Data Mining: Concepts and Techniques

— Chapter 7 —

Jiawei Han
Department of Computer Science
University of Illinois at Urbana-Champaign
www.cs.uiuc.edu/~hanj

Cluster Analysis

1. What is Cluster Analysis?
2. Types of Data in Cluster Analysis
3. A Categorization of Major Clustering Methods
4. Partitioning Methods
5. Hierarchical Methods
6. Density-Based Methods

Major Clustering Approaches (I)

- Partitioning approach:
  - Construct various partitions and then evaluate them by some criterion, e.g., minimizing the sum of square errors
  - Typical methods: k-means, k-medoids, CLARANS
- Hierarchical approach:
  - Create a hierarchical decomposition of the set of data (or objects) using some criterion
  - Typical methods: Diana, Agnes, BIRCH, ROCK, CHAMELEON
- Density-based approach:
  - Based on connectivity and density functions
  - Typical methods: DBSCAN, OPTICS, DenClue, CHAMELEON

Major Clustering Approaches (II)

- Grid-based approach:
  - based on a multiple-level granularity structure
  - Typical methods: STING, WaveCluster, CLIQUE
- Model-based:
  - A model is hypothesized for each of the clusters and tries to find the best fit of that model to each other
  - Typical methods: EM, SOM, COBWEB
- Frequent pattern-based:
  - Based on the analysis of frequent patterns
  - Typical methods: pCluster
- User-guided or constraint-based:
  - Clustering by considering user-specified or application-specific constraints
  - Typical methods: COD (obstacles), constrained clustering

Cluster Analysis

1. What is Cluster Analysis?
2. Types of Data in Cluster Analysis
3. A Categorization of Major Clustering Methods
4. Partitioning Methods
5. Hierarchical Methods
6. Density-Based Methods

Partitioning Algorithms: Basic Concept

- Partitioning method: Construct a partition of a database ***D*** of ***n*** objects into a set of ***k*** clusters, s.t., **minimal** sum of squared distance

$$\Sigma_{m=1}^{k}\Sigma_{t_{mi}\in Km}(C_m - t_{mi})^2$$

Figure on whiteboard:

Partitioning Algorithms: Basic Concept

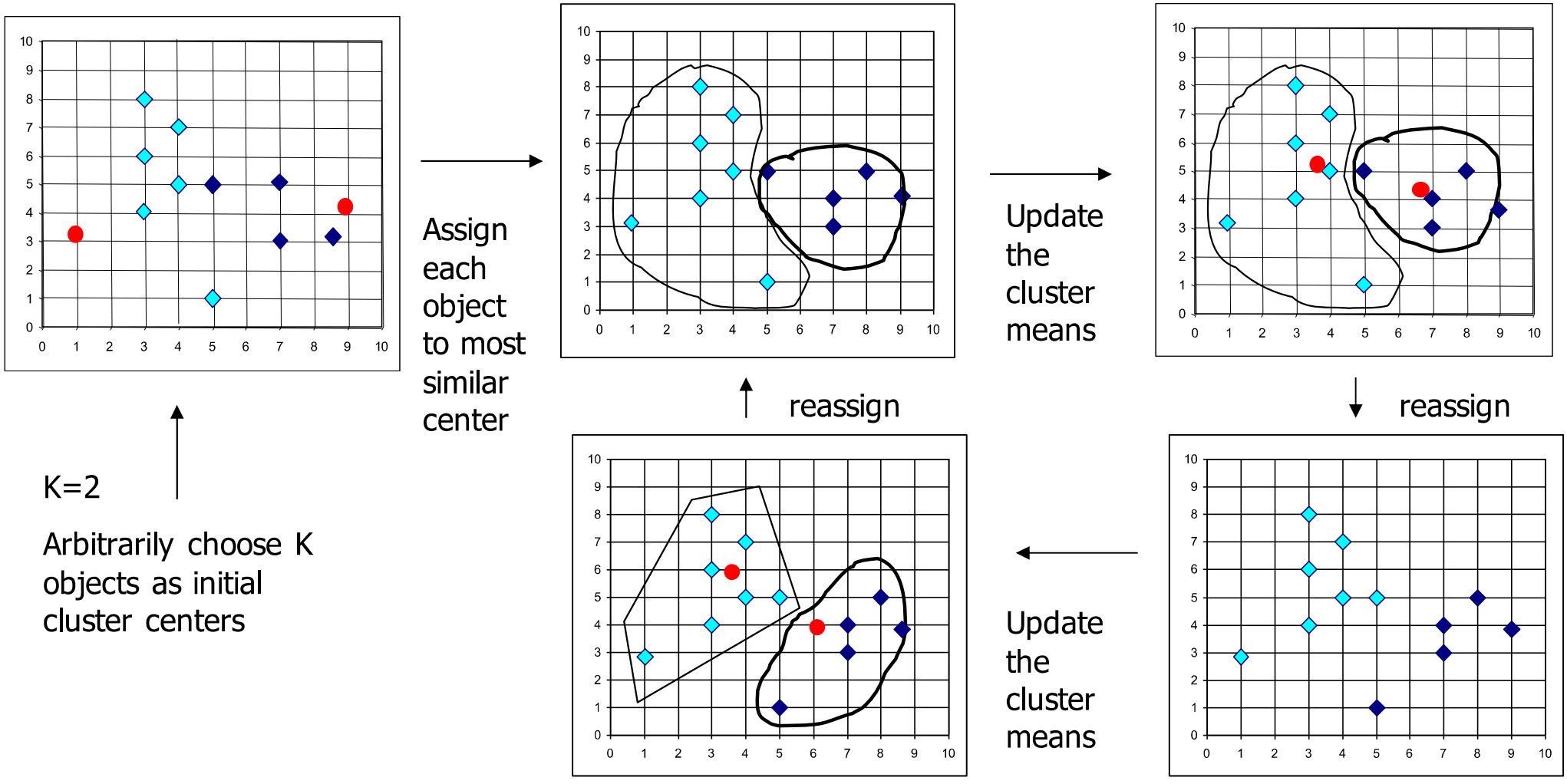
- Given a *k*, find a partition of *k clusters* that optimizes the chosen partitioning criterion
  - Global optimal: exhaustively enumerate all partitions
  - Heuristic methods: *k-means* and *k-medoids* algorithms
  - *k-means* (MacQueen'67): Each cluster is represented by the center of the cluster
  - *k-medoids* or PAM (Partition around medoids) (Kaufman & Rousseeuw'87): Each cluster is represented by one of the objects in the cluster

