

# **Data Mining:**

---

## **Concepts and Techniques**

**(3<sup>rd</sup> ed.)**

### **— Chapter 11 —**

Jiawei Han, Micheline Kamber, and Jian Pei  
University of Illinois at Urbana-Champaign &  
Simon Fraser University

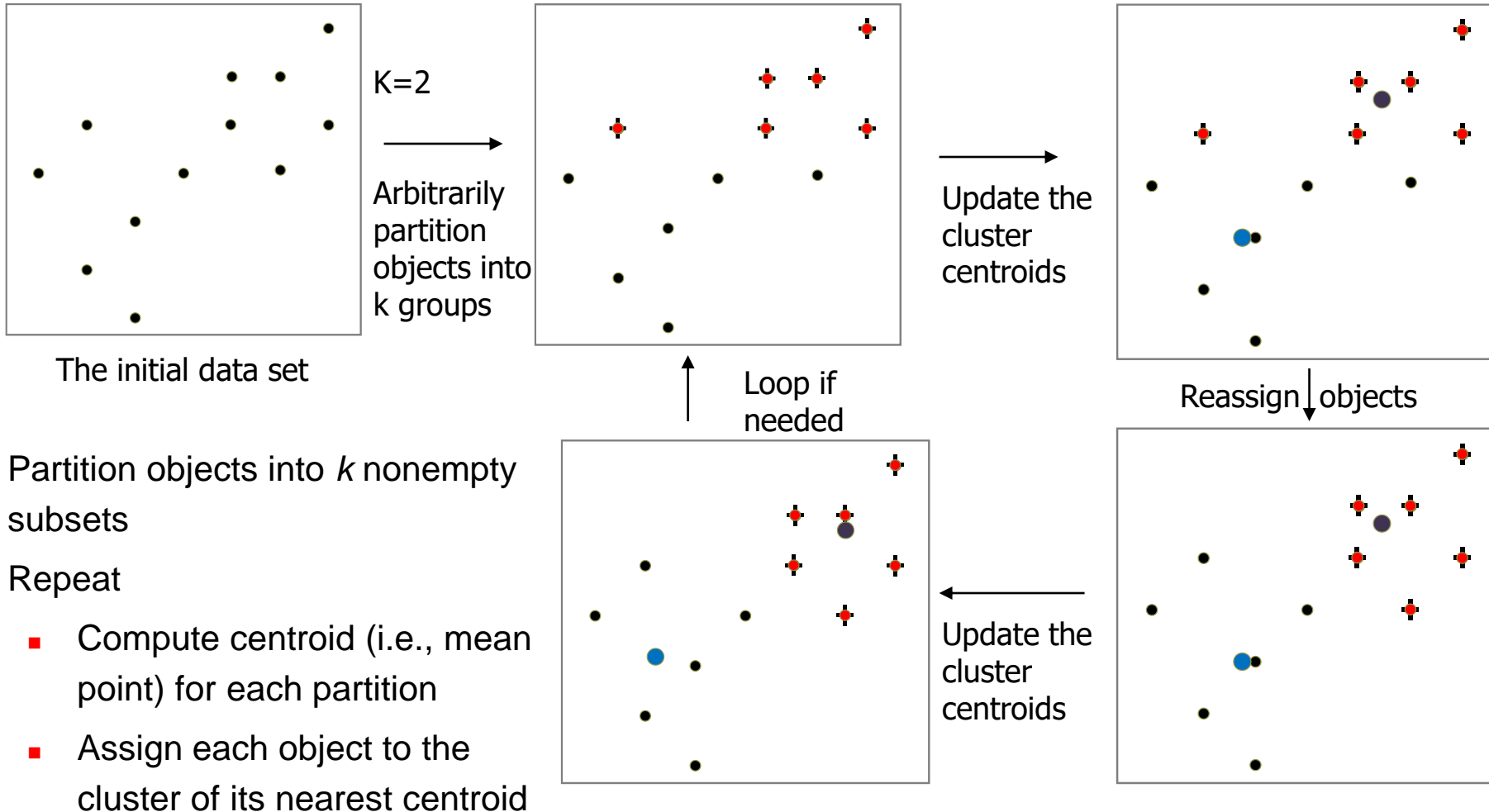
©2011 Han, Kamber & Pei. All rights reserved.

# Review: Basic Cluster Analysis Methods (Chap. 10)

---

- Cluster Analysis: Basic Concepts
  - Group data so that object similarity is high within clusters but low across clusters
- Partitioning Methods
  - K-means and k-medoids algorithms and their refinements
- Hierarchical Methods
  - Agglomerative and divisive method, Birch, Cameleon
- Density-Based Methods
  - DBScan, Optics and DenCLu
- Grid-Based Methods
  - STING and CLIQUE (subspace clustering)
- Evaluation of Clustering
  - Assess clustering tendency, determine # of clusters, and measure clustering quality

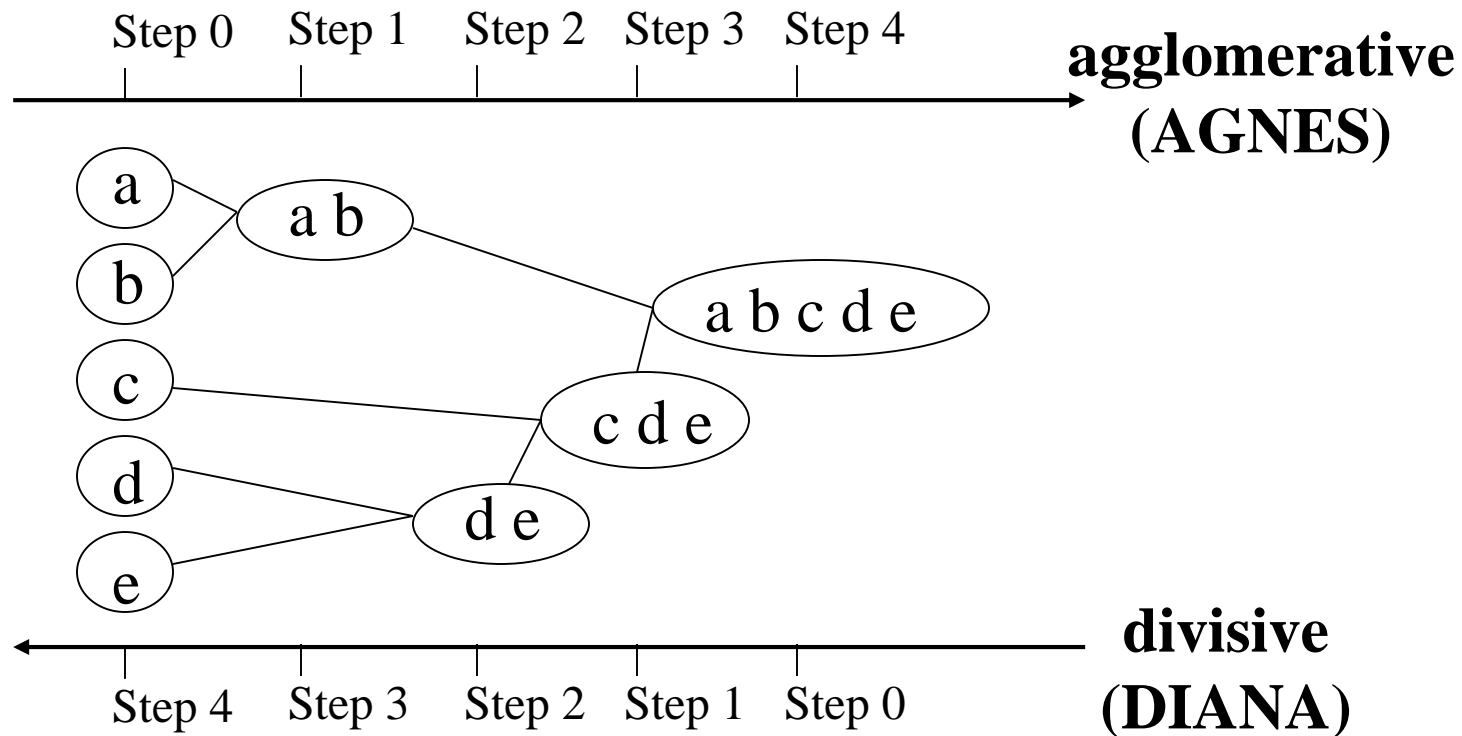
# *K-Means* Clustering



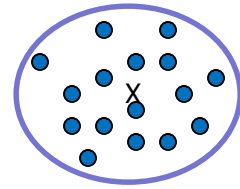
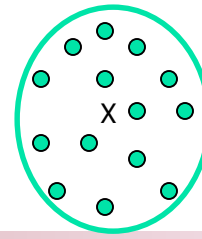
- Partition objects into  $k$  nonempty subsets
- Repeat
  - Compute centroid (i.e., mean point) for each partition
  - Assign each object to the cluster of its nearest centroid
- Until no change

# Hierarchical Clustering

- Use distance matrix as clustering criteria. This method does not require the number of clusters  $k$  as an input, but needs a termination condition



# Distance between Clusters

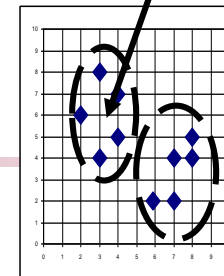
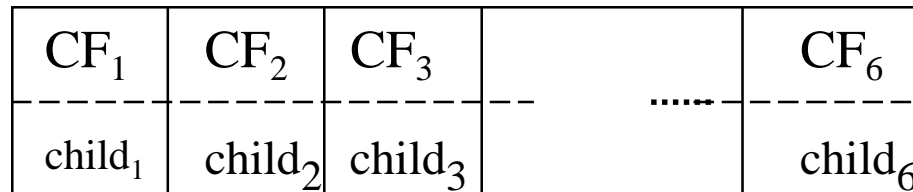


- Single link: smallest distance between an element in one cluster and an element in the other, i.e.,  $\text{dist}(K_i, K_j) = \min(t_{ip}, t_{jq})$
- Complete link: largest distance between an element in one cluster and an element in the other, i.e.,  $\text{dist}(K_i, K_j) = \max(t_{ip}, t_{jq})$
- Average: avg distance between an element in one cluster and an element in the other, i.e.,  $\text{dist}(K_i, K_j) = \text{avg}(t_{ip}, t_{jq})$
- Centroid: distance between the centroids of two clusters, i.e.,  $\text{dist}(K_i, K_j) = \text{dist}(C_i, C_j)$
- Medoid: distance between the medoids of two clusters, i.e.,  $\text{dist}(K_i, K_j) = \text{dist}(M_i, M_j)$ 
  - Medoid: a chosen, centrally located object in the cluster

# BIRCH and the Clustering Feature (CF) Tree Structure

$B = 7$  Root

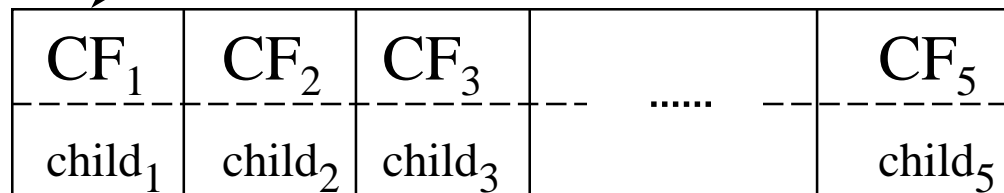
$L = 6$



CF = (5,  
(16,30),(54,190))

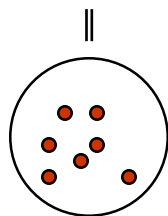
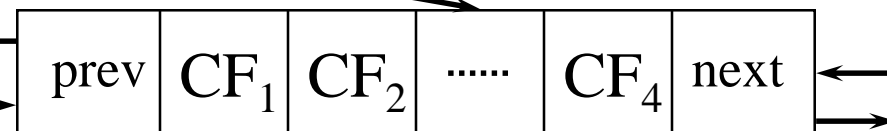
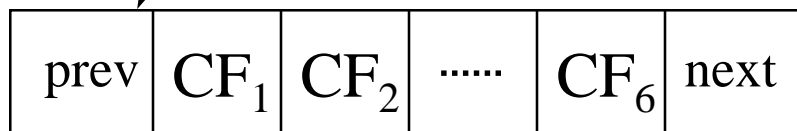
(3,4)  
(2,6)  
(4,5)  
(4,7)  
(3,8)

Non-leaf node

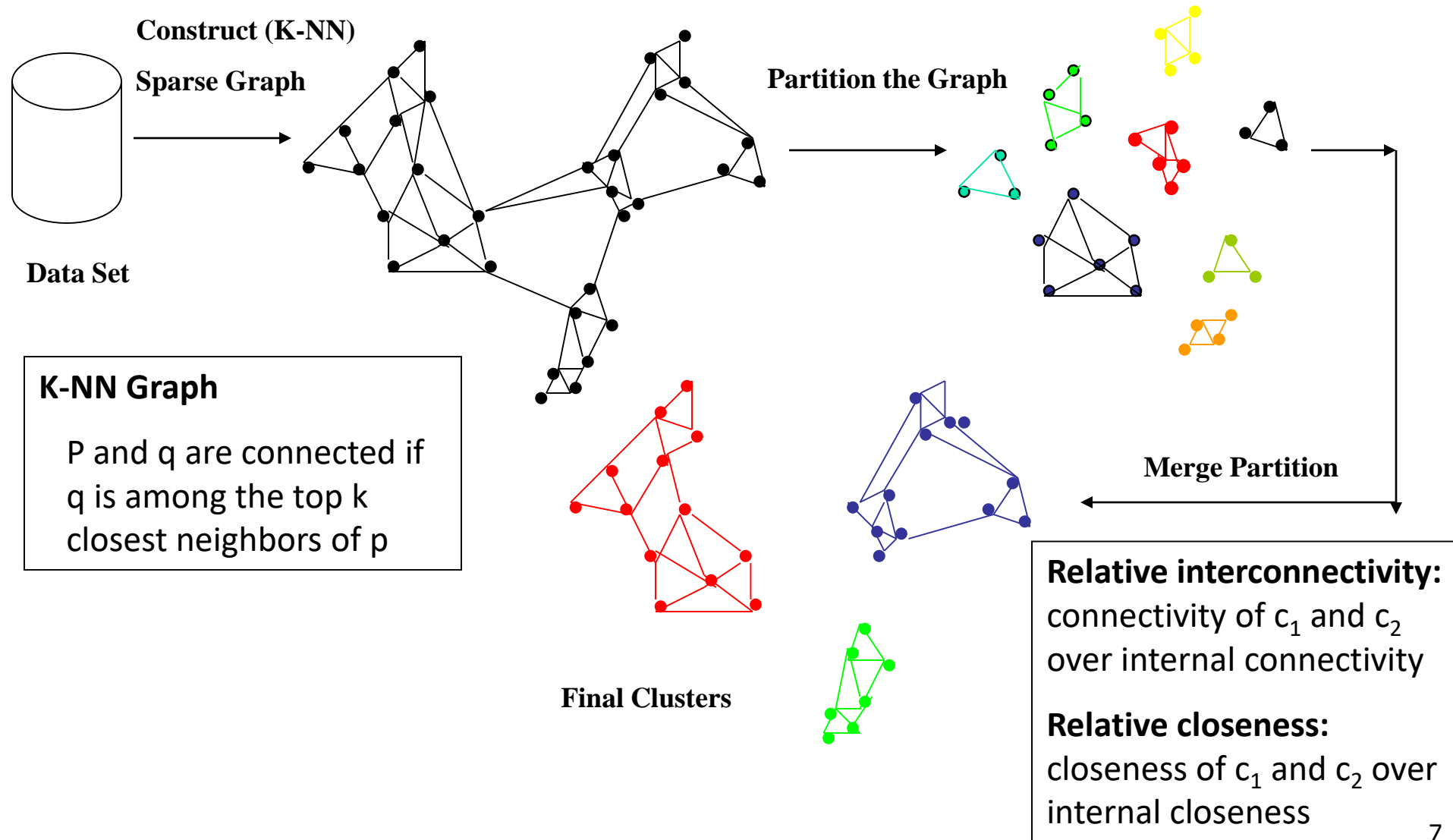


Leaf node

Leaf node



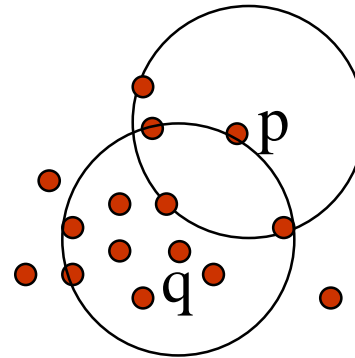
# Overall Framework of CHAMELEON



# Density-Based Clustering: DBSCAN

- Two parameters:
  - **Eps**: Maximum radius of the neighbourhood
  - **MinPts**: Minimum number of points in an Eps-neighbourhood of that point
- $N_{Eps}(p)$ :  $\{q \text{ belongs to } D \mid \text{dist}(p,q) \leq Eps\}$
- **Directly density-reachable**: A point  $p$  is directly density-reachable from a point  $q$  w.r.t.  $Eps$ ,  $MinPts$  if
  - $p$  belongs to  $N_{Eps}(q)$
  - core point condition:

$$|N_{Eps}(q)| \geq MinPts$$

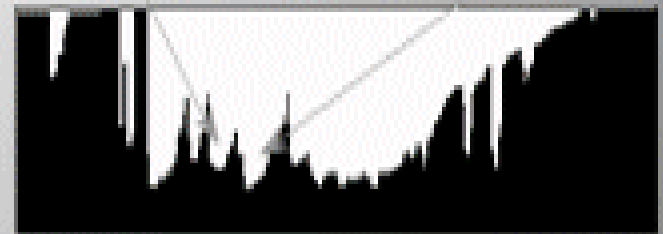
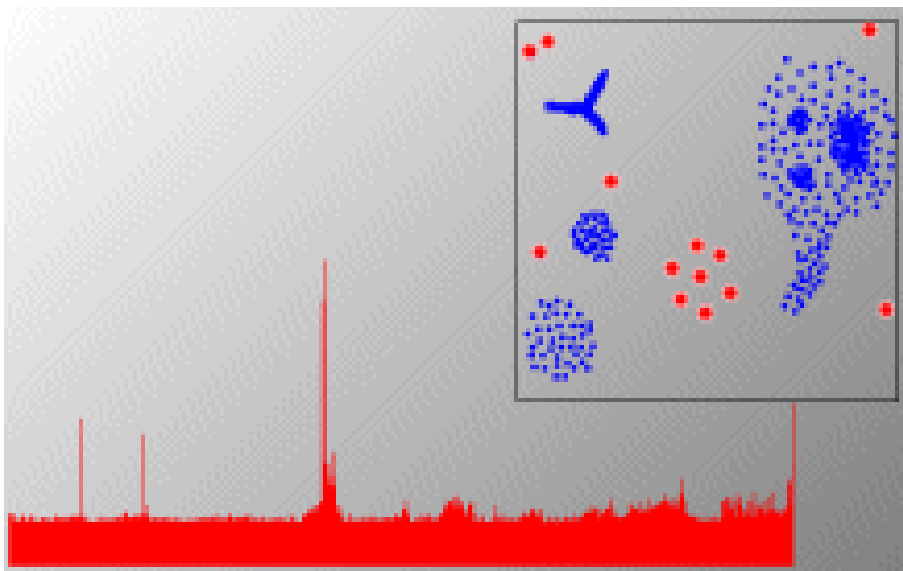
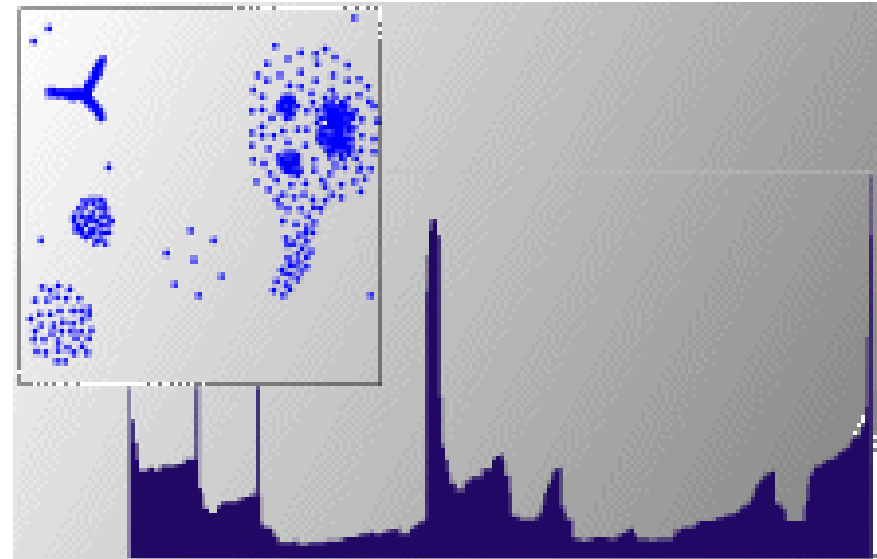
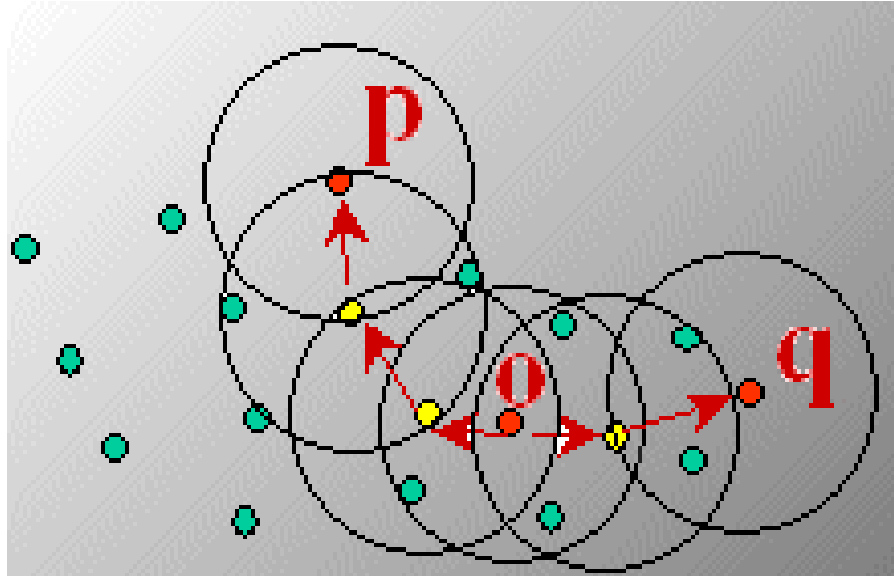


MinPts = 5

Eps = 1 cm



# Density-Based Clustering: OPTICS & Its Applications



# DENCLU: Center-Defined and Arbitrary

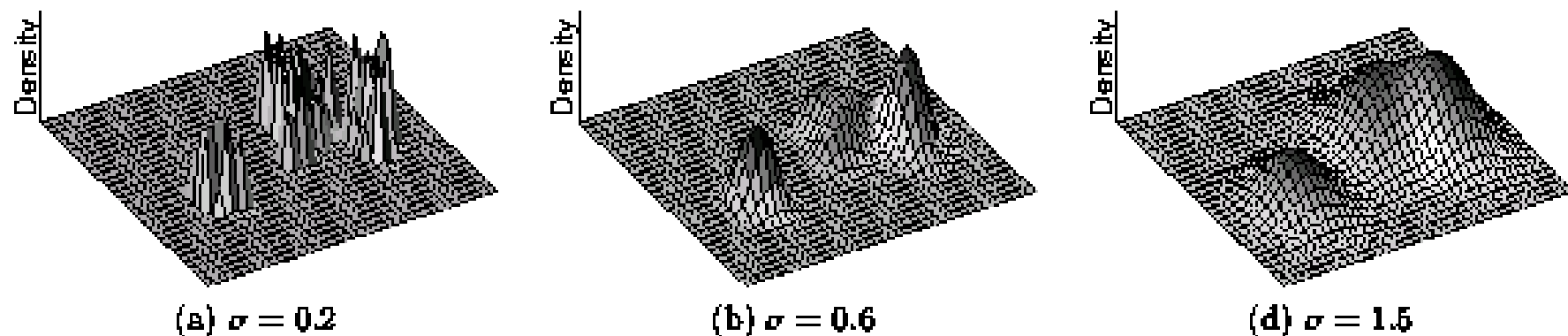


Figure 3: Example of Center-Defined Clusters for different  $\sigma$

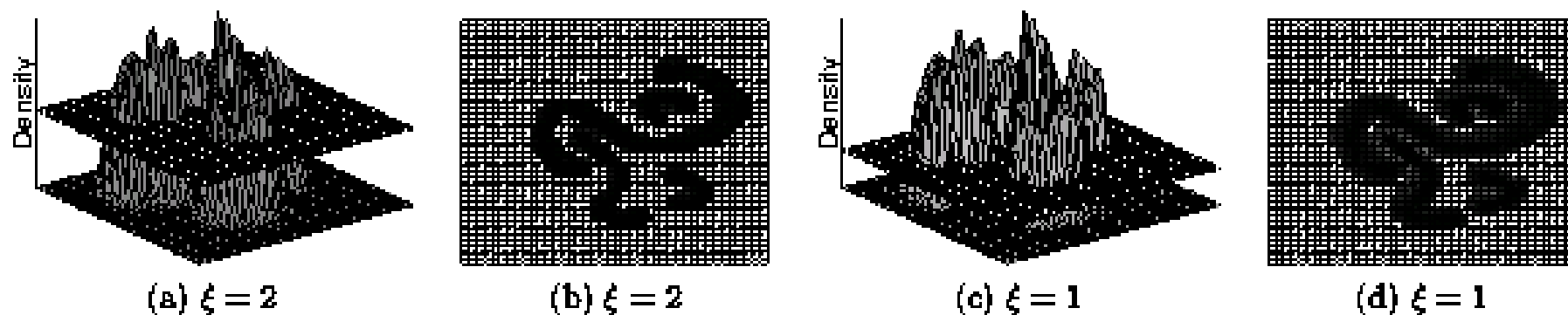
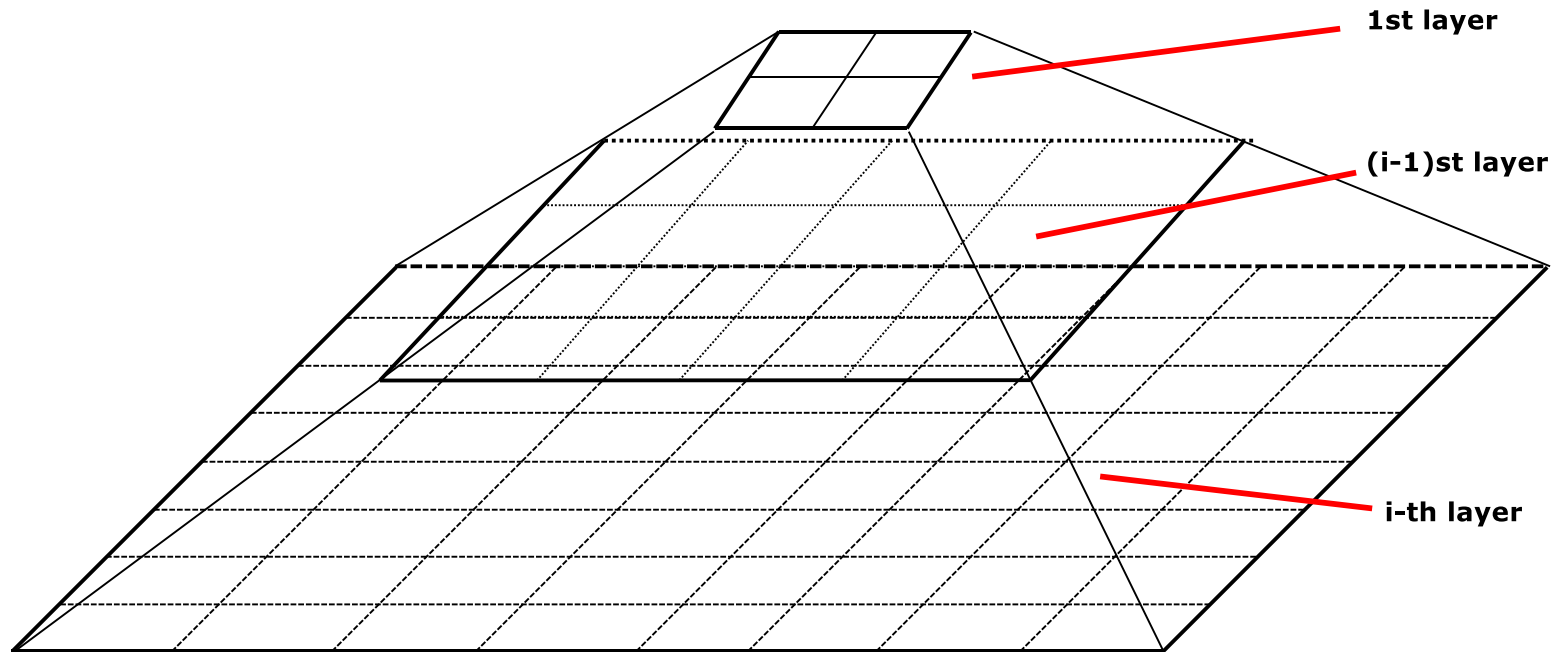


