_{v \in B_u} \frac{R'(v)}{N_v} + cE(u)$$

rank source

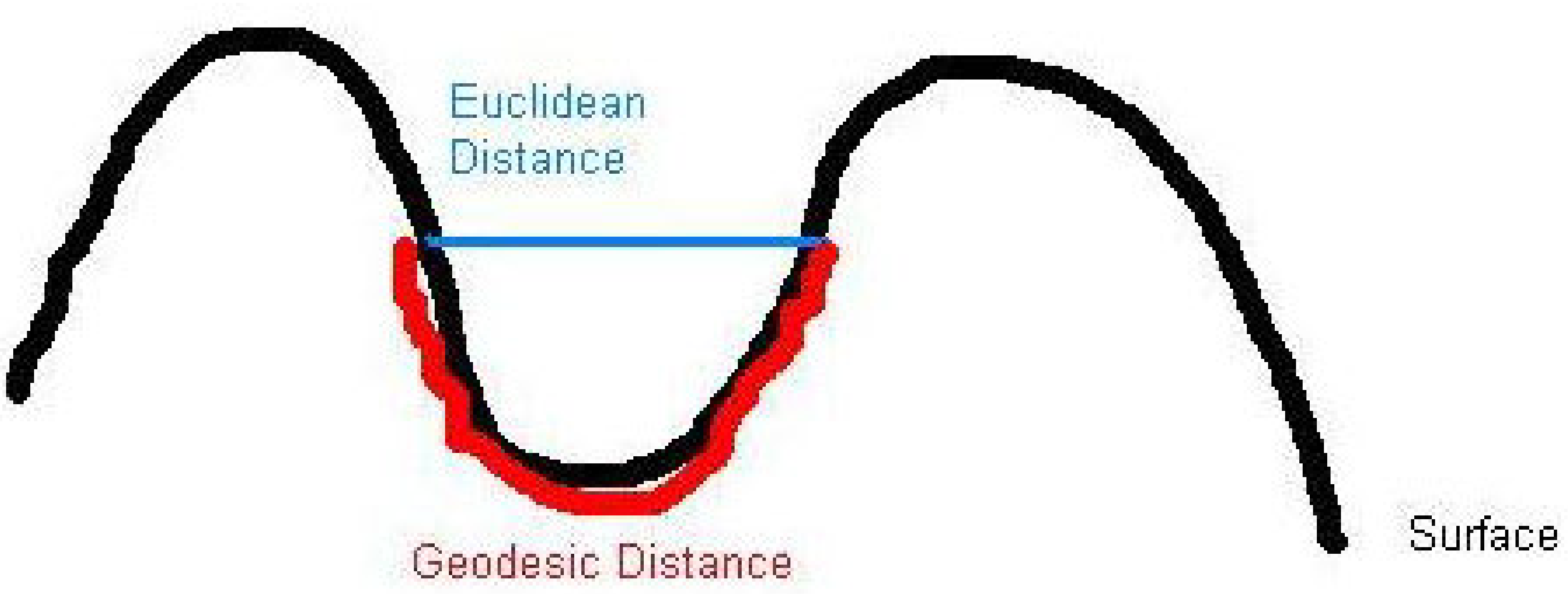
Google (4)



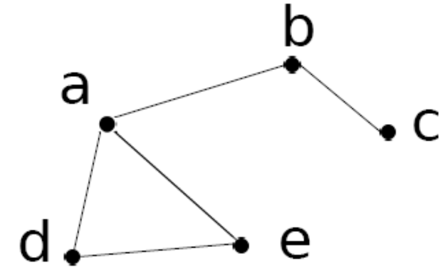
Brin, Page (1998) The anatomy of a large-scale hypertextual Web search engine. Comp Netw and ISDN Syst

Similarity Measures: Geodesic Distances (1)

- ***Geodesic distance***: distance along curved spaces
- May be approximated by adding many short straight segments, using the Euclidean distance for each of these



Geodesic Distances (2)

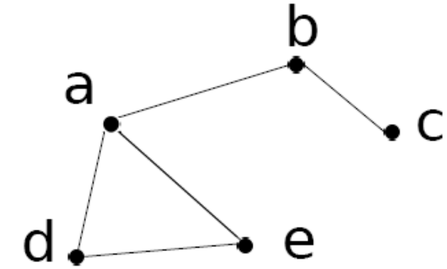


- ***Geodesic distance*** (A, B): length (i.e., # of edges) of the shortest path between A and B (if not connected, defined as infinite)
- ***Eccentricity*** of v, eccen(v): The largest geodesic distance between v and any other vertex u ∈ V − {v}.
  - E.g.,

    eccen(a) = eccen(b) = 2;

    eccen(c) = eccen(d) = eccen(e) = 3
- A ***peripheral vertex*** is a vertex that achieves the diameter.
  - **E.g.**, Vertices c, d, and e are peripheral vertices

Geodesic Distances (3)



- ***Radius*** of graph G:
  The minimum eccentricity of all vertices,
  i.e., the distance between the
  “most central point” and the “farthest border”
  - $r = \min_{v \in V} \text{eccen}(v)$
  - **E.g.**, radius (g) = 2
- ***Diameter*** of graph G: The maximum eccentricity of all vertices, i.e., the largest distance between any pair of vertices in G
  - $d = \max_{v \in V} \text{eccen}(v)$
  - **E.g.**, diameter (g) = 3

SimRank

- SimRank: ***structural-context similarity*** – based on similarity of its neighbors
- In a ***directed graph*** G = (V,E),
  - individual in-neighborhood of *v*: *I(v) = {u | (u, v) ∈ E}*
  - individual out-neighborhood of *v*: *O(v) = {w | (v, w) ∈ E}*
- ***Similarity*** in SimRank:

$$s(u,v) = \frac{C}{|I(u)| \cdot |I(v)|} \sum_{x \in I(u)} \sum_{y \in I(v)} s(x,y)$$

remember google's page rank: $R(u) = c \sum_{v \in B_u} \frac{R(v)}{N_v}$

SimRank: Similarity by Fix Point Iteration

$$s(u,v) = \frac{C}{|I(u)| \cdot |I(v)|} \sum_{x \in I(u)} \sum_{y \in I(v)} s(x,y)$$

- Problem: recursive formula
- Solution: Iteration to a fixed point:
  - Initialization:

$$s_0(u,v) = \begin{cases} 0 & \text{if } u \neq v \\ 1 & \text{if } u = v \end{cases}$$

  - Then we can compute s_{t+1} from s_t based on the definition:

$$s_{t+1}(u,v) = \frac{C}{|I(u)| \cdot |I(v)|} \sum_{x \in I(u)} \sum_{y \in I(v)} s_t(x,y)$$

  - Typical parameters: decay factor *C=0.8,* number of iterations *T=5.*

SimRank: Similarity by Random Walk

- The probability *P* of a tour *t* of length *l* is defined as:

$$P[t] = \begin{cases} \prod_{i=1}^{l(t)} \frac{1}{|O(w_i)|} & \text{if } l(t) > 0 \\ 1 & \text{if } l(t) = 0 \end{cases}$$

- Similarity based on ***random walk***: in a strongly connected component
  - Expected distance:

$$d(u,v) = \sum_{t:u \to v} P[t] \cdot l(t)$$

  - Expected meeting distance for a pair of tours:

$$m(u,v) = \sum_{t:(u,v) \to (x,x)} P[t] \cdot l(t)$$

  - Expected meeting probability:

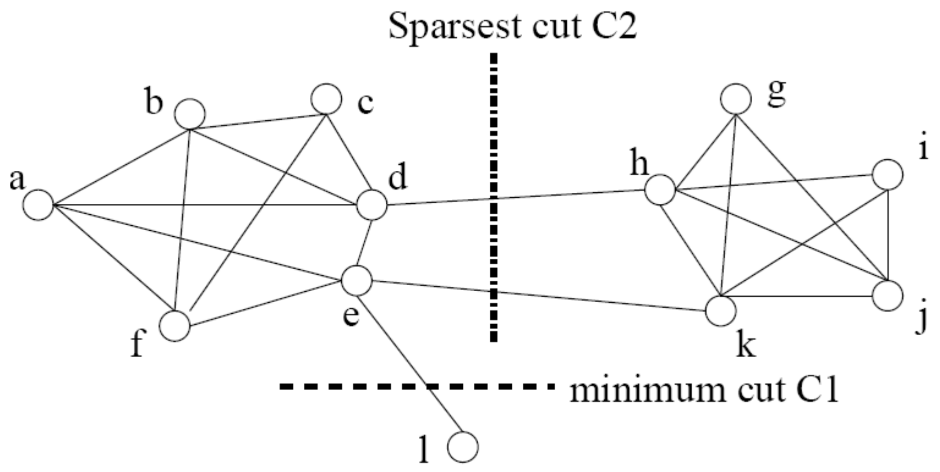
$$p(u,v) = \sum_{t:(u,v) \to (x,x)} P[t] \cdot C^{l(t)}$$

  where *C* defines the probability of continuing the walk

Jeh, Widom (ca. 2002) SimRank: A Measure of Structural-Context Similarity

Graph Clustering: Sparsest Cut (1)

- ***Undirected graph*** G = (V,E). The *cut set* of a cut is the set of edges {(u, v) ∈ E | u ∈ S, v ∈ T } and S and T are in the two partitions



- ***Size*** of the cut:
# of edges in the cut set

- Min-cut (e.g., C_1)
is not a good partition

- A better measure: ***Sparsity***

$$\Phi = \frac{\text{the size of the cut}}{\min\{|S|, |T|\}}$$

Graph Clustering: Sparsest Cut (2)

- A cut is ***sparsest*** if its sparsity is not greater than that of any other cut
- **Ex**. Cut C2 = ({a, b, c, d, e, f, l}, {g, h, i, j, k}) is the sparsest cut
- For k clusters, the ***modularity*** of a clustering assesses the quality of the clustering:

$$Q = \sum_{i=1}^{k} \left(\frac{l_i}{|E|} - \left(\frac{d_i}{2|E|} \right)^2 \right)$$

l_i: # edges between vertices *within* i-th cluster
d_i: # *all* edges connecting to vertices in i-th cluster

- The *modularity* of a clustering of a graph is the difference between the fraction of all edges that fall into individual clusters and the fraction that would do so if the graph vertices were randomly connected
- The optimal clustering of graphs maximizes the modularity

Graph Clustering: Challenges of Finding Good Cuts

- High computational cost
  - Many graph cut problems are computationally expensive
  - The sparsest cut problem is NP-hard
  - Need to tradeoff between efficiency/scalability and quality
- Sophisticated graphs
  - May involve weights and/or cycles
- High dimensionality
  - A graph can have many vertices. In a similarity matrix, a vertex is represented as a vector (a row in the matrix) whose dimensionality is the number of vertices in the graph
- Sparsity
  - A large graph is often sparse, meaning each vertex on average connects to only a small number of other vertices
  - A similarity matrix from a large sparse graph can also be sparse

Two Approaches for Graph Clustering

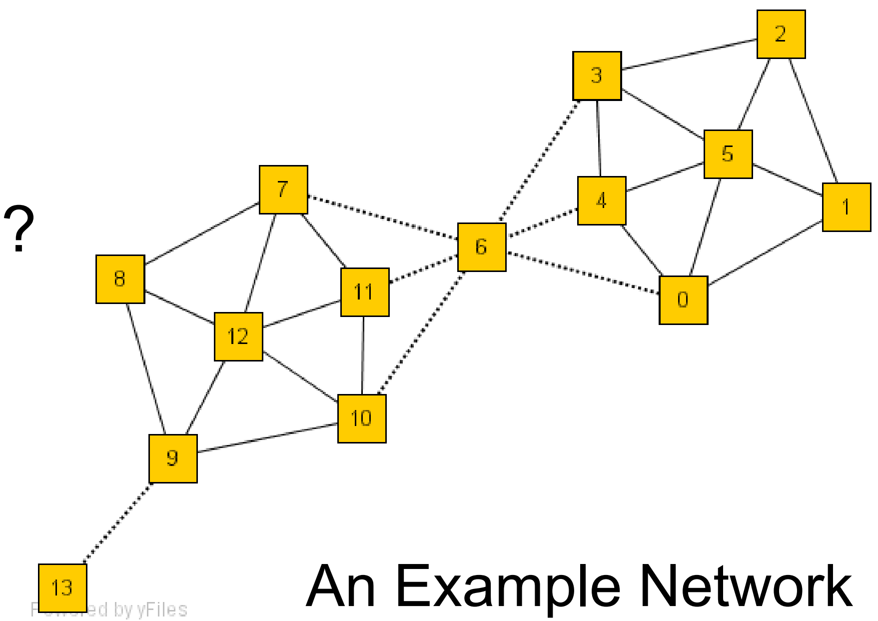
1. Using generic clustering methods for high-dimensional data
   - Extract a similarity matrix from a graph using a similarity measure
   - A generic clustering method can then be applied on the similarity matrix to discover clusters
   - Ex. Spectral clustering: approximate optimal graph cut solutions

2. Methods specifically designed for clustering graphs
   - Search the graph to find well-connected components as clusters
   - Ex. SCAN (Structural Clustering Algorithm for Networks)

     [X. Xu, N. Yuruk, Z. Feng, and T. A. J. Schweiger, “SCAN: A Structural Clustering Algorithm for Networks”, KDD'07]

SCAN: Density-Based Clustering of Networks

- How many clusters?
- What size should they be?
- What is the best partitioning?
- Should some points be segregated?