The *K-Means* Clustering Method

- Given *k*, data D
- 1. arbitrarily choose *k* objects as initial cluster centers
- 2. Repeat
  - Assign each object to the cluster to which the object is most similar based on mean values of the objects in the cluster
  - Update cluster means (calculate mean value of the objects for each cluster)
- Until no change

The *K-Means* Clustering Method

- Example



Comments on the *K-Means* Method

- Strength: *Relatively efficient*: *O*(*tkn*), where *n* is # objects, *k* is # clusters, and *t* is # iterations. Normally, *k*, *t* << *n*.
  - Comparing: PAM: $O(k(n-k)^2)$, CLARA: $O(ks^2 + k(n-k))$
- Comment: Often terminates at a *local optimum*. The *global optimum* may be found using techniques such as: *deterministic annealing* and *genetic algorithms*
- Weakness
  - Applicable only when *mean* is defined, then what about categorical data?
  - Need to specify *k,* the *number* of clusters, in advance
  - Unable to handle noisy data and *outliers*
  - Not suitable to discover clusters with *non-convex shapes*

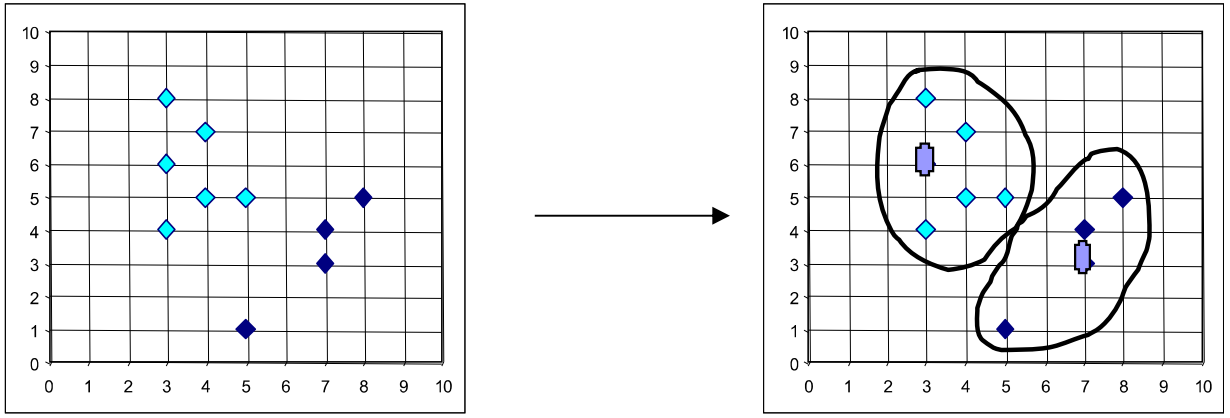
K-means is sensitive to outliers

- The k-means algorithm is sensitive to outliers !
  - Since an object with an extremely large value may substantially distort the distribution of the data.