Figure 4: Example of Arbitrary-Shape Clusters for different  $\xi$

# STING: A Statistical Information Grid Approach

- Wang, Yang and Muntz (VLDB'97)
- The spatial area is divided into rectangular cells
- There are several levels of cells corresponding to different levels of resolution



# Evaluation of Clustering Quality

---

- Assessing Clustering Tendency
  - Assess if non-random structure exists in the data by measuring the probability that the data is generated by a uniform data distribution
- Determine the Number of Clusters
  - Empirical method: # of clusters  $\approx \sqrt{n/2}$
  - Elbow method: Use the turning point in the curve of sum of within cluster variance w.r.t # of clusters
  - Cross validation method
- Measuring Clustering Quality
  - Extrinsic: supervised
    - Compare a clustering against the ground truth using certain clustering quality measure
  - Intrinsic: unsupervised
    - Evaluate the goodness of a clustering by considering how well the clusters are separated, and how compact the clusters are

# Outline of Advanced Clustering Analysis

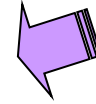
---

- Probability Model-Based Clustering
  - Each object may take a probability to belong to a cluster
- Clustering High-Dimensional Data
  - Curse of dimensionality: Difficulty of distance measure in high-D space
- Clustering Graphs and Network Data
  - Similarity measurement and clustering methods for graph and networks
- Clustering with Constraints
  - Cluster analysis under different kinds of constraints, e.g., that raised from background knowledge or spatial distribution of the objects

# Chapter 11. Cluster Analysis: Advanced Methods

---

- Probability Model-Based Clustering
- Clustering High-Dimensional Data
- Clustering Graphs and Network Data
- Clustering with Constraints
- Summary



# Fuzzy Set and Fuzzy Cluster

- Clustering methods discussed so far
  - Every data object is assigned to exactly one cluster
- Some applications may need for fuzzy or soft cluster assignment
  - Ex. An e-game could belong to both entertainment and software
- Methods: fuzzy clusters and probabilistic model-based clusters
- Fuzzy cluster: A fuzzy set  $S$ :  $F_S : X \rightarrow [0, 1]$  (value between 0 and 1)
- Example: Popularity of cameras is defined as a fuzzy mapping

Camera	Sales (units)
$A$	50
$B$	1320
$C$	860
$D$	270

$$\text{Pop}(o) = \begin{cases} 1 & \text{if 1,000 or more units of } o \text{ are sold} \\ \frac{i}{1000} & \text{if } i \text{ } (i < 1000) \text{ units of } o \text{ are sold} \end{cases}$$

- Then,  $A(0.05)$ ,  $B(1)$ ,  $C(0.86)$ ,  $D(0.27)$

# Fuzzy (Soft) Clustering

- Example: Let cluster features be
  - $C_1$  :“digital camera” and “lens”
  - $C_2$ : “computer”

Review-id	Keywords
$R_1$	digital camera, lens
$R_2$	digital camera
$R_3$	lens
$R_4$	digital camera, lens, computer
$R_5$	computer, CPU
$R_6$	computer, computer game

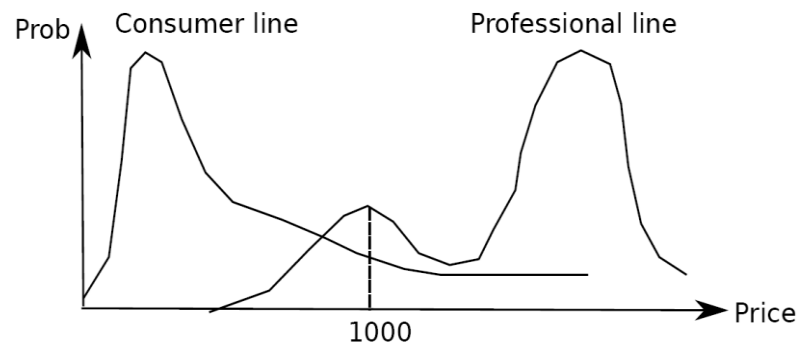
$$M = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ \frac{2}{3} & \frac{1}{3} \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$$

- Fuzzy clustering
  - k fuzzy clusters  $C_1, \dots, C_k$ , represented as a partition matrix  $M = [w_{ij}]$
  - P1: for each object  $o_i$  and cluster  $C_j$ ,  $0 \leq w_{ij} \leq 1$  (fuzzy set)
  - P2: for each object  $o_i$ ,  $\sum_{j=1}^k w_{ij} = 1$ , equal participation in the clustering
  - P3: for each cluster  $C_j$ ,  $0 < \sum_{i=1}^n w_{ij} < n$  ensures there is no empty cluster
- Let  $c_1, \dots, c_k$  as the center of the k clusters
- For an object  $o_i$ , sum of the squared error (SSE), p is a parameter:
- For a cluster  $C_j$  SSE: 
$$SSE(C_j) = \sum_{i=1}^n w_{ij}^p \text{dist}(o_i, c_j)^2 \quad SSE(o_i) = \sum_{j=1}^k w_{ij}^p \text{dist}(o_i, c_j)^2$$
- Measure how well a clustering fits the data: 
$$SSE(C) = \sum_{i=1}^n \sum_{j=1}^k w_{ij}^p \text{dist}(o_i, c_j)^2$$



# Probabilistic Model-Based Clustering

- Cluster analysis is to find hidden categories.
- A hidden category (i.e., *probabilistic cluster*) is a distribution over the data space, which can be mathematically represented using a probability density function (or distribution function).
- Ex. 2 categories for digital cameras sold
  - consumer line vs. professional line
  - density functions  $f_1, f_2$  for  $C_1, C_2$
  - obtained by probabilistic clustering
- A **mixture model** assumes that a set of observed objects is a mixture of instances from multiple probabilistic clusters, and conceptually each observed object is generated independently
- **Out task**: infer a set of  $k$  probabilistic clusters that is mostly likely to generate  $D$  using the above data generation process



# Model-Based Clustering

- A set  $C$  of  $k$  probabilistic clusters  $C_1, \dots, C_k$  with probability density functions  $f_1, \dots, f_k$ , respectively, and their probabilities  $\omega_1, \dots, \omega_k$ .
- Probability of an object  $o$  generated by cluster  $C_j$  is  $P(o|C_j) = \omega_j f_j(o)$
- Probability of  $o$  generated by the set of cluster  $C$  is  $P(o|C) = \sum_{j=1}^k \omega_j f_j(o)$
- Since objects are assumed to be generated independently, for a data set  $D = \{o_1, \dots, o_n\}$ , we have,

$$P(D|C) = \prod_{i=1}^n P(o_i|C) = \prod_{i=1}^n \sum_{j=1}^k \omega_j f_j(o_i)$$

- Task: Find a set  $C$  of  $k$  probabilistic clusters s.t.  $P(D|C)$  is maximized
- However, maximizing  $P(D|C)$  is often intractable since the probability density function of a cluster can take an arbitrarily complicated form
- To make it computationally feasible (as a compromise), assume the probability density functions being some parameterized distributions

# Univariate Gaussian Mixture Model

- $O = \{o_1, \dots, o_n\}$  ( $n$  observed objects),  $\Theta = \{\theta_1, \dots, \theta_k\}$  (parameters of the  $k$  distributions), and  $P_j(o_i | \theta_j)$  is the probability that  $o_i$  is generated from the  $j$ -th distribution using parameter  $\theta_j$ , we have

$$P(o_i | \Theta) = \sum_{j=1}^k \omega_j P_j(o_i | \theta_j) \quad P(O | \Theta) = \prod_{i=1}^n \sum_{j=1}^k \omega_j P_j(o_i | \theta_j)$$

- Univariate Gaussian mixture model
  - Assume the probability density function of each cluster follows a 1-d Gaussian distribution. Suppose that there are  $k$  clusters.
  - The probability density function of each cluster are centered at  $\mu_j$  with standard deviation  $\sigma_j$ ,  $\theta_j = (\mu_j, \sigma_j)$ , we have

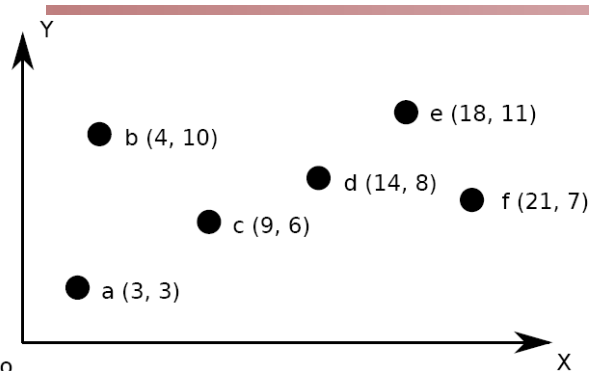
$$P(o_i | \theta_j) = \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{(o_i - \mu_j)^2}{2\sigma_j^2}} \quad P(o_i | \Theta) = \sum_{j=1}^k \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{(o_i - \mu_j)^2}{2\sigma_j^2}}$$
$$P(O | \Theta) = \prod_{i=1}^n \sum_{j=1}^k \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{(o_i - \mu_j)^2}{2\sigma_j^2}}$$

# The EM (Expectation Maximization) Algorithm

---

- The k-means algorithm has two steps at each iteration:
  - **Expectation Step** (E-step): Given the current cluster centers, each object is assigned to the cluster whose center is closest to the object: An object is *expected to belong to the closest cluster*
  - **Maximization Step** (M-step): Given the cluster assignment, for each cluster, the algorithm *adjusts the center* so that *the sum of distance* from the objects assigned to this cluster and the new center is minimized
- **The (EM) algorithm:** A framework to approach maximum likelihood or maximum a posteriori estimates of parameters in statistical models.
  - **E-step** assigns objects to clusters according to the current fuzzy clustering or parameters of probabilistic clusters
  - **M-step** finds the new clustering or parameters that maximize the sum of squared error (SSE) or the expected likelihood

# Fuzzy Clustering Using the EM Algorithm



Iteration	E-step	M-step
1	$M^T = \begin{bmatrix} 1 & 0 & 0.48 & 0.42 & 0.41 & 0.47 \\ 0 & 1 & 0.52 & 0.58 & 0.59 & 0.53 \end{bmatrix}$	$c_1 = (8.47, 5.12),$ $c_2 = (10.42, 8.99)$
2	$M^T = \begin{bmatrix} 0.73 & 0.49 & 0.91 & 0.26 & 0.33 & 0.42 \\ 0.27 & 0.51 & 0.09 & 0.74 & 0.67 & 0.58 \end{bmatrix}$	$c_1 = (8.51, 6.11),$ $c_2 = (14.42, 8.69)$
3	$M^T = \begin{bmatrix} 0.80 & 0.76 & 0.99 & 0.02 & 0.14 & 0.23 \\ 0.20 & 0.24 & 0.01 & 0.98 & 0.86 & 0.77 \end{bmatrix}$	$c_1 = (6.40, 6.24),$ $c_2 = (16.55, 8.64)$

- Initially, let  $c_1 = a$  and  $c_2 = b$

- 1<sup>st</sup> E-step: assign  $o$  to  $c_1$ , w. wt =  $\frac{\frac{1}{\text{dist}(o, c_1)^2}}{\frac{1}{\text{dist}(o, c_1)^2} + \frac{1}{\text{dist}(o, c_2)^2}} = \frac{\text{dist}(o, c_2)^2}{\text{dist}(o, c_1)^2 + \text{dist}(o, c_2)^2}$ 
  - $w_{c, c_1} = \frac{41}{45+41} = 0.48$

- 1<sup>st</sup> M-step: recalculate the centroids according to the partition matrix, minimizing the sum of squared error (SSE)

$$c_j = \frac{\sum_{\text{each point } o} w_{o, c_j}^2 o}{\sum_{\text{each point } o} w_{o, c_j}^2} \quad c_1 = \left( \frac{1^2 \times 3 + 0^2 \times 4 + 0.48^2 \times 9 + 0.42^2 \times 14 + 0.41^2 \times 18 + 0.47^2 \times 21}{1^2 + 0^2 + 0.48^2 + 0.42^2 + 0.41^2 + 0.47^2}, \frac{1^2 \times 3 + 0^2 \times 10 + 0.48^2 \times 6 + 0.42^2 \times 8 + 0.41^2 \times 11 + 0.47^2 \times 7}{1^2 + 0^2 + 0.48^2 + 0.42^2 + 0.41^2 + 0.47^2} \right) = (8.47, 5.12)$$

- Iteratively calculate this until the cluster centers converge or the change is small enough

# Univariate Gaussian Mixture Model

- $O = \{o_1, \dots, o_n\}$  ( $n$  observed objects),  $\Theta = \{\theta_1, \dots, \theta_k\}$  (parameters of the  $k$  distributions), and  $P_j(o_i | \theta_j)$  is the probability that  $o_i$  is generated from the  $j$ -th distribution using parameter  $\theta_j$ , we have

$$P(o_i | \Theta) = \sum_{j=1}^k \omega_j P_j(o_i | \theta_j) \quad P(O | \Theta) = \prod_{i=1}^n \sum_{j=1}^k \omega_j P_j(o_i | \theta_j)$$

- Univariate Gaussian mixture model
  - Assume the probability density function of each cluster follows a 1-d Gaussian distribution. Suppose that there are  $k$  clusters.
  - The probability density function of each cluster are centered at  $\mu_j$  with standard deviation  $\sigma_j$ ,  $\theta_j = (\mu_j, \sigma_j)$ , we have

$$P(o_i | \theta_j) = \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{(o_i - \mu_j)^2}{2\sigma_j^2}} \quad P(o_i | \Theta) = \sum_{j=1}^k \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{(o_i - \mu_j)^2}{2\sigma_j^2}}$$
$$P(O | \Theta) = \prod_{i=1}^n \sum_{j=1}^k \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{(o_i - \mu_j)^2}{2\sigma_j^2}}$$

# Computing Mixture Models with EM

- Given  $n$  objects  $\mathbf{O} = \{o_1, \dots, o_n\}$ , we want to mine a set of parameters  $\Theta = \{\theta_1, \dots, \theta_k\}$  s.t.,  $P(\mathbf{O}|\Theta)$  is maximized, where  $\theta_j = (\mu_j, \sigma_j)$  are the mean and standard deviation of the  $j$ -th univariate Gaussian distribution
- We initially assign random values to parameters  $\theta_j$ , then iteratively conduct the E- and M- steps until converge or sufficiently small change
- At the E-step, for each object  $o_i$ , calculate the probability that  $o_i$  belongs to each distribution,

$$P(\theta_j|o_i, \Theta) = \frac{P(o_i|\theta_j)}{\sum_{l=1}^k P(o_i|\theta_l)}$$

- At the M-step, adjust the parameters  $\theta_j = (\mu_j, \sigma_j)$  so that the expected likelihood  $P(\mathbf{O}|\Theta)$  is maximized

$$\mu_j = \sum_{i=1}^n o_i \frac{P(\theta_j|o_i, \Theta)}{\sum_{l=1}^k P(\theta_j|o_l, \Theta)} = \frac{\sum_{i=1}^n o_i P(\theta_j|o_i, \Theta)}{\sum_{i=1}^n P(\theta_j|o_i, \Theta)} \quad \sigma_j = \sqrt{\frac{\sum_{i=1}^n P(\theta_j|o_i, \Theta)(o_i - \mu_j)^2}{\sum_{i=1}^n P(\theta_j|o_i, \Theta)}}$$

# Advantages and Disadvantages of Mixture Models

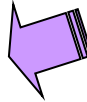
---

- Strength
  - Mixture models are more general than partitioning and fuzzy clustering
  - Clusters can be characterized by a small number of parameters
  - The results may satisfy the statistical assumptions of the generative models
- Weakness
  - Converge to local optimal (overcome: run multi-times w. random initialization)
  - Computationally expensive if the number of distributions is large, or the data set contains very few observed data points
  - Need large data sets
  - Hard to estimate the number of clusters



# Chapter 11. Cluster Analysis: Advanced Methods

---

- Probability Model-Based Clustering
- Clustering High-Dimensional Data 
- Clustering Graphs and Network Data
- Clustering with Constraints
- Summary

# Clustering High-Dimensional Data

---

- Clustering high-dimensional data (How high is high-D in clustering?)
  - Many applications: text documents, DNA micro-array data
  - Major challenges:
    - Many irrelevant dimensions may mask clusters
    - Distance measure becomes meaningless—due to equi-distance
    - Clusters may exist only in some subspaces
- Methods
  - **Subspace-clustering:** Search for clusters existing in subspaces of the given high dimensional data space
    - CLIQUE, ProClus, and bi-clustering approaches
  - **Dimensionality reduction approaches:** Construct a much lower dimensional space and search for clusters there (may construct new dimensions by combining some dimensions in the original data)
    - Dimensionality reduction methods and spectral clustering

# Traditional Distance Measures May Not Be Effective on High-D Data

- Traditional distance measure could be dominated by noises in many dimensions
- Ex. Which pairs of customers are more similar?

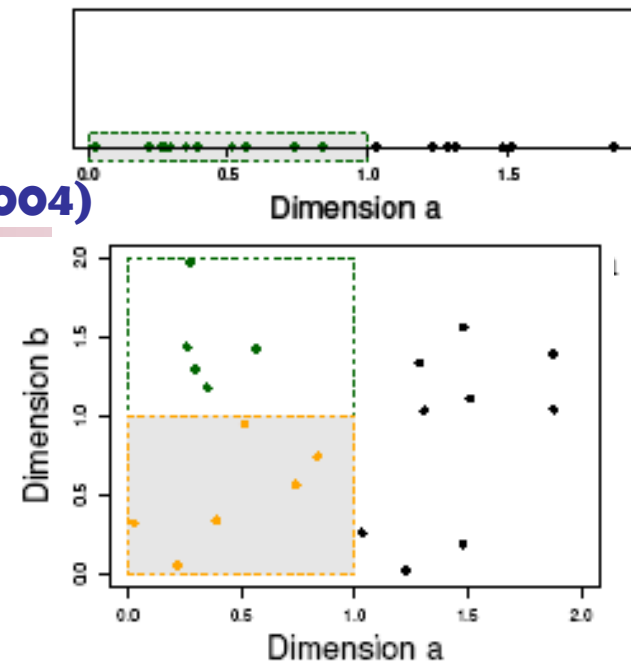
Customer	$P_1$	$P_2$	$P_3$	$P_4$	$P_5$	$P_6$	$P_7$	$P_8$	$P_9$	$P_{10}$
Ada	1	0	0	0	0	0	0	0	0	0
Bob	0	0	0	0	0	0	0	0	0	1
Cathy	1	0	0	0	1	0	0	0	0	1

- By Euclidean distance, we get,  
$$\text{dist}(\text{Ada}, \text{Bob}) = \text{dist}(\text{Bob}, \text{Cathy}) = \text{dist}(\text{Ada}, \text{Cathy}) = \sqrt{2}$$
  - despite Ada and Cathy look more similar
- Clustering should not only consider dimensions but also attributes (features)
  - Feature transformation: effective if most dimensions are relevant (PCA & SVD useful when features are highly correlated/redundant)
  - Feature selection: useful to find a subspace where the data have nice clusters

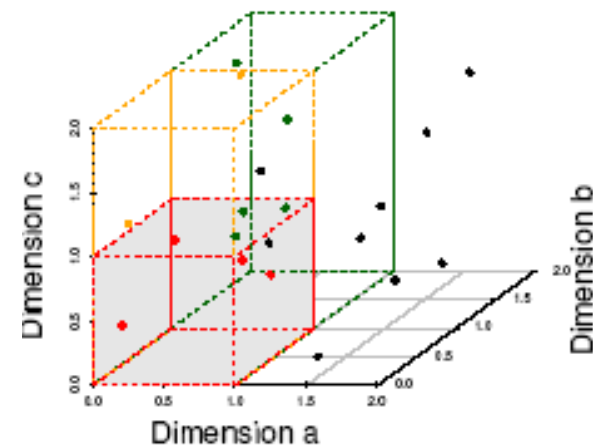
# The Curse of Dimensionality

(graphs adapted from Parsons et al. KDD Explorations 2004)

- Data in only one dimension is relatively packed
- Adding a dimension “stretch” the points across that dimension, making them further apart
- Adding more dimensions will make the points further apart—high dimensional data is extremely sparse
- Distance measure becomes meaningless—due to equi-distance



(b) 6 Objects in One Unit Bin

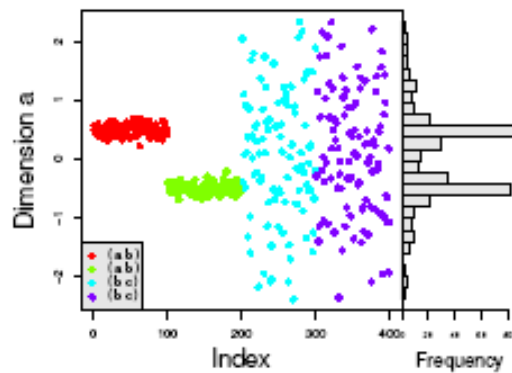
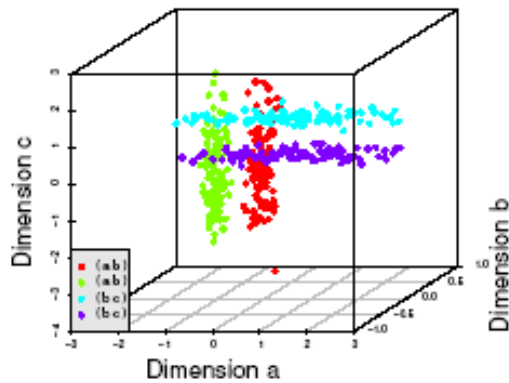


(c) 4 Objects in One Unit Bin

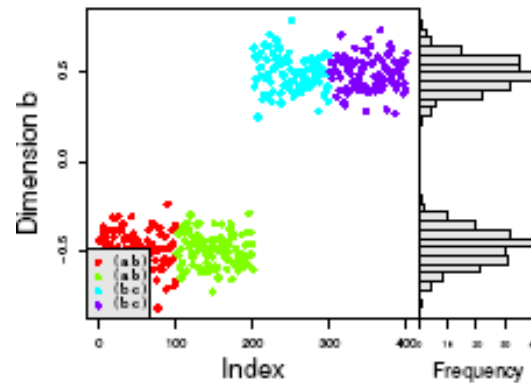
# Why Subspace Clustering?