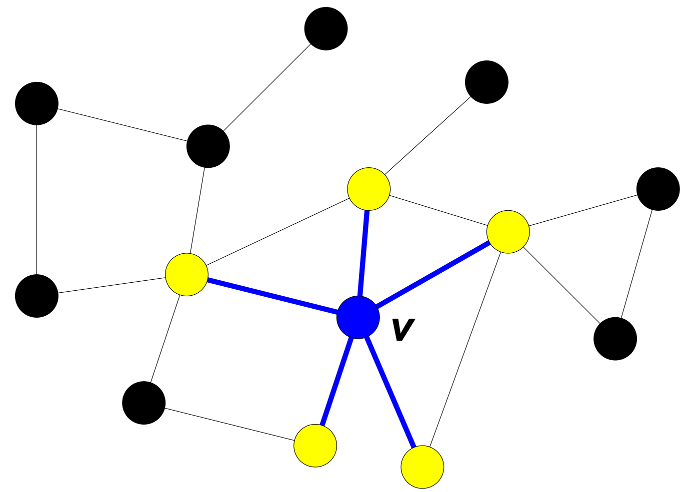
An Example Network

- Application:

  Given simply information of who associates with whom. Identify clusters of individuals with common interests or special relationships (families, cliques, terrorist cells) …

A Social Network Model

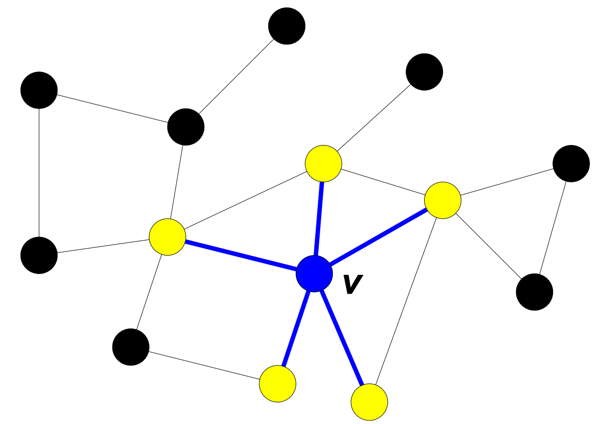
- Characteristics:
  - Individuals in a tight social group, or ***clique***, know many of the same people, regardless of the size of the group
  - Individuals who are ***hubs*** know many people in different groups but belong to no single group. E.g., politicians bridge multiple groups
  - Individuals who are ***outliers*** reside at the margins of society. E.g., hermits know few people and belong to no group
- The neighborhood of a vertex
  - Define $\Gamma(\nu)$ as the immediate neighborhood of a vertex (i.e. the set of people that an individual knows)



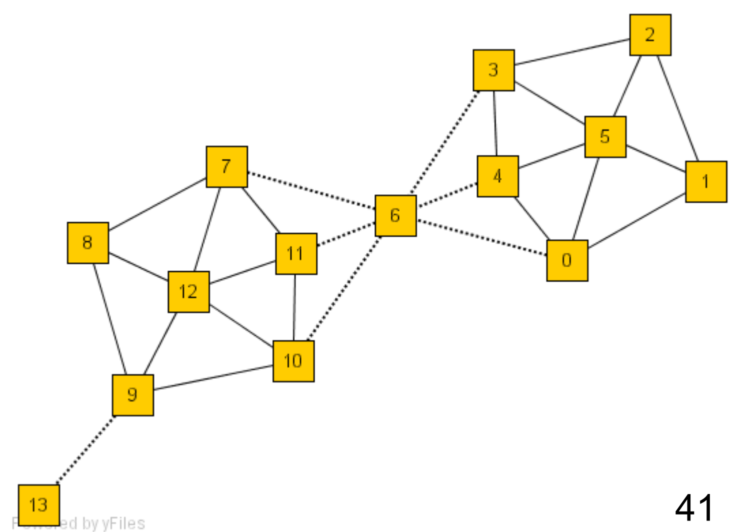
Structure Similarity

- The desired characteristics tend to be captured by a measure we call Structural Similarity

$$\sigma(v,w) = \frac{|\Gamma(v) \cap \Gamma(w)|}{\sqrt{|\Gamma(v)| \cdot |\Gamma(w)|}}$$



- Structural similarity is large for members of a clique and small for hubs and outliers



Structural Connectivity

- ε-neighborhood: $N_\varepsilon(v) = \{w \in \Gamma(v) \mid \sigma(v,w) \geq \varepsilon\}$
- Vertex is a core: $CORE_{\varepsilon,\mu}(v) \Leftrightarrow |N_\varepsilon(v)| \geq \mu$ μ integer

  we will let structures grow starting from the core

- Direct structure reachable:

$$DirREACH_{\varepsilon,\mu}(v,w) \Leftrightarrow CORE_{\varepsilon,\mu}(v) \wedge w \in N_\varepsilon(v)$$

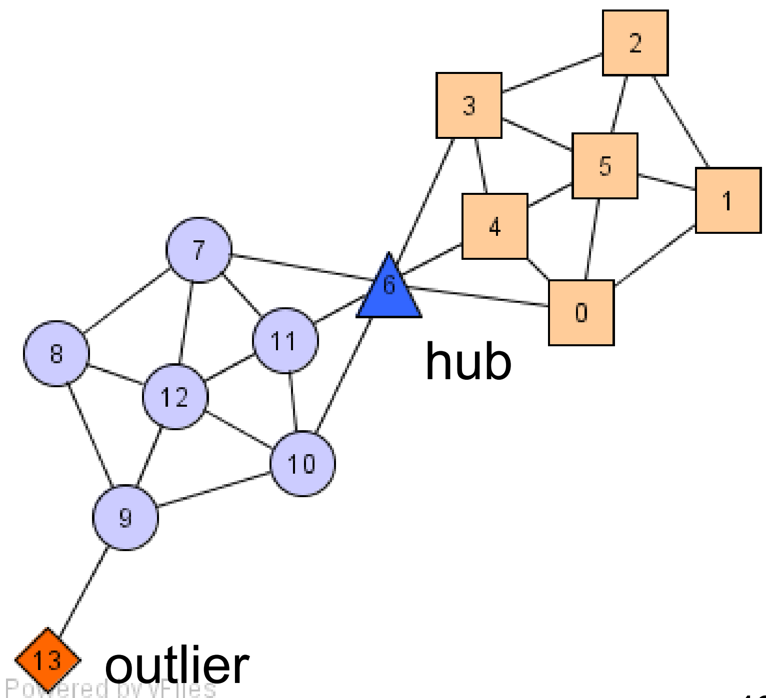
- Structure reachable: $REACH_{\varepsilon,\mu}(v,w)$ transitive closure of direct structure reachability
- Structure connected:

$$CONNECT_{\varepsilon,\mu}(v,w) \Leftrightarrow \exists u \in V : REACH_{\varepsilon,\mu}(u,v) \wedge REACH_{\varepsilon,\mu}(u,w)$$

Xu, Yuruk, Feng, Schweiger (SIGKDD'07) "SCAN: A Structural Clustering Algorithm for Networks".
See also: Ester, Kriegel, Sander, Xu (KDD'96) "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases".

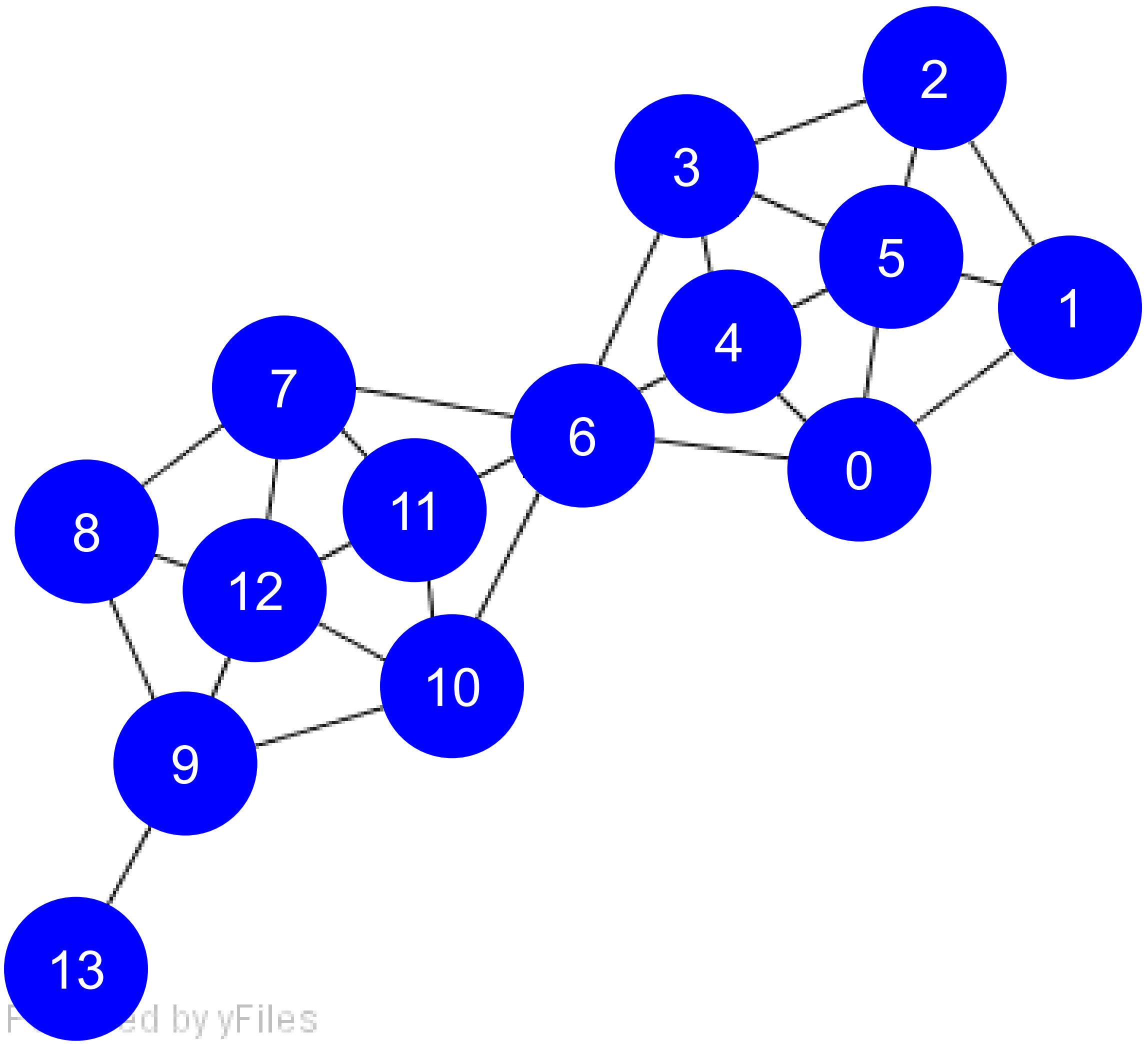
Structure-Connected Clusters

- Define a structure-connected cluster *C:*
  - Connectivity: $\forall v, w \in C : CONNECT_{\varepsilon,\mu}(v,w)$
  - Maximality: $\forall v, w \in V : v \in C \wedge REACH_{\varepsilon,\mu}(v,w) \Rightarrow w \in C$
- **Hubs**:
  - Not belong to any cluster
  - Bridge to many clusters
- **Outliers**:
  - Not belong to any cluster
  - Connect to less clusters