Figure on whiteboard:

K-medoids or PAM

- K-Medoids: Instead of taking the **mean** value of the object in a cluster as a reference point, **medoids** can be used, which is the **most centrally located** object in a cluster.

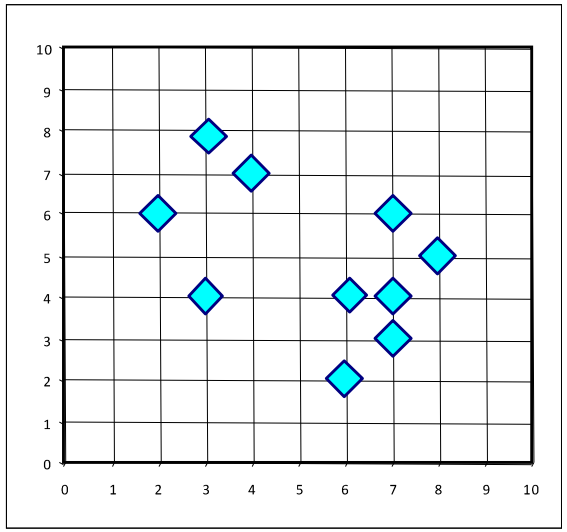


The *K-Medoids* Clustering Method

- Find *representative* objects, called medoids, in clusters
- *PAM* (Partitioning Around Medoids, 1987)
  - starts from an initial set of medoids and iteratively replaces one of the medoids by one of the non-medoids if it improves the total distance of the resulting clustering
  - *PAM* works effectively for small data sets, but does not scale well for large data sets
- Optimizations:
  - *CLARA* (Kaufmann & Rousseeuw, 1990)
  - *CLARANS* (Ng & Han, 1994): Randomized sampling