(adapted from Parsons et al. SIGKDD Explorations 2004)

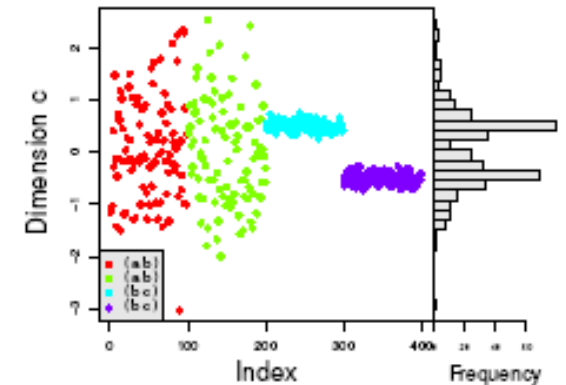
- Clusters may exist only in some subspaces
- Subspace-clustering: find clusters in all the subspaces



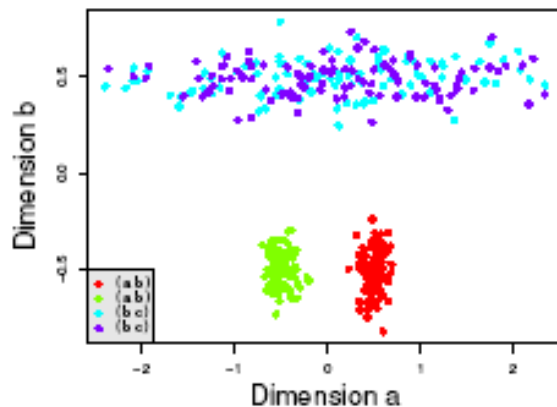
(a) Dimension *a*



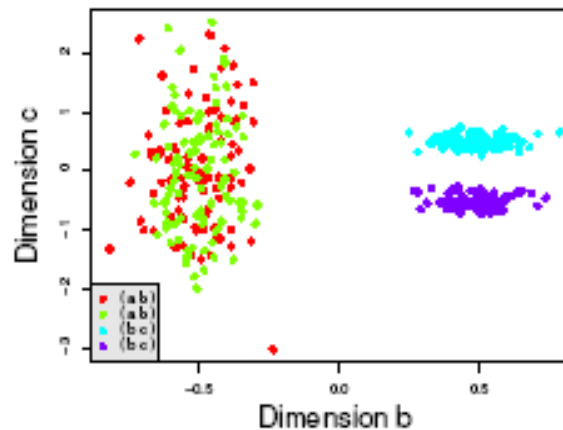
(b) Dimension *b*



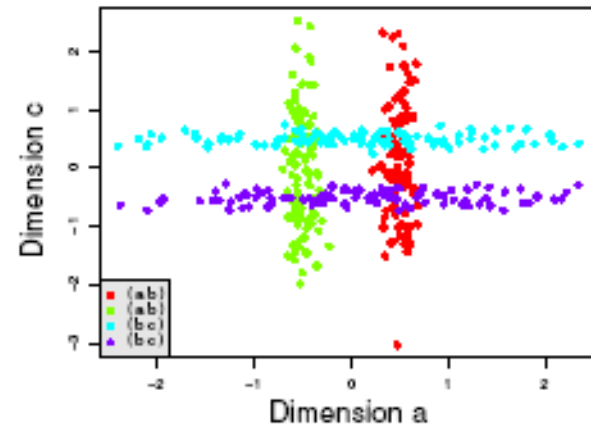
(c) Dimension *c*



(a) Dims *a* & *b*



(b) Dims *b* & *c*



(c) Dims *a* & *c*

# Subspace Clustering Methods

---

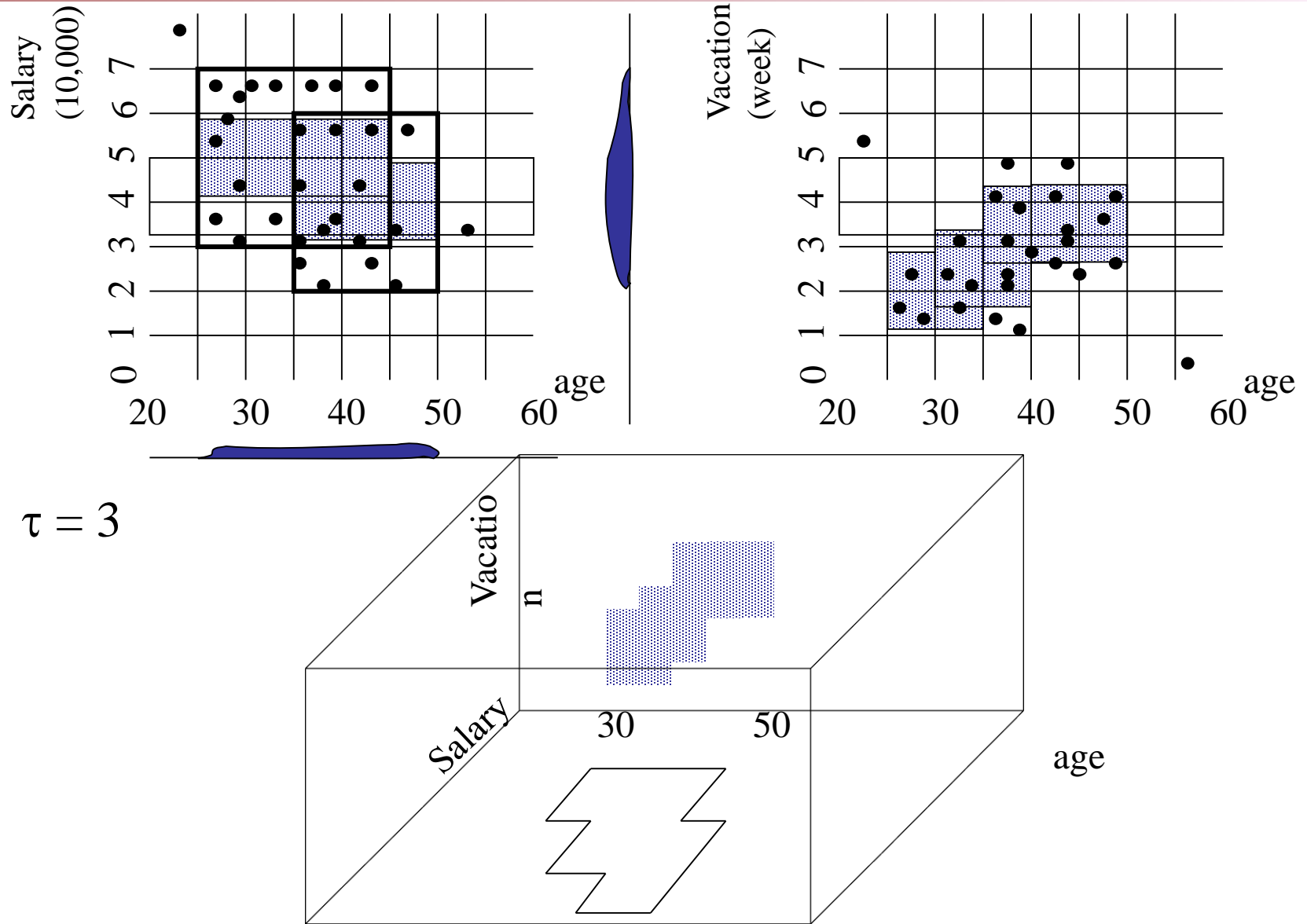
- Subspace search methods: Search various subspaces to find clusters
  - Bottom-up approaches
  - Top-down approaches
- Correlation-based clustering methods
  - E.g., PCA based approaches
- Bi-clustering methods
  - Optimization-based methods
  - Enumeration methods

# Subspace Clustering Method (I): Subspace Search Methods

---

- Search various subspaces to find clusters
- *Bottom-up approaches*
  - Start from low-D subspaces and search higher-D subspaces only when there may be clusters in such subspaces
  - Various pruning techniques to reduce the number of higher-D subspaces to be searched
  - Ex. CLIQUE (Agrawal et al. 1998)
- *Top-down approaches*
  - Start from full space and search smaller subspaces recursively
  - Effective only if the *locality assumption* holds: restricts that the subspace of a cluster can be determined by the local neighborhood
  - Ex. PROCLUS (Aggarwal et al. 1999): a  $k$ -medoid-like method

# CLIQUE: SubSpace Clustering with Aprori Pruning





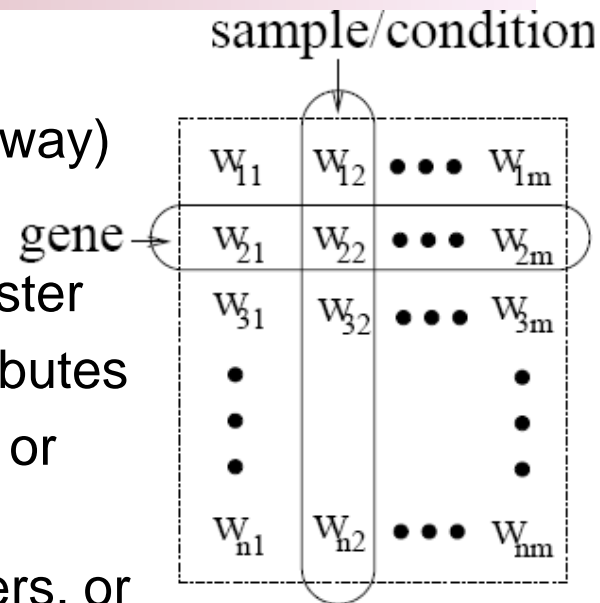
# Subspace Clustering Method (II): Correlation-Based Methods

---

- Subspace search method: similarity based on distance or density
- Correlation-based method: based on advanced correlation models
- Ex. PCA-based approach:
  - Apply PCA (for Principal Component Analysis) to derive a set of new, uncorrelated dimensions,
  - then mine clusters in the new space or its subspaces
- Other space transformations:
  - Hough transform
  - Fractal dimensions

# Subspace Clustering Method (III): Bi-Clustering Methods

- Bi-clustering: Cluster both objects and attributes simultaneously (treat objs and attrs in symmetric way)
- Four requirements:
  - Only a small set of objects participate in a cluster
  - A cluster only involves a small number of attributes
  - An object may participate in multiple clusters, or does not participate in any cluster at all
  - An attribute may be involved in multiple clusters, or is not involved in any cluster at all



- Ex 1. *Gene expression or microarray data*: a gene sample/condition matrix.

- Each element in the matrix, a real number, records the expression level of a gene under a specific condition

customers

products

$w_{11}$	$w_{12}$	$\dots$	$w_{1m}$
$w_{21}$	$w_{22}$	$\dots$	$w_{2m}$
$\dots$	$\dots$	$\dots$	$\dots$
$w_{n1}$	$w_{n2}$	$\dots$	$w_{nm}$

- Ex. 2. Clustering customers and products

- Another bi-clustering problem

# Types of Bi-clusters

- Let  $A = \{a_1, \dots, a_n\}$  be a set of genes,  $B = \{b_1, \dots, b_n\}$  a set of conditions
- A bi-cluster: A submatrix where genes and conditions follow some consistent patterns

- 4 types of bi-clusters (ideal cases)

- Bi-clusters with constant values:

- for any  $i$  in  $I$  and  $j$  in  $J$ ,  $e_{ij} = c$

- Bi-clusters with constant values on rows:

- $e_{ij} = c + \alpha_i$

- Also, it can be constant values on columns

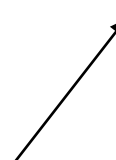
- Bi-clusters with *coherent values* (aka. *pattern-based clusters*)

- $e_{ij} = c + \alpha_i + \beta_j$

- Bi-clusters with *coherent* evolutions on rows


- $e_{ij}(e_{i1j1} - e_{i1j2})(e_{i2j1} - e_{i2j2}) \geq 0$

- i.e., only interested in the up- or down- regulated changes across genes or conditions without constraining on the exact values



10	10	10	10	10
20	20	20	20	20
50	50	50	50	50
0	0	0	0	0

10	50	30	70	20
20	60	40	80	30
50	90	70	110	60
0	40	20	60	10



10	50	30	70	20
20	100	50	1000	30
50	100	90	120	80
0	80	20	100	10

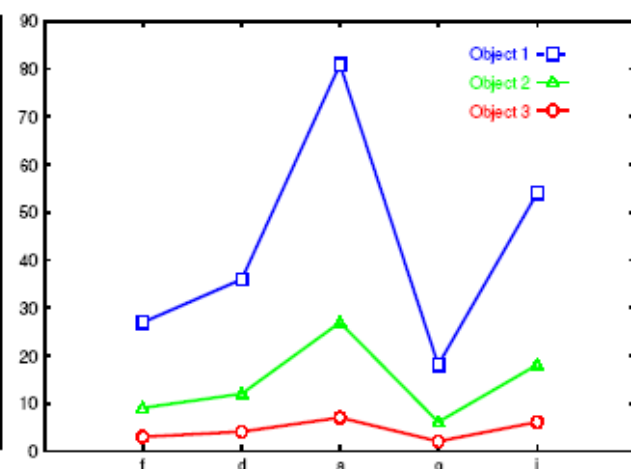
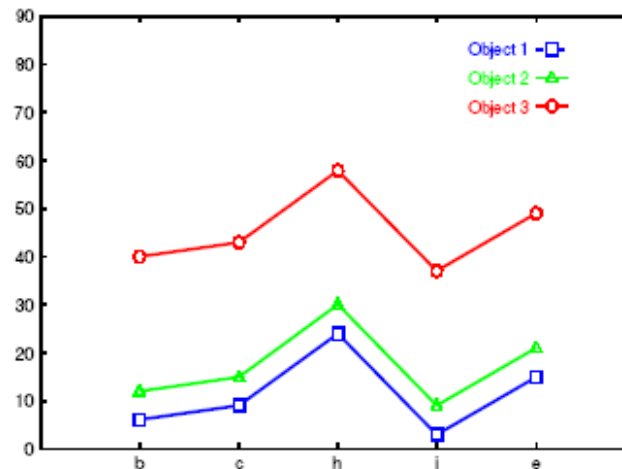
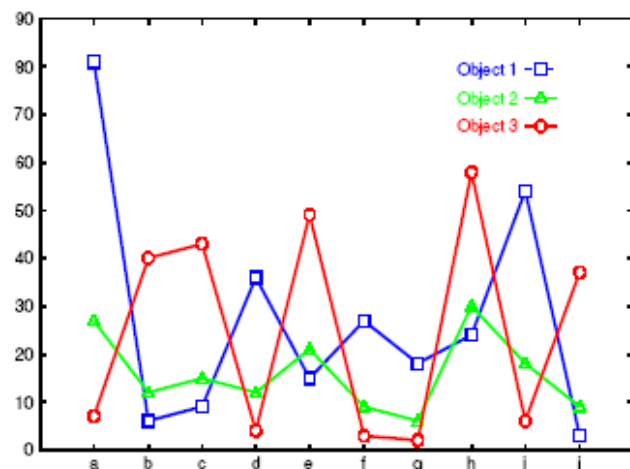
# Bi-Clustering Methods

---

- Real-world data is noisy: Try to find approximate bi-clusters
- Methods: Optimization-based methods vs. enumeration methods
- Optimization-based methods
  - Try to find a submatrix at a time that achieves the best significance as a bi-cluster
  - Due to the cost in computation, greedy search is employed to find local optimal bi-clusters
  - Ex.  $\delta$ -Cluster Algorithm (Cheng and Church, ISMB'2000)
- Enumeration methods
  - Use a tolerance threshold to specify the degree of noise allowed in the bi-clusters to be mined
  - Then try to enumerate all submatrices as bi-clusters that satisfy the requirements
  - Ex.  $\delta$ -pCluster Algorithm (H. Wang et al.' SIGMOD'2002, MaPle: Pei et al., ICDM'2003)

# Bi-Clustering for Micro-Array Data Analysis

- Left figure: Micro-array “raw” data shows 3 genes and their values in a multi-D space: Difficult to find their patterns
- Right two: Some subsets of dimensions form nice **shift** and **scaling** patterns
- No globally defined similarity/distance measure
- Clusters may not be exclusive
  - An object can appear in multiple clusters



# Bi-Clustering (I): $\delta$ -Bi-Cluster

- For a submatrix  $I \times J$ , the mean of the  $i$ -th row:  $e_{iJ} = \frac{1}{|J|} \sum_{j \in J} e_{ij}$ 
  - The mean of the  $j$ -th column:  $e_{Ij} = \frac{1}{|I|} \sum_{i \in I} e_{ij}$
  - The mean of all elements in the submatrix is

$$e_{IJ} = \frac{1}{|I||J|} \sum_{i \in I, j \in J} e_{ij} = \frac{1}{|I|} \sum_{i \in I} e_{iJ} = \frac{1}{|J|} \sum_{j \in J} e_{Ij}$$

- The quality of the submatrix as a bi-cluster can be measured by the *mean squared residue* value

$$H(I \times J) = \frac{1}{|I||J|} \sum_{i \in I, j \in J} (e_{ij} - e_{iJ} - e_{Ij} + e_{IJ})^2$$

- A submatrix  $I \times J$  is  **$\delta$ -bi-cluster** if  $H(I \times J) \leq \delta$  where  $\delta \geq 0$  is a threshold. When  $\delta = 0$ ,  $I \times J$  is a perfect bi-cluster with coherent values. By setting  $\delta > 0$ , a user can specify the tolerance of average noise per element against a perfect bi-cluster
  - $\text{residue}(e_{ij}) = e_{ij} - e_{iJ} - e_{Ij} + e_{IJ}$

# Bi-Clustering (I): The $\delta$ -Cluster Algorithm

- **Maximal  $\delta$ -bi-cluster** is a  $\delta$ -bi-cluster  $I \times J$  such that there does not exist another  $\delta$ -bi-cluster  $I' \times J'$  which contains  $I \times J$
- Computing is costly: Use heuristic greedy search to obtain local optimal clusters
- Two phase computation: deletion phase and additional phase
- Deletion phase: Start from the whole matrix, iteratively remove rows and columns while the mean squared residue of the matrix is over  $\delta$ 
  - At each iteration, for each row/column, compute the *mean squared residue*:
$$d(i) = \frac{1}{|J|} \sum_{j \in J} (e_{ij} - e_{iJ} - e_{Ij} + e_{IJ})^2 \quad d(j) = \frac{1}{|I|} \sum_{i \in I} (e_{ij} - e_{iJ} - e_{Ij} + e_{IJ})^2$$
  - Remove the row or column of the largest mean squared residue
- Addition phase:
  - Expand iteratively the  $\delta$ -bi-cluster  $I \times J$  obtained in the deletion phase as long as the  $\delta$ -bi-cluster requirement is maintained
  - Consider all the rows/columns not involved in the current bi-cluster  $I \times J$  by calculating their mean squared residues
  - A row/column of the smallest mean squared residue is added into the current  $\delta$ -bi-cluster
- It finds only one  $\delta$ -bi-cluster, thus needs to run multiple times: replacing the elements in the output bi-cluster by random numbers

# Bi-Clustering (II): $\delta$ -pCluster

- Enumerating all bi-clusters ( $\delta$ -pClusters) [H. Wang, et al., Clustering by pattern similarity in large data sets. SIGMOD'02]
- Since a submatrix  $I \times J$  is a bi-cluster with (perfect) coherent values iff  $e_{i_1j_1} - e_{i_2j_1} = e_{i_1j_2} - e_{i_2j_2}$ . For any  $2 \times 2$  submatrix of  $I \times J$ , define  $p$ -score

$$p\text{-score} \begin{pmatrix} e_{i_1j_1} & e_{i_1j_2} \\ e_{i_2j_1} & e_{i_2j_2} \end{pmatrix} = |(e_{i_1j_1} - e_{i_2j_1}) - (e_{i_1j_2} - e_{i_2j_2})|$$

- A submatrix  $I \times J$  is a  **$\delta$ -pCluster** (pattern-based cluster) if the  $p$ -score of every  $2 \times 2$  submatrix of  $I \times J$  is at most  $\delta$ , where  $\delta \geq 0$  is a threshold specifying a user's tolerance of noise against a perfect bi-cluster
- The  $p$ -score controls the noise on every element in a bi-cluster, while the mean squared residue captures the average noise
- **Monotonicity:** If  $I \times J$  is a  $\delta$ -pClusters, every  $x \times y$  ( $x, y \geq 2$ ) submatrix of  $I \times J$  is also a  $\delta$ -pClusters.
- A  $\delta$ -pCluster is **maximal** if no more row or column can be added into the cluster and retain  $\delta$ -pCluster: We only need to compute all maximal  $\delta$ -pClusters.

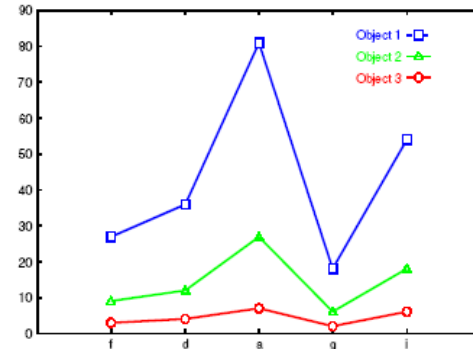
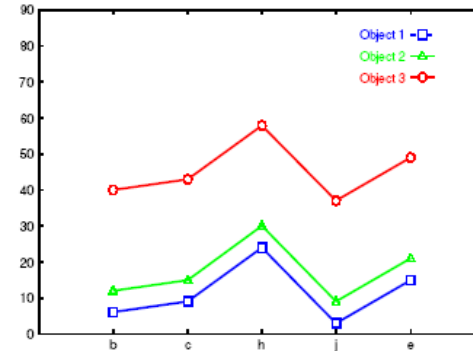
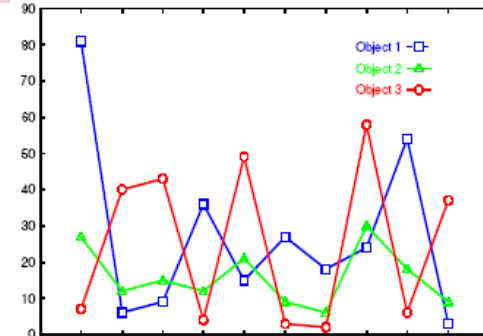


# MaPle: Efficient Enumeration of $\delta$ -pClusters

- Pei et al., MaPle: Efficient enumerating all maximal  $\delta$ -pClusters. ICDM'03
- Framework: Same as pattern-growth in frequent pattern mining (based on the downward closure property)
- For each condition combination J, find the maximal subsets of genes I such that I x J is a  $\delta$ -pClusters
  - If I x J is not a submatrix of another  $\delta$ -pClusters
  - then I x J is a maximal  $\delta$ -pCluster.
- Algorithm is very similar to mining frequent closed itemsets
- Additional advantages of  $\delta$ -pClusters:
  - Due to averaging of  $\delta$ -cluster, it may contain outliers but still within  $\delta$ -threshold
  - Computing bi-clusters for scaling patterns, take logarithmic on

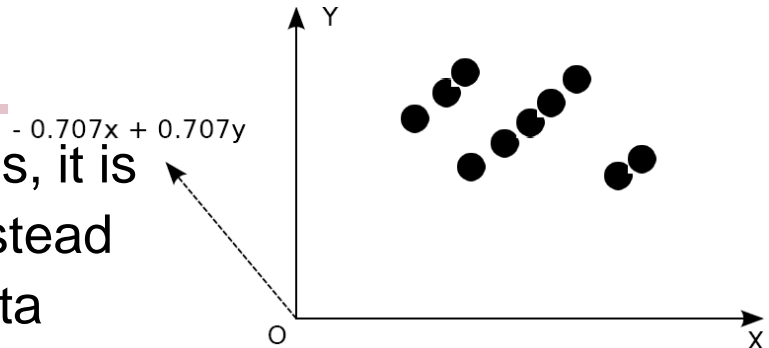
$$\frac{d_{xa} / d_{ya}}{d_{xb} / d_{yb}} < \delta$$

will lead to the p-score form



# Dimensionality-Reduction Methods

- Dimensionality reduction: In some situations, it is more effective to construct a new space instead of using some subspaces of the original data
- Ex. To cluster the points in the right figure, any subspace of the original one, X and Y, cannot help, since all the three clusters will be projected into the overlapping areas in X and Y axes.
  - Construct a new dimension as the dashed one, the three clusters become apparent when the points projected into the new dimension
- Dimensionality reduction methods
  - Feature selection and extraction: But may not focus on clustering structure finding
  - Spectral clustering: Combining feature extraction and clustering (i.e., use the *spectrum* of the similarity matrix of the data to perform dimensionality reduction for clustering in fewer dimensions)
    - Normalized Cuts (Shi and Malik, CVPR'97 or PAMI'2000)
    - The Ng-Jordan-Weiss algorithm (NIPS'01)