SCAN Algorithm

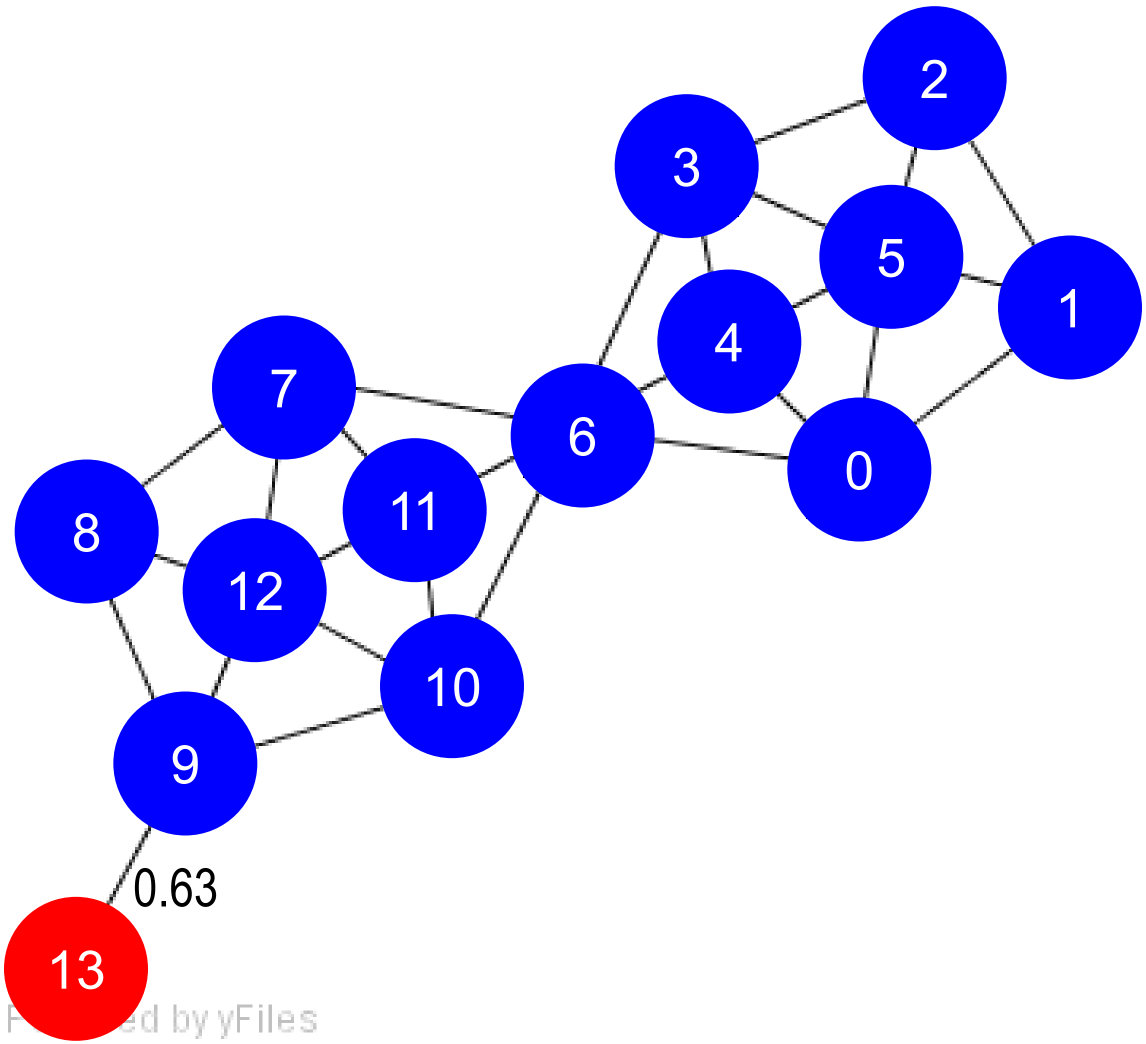
$\mu = 2$
$\varepsilon = 0.7$



SCAN Algorithm

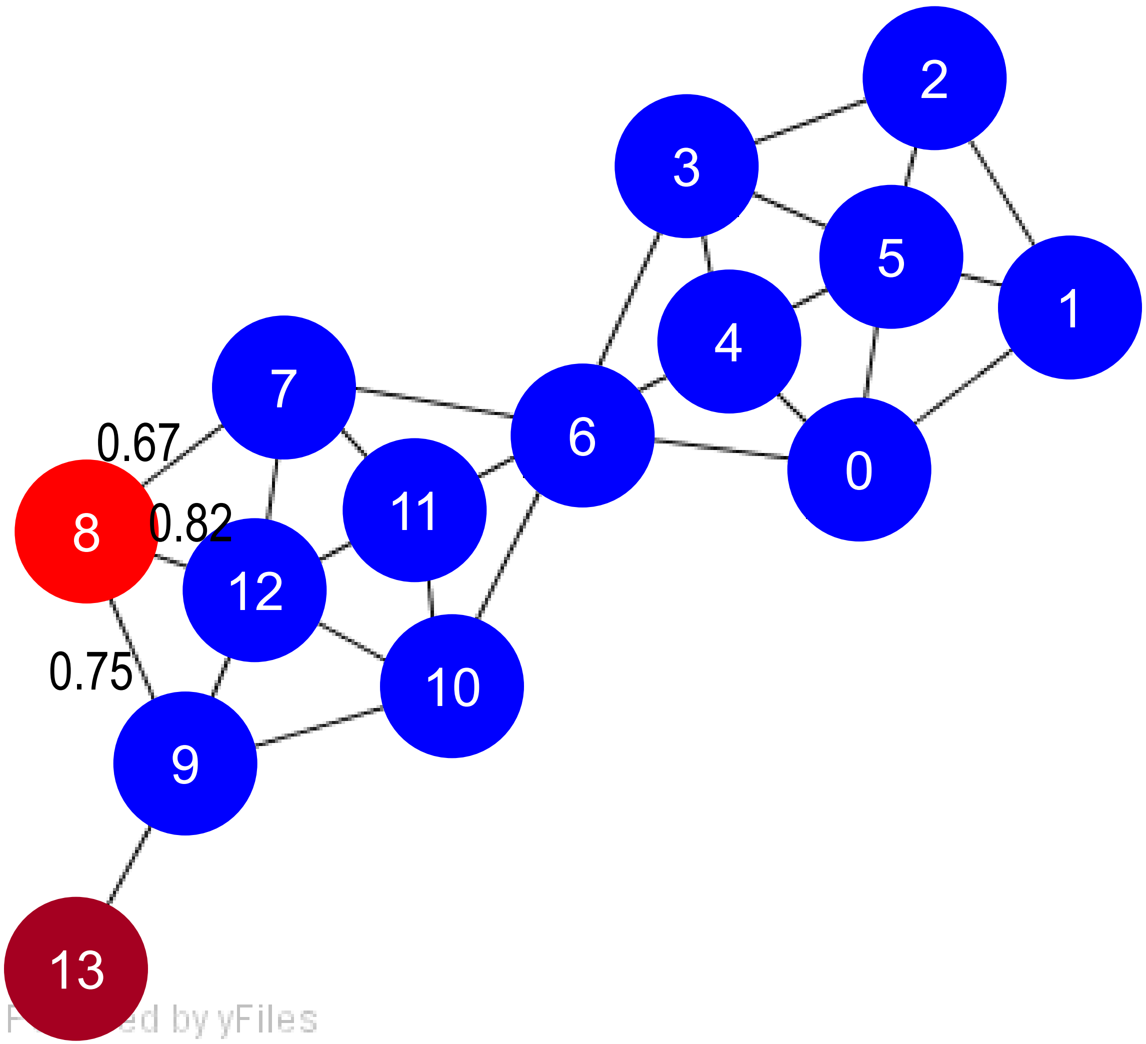
$\mu = 2$

$\varepsilon = 0.7$



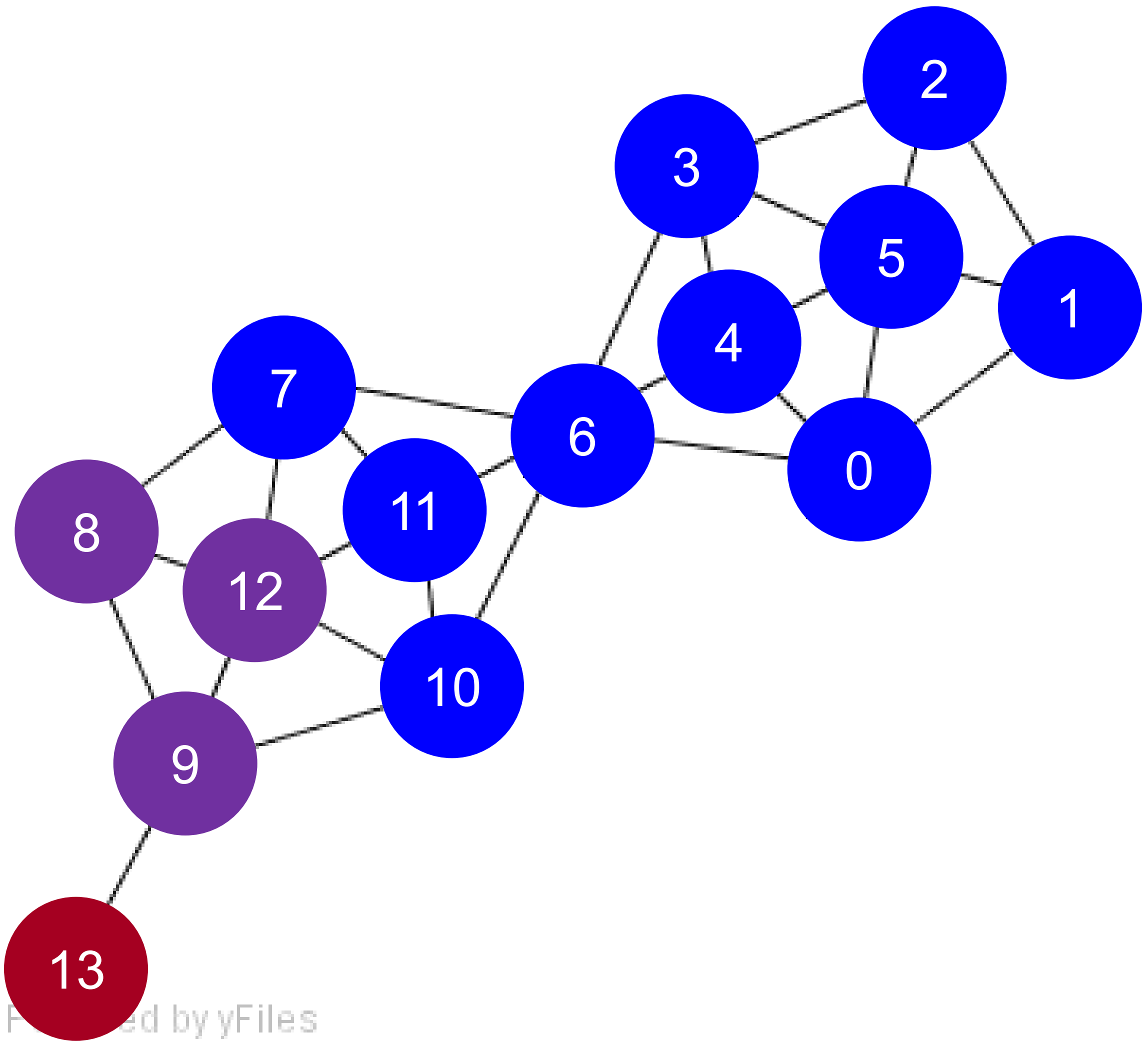
SCAN Algorithm

$\mu = 2$
$\varepsilon = 0.7$



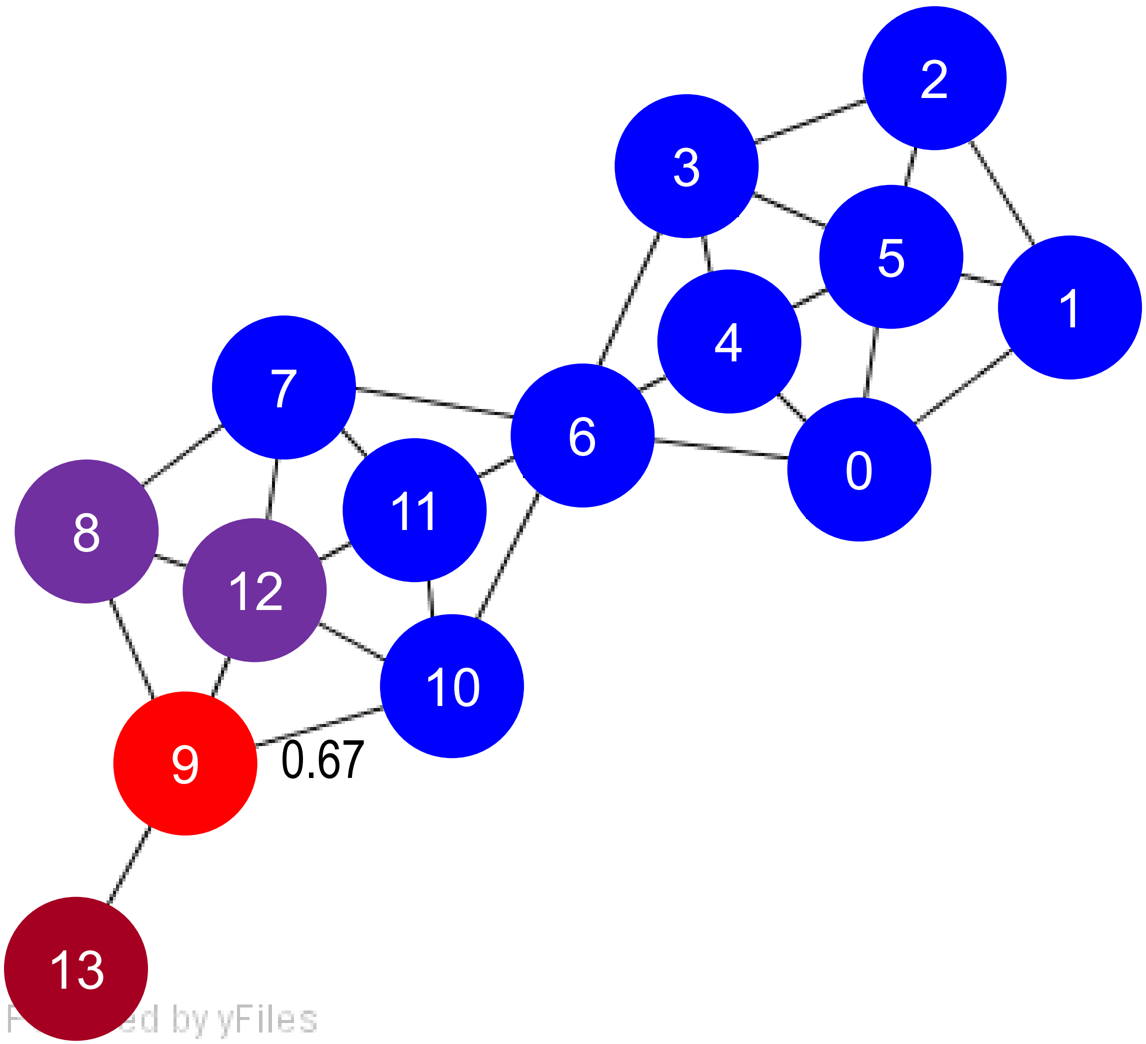
SCAN Algorithm

$\mu = 2$
$\varepsilon = 0.7$



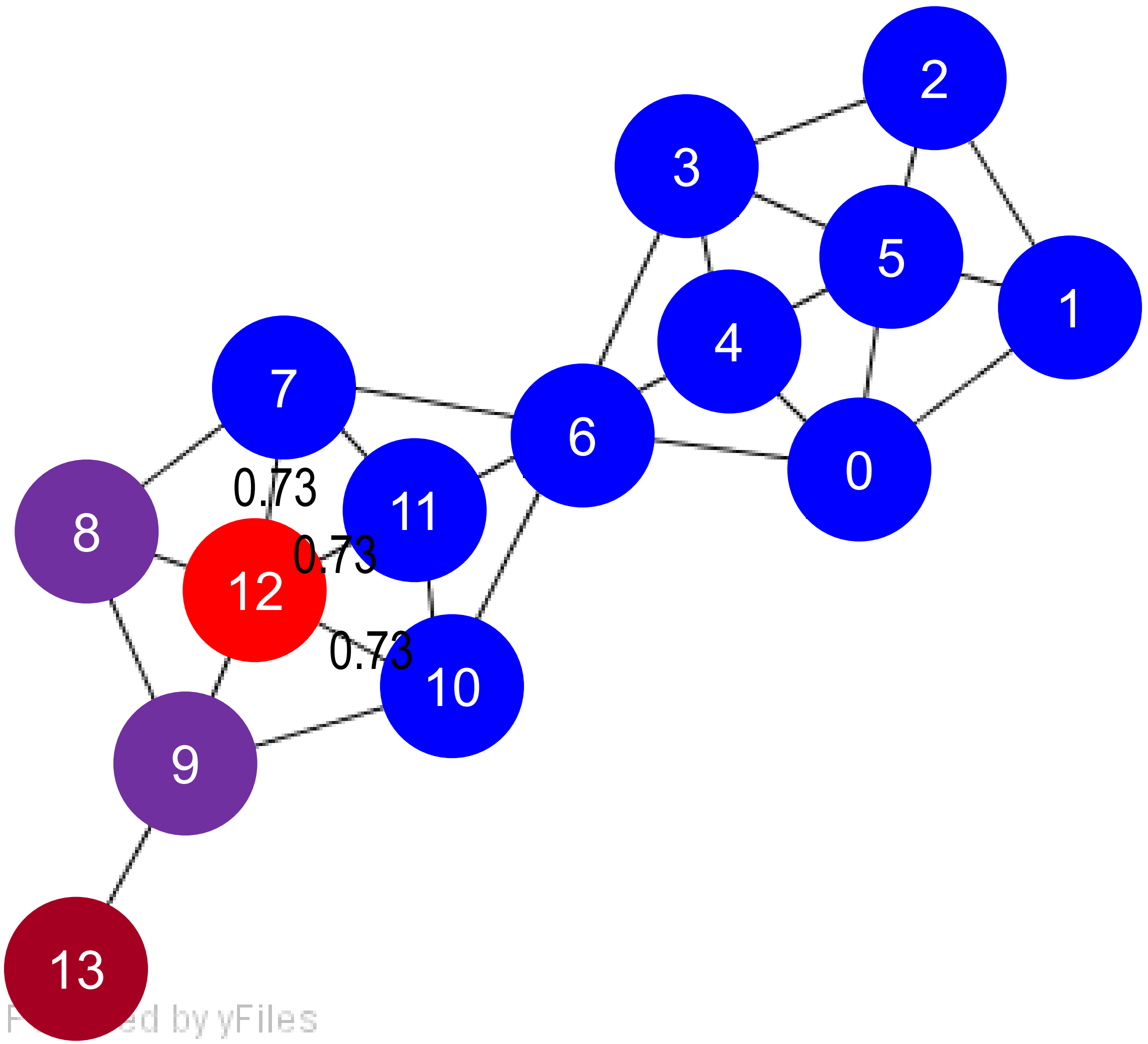
SCAN Algorithm

$\mu = 2$
$\varepsilon = 0.7$



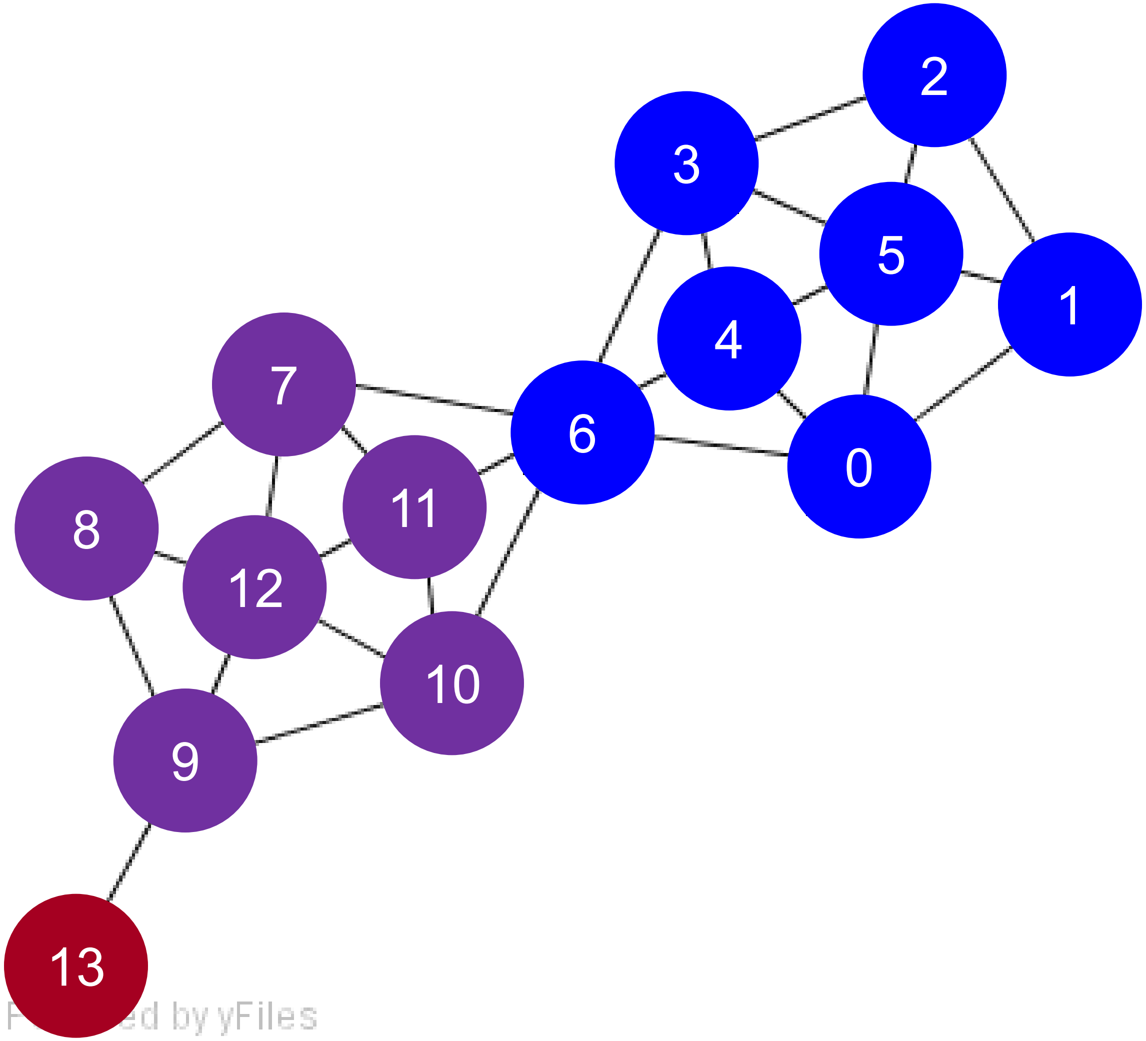
SCAN Algorithm

$\mu = 2$
$\varepsilon = 0.7$



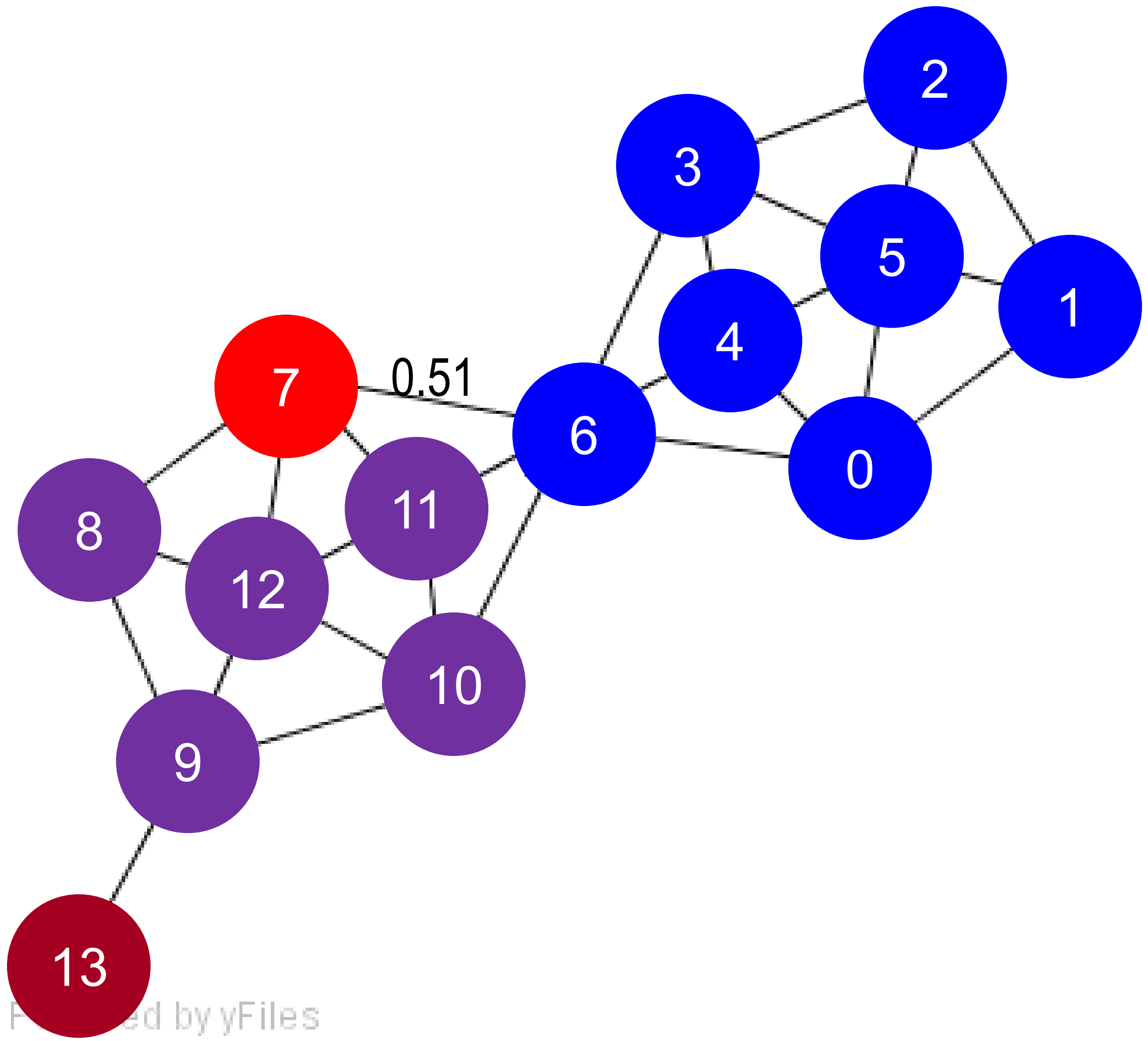
SCAN Algorithm

$\mu = 2$
$\varepsilon = 0.7$