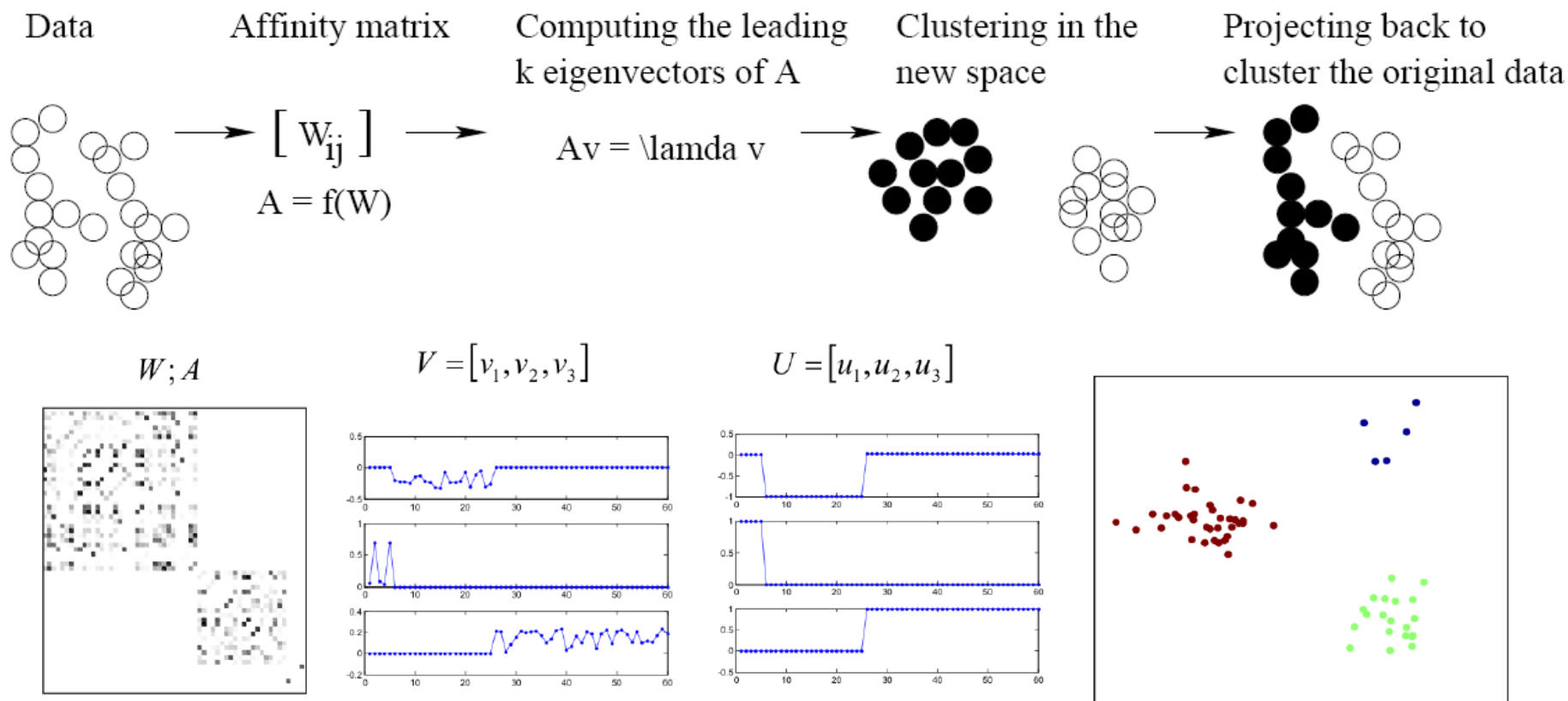


# Spectral Clustering:

## The Ng-Jordan-Weiss (NJW) Algorithm

- Given a set of objects  $o_1, \dots, o_n$ , and the distance between each pair of objects,  $\text{dist}(o_i, o_j)$ , find the desired number  $k$  of clusters
- Calculate an affinity matrix  $W$ , where  $\sigma$  is a scaling parameter that controls how fast the affinity  $W_{ij}$  decreases as  $\text{dist}(o_i, o_j)$  increases. In NJW, set  $W_{ij} = 0$ 
$$D_{ii} = \sum_{j=1}^n W_{ij}$$
- Derive a matrix  $A = f(W)$ . NJW defines a matrix  $D$  to be a diagonal matrix s.t.  $D_{ii}$  is the sum of the  $i$ -th row of  $W$ , i.e.,  $W_{ij} = e^{-\frac{\text{dist}(o_i, o_j)}{\sigma^2}}$   
Then,  $A$  is set to  $A = D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$
- A spectral clustering method finds the  $k$  leading eigenvectors of  $A$ 
  - A vector  $v$  is an eigenvector of matrix  $A$  if  $Av = \lambda v$ , where  $\lambda$  is the corresponding eigen-value
- Using the  $k$  leading eigenvectors, project the original data into the new space defined by the  $k$  leading eigenvectors, and run a clustering algorithm, such as  $k$ -means, to find  $k$  clusters
- Assign the original data points to clusters according to how the transformed points are assigned in the clusters obtained

# Spectral Clustering: Illustration and Comments

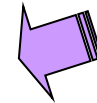


- Spectral clustering: Effective in tasks like image processing
- Scalability challenge: Computing eigenvectors on a large matrix is costly
- Can be combined with other clustering methods, such as bi-clustering

# Chapter 11. Cluster Analysis: Advanced Methods

---

- Probability Model-Based Clustering
- Clustering High-Dimensional Data
- Clustering Graphs and Network Data
- Clustering with Constraints
- Summary

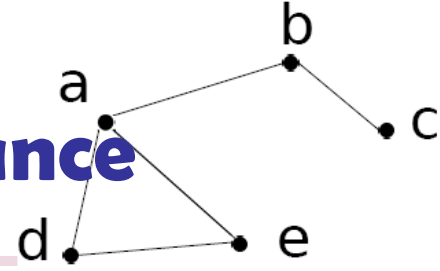


# Clustering Graphs and Network Data

---

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

# Similarity Measure (I): Geodesic Distance



- 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  $v$ ,  $\text{eccen}(v)$ : The largest geodesic distance between  $v$  and any other vertex  $u \in V - \{v\}$ .
  - E.g.,  $\text{eccen}(a) = \text{eccen}(b) = 2$ ;  $\text{eccen}(c) = \text{eccen}(d) = \text{eccen}(e) = 3$
- **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.,  $\text{radius}(g) = 2$
- **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.,  $\text{diameter}(g) = 3$
- A **peripheral vertex** is a vertex that achieves the diameter.
  - E.g., Vertices  $c$ ,  $d$ , and  $e$  are peripheral vertices

# SimRank: Similarity Based on Random Walk and Structural Context

- SimRank: structural-context similarity, i.e., based on the similarity of its neighbors
- In a directed graph  $G = (V, E)$ ,
  - *individual in-neighborhood* of  $v$ :  $I(v) = \{u \mid (u, v) \in E\}$
  - *individual out-neighborhood* of  $v$ :  $O(v) = \{w \mid (v, w) \in E\}$
- Similarity in SimRank: 
$$s(u, v) = \frac{C}{|I(u)||I(v)|} \sum_{x \in I(u)} \sum_{y \in I(v)} s(x, y)$$
  - Initialization:  $s_0(u, v) = \begin{cases} 0 & \text{if } u \neq v \\ 1 & \text{if } u = v \end{cases}$ 

$$P[t] = \begin{cases} \prod_{i=1}^{k-1} \frac{1}{|O(w_i)|} & \text{if } l(t) > 0 \\ 0 & \text{if } l(t) = 0. \end{cases}$$
  - Then we can compute  $s_{i+1}$  from  $s_i$  based on the definition
- Similarity based on random walk: in a strongly connected component
  - Expected distance:  $d(u, v) = \sum_{t: u \rightsquigarrow v} P[t]l(t)$   $P[t]$  is the probability of the tour
  - Expected meeting distance:  $m(u, v) = \sum_{t: (u, v) \rightsquigarrow (x, x)} P[t]l(t)$
  - Expected meeting probability:  $p(u, v) = \sum_{t: (u, v) \rightsquigarrow (x, x)} P[t]C^{l(t)}$



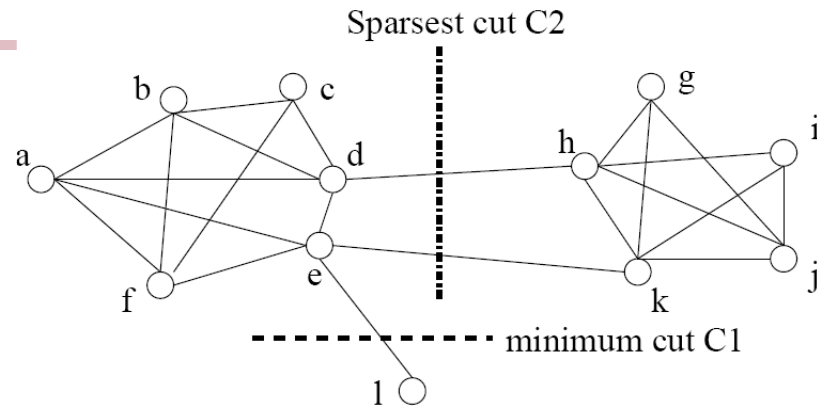
# Graph Clustering: Sparsest Cut

- $G = (V, E)$ . The *cut set* of a cut is the set of edges  $\{(u, v) \in E \mid u \in S, v \in T\}$  and  $S$  and  $T$  are in 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**:  $\Phi = \frac{\text{the size of the cut}}{\min\{|S|, |T|\}}$



- A cut is **sparsest** if its sparsity is not greater than that of any other cut

- Ex. Cut  $C_2 = (\{a, b, c, d, e, f, l\}, \{g, h, i, j, k\})$  is the sparsest cut

- For  $k$  clusters, the **modularity** of a clustering assesses the quality of the clustering:

$$Q = \sum_{i=1}^k \left( \frac{l_i}{|E|} - \left( \frac{d_i}{2|E|} \right)^2 \right) \quad \begin{array}{l} l_i: \text{\# edges between vertices in the } i\text{-th cluster} \\ d_i: \text{the sum of the degrees of the vertices in the } i\text{-th cluster} \end{array}$$

- 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

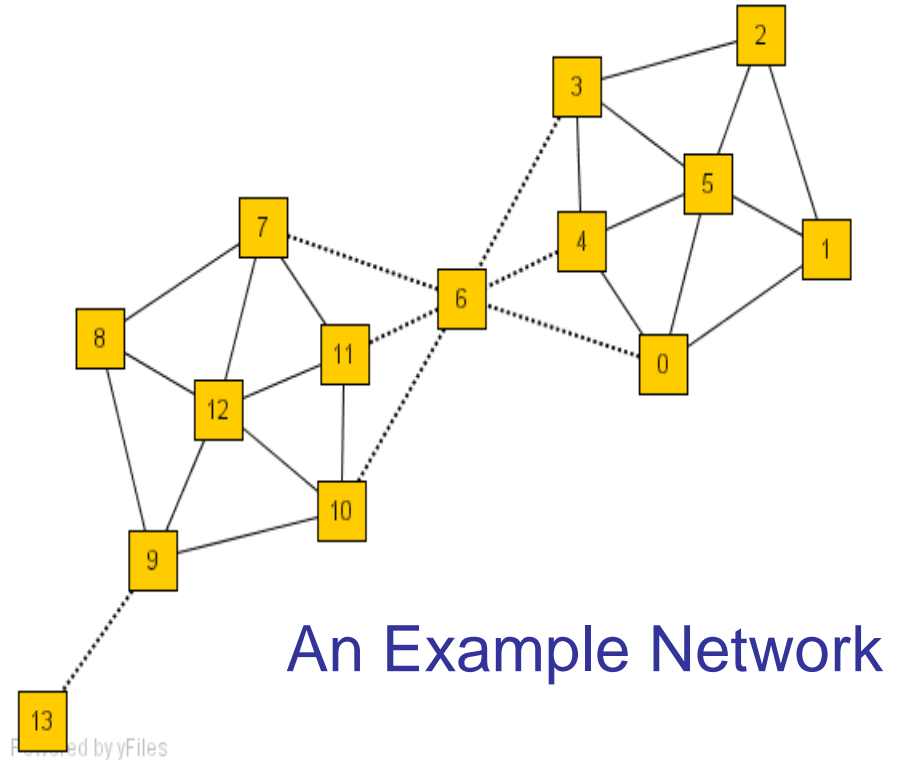
# Two Approaches for Graph Clustering

---

- Two approaches for clustering graph data
  - Use *generic clustering methods* for high-dimensional data
  - *Designed specifically for clustering graphs*
- Using 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
- Methods specific to 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

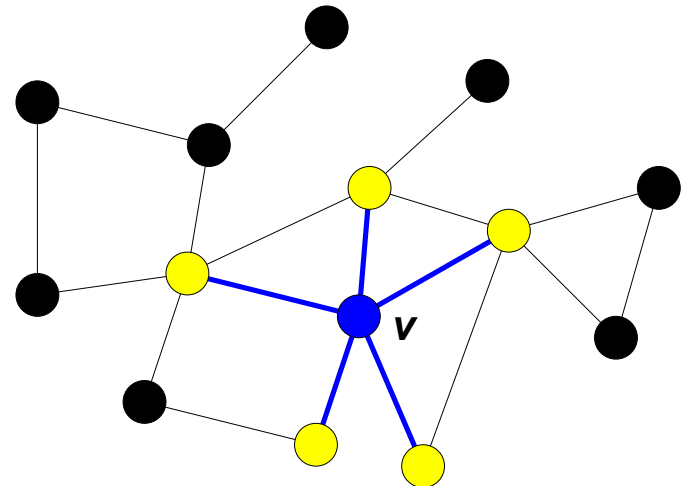
- How many clusters?
- What size should they be?
- What is the best partitioning?
- Should some points be segregated?



- Application: Given simply information of who associates with whom, could one identify clusters of individuals with common interests or special relationships (families, cliques, terrorist cells)?

# A Social Network Model

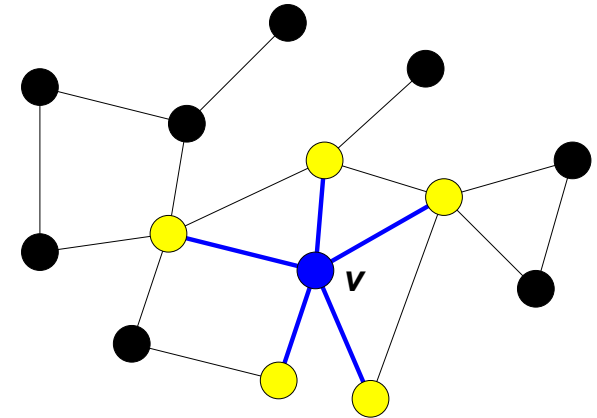
- Cliques, hubs and outliers
  - Individuals in a tight social group, or **clique**, know many of the same people, regardless of the size of the group
  - Individuals who are **hubs** know many people in different groups but belong to no single group. Politicians, for example bridge multiple groups
  - Individuals who are **outliers** reside at the margins of society. Hermits, for example, know few people and belong to no group
- The Neighborhood of a Vertex
  - Define  $\Gamma(v)$  as the immediate neighborhood of a vertex (i.e. the set of people that an individual knows )



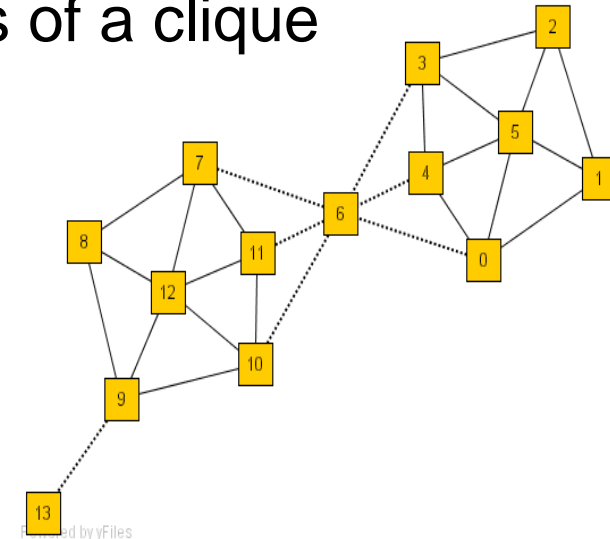
# Structure Similarity

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

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



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



# Structural Connectivity [1]

- $\varepsilon$ -Neighborhood:  $N_{\varepsilon}(v) = \{w \in \Gamma(v) \mid \sigma(v, w) \geq \varepsilon\}$

- Core:  $CORE_{\varepsilon, \mu}(v) \Leftrightarrow |N_{\varepsilon}(v)| \geq \mu$

- Direct structure reachable:

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

- Structure reachable: transitive closure of direct structure reachability

- Structure connected:

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

[1] M. Ester, H. P. Kriegel, J. Sander, & X. Xu (KDD'96) “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases

# Structure-Connected Clusters

- Structure-connected cluster C

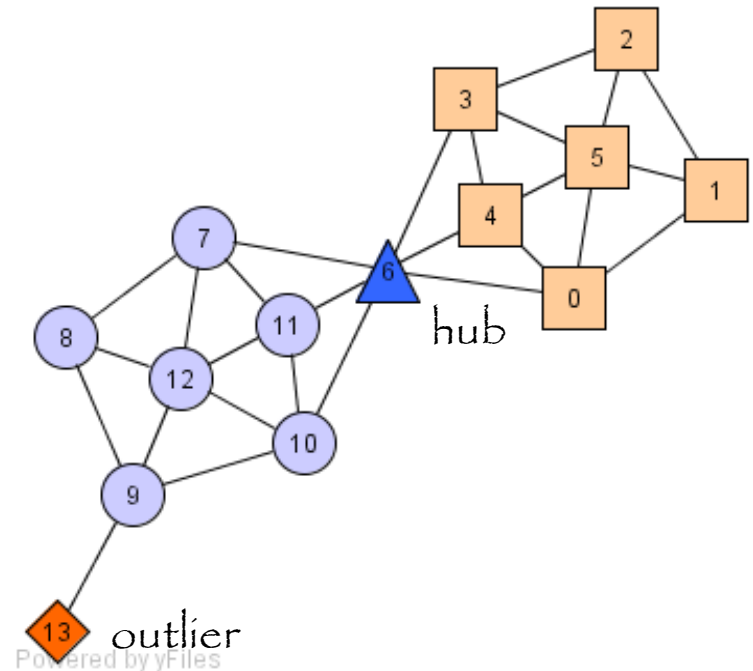
- Connectivity:  $\forall v, w \in C : CONNECT_{\varepsilon, \mu}(v, w)$
- Maximality:  $\forall v, w \in V : v \in C \wedge REACH_{\varepsilon, \mu}(v, w) \Rightarrow w \in C$

- Hubs:

- Not belong to any cluster
- Bridge to many clusters

- Outliers:

- Not belong to any cluster
- Connect to less clusters

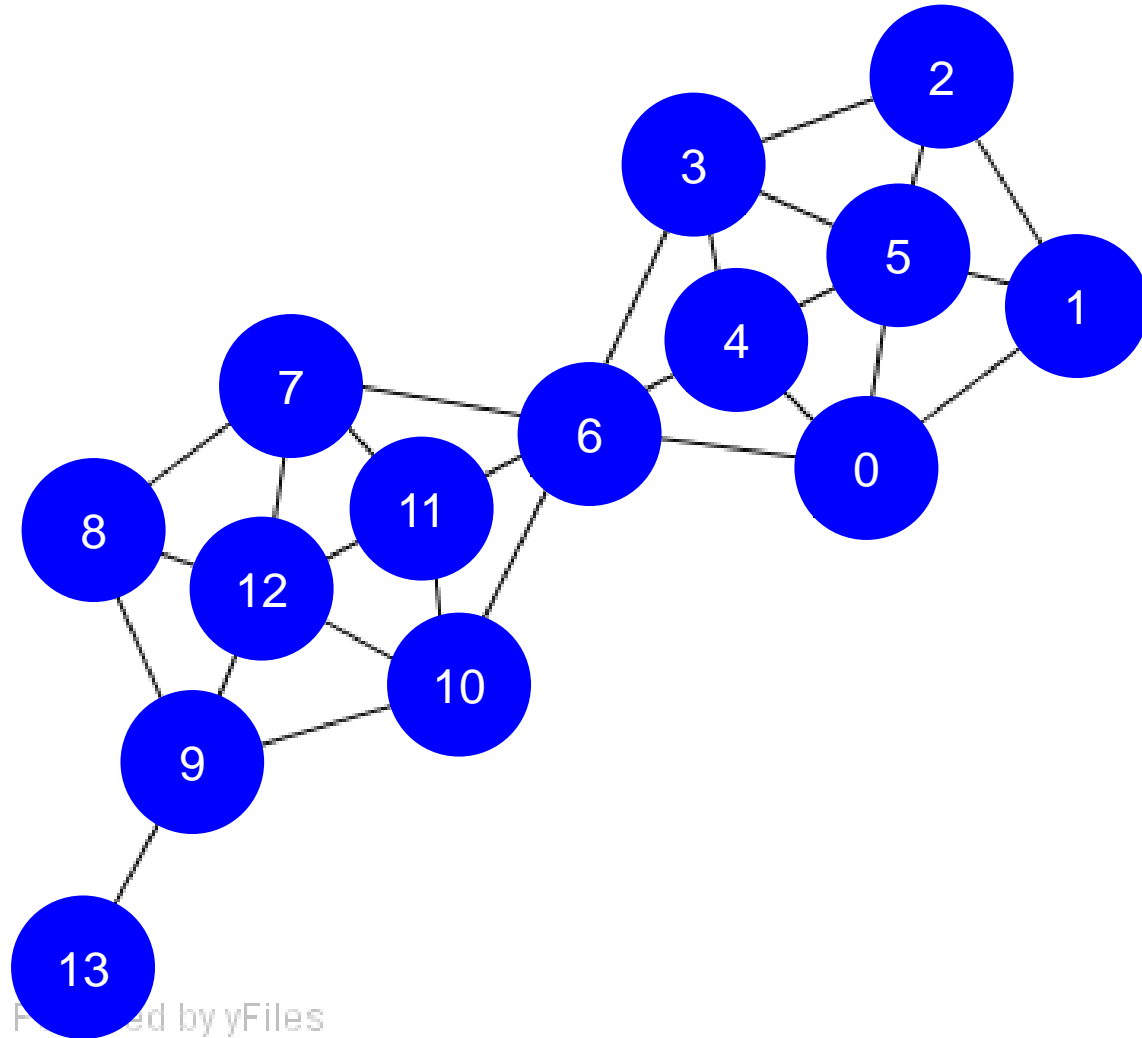




# Algorithm

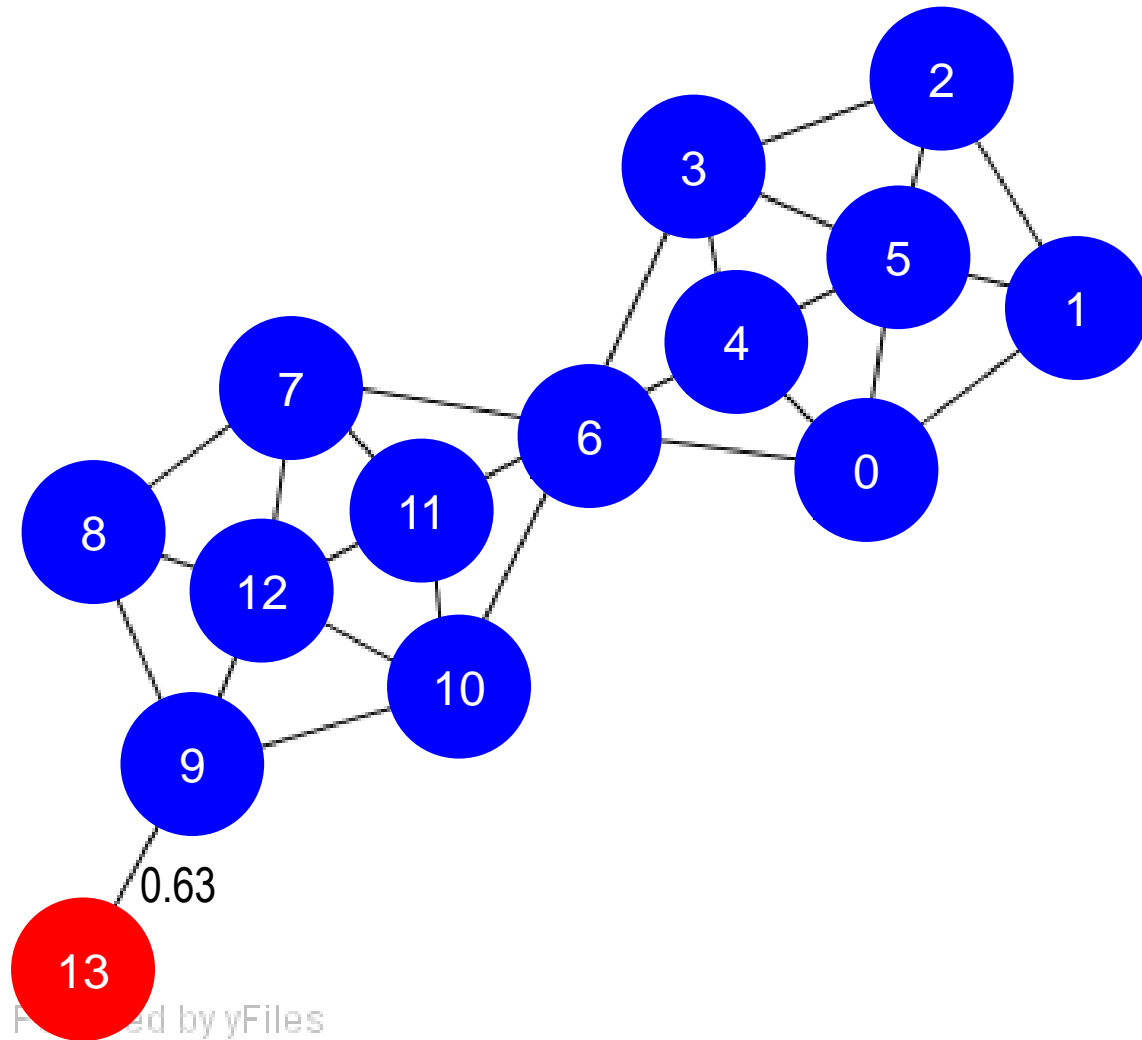
---

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



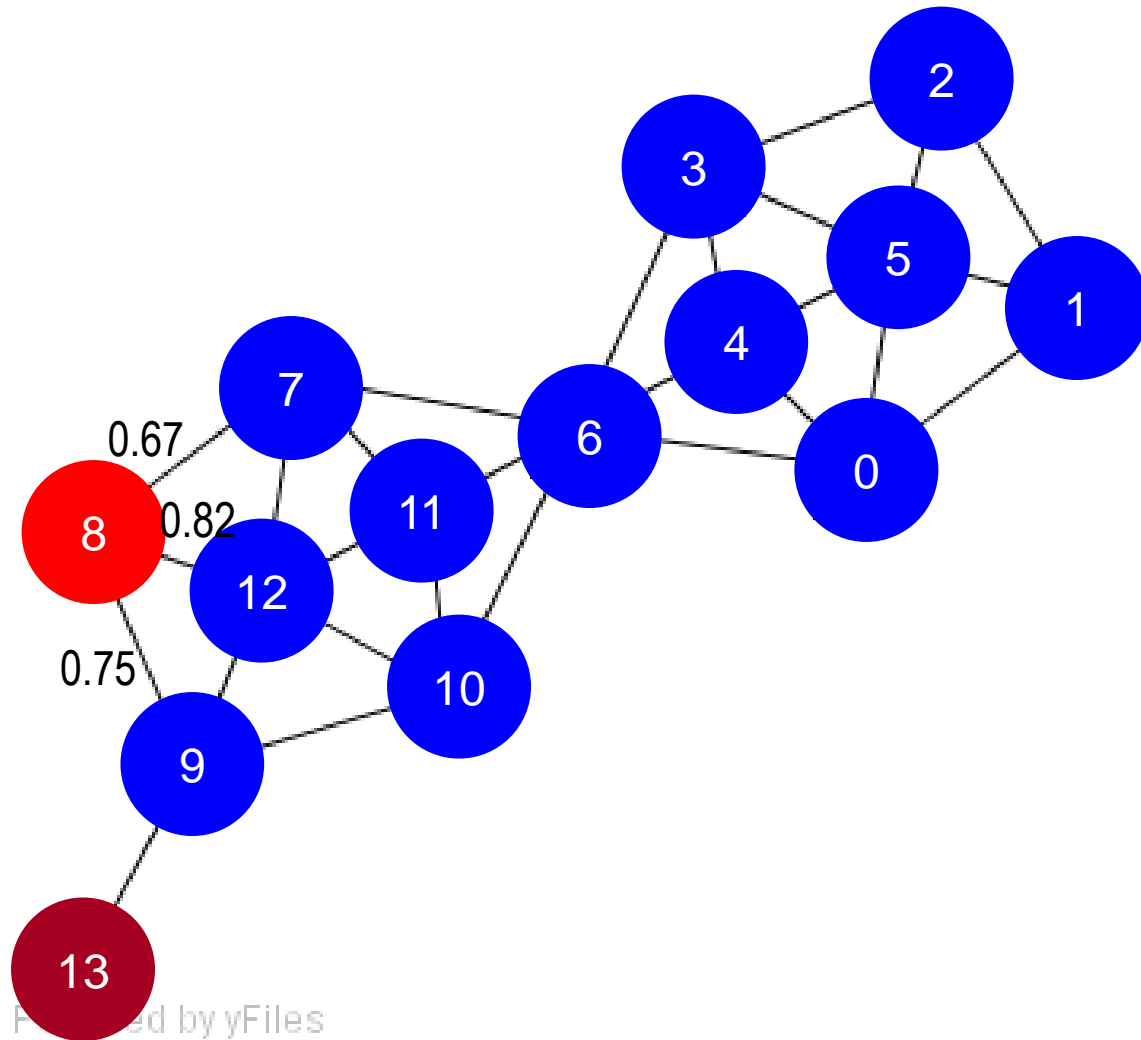
# Algorithm

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



# Algorithm

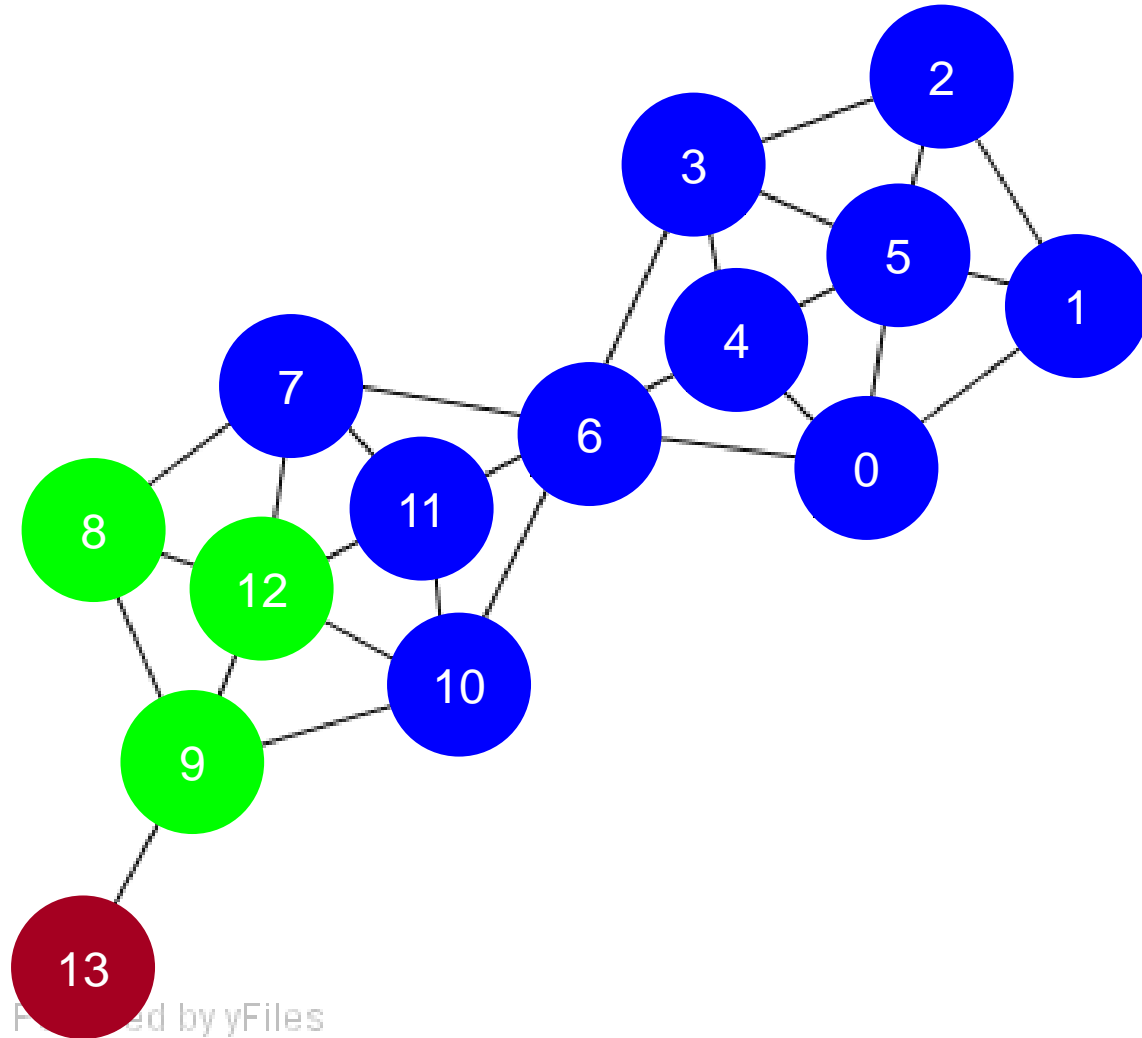
$$\mu = 2$$
$$\varepsilon = 0.7$$



Powered by yFiles

# Algorithm

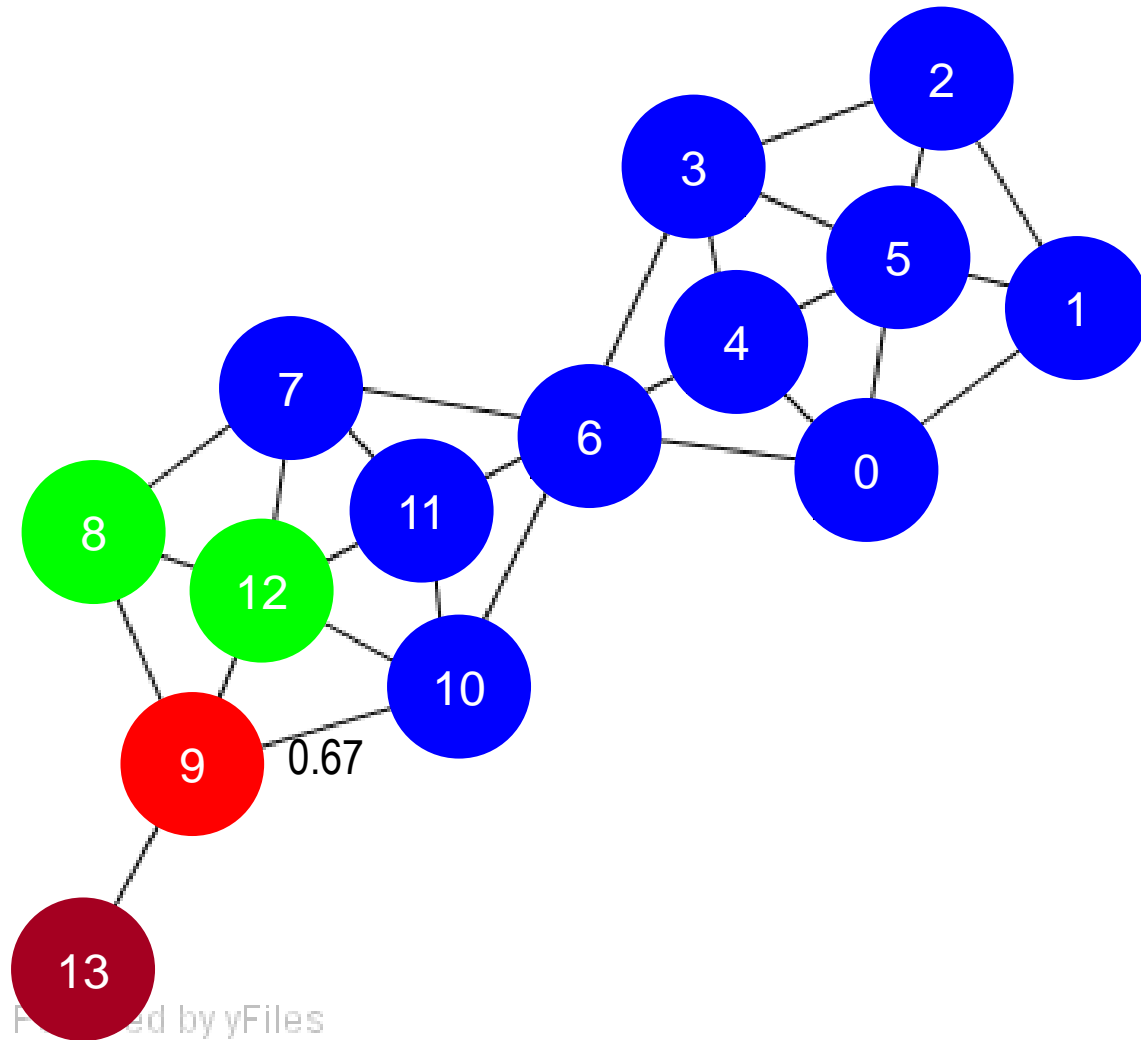
$$\mu = 2$$
$$\varepsilon = 0.7$$



Powered by yFiles

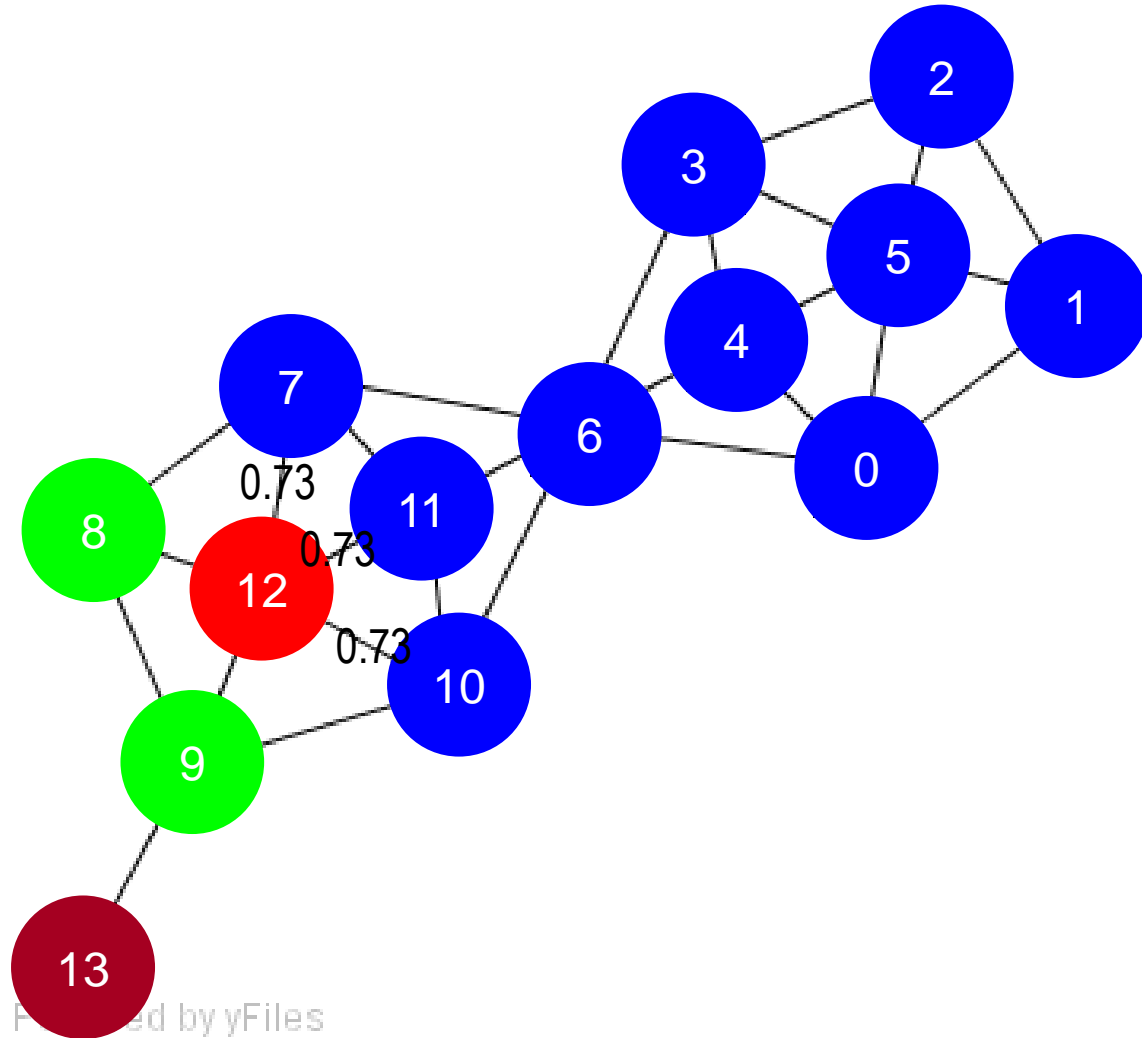
# Algorithm

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



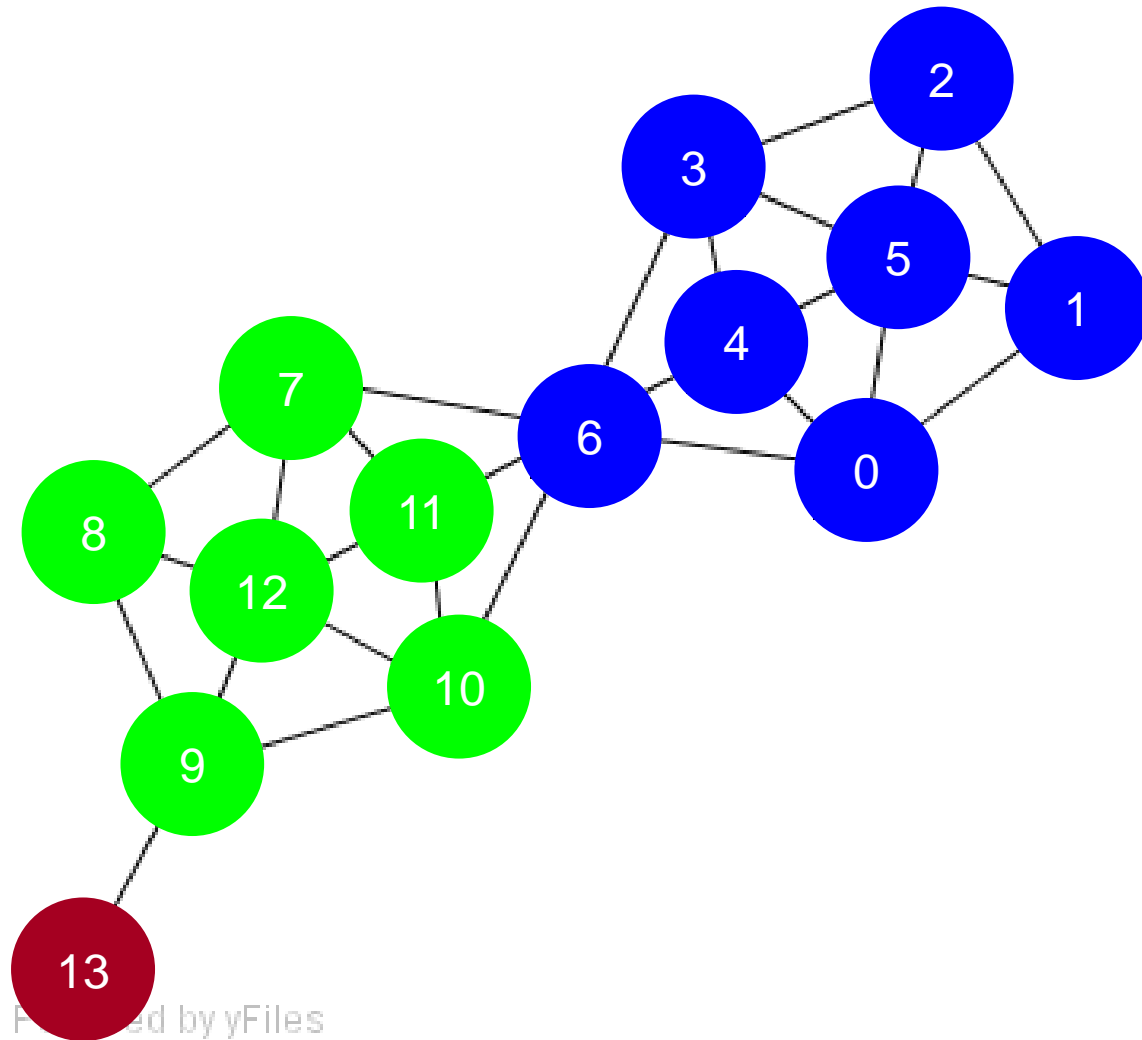
# Algorithm

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



# Algorithm

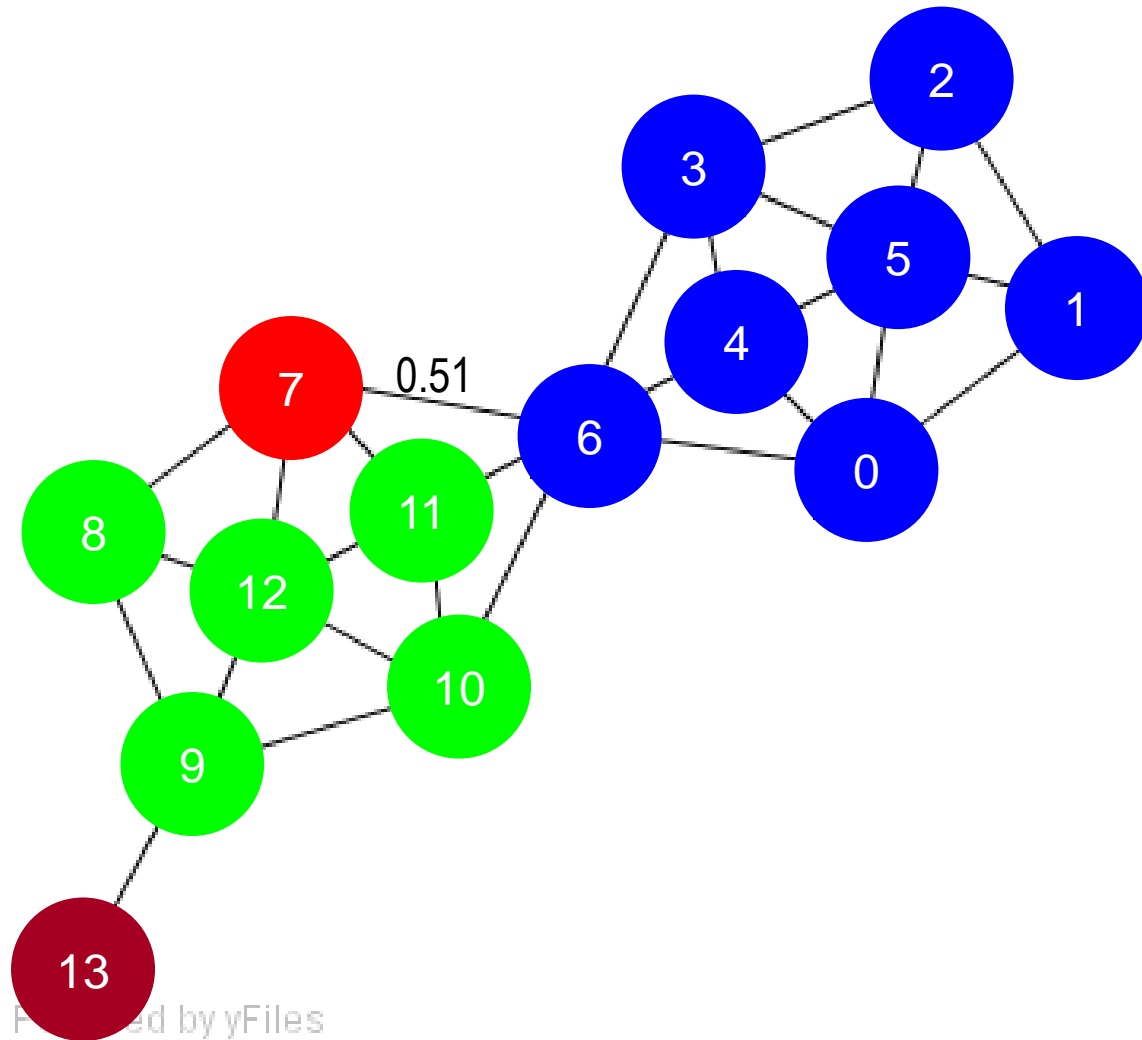
$$\mu = 2$$
$$\varepsilon = 0.7$$



Powered by yFiles

# Algorithm

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

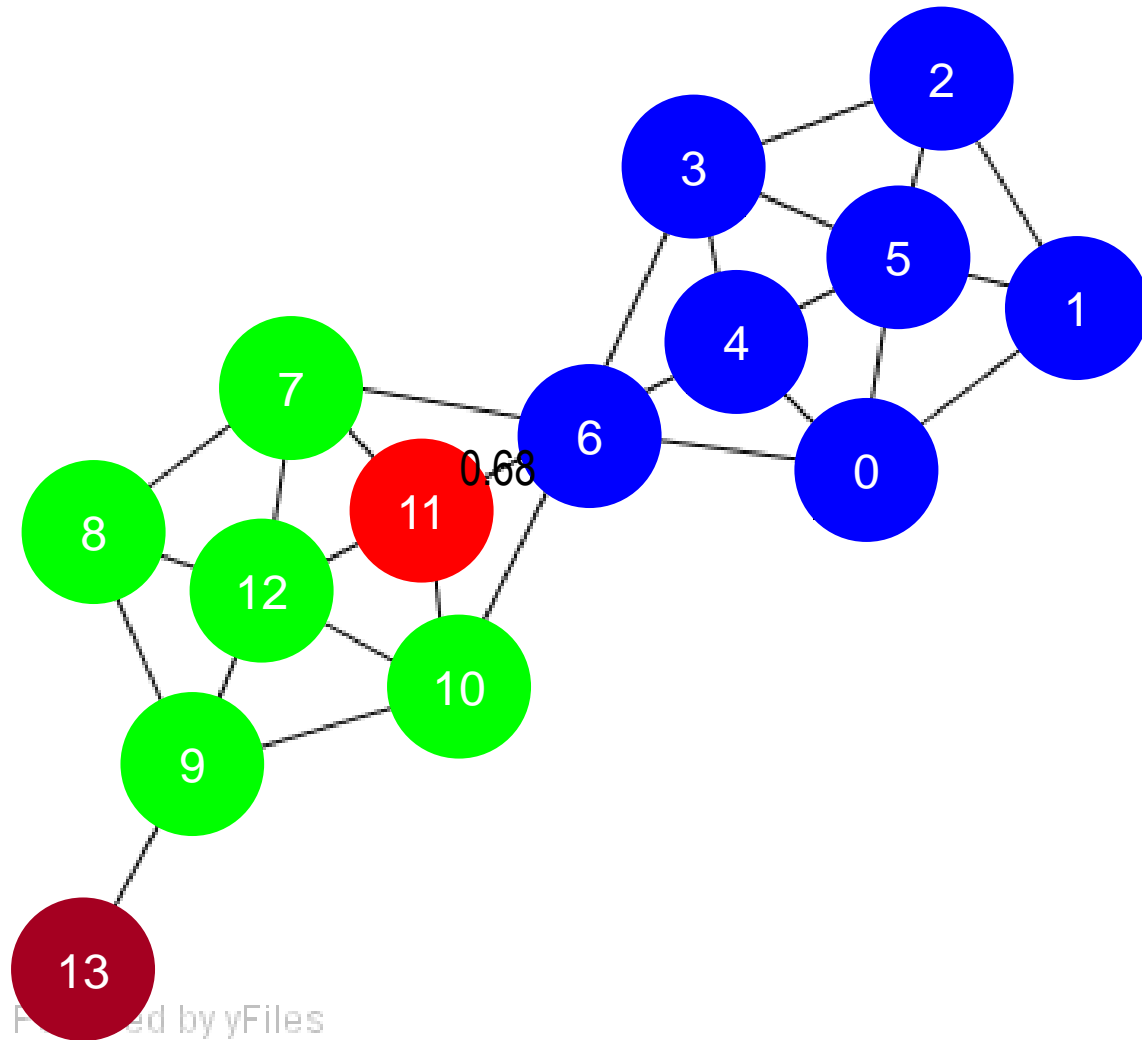


Powered by yFiles



# Algorithm

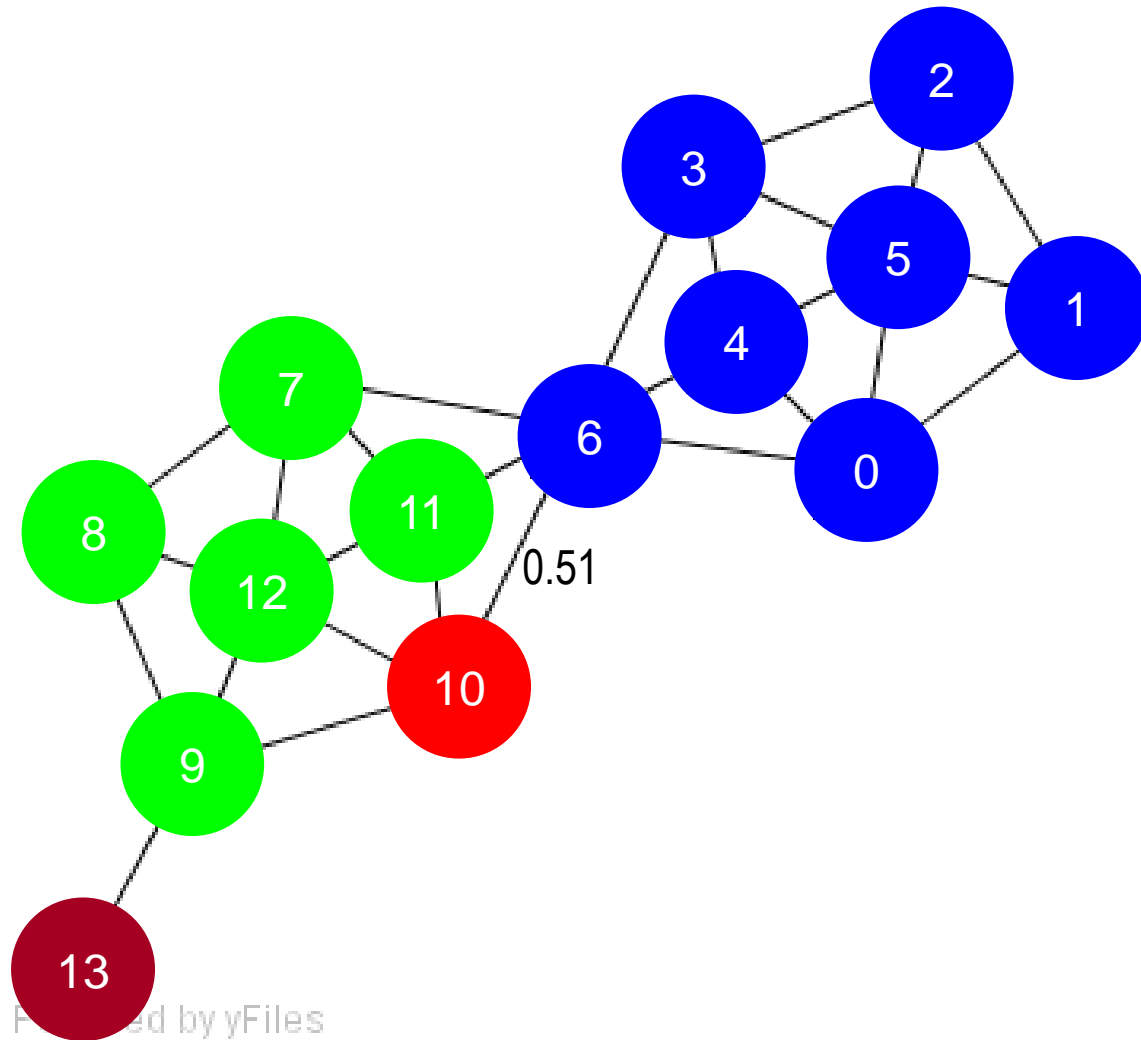
$$\mu = 2$$
$$\varepsilon = 0.7$$



Powered by yFiles

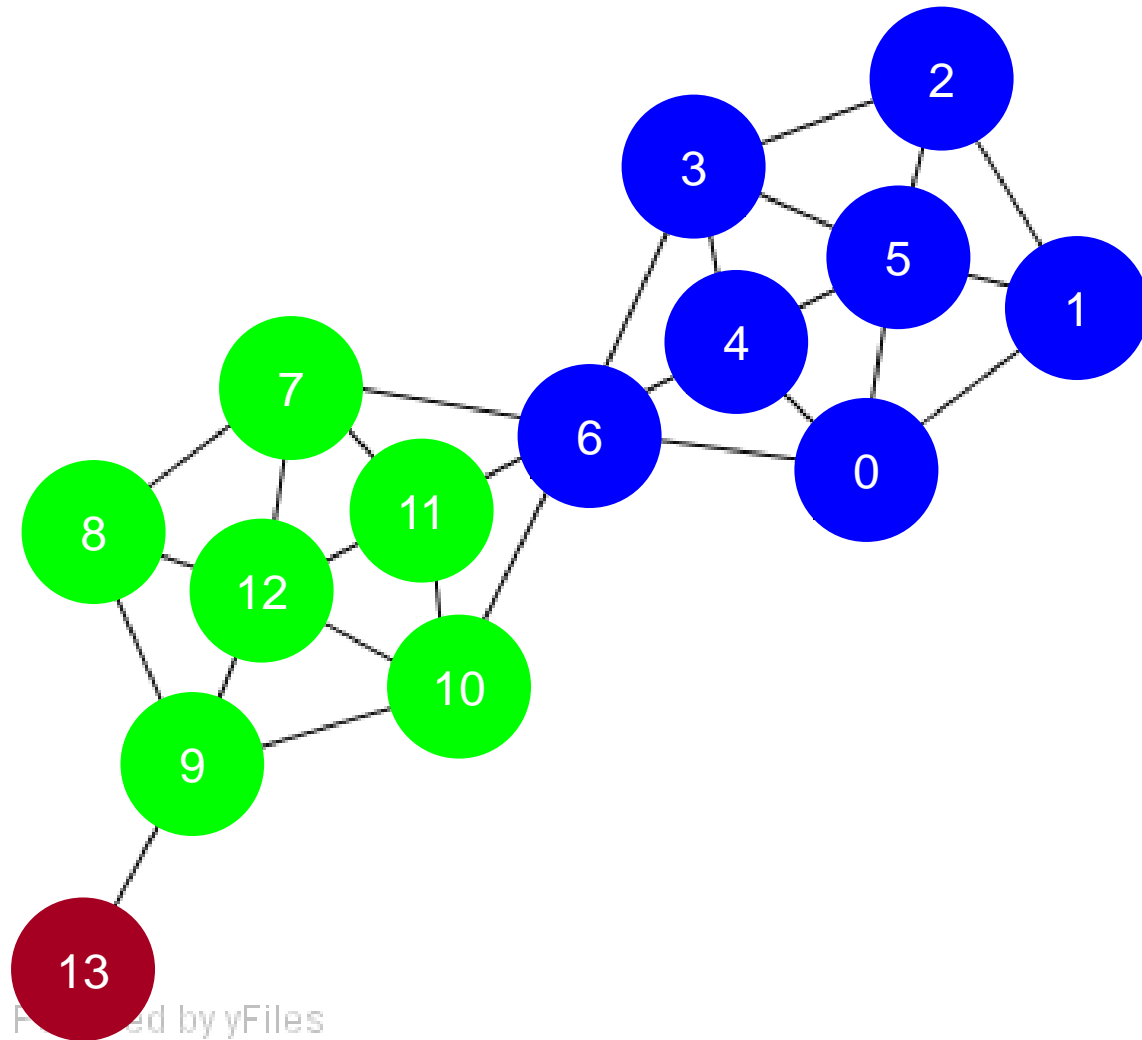
# Algorithm

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



# Algorithm

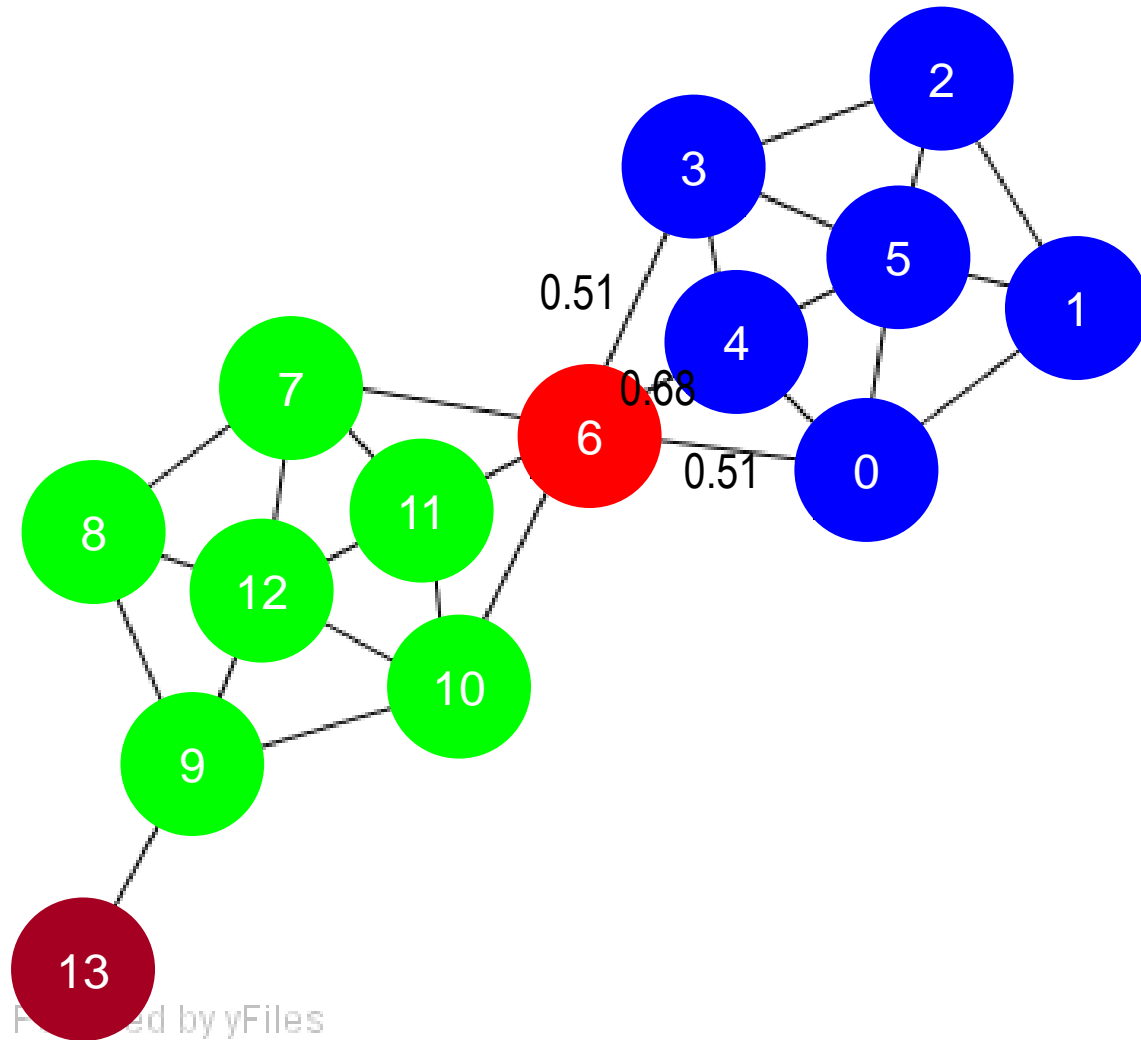
$$\mu = 2$$
$$\varepsilon = 0.7$$



Powered by yFiles

# Algorithm

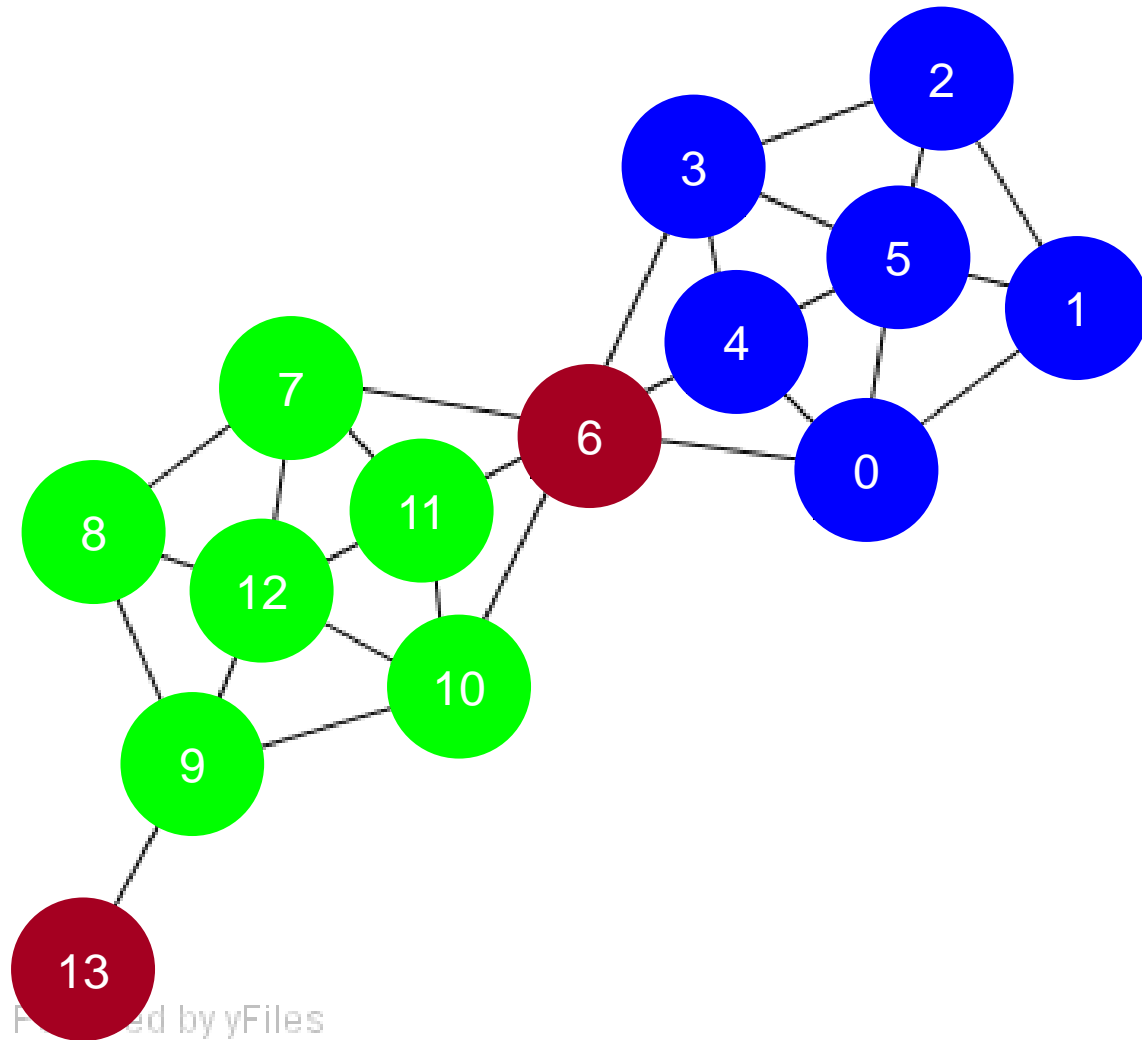
$$\mu = 2$$
$$\varepsilon = 0.7$$



Powered by yFiles

# Algorithm

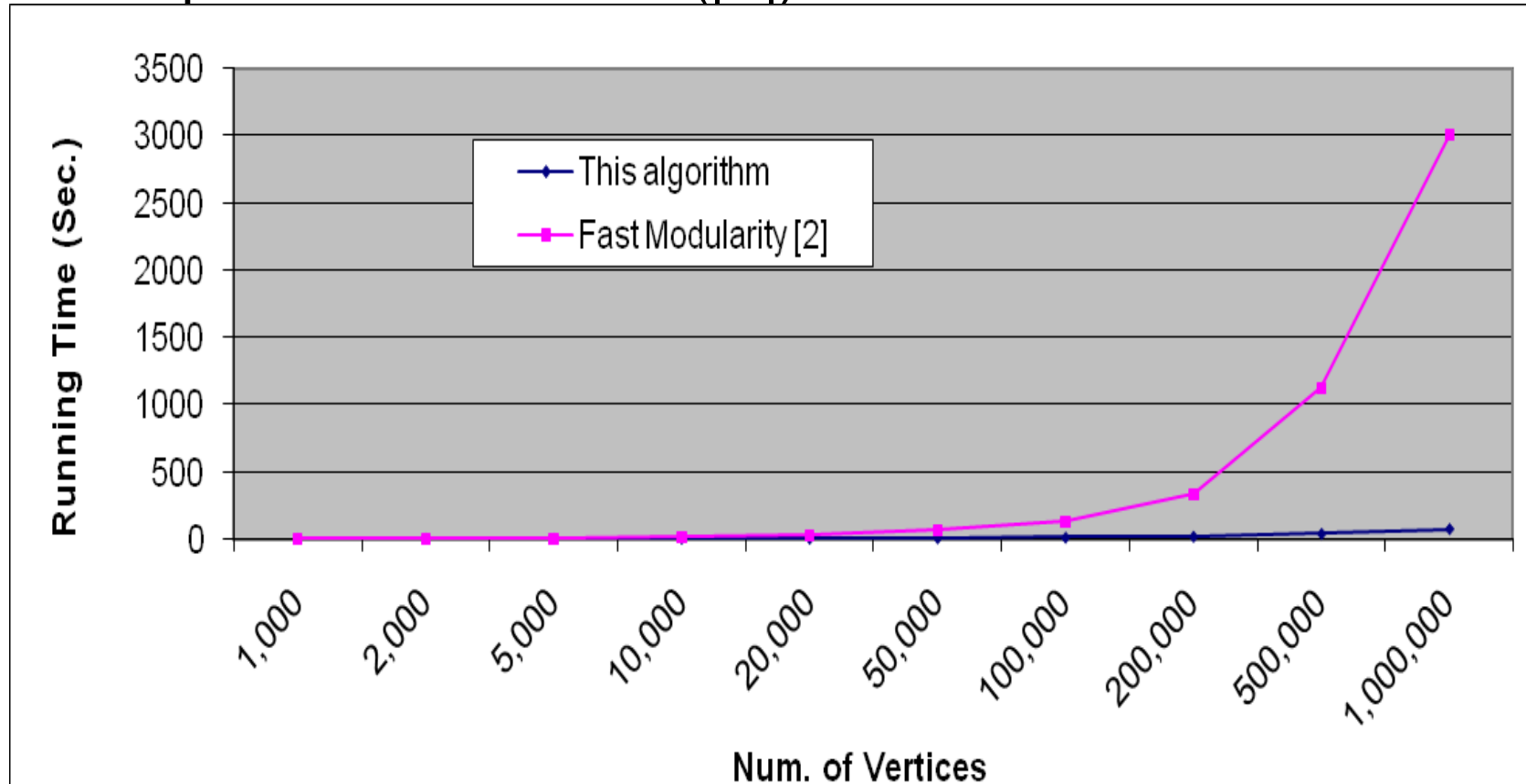
$$\mu = 2$$
$$\varepsilon = 0.7$$



Powered by yFiles

# Running Time

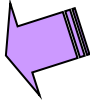
- Running time =  $O(|E|)$
- For sparse networks =  $O(|V|)$



[2] A. Clauset, M. E. J. Newman, & C. Moore, *Phys. Rev. E* **70**, 066111 (2004).

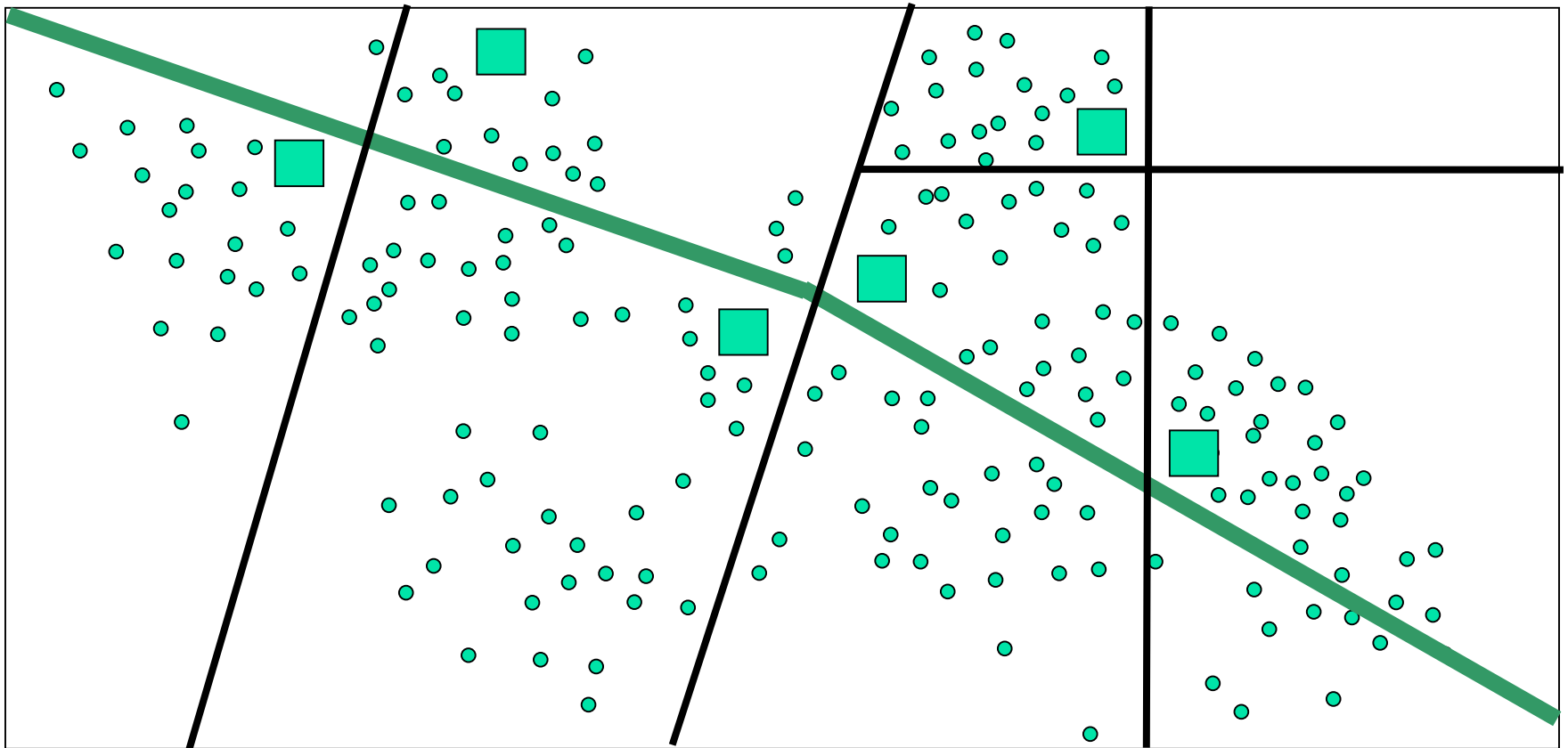
# Chapter 11. Cluster Analysis: Advanced Methods

---

- Probability Model-Based Clustering
- Clustering High-Dimensional Data
- Clustering Graphs and Network Data
- Clustering with Constraints 
- Summary