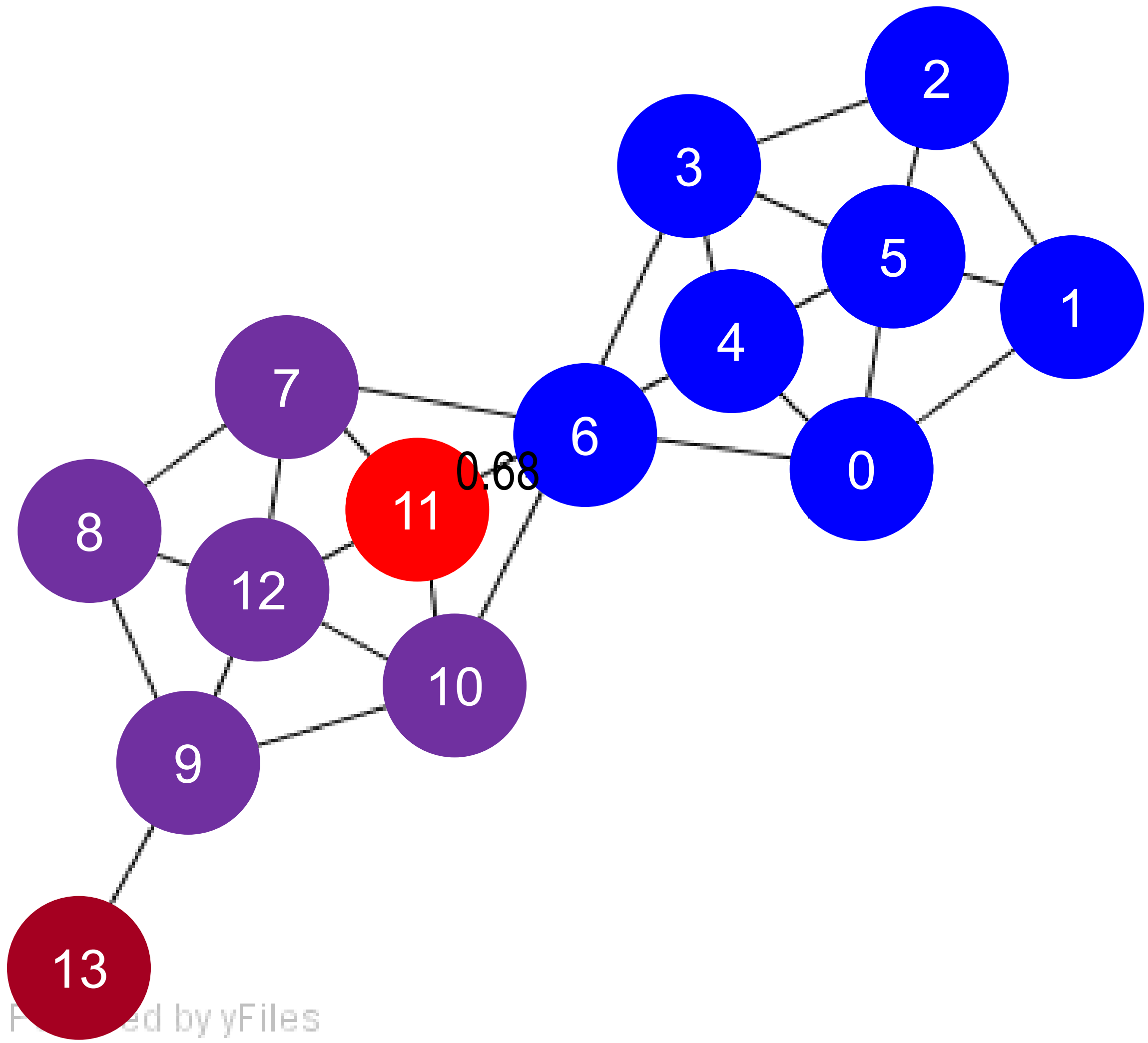
SCAN Algorithm

$\mu = 2$
$\varepsilon = 0.7$



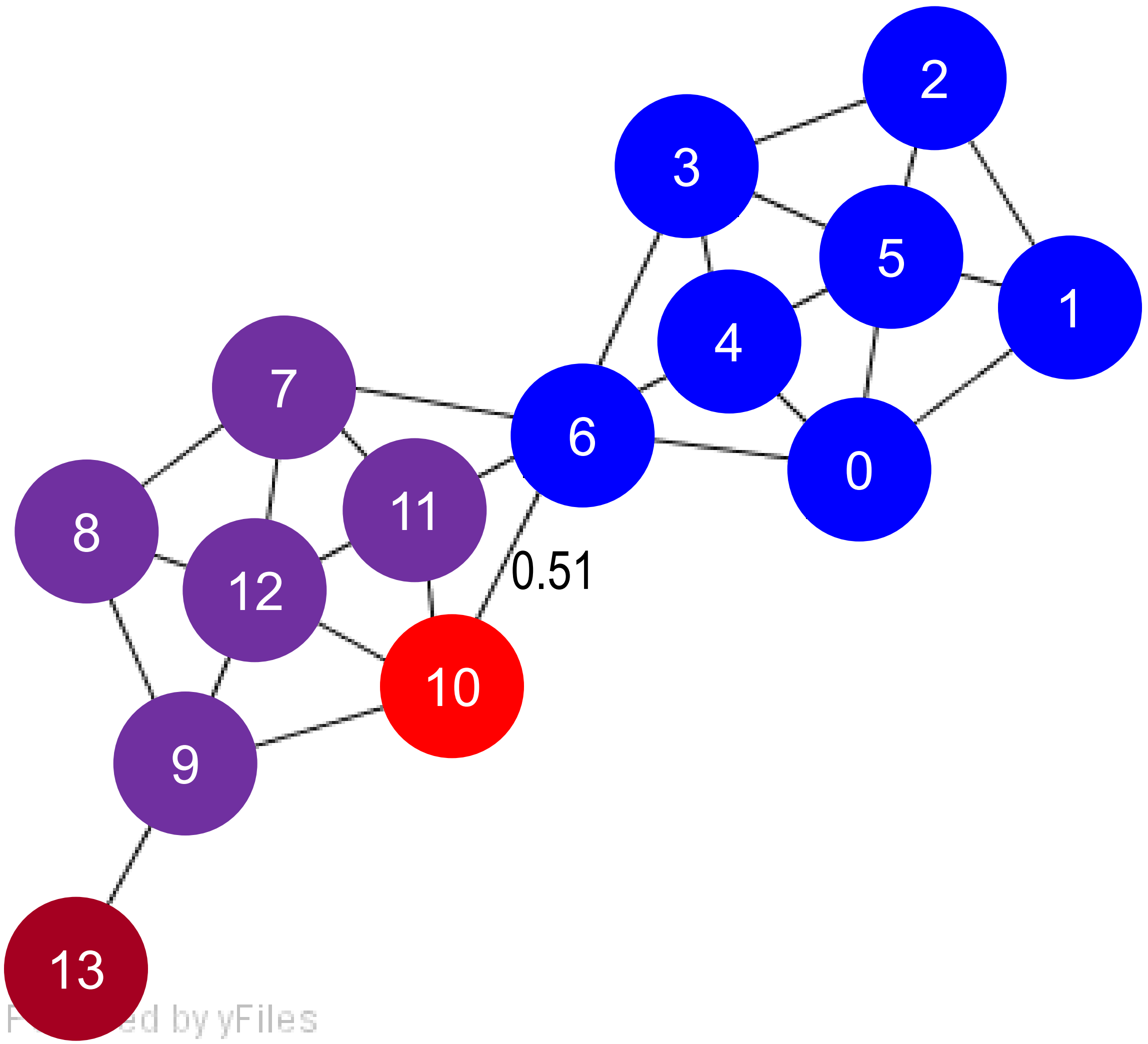
SCAN Algorithm

$\mu = 2$
$\varepsilon = 0.7$



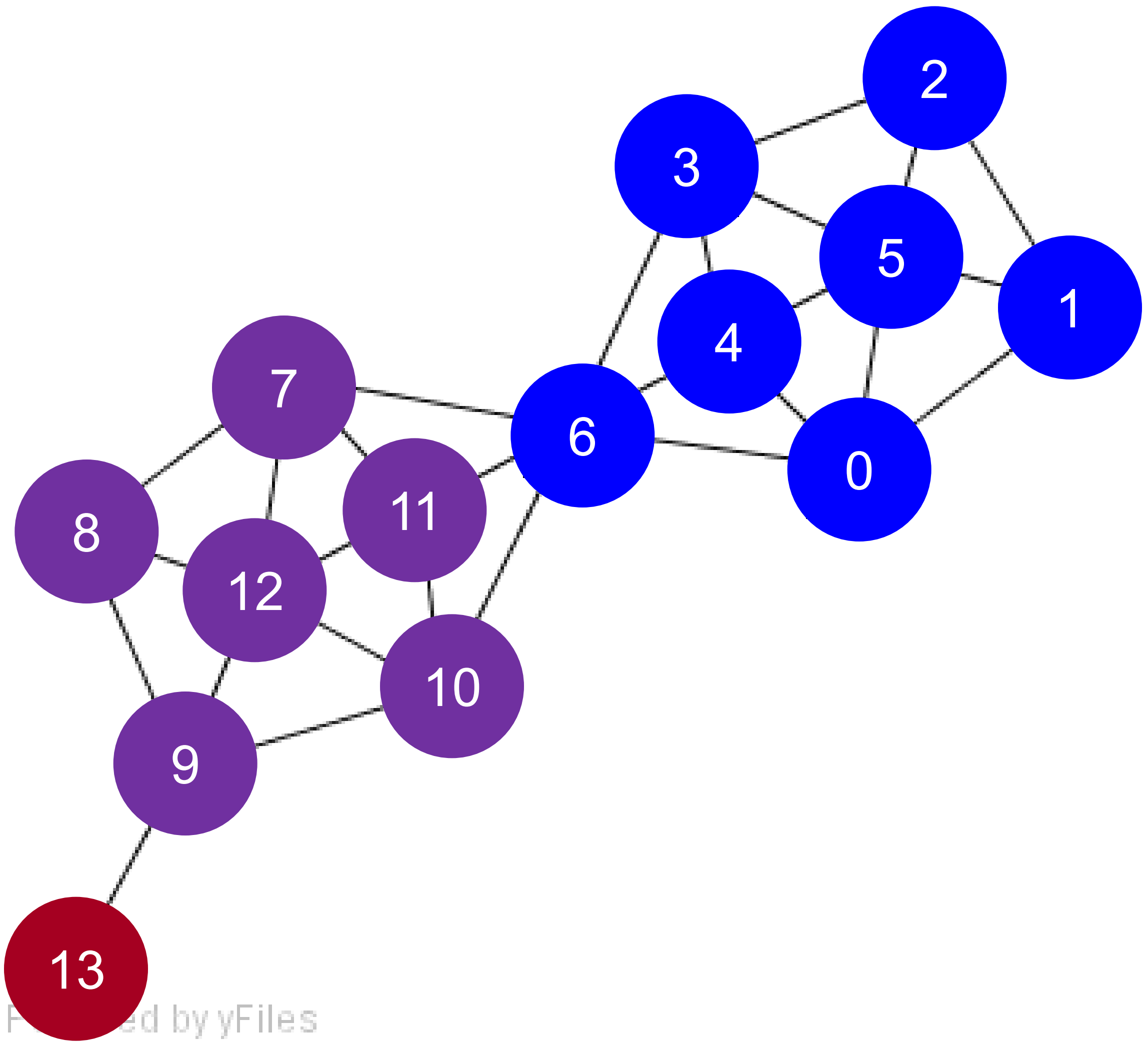
SCAN Algorithm

$\mu = 2$
$\varepsilon = 0.7$



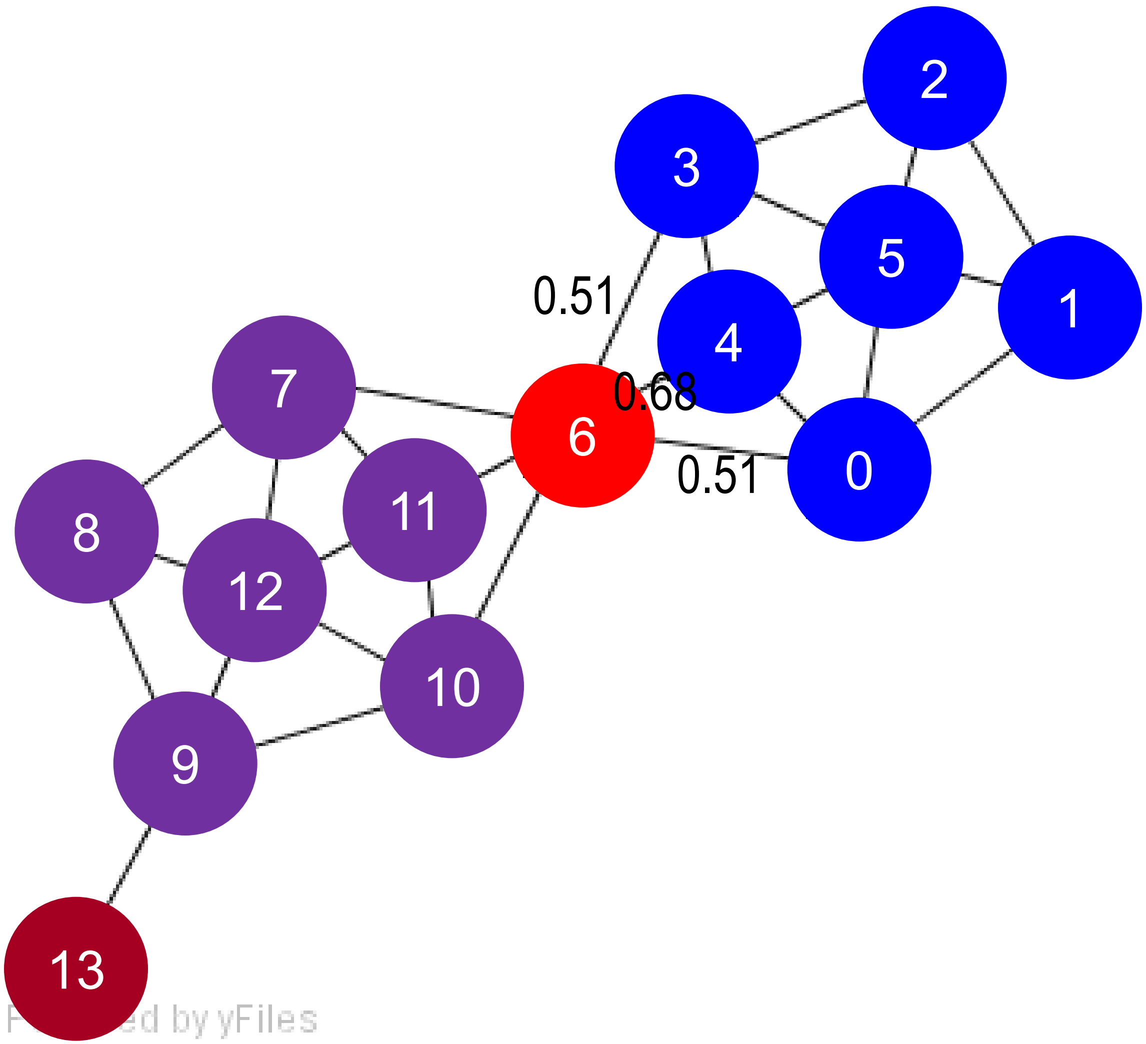
SCAN Algorithm

$\mu = 2$
$\varepsilon = 0.7$



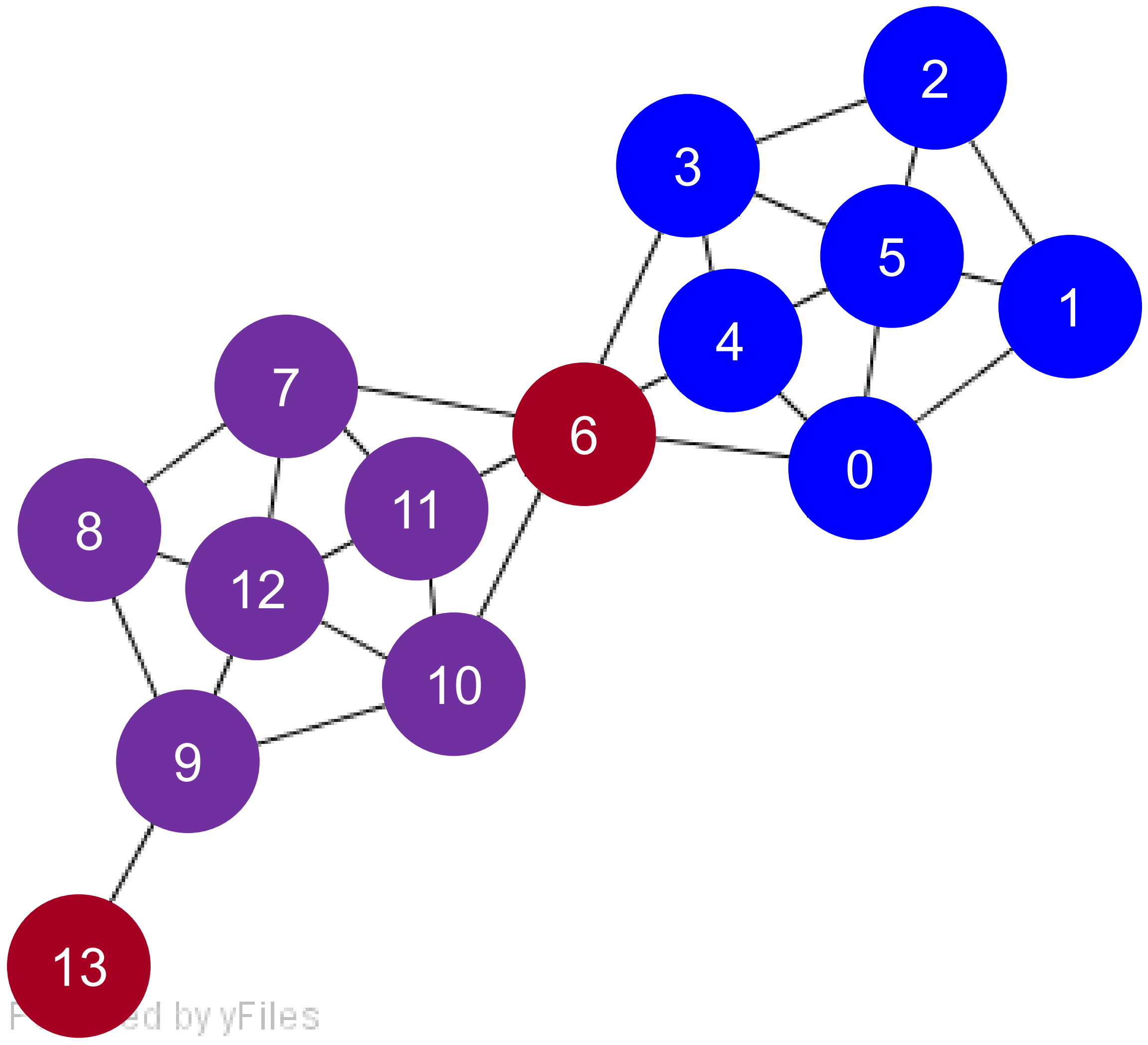
SCAN Algorithm

$\mu = 2$
$\varepsilon = 0.7$



SCAN Algorithm

$\mu = 2$
$\varepsilon = 0.7$



Summary Clustering Graphs

- Directed and undirected graphs
- Distances and structure similarities
- Clustering: sparsest cut, SCAN