# Why Constraint-Based Cluster Analysis?

- Need user feedback: Users know their applications the best
- Less parameters but more user-desired constraints, e.g., an ATM allocation problem: obstacle & desired clusters





# Categorization of Constraints

---

- Constraints on instances: specifies how a pair or a set of instances should be grouped in the cluster analysis
  - Must-link vs. cannot link constraints
    - $\text{must-link}(x, y)$ :  $x$  and  $y$  should be grouped into one cluster
  - Constraints can be defined using variables, e.g.,
    - $\text{cannot-link}(x, y)$  if  $\text{dist}(x, y) > d$
- Constraints on clusters: specifies a requirement on the clusters
  - E.g., specify the min # of objects in a cluster, the max diameter of a cluster, the shape of a cluster (e.g., a convex), # of clusters (e.g.,  $k$ )
- Constraints on similarity measurements: specifies a requirement that the similarity calculation must respect
  - E.g., driving on roads, obstacles (e.g., rivers, lakes)
- Issues: Hard vs. soft constraints; conflicting or redundant constraints

# Constraint-Based Clustering Methods (I): Handling Hard Constraints

---

- Handling hard constraints: Strictly respect the constraints in cluster assignments
- Example: The COP-k-means algorithm
  - Generate super-instances for must-link constraints
    - Compute the transitive closure of the must-link constraints
    - To represent such a subset, replace all those objects in the subset by the mean.
    - The super-instance also carries a weight, which is the number of objects it represents
  - Conduct modified k-means clustering to respect cannot-link constraints
    - Modify the center-assignment process in k-means to a *nearest feasible center assignment*
    - An object is assigned to the nearest center so that the assignment respects all cannot-link constraints

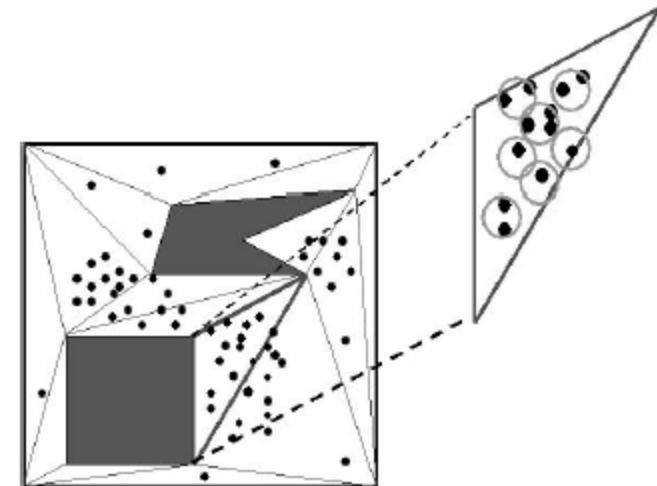
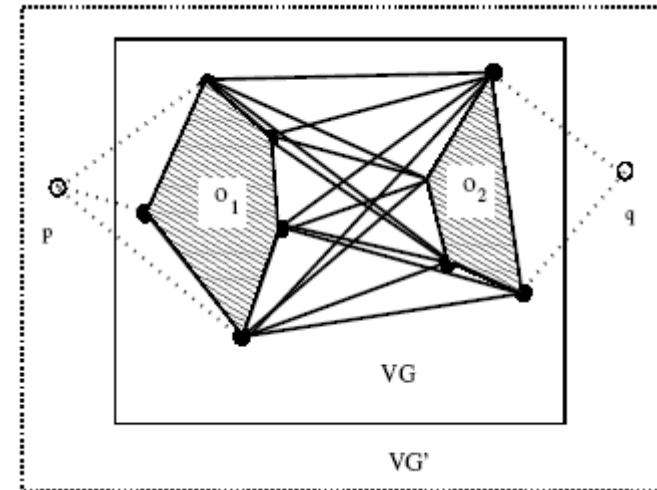
# Constraint-Based Clustering Methods (II): Handling Soft Constraints

---

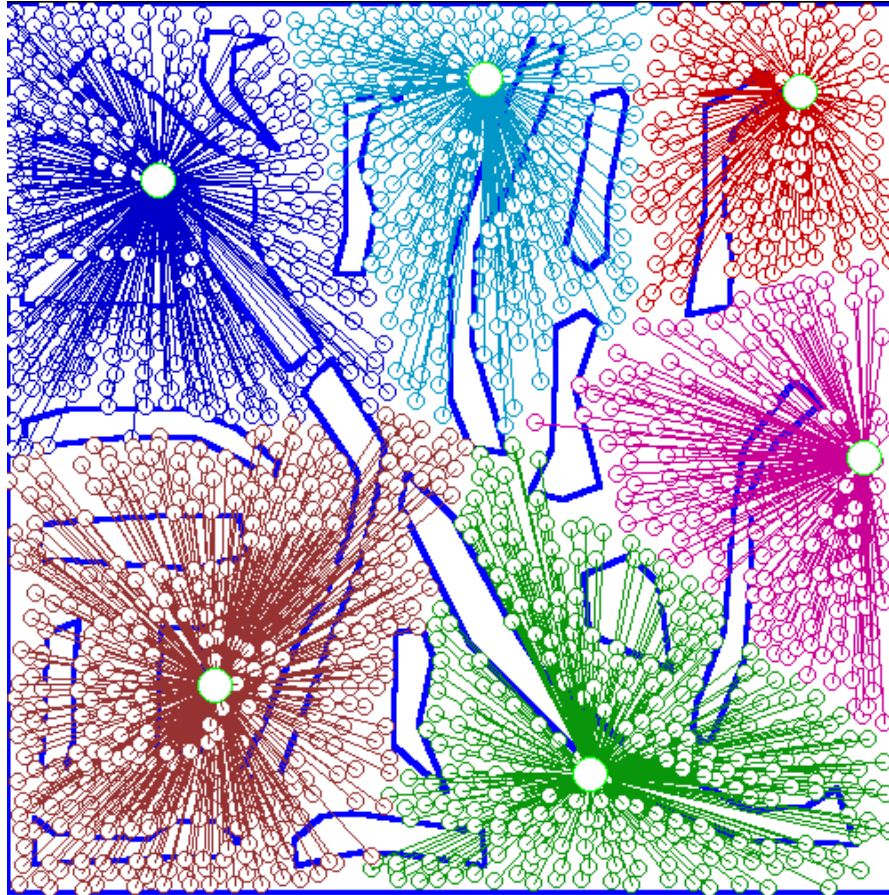
- Treated as an optimization problem: When a clustering violates a soft constraint, a penalty is imposed on the clustering
- Overall objective: Optimizing the clustering quality, and minimizing the constraint violation penalty
- Ex. CVQE (Constrained Vector Quantization Error) algorithm: Conduct k-means clustering while enforcing constraint violation penalties
- Objective function: Sum of distance used in k-means, adjusted by the constraint violation penalties
  - Penalty of a *must-link* violation
    - If objects  $x$  and  $y$  must-be-linked but they are assigned to two different centers,  $c_1$  and  $c_2$ ,  $\text{dist}(c_1, c_2)$  is added to the objective function as the penalty
  - Penalty of a cannot-link violation
    - If objects  $x$  and  $y$  cannot-be-linked but they are assigned to a common center  $c$ ,  $\text{dist}(c, c')$ , between  $c$  and  $c'$  is added to the objective function as the penalty, where  $c'$  is the closest cluster to  $c$  that can accommodate  $x$  or  $y$

# Speeding Up Constrained Clustering

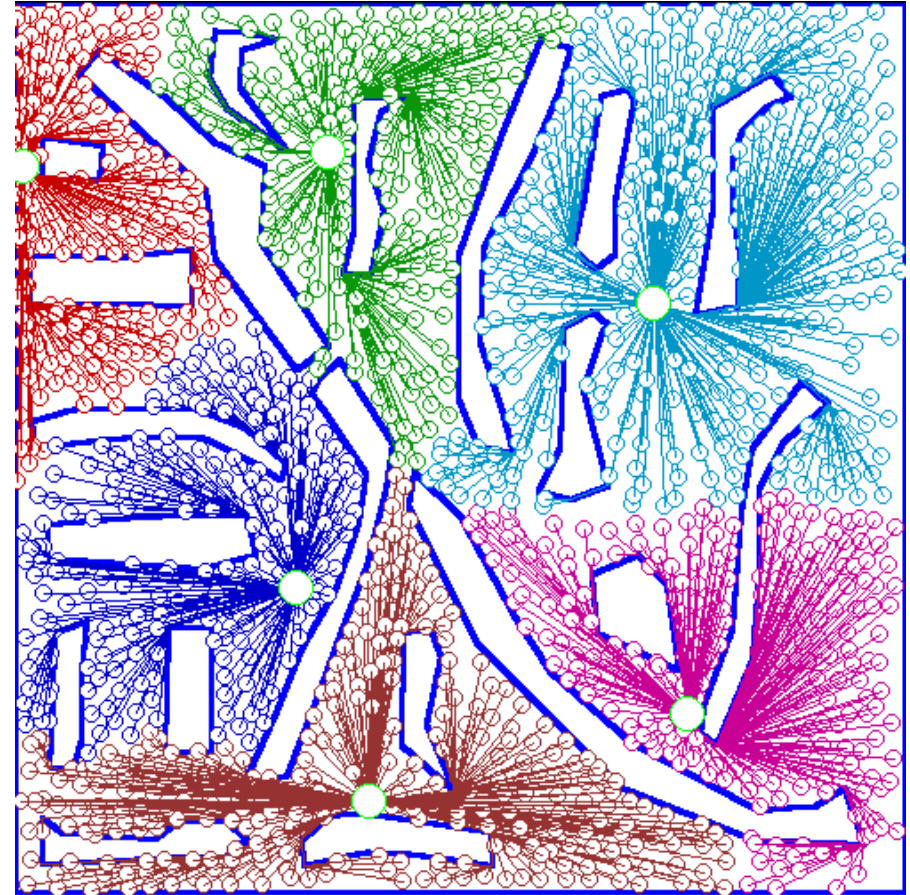
- It is costly to compute some constrained clustering
- Ex. Clustering with obstacle objects: Tung, Hou, and Han. Spatial clustering in the presence of obstacles, ICDE'01
- K-medoids is more preferable since k-means may locate the ATM center in the middle of a lake
- Visibility graph and shortest path
- Triangulation and micro-clustering
- Two kinds of join indices (shortest-paths) worth pre-computation
  - VV index: indices for any pair of obstacle vertices
  - MV index: indices for any pair of micro-cluster and obstacle indices



# An Example: Clustering With Obstacle Objects



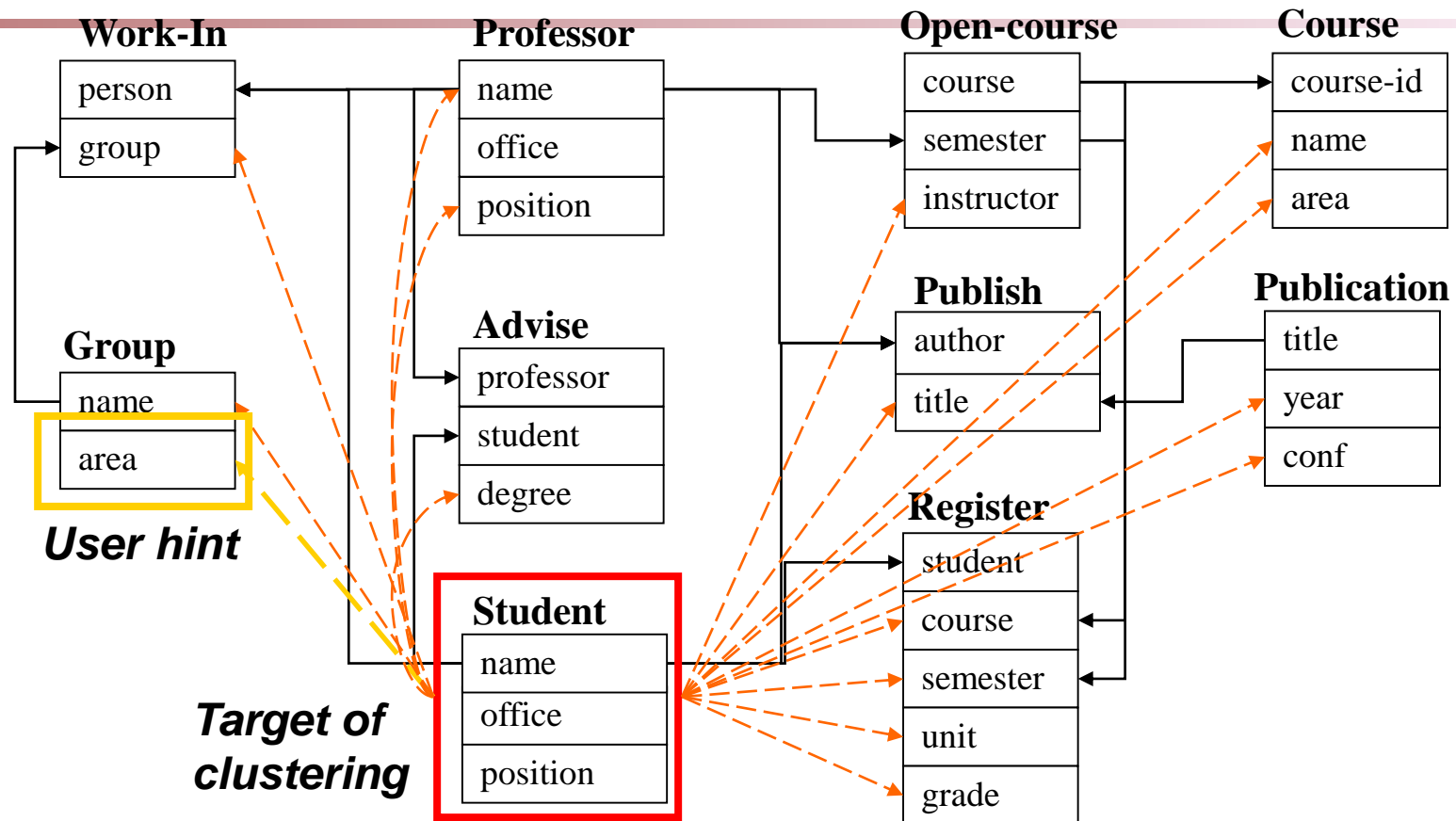
**Not** Taking obstacles into account



Taking obstacles into account



# User-Guided Clustering: A Special Kind of Constraints



- X. Yin, J. Han, P. S. Yu, "Cross-Relational Clustering with User's Guidance", KDD'05
- User usually has a goal of clustering, e.g., clustering students by research area
- User specifies his clustering goal to CrossClus

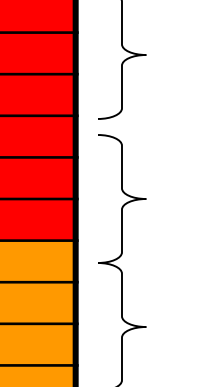
## Comparing with Classification

User hint

			Red
			Red
			Red
			Red
			Red
			Orange
			Orange
			Orange
			Orange
			Orange
			Magenta
			Magenta
			Magenta
			Magenta
			Magenta

## All tuples for clustering

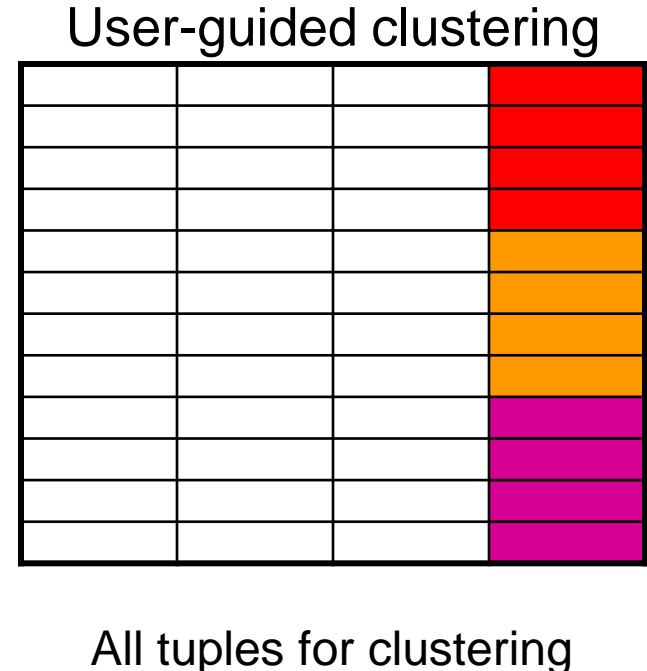
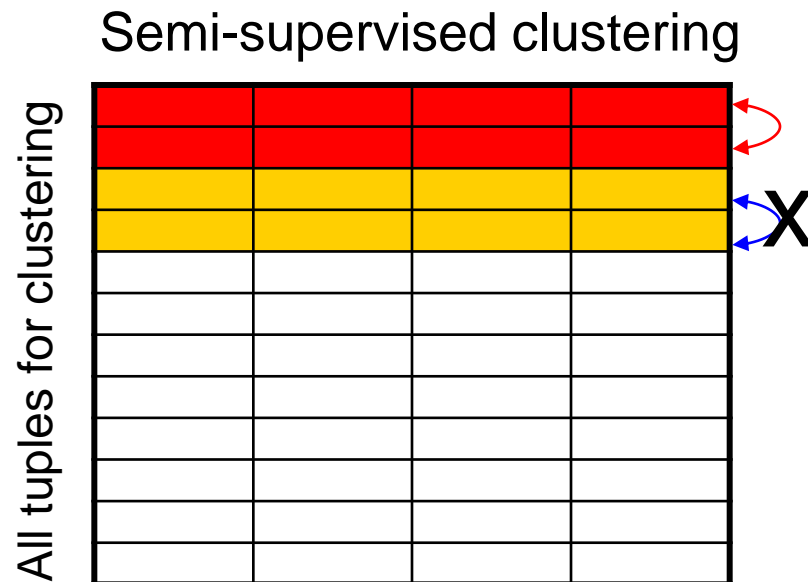
- User hint ■ User-specified *feature* (in the form of *attribute*) is used as a hint, not class labels

  - The attribute may contain too many or too few distinct values, e.g., a user may want to cluster students into 20 clusters instead of 3
  - Additional features need to be included in cluster analysis

The diagram shows a data matrix with 15 rows and 2 columns. The first column is labeled 'User hint' and contains 15 red cells. The second column is labeled 'clustering' and contains 15 cells, each with a different color (yellow, orange, green, blue, purple, etc.). Brackets on the right side of the matrix group the rows into three clusters of five rows each, corresponding to the three colors in the 'clustering' column.

# Comparing with Semi-Supervised Clustering

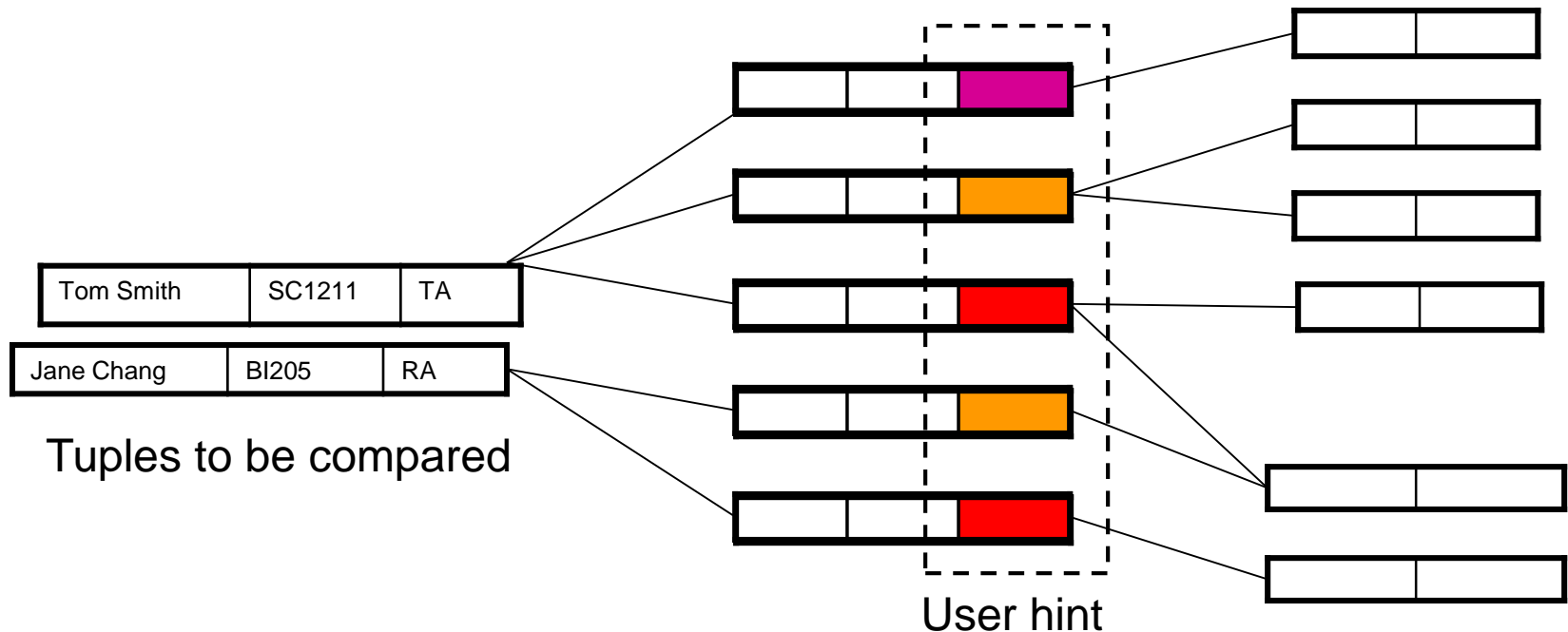
- Semi-supervised clustering: User provides a training set consisting of “similar” (“must-link”) and “dissimilar” (“cannot link”) pairs of objects
- User-guided clustering: User specifies an attribute as a hint, and more relevant features are found for clustering





# Why Not Semi-Supervised Clustering?

- Much information (in multiple relations) is needed to judge whether two tuples are similar
- A user may not be able to provide a good training set
- It is much easier for a user to specify an attribute as a hint, such as a student's *research area*



# CrossClus: An Overview

---

- Measure *similarity between features* by how they group objects into clusters
- Use a heuristic method to search for pertinent features
  - *Start from user-specified feature and gradually expand search range*
- Use *tuple ID propagation* to create feature values
  - Features can be easily created during the expansion of search range, by propagating IDs
- Explore three clustering algorithms: *k*-means, *k*-medoids, and hierarchical clustering

# Multi-Relational Features

- A multi-relational feature is defined by:
  - A join path, e.g.,  $Student \rightarrow Register \rightarrow OpenCourse \rightarrow Course$
  - An attribute, e.g.,  $Course.area$
  - (For numerical feature) an aggregation operator, e.g., sum or average
- Categorical feature  $f = [Student \rightarrow Register \rightarrow OpenCourse \rightarrow Course, Course.area, null]$

areas of courses of each student

Tuple	Areas of courses		
	<i>DB</i>	<i>AI</i>	<i>TH</i>
$t_1$	5	5	0
$t_2$	0	3	7
$t_3$	1	5	4
$t_4$	5	0	5
$t_5$	3	3	4

Values of feature  $f$

Tuple	Feature $f$		
	<i>DB</i>	<i>AI</i>	<i>TH</i>
$t_1$	0.5	0.5	0
$t_2$	0	0.3	0.7
$t_3$	0.1	0.5	0.4
$t_4$	0.5	0	0.5
$t_5$	0.3	0.3	0.4

$f(t_1)$



$f(t_2)$



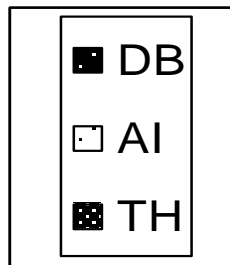
$f(t_3)$



$f(t_4)$



$f(t_5)$

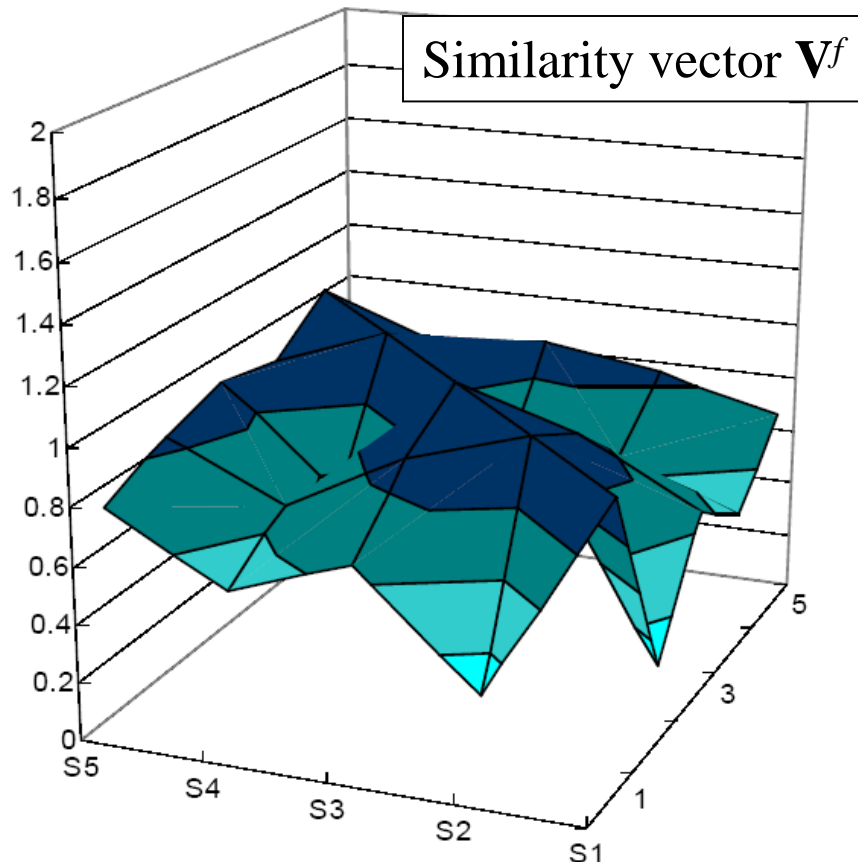


# Representing Features

- Similarity between tuples  $t_1$  and  $t_2$  w.r.t. categorical feature  $f$

- Cosine similarity between vectors  $f(t_1)$  and  $f(t_2)$

$$\text{sim}_f(t_1, t_2) = \frac{\sum_{k=1}^L f(t_1).p_k \cdot f(t_2).p_k}{\sqrt{\sum_{k=1}^L f(t_1).p_k^2} \cdot \sqrt{\sum_{k=1}^L f(t_2).p_k^2}}$$



- Most important information of a feature  $f$  is how  $f$  groups tuples into clusters
- $f$  is represented by similarities between every pair of tuples indicated by  $f$
- The horizontal axes are the tuple indices, and the vertical axis is the similarity
- This can be considered as a vector of  $N \times N$  dimensions

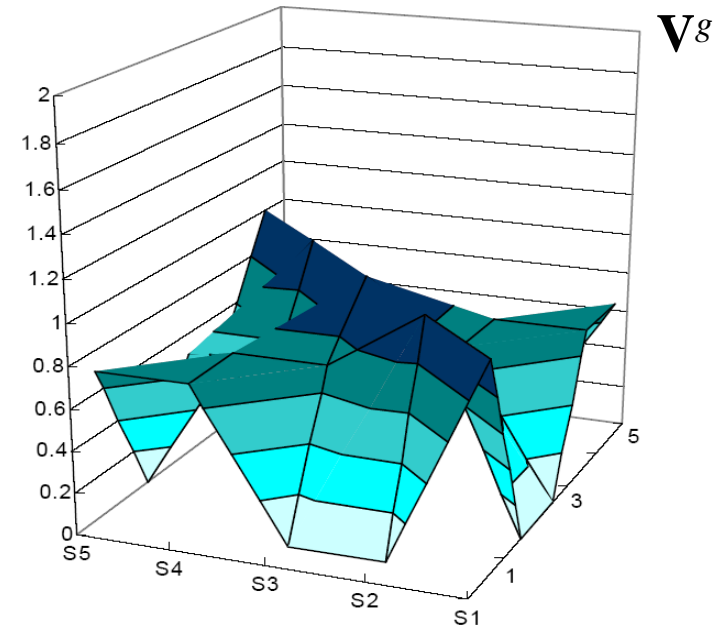
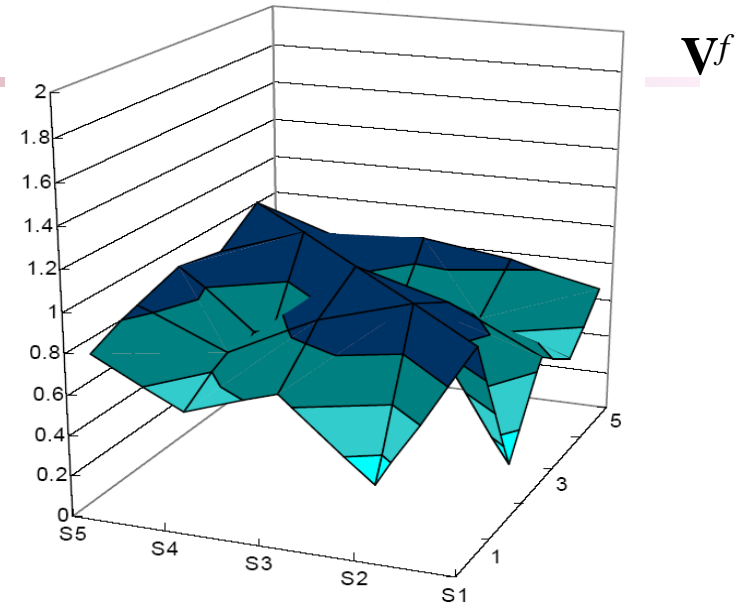
# Similarity Between Features