Probabilistic Graphical Models

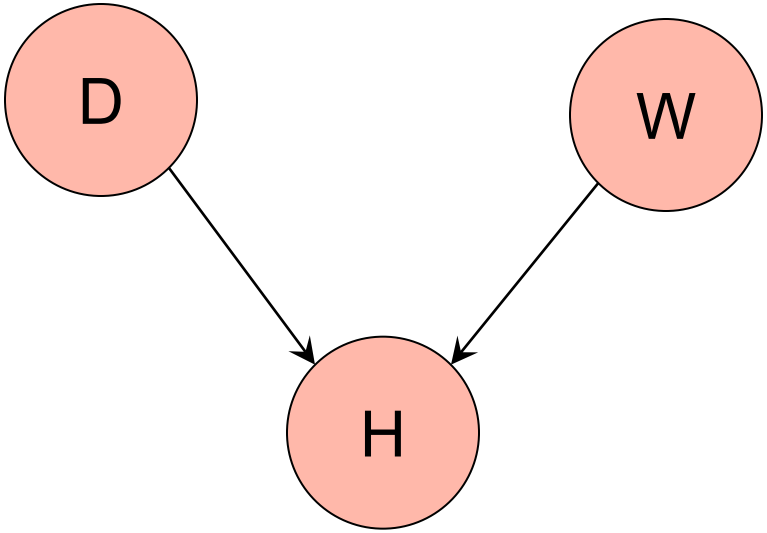
- Modelling of observations and their relationships
- Until now: semantic networks
- Reasoning over properties inherited from other instance(s)
  - *is*-*a*, *has*-*a* relationships

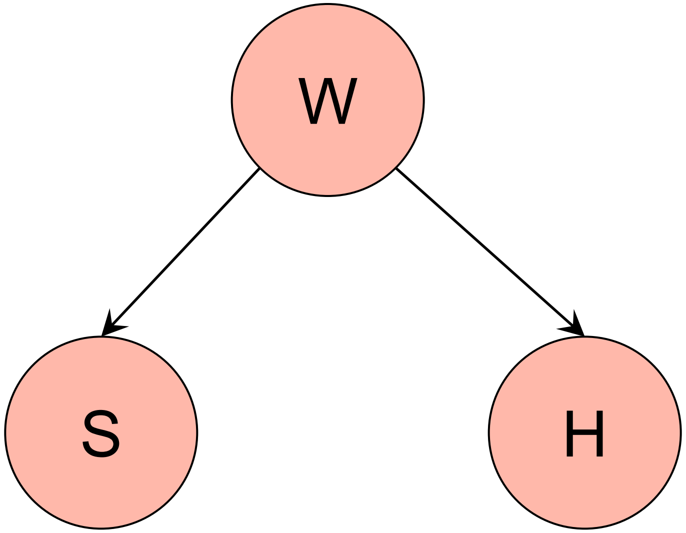
But how certain are we about the resulting statements?

- Make inference under uncertainty
- Stochastics helps us with that
- Probabilistic Graphical Model can be
  directed or ***undirected***

Bayes Theorem

- Computation of the conditional probability
- Priors often fixed
- Computation of the ***posterior distribution***
- Formula allows queries of probable event occurences





Bayesian Belief Network

- also known as ***Bayesian network***, probabilistic network
- Components:

  (1) A *directed acyclic graph* (called a structure)

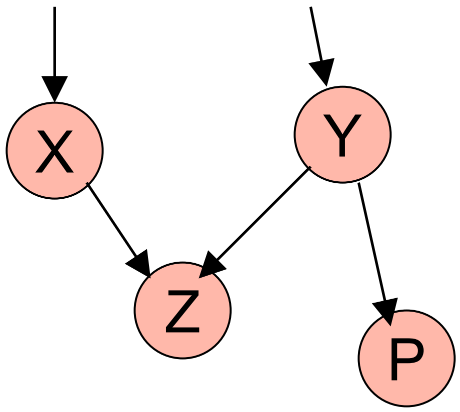
  models of *causal influence* relationships

  represents *dependencies* among the variables

  allows *class conditional independencies* between *subsets* of variables

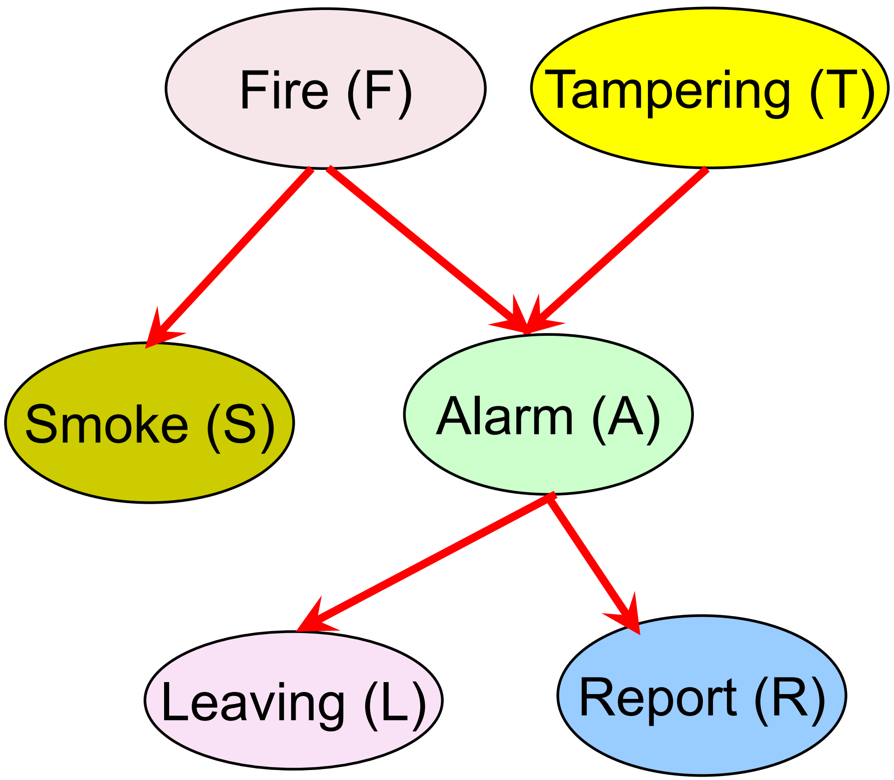
  (2) A set of *conditional probability tables* (CPTs)

  gives a specification of joint probability distribution



- Nodes: random variables
- Links: dependency
- X and Y are the parents of Z, and Y is the parent of P
- No dependency between Z and P
- Has no loops/cycles

A Bayesian Network and Some of Its CPTs



CPT: Conditional Probability Tables

Fire	Smoke	$\Theta_{s\|f}$
True	True	.90
False	True	.01

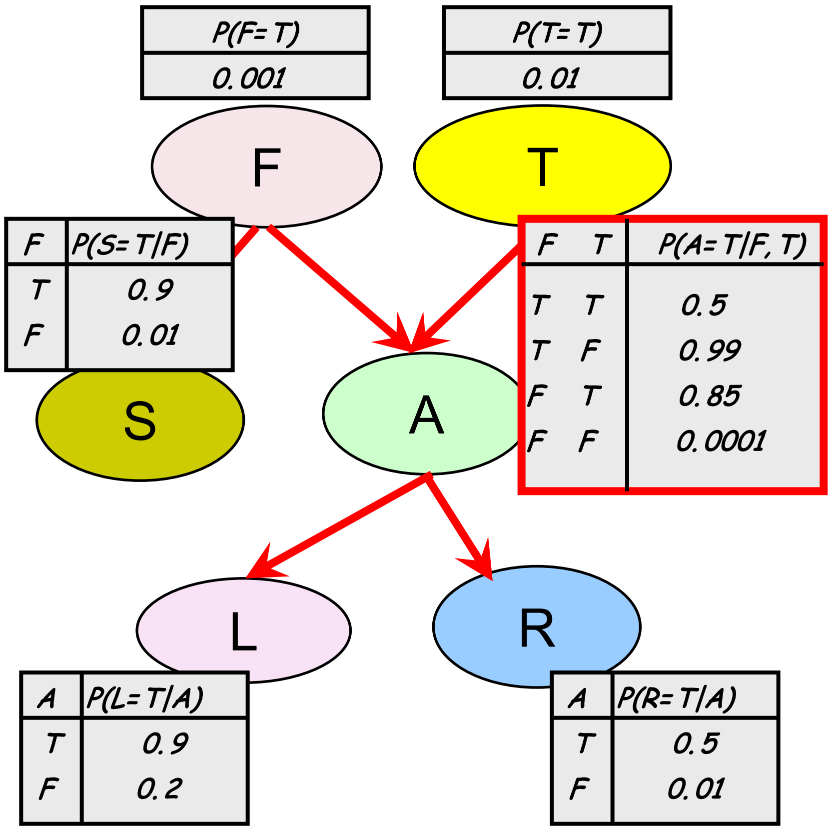
Fire	Tampering	Alarm	$\Theta_{a\|f,t}$
True	True	True	.5
True	False	True	.99
False	True	True	.85
False	False	True	.0001

CPT shows the conditional probability for each possible combination of its parents

Derivation of the probability of a particular combination of values of X, from CPT:

$$P(x_1,\ldots,x_n) = \prod_{i=1}^{n} P(x_i \mid Parents(x_i))$$

A Bayesian Network and Some of Its CPTs



has $O(2^6)$ combinations

$P(S,F,T,A,L,R)$

product rule

$= P(T)\,P(S,F,T,A,L|T)$

F and S are independent of T

$= P(T)\,P(S,F\cancel{|T})\,P(A,L,R|S,F,T)$

product rule

$= P(T)\,P(F)\,P(S|F)\,P(A,L,R|F,T)$

L and R are conditionally independent of F and T given A

$= P(T)\,P(F)\,P(S|F)\,P(A|F,T)\,P(L,R|A\cancel{,F,T})$

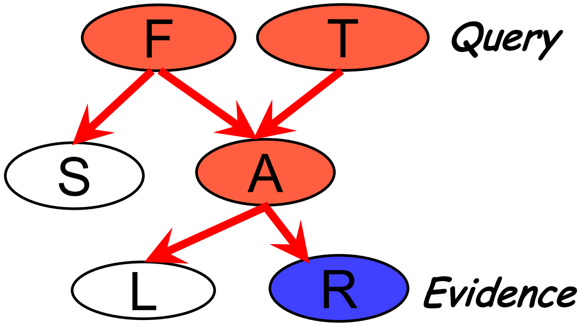
L and R are independent given A

$= P(T)\,P(F)\,P(S|F)\,P(A|F,T)\,P(L|A)\,P(R|A)$

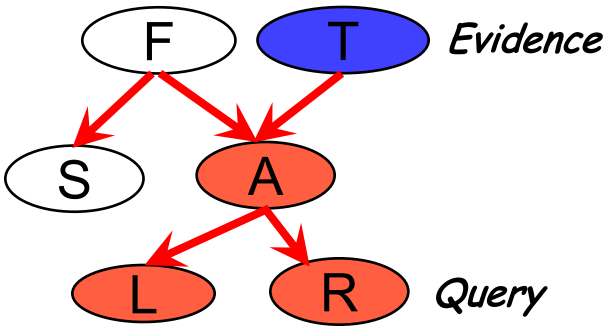
has $O(2^2)$ combinations only

Types of Reasoning in Bayesian Networks

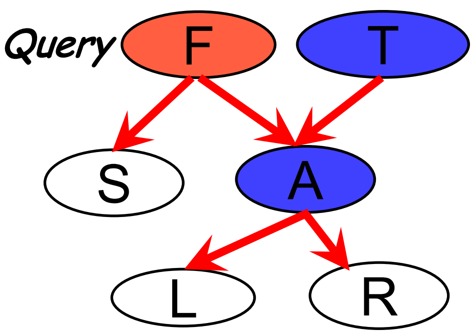
- **Diagnostic**
  - From symptoms to causes, e.g., doctor infers diseases from symptoms. Reasoning occurs in opposite direction to network arcs.