Values of Feature  $f$  and  $g$

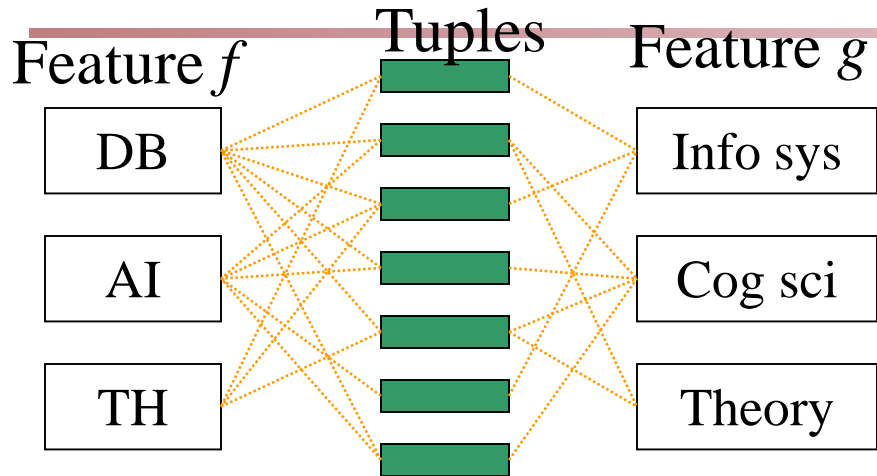
	Feature $f$ (course)				Feature $g$ (group)		
	DB	AI	TH		Info sys	Cog sci	Theory
$t_1$	0.5	0.5	0		1	0	0
$t_2$	0	0.3	0.7		0	0	1
$t_3$	0.1	0.5	0.4		0	0.5	0.5
$t_4$	0.5	0	0.5		0.5	0	0.5
$t_5$	0.3	0.3	0.4		0.5	0.5	0

Similarity between two features –  
cosine similarity of two vectors

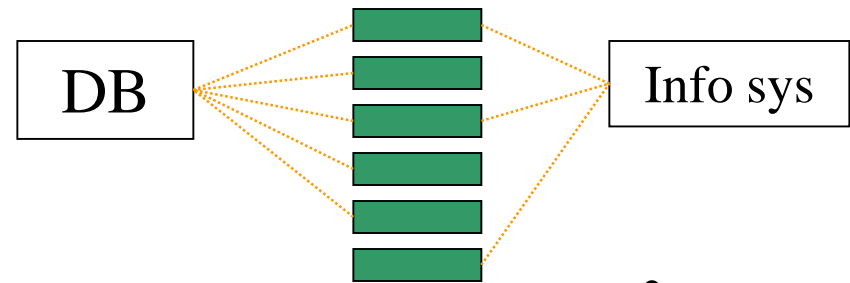
$$\text{sim}(f, g) = \frac{V^f \cdot V^g}{|V^f| |V^g|}$$



# Computing Feature Similarity



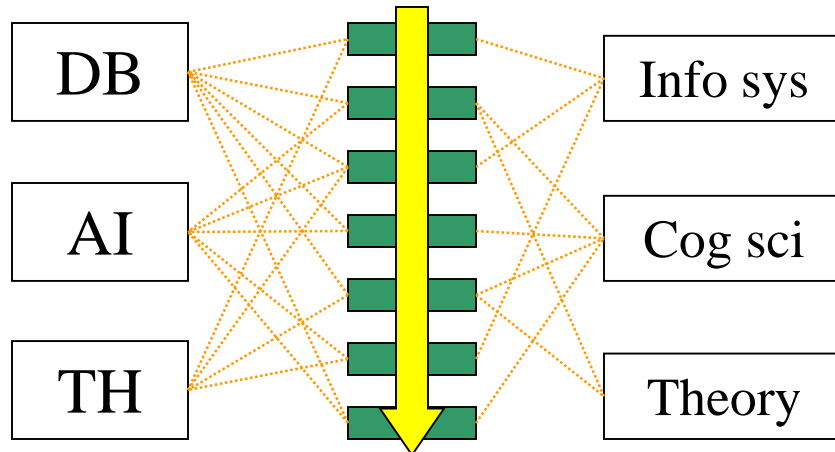
Similarity between feature values w.r.t. the tuples

$$\text{sim}(f_k, g_q) = \sum_{i=1 \text{ to } N} f(t_i) \cdot p_k \cdot g(t_i) \cdot p_q$$


$$V^f \cdot V^g = \sum_{i=1}^N \sum_{j=1}^N \text{sim}_g(t_i, t_j) = \sum_{k=1}^l \sum_{q=1}^m \text{sim}(f_k, g_q)^2$$

*sim(t<sub>i</sub>, t<sub>j</sub>): sim<sub>g</sub>(t<sub>i</sub>, t<sub>j</sub>) = Tuple similarities, hard to compute*

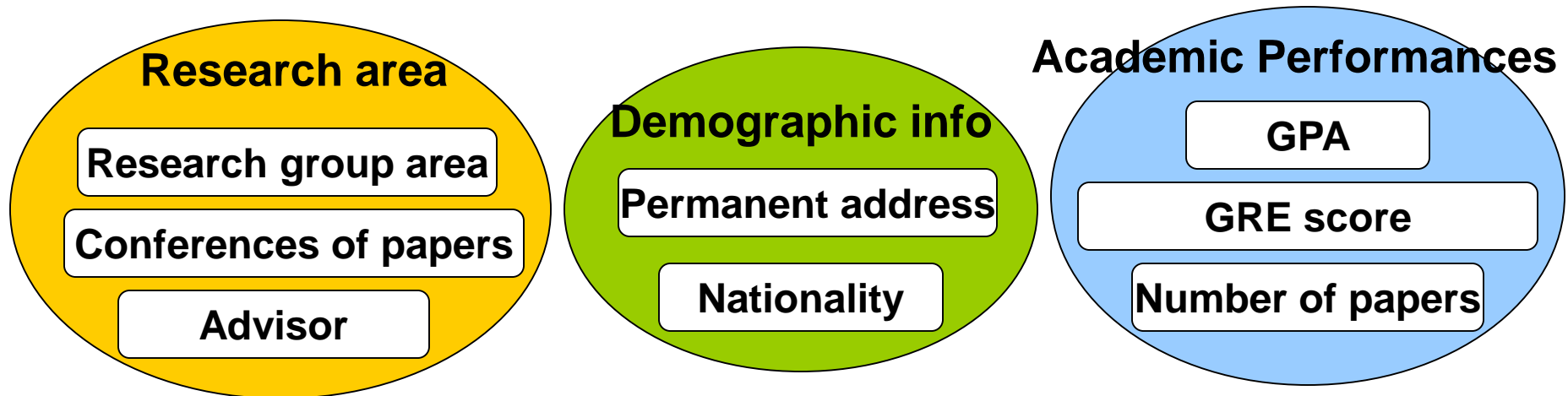
*sim(f<sub>k</sub>, g<sub>q</sub>): sim(f<sub>k</sub>, g<sub>q</sub>) = Feature value similarities, easy to compute*



Compute similarity between each pair of feature values by one scan on data

# Searching for Pertinent Features

- Different features convey different aspects of information

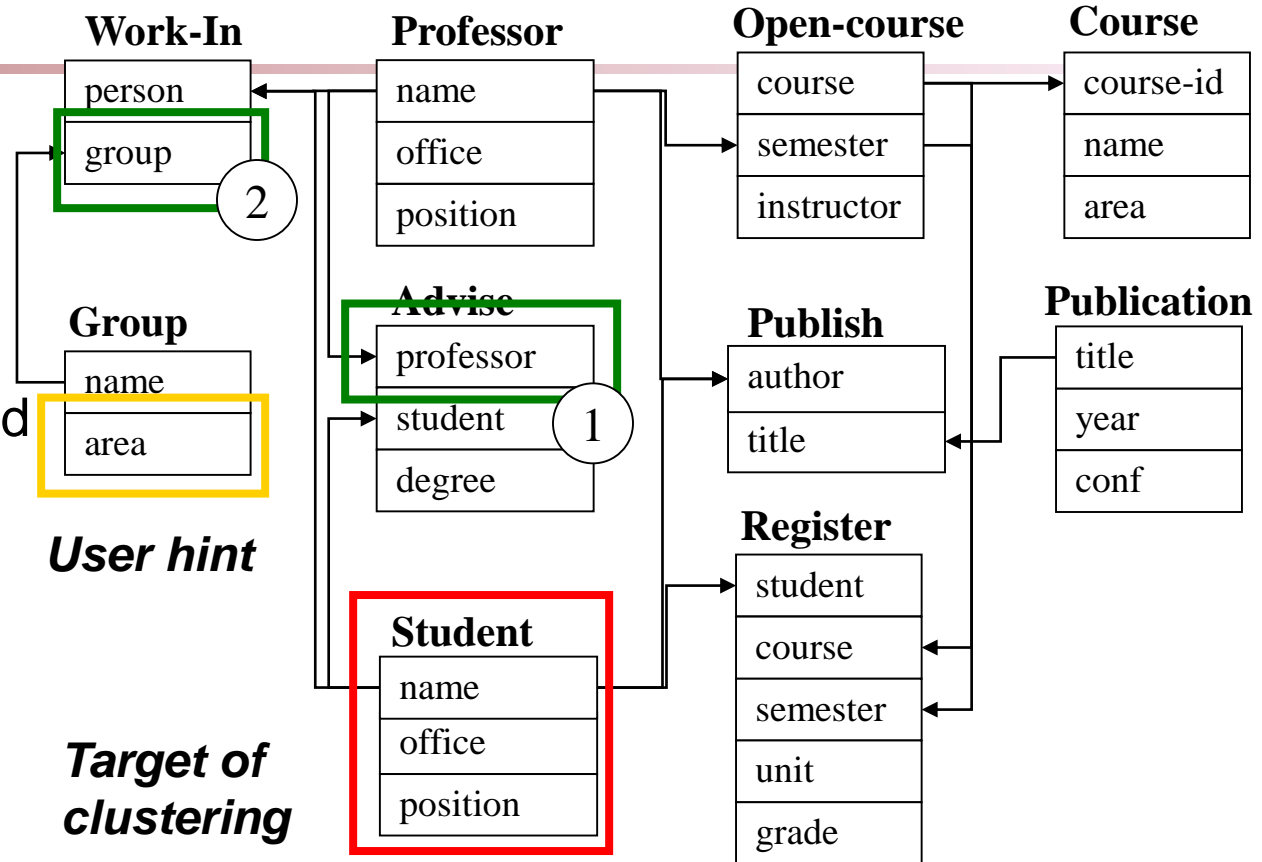


- Features conveying same aspect of information usually cluster tuples in more similar ways
  - Research group areas vs. conferences of publications
- Given user specified feature
  - Find pertinent features by computing feature similarity

# Heuristic Search for Pertinent Features

## Overall procedure

1. Start from the user-specified feature
2. Search in neighborhood of existing pertinent features
3. Expand search range gradually



- Tuple ID propagation is used to create multi-relational features
  - IDs of target tuples can be propagated along any join path, from which we can find tuples joinable with each target tuple



# Clustering with Multi-Relational Features

---

- Given a set of  $L$  pertinent features  $f_1, \dots, f_L$ , similarity between two tuples

$$\text{sim}(t_1, t_2) = \sum_{i=1}^L \text{sim}_{f_i}(t_1, t_2) \cdot f_i.\text{weight}$$

- Weight of a feature is determined in feature search by its similarity with other pertinent features
- Clustering methods
  - CLARANS [Ng & Han 94], a scalable clustering algorithm for non-Euclidean space
  - K-means
  - Agglomerative hierarchical clustering

# Experiments: Compare CrossClus with

---

- Baseline: Only use the user specified feature
- PROCLUS [Aggarwal, et al. 99]: a state-of-the-art subspace clustering algorithm
  - Use a subset of features for each cluster
  - We convert relational database to a table by propositionalization
  - User-specified feature is forced to be used in every cluster
- RDBC [Kirsten and Wrobel'00]
  - A representative ILP clustering algorithm
  - Use neighbor information of objects for clustering
  - User-specified feature is forced to be used

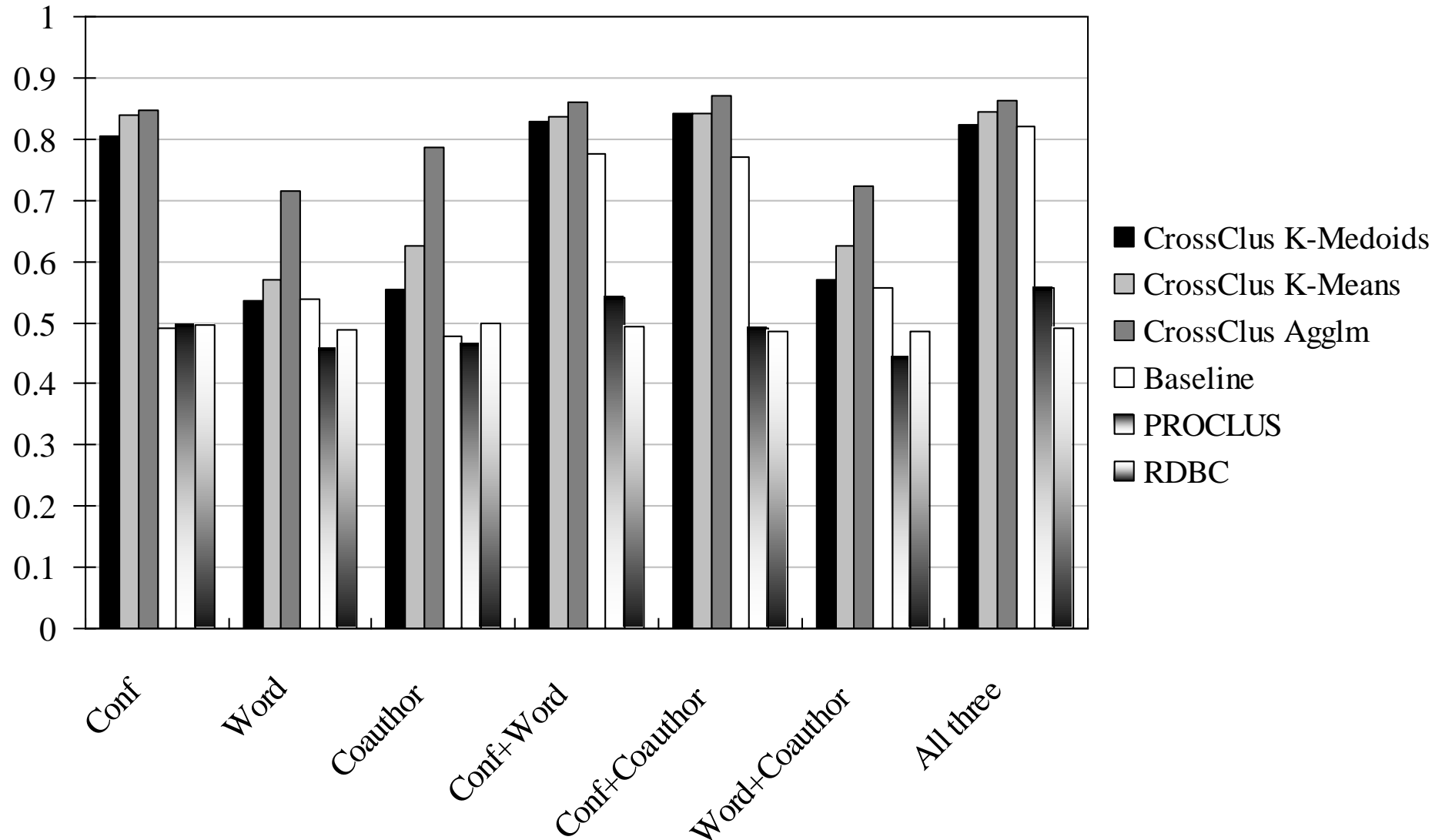
# Measure of Clustering Accuracy

---

- Accuracy
  - Measured by manually labeled data
    - We manually assign tuples into clusters according to their properties (e.g., professors in different research areas)
  - Accuracy of clustering: Percentage of pairs of tuples in the same cluster that share common label
    - This measure favors many small clusters
    - We let each approach generate the same number of clusters


# DBLP Dataset

Clustering Accuracy - DBLP



# Chapter 11. Cluster Analysis: Advanced Methods

---

- Probability Model-Based Clustering
- Clustering High-Dimensional Data
- Clustering Graphs and Network Data
- Clustering with Constraints
- Summary 

# Summary

---

- Probability Model-Based Clustering
  - Fuzzy clustering
  - Probability-model-based clustering
  - The EM algorithm
- Clustering High-Dimensional Data
  - Subspace clustering: bi-clustering methods
  - Dimensionality reduction: Spectral clustering
- Clustering Graphs and Network Data
  - Graph clustering: min-cut vs. sparsest cut
  - High-dimensional clustering methods
  - Graph-specific clustering methods, e.g., SCAN
- Clustering with Constraints
  - Constraints on instance objects, e.g., Must link vs. Cannot Link
  - Constraint-based clustering algorithms

# References (I)

---

- R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. *SIGMOD'98*
- C. C. Aggarwal, C. Procopiuc, J. Wolf, P. S. Yu, and J.-S. Park. Fast algorithms for projected clustering. *SIGMOD'99*
- S. Arora, S. Rao, and U. Vazirani. Expander flows, geometric embeddings and graph partitioning. *J. ACM*, 56:5:1–5:37, 2009.
- J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, 1981.
- K. S. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? *ICDT'99*
- Y. Cheng and G. Church. Biclustering of expression data. *ISMB'00*
- I. Davidson and S. S. Ravi. Clustering with constraints: Feasibility issues and the k-means algorithm. *SDM'05*
- I. Davidson, K. L. Wagstaff, and S. Basu. Measuring constraint-set utility for partitional clustering algorithms. *PKDD'06*
- C. Fraley and A. E. Raftery. Model-based clustering, discriminant analysis, and density estimation. *J. American Stat. Assoc.*, 97:611–631, 2002.
- F. H"oppner, F. Klawonn, R. Kruse, and T. Runkler. *Fuzzy Cluster Analysis: Methods for Classification, Data Analysis and Image Recognition*. Wiley, 1999.
- G. Jeh and J. Widom. SimRank: a measure of structural-context similarity. *KDD'02*
- H.-P. Kriegel, P. Kroeger, and A. Zimek. Clustering high dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Trans. Knowledge Discovery from Data (TKDD)*, 3, 2009.
- U. Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17:395–416, 2007

# References (II)

---

- G. J. McLachlan and K. E. Bkassfod. *Mixture Models: Inference and Applications to Clustering*. John Wiley & Sons, 1988.
- B. Mirkin. Mathematical classification and clustering. *J. of Global Optimization*, 12:105–108, 1998.
- S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 1, 2004.
- A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. NIPS'01
- J. Pei, X. Zhang, M. Cho, H. Wang, and P. S. Yu. Maple: A fast algorithm for maximal pattern-based clustering. *ICDM'03*
- M. Radovanović, A. Nanopoulos, and M. Ivanović. Nearest neighbors in high-dimensional data: the emergence and influence of hubs. *ICML'09*
- S. E. Schaeffer. Graph clustering. *Computer Science Review*, 1:27–64, 2007.
- A. K. H. Tung, J. Hou, and J. Han. Spatial clustering in the presence of obstacles. *ICDE'01*
- A. K. H. Tung, J. Han, L. V. S. Lakshmanan, and R. T. Ng. Constraint-based clustering in large databases. *ICDT'01*
- A. Tanay, R. Sharan, and R. Shamir. Biclustering algorithms: A survey. In *Handbook of Computational Molecular Biology*, Chapman & Hall, 2004.
- K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl. Constrained k-means clustering with background knowledge. *ICML'01*
- H. Wang, W. Wang, J. Yang, and P. S. Yu. Clustering by pattern similarity in large data sets. *SIGMOD'02*
- X. Xu, N. Yuruk, Z. Feng, and T. A. J. Schweiger. SCAN: A structural clustering algorithm for networks. *KDD'07*
- X. Yin, J. Han, and P.S. Yu, “Cross-Relational Clustering with User's Guidance”, KDD'05



---

# **Slides Not to Be Used in Class**

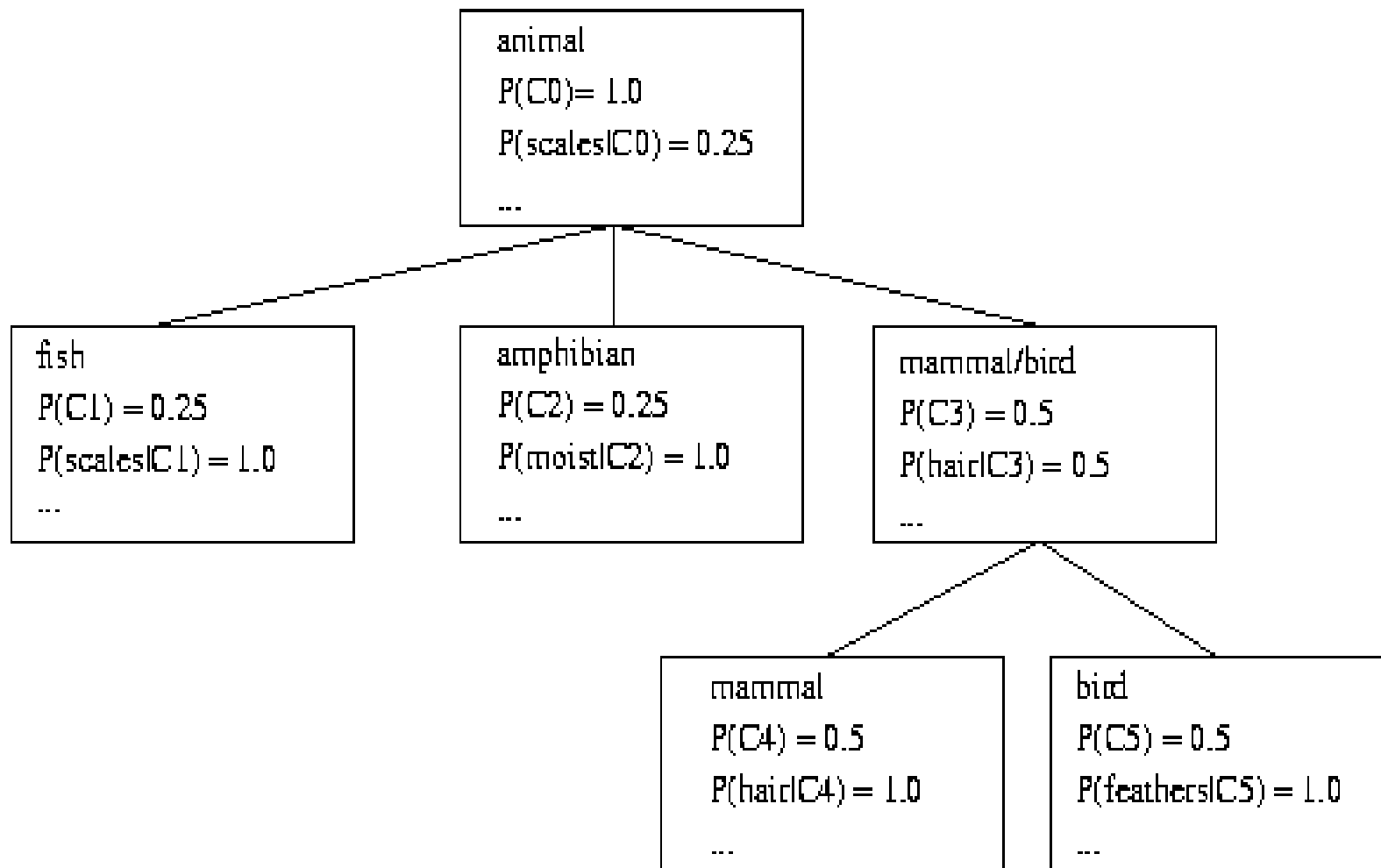
# Conceptual Clustering

---

- Conceptual clustering
  - A form of clustering in machine learning
  - Produces a classification scheme for a set of unlabeled objects
  - Finds characteristic description for each concept (class)
- COBWEB (Fisher'87)
  - A popular a simple method of incremental conceptual learning
  - Creates a hierarchical clustering in the form of a **classification tree**
  - Each node refers to a concept and contains a probabilistic description of that concept

# COBWEB Clustering Method

## A classification tree



# More on Conceptual Clustering

---

- Limitations of COBWEB
  - The assumption that the attributes are independent of each other is often too strong because correlation may exist
  - Not suitable for clustering large database data – skewed tree and expensive probability distributions
- CLASSIT
  - an extension of COBWEB for incremental clustering of continuous data
  - suffers similar problems as COBWEB
- AutoClass (Cheeseman and Stutz, 1996)
  - Uses Bayesian statistical analysis to estimate the number of clusters
  - Popular in industry

# Neural Network Approaches

---

- Neural network approaches
  - Represent each cluster as an exemplar, acting as a “prototype” of the cluster
  - New objects are distributed to the cluster whose exemplar is the most similar according to some distance measure
- Typical methods
  - SOM (Soft-Organizing feature Map)
  - Competitive learning
    - Involves a hierarchical architecture of several units (neurons)
    - Neurons compete in a “winner-takes-all” fashion for the object currently being presented

# Self-Organizing Feature Map (SOM)

---

- SOMs, also called topological ordered maps, or Kohonen Self-Organizing Feature Map (KSOMs)
- It maps all the points in a high-dimensional source space into a 2 to 3-d target space, s.t., the distance and proximity relationship (i.e., topology) are preserved as much as possible
- Similar to k-means: cluster centers tend to lie in a low-dimensional manifold in the feature space
- Clustering is performed by having several units competing for the current object
  - The unit whose weight vector is closest to the current object wins
  - The winner and its neighbors learn by having their weights adjusted
- SOMs are believed to resemble processing that can occur in the brain
- Useful for visualizing high-dimensional data in 2- or 3-D space

# Web Document Clustering Using SOM

- The result of SOM clustering of 12088 Web articles
- The picture on the right: drilling down on the keyword “mining”
- Based on websom.hut.fi Web page