- **Predictive**
  - Reasoning from new information about causes to new beliefs about effects, follows the directions of the networks arcs.



- **Inter-causal (explaining away)**
  - Mutual causes of a common effect. Initially, causes may be independent. But if a common effect is observed and we learn that one cause is true, then the other is less likely – it has been "explained away".



- **Combined**
  - Simultaneous use of diagnostic and predictive reasoning

How Are Bayesian Networks Constructed?

- ***Subjective construction***: Identification of (direct) causal structure
  - People are quite good at identifying direct causes from a given set of variables & whether the set contains all relevant direct causes
  - *Markovian assumption*: Each variable becomes independent of its non-effects once its direct causes are known
  - E.g., $S \leftarrow F \rightarrow A \leftarrow T$, path $S \rightarrow A$ is blocked once we know $F \rightarrow A$
  - HMM (Hidden Markov Model): often used to model dynamic systems whose states are not observable, yet their outputs are
- ***Synthesis*** from other specifications
  - E.g., from a formal system design: block diagrams & info flow
- ***Learning from data***
  - E.g., from medical records or student admission record
  - Learn parameters given its structure or learn both structure and params
  - Maximum likelihood principle: favors Bayesian networks that maximize the probability of observing the given data set

Training Bayesian Networks

- Scenario 1: Given the network structure and all variables observable: → *compute only the CPT entries*
- Scenario 2: Network structure known, some variables hidden: → *gradient descent* (greedy hill-climbing) method, i.e., search for a solution along the steepest descent of a criterion function
  - Weights are initialized to random probability values
  - At each iteration, it moves towards what appears to be the best solution at the moment
  - Weights are updated at each iteration & converge to local optimum
- Scenario 3: Network structure unknown, all variables observable: → search through the model space to *reconstruct network topology*
- Scenario 4: Unknown structure, all hidden variables: → no good algorithms known for this purpose

[D. Heckerman. A Tutorial on Learning with Bayesian Networks. In *Learning in Graphical Models,* M. Jordan, ed. MIT Press, 1999]

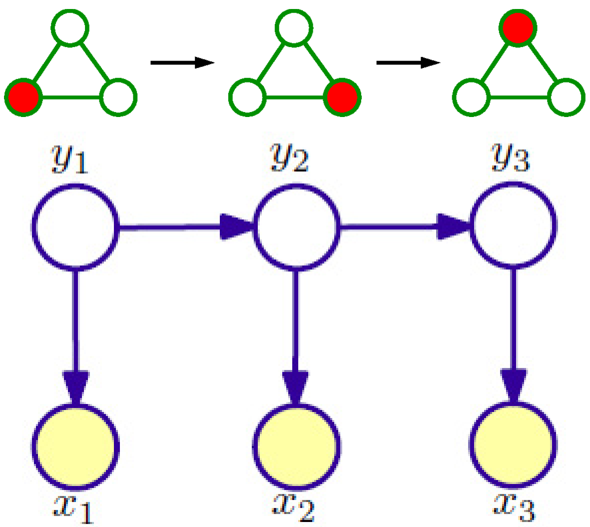
Bayesian Cars

Pradalier, Bessiere (ca. 2005) The CyCab: Bayesian navigation on sensory–motor trajectories

Hidden Markov Models

- Sequential Data modelling
- Applications:
  - Gene prediction
  - Weather forcecasting
  - Automatic speech recognition (Rabiner)
  - Gesture Recognition
  - Information Extraction
  - EEG activity for sleep monitoring
- HMM packages in Matlab (free: Kevin Murphy's), R, Java

Hidden Markov Models



- Model λ:(A, B, π)
  - A: State-transition matrix
  - B: Symbol-emission matrix
  - π: initial state probability vector
- Only emissions are observable, but unknown which state produced them (so: states are ***hidden***)
- State sequence can only be inferred from observed events
- Again: Markov assumption
- State structure typically drawn "unrolled" in time

Weather example

- Given the current weather condition, predict the next possible state (Markov assumption)

		Time $t+1$			
	State	Sunny	Cloudy	Rainy	Snowing
Time t	Sunny	0.6	0.3	0.1	0.0
	Cloudy	0.2	0.4	0.3	0.1
	Rainy	0.1	0.2	0.5	0.2
	Snowing	0.0	0.3	0.2	0.5

Weather Example: Emissions

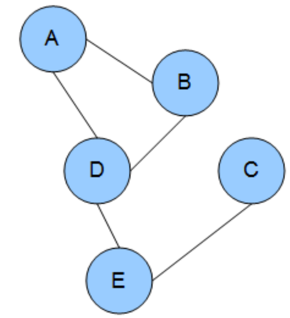
- Classification of weather conditions into {good, bad, variable} is only observable

		Weather			
		Sunny	Cloudy	Rainy	Snowy
Met condition	Good	0.5	0.3	0.2	0.0
	Variable	0.2	0.3	0.3	0.2
	Bad	0.1	0.2	0.3	0.4

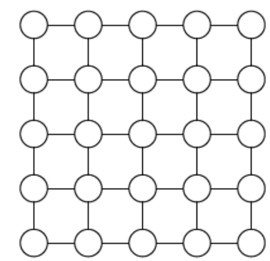
Hidden Markov Models - Problems

- ***Decoding***: Given the HMM and the observation sequence, what is the most probable state sequence?
  - Viterbi algorithm
- ***Evaluation***: Given the HMM and the observation sequence, how probable is it that this HMM generated it?
  - Forward-backward algorithm
- ***Learning***: Find the HMM parameters which best fit to the observation sequence (training data)
  - Baum-Welch algorithm

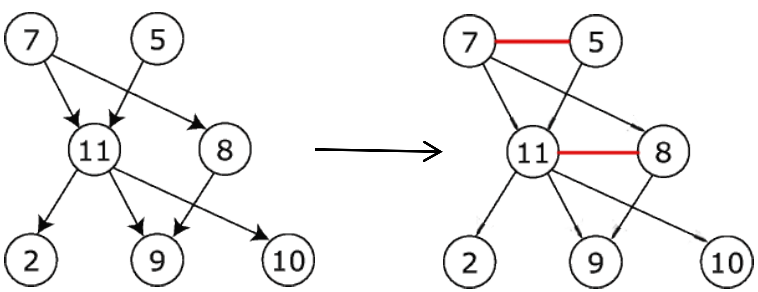
Undirected Graphical Models



- Markov Random Fields or Markov Network
- Originated from statistical physics
- Models correlations, not necessarily causality
  - Image segmentation: pixel correlation

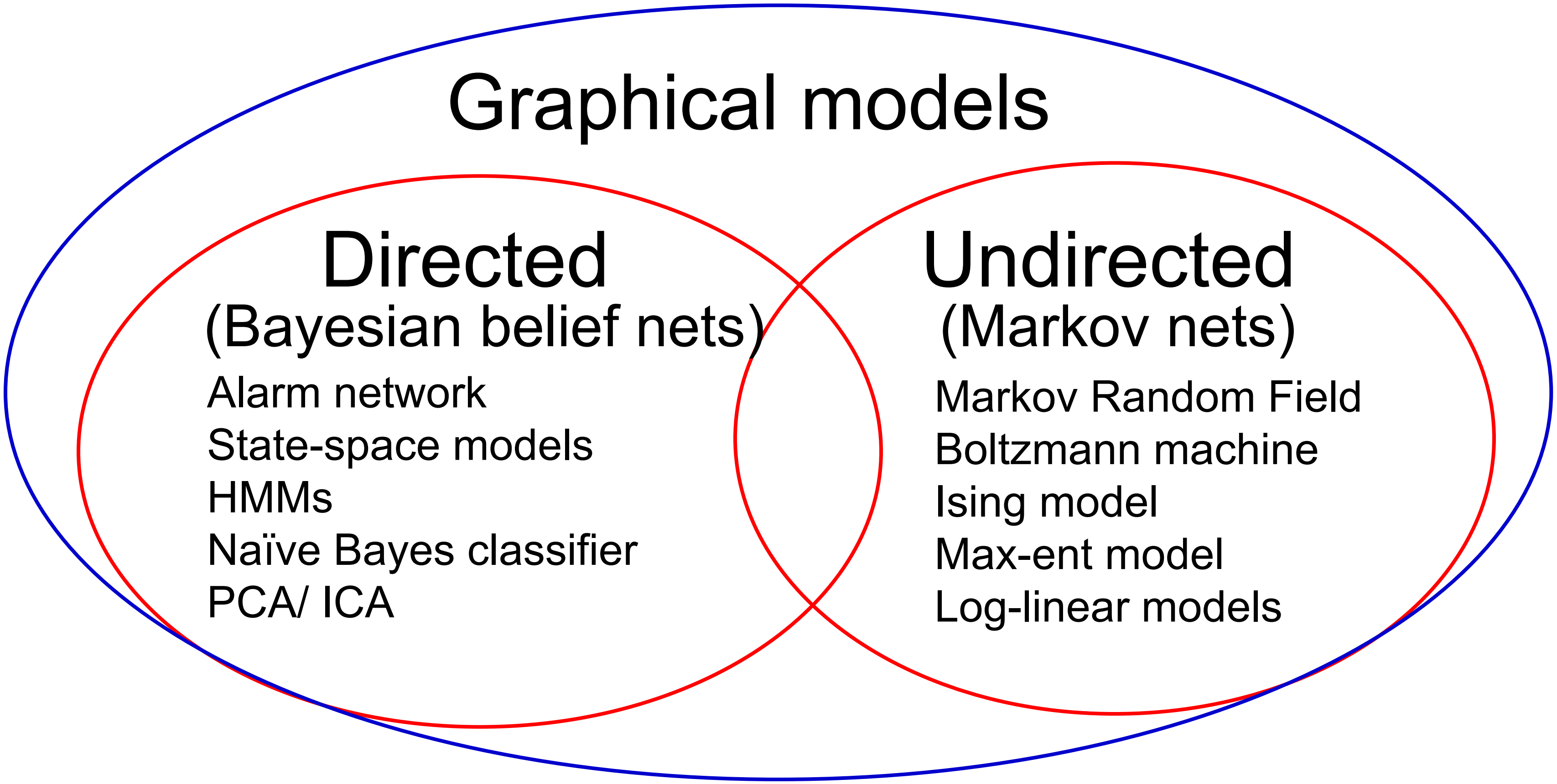


- No modeling of observations
- Definition of ***potentials*** of nodes and edges
- Moralization: every directed model can be transformed into an undirected model



- Inference algorithms as for HMM still applicable

Probabilistic Graphical Models



Summary

- Case Based Reasoning – associating cases from the past
- Semantic Networks – Graphical models for knowledge representation and classification
- Intuitive access to graphical visualization, but usually need experts for the specification
- Time series: Hidden Markov Models
- Probabilistic Reasoning: Bayesian Belief Networks