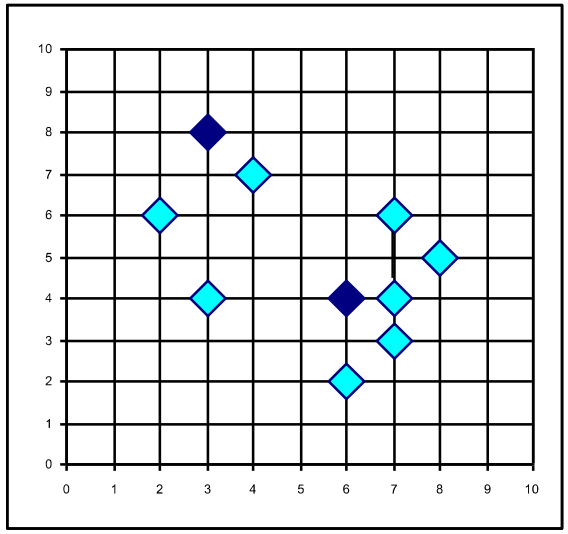
A Typical K-Medoids Algorithm (PAM) –

basic idea, NOT the algorithm

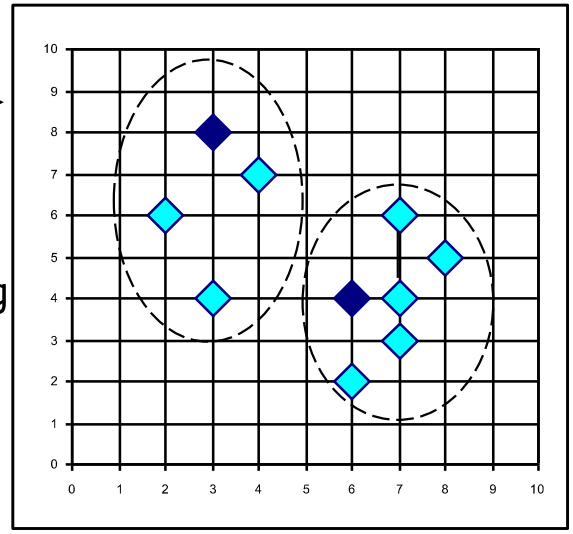


K=2

Arbitrary choose k objects as initial medoids



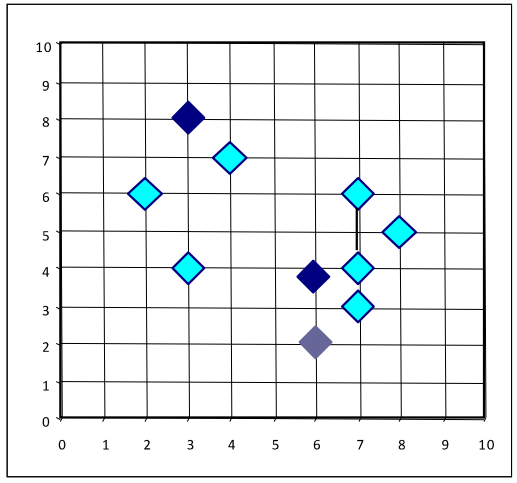
Assign each remaining object to nearest medoids



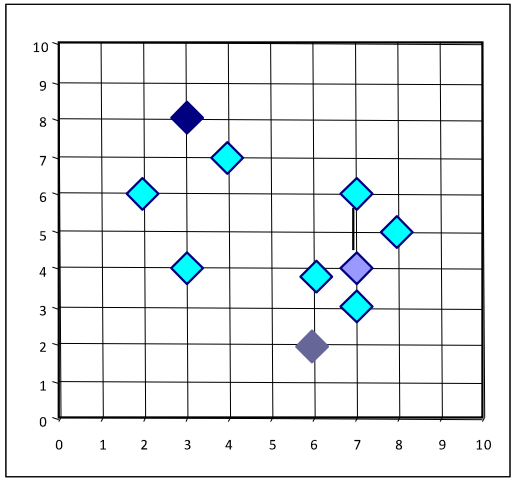
Total Cost = 20

All pairs

Randomly select a nonmedoid object, O_{ramdom}



Compute total cost of swapping



Total Cost = 26

Swapping O and O_{ramdom}

For **best pair**

If quality is improved.

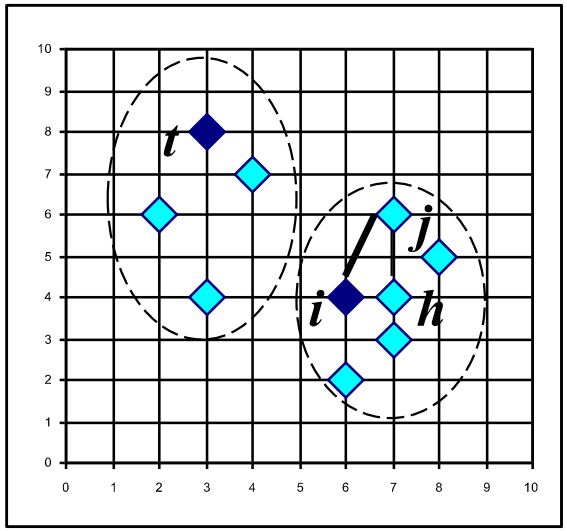
Do loop

Until no change

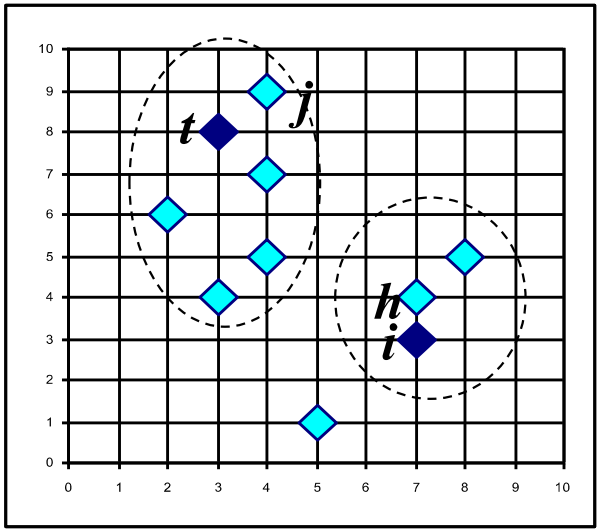
PAM (Partitioning Around Medoids)

- PAM (Kaufman and Rousseeuw, 1987)
- Algorithm
  - Select ***k*** representative objects arbitrarily
  - For each pair of non-selected object ***h*** and selected object ***i***, calculate the total swapping cost TC_{ih}
  - Select a pair ***i*** and ***h***, which corresponds to the minimum swapping cost
    - If $TC_{ih} < 0$, ***i*** is replaced by ***h***
    - Then assign each non-selected object to the most similar representative object
  - repeat steps 2-3 until there is no change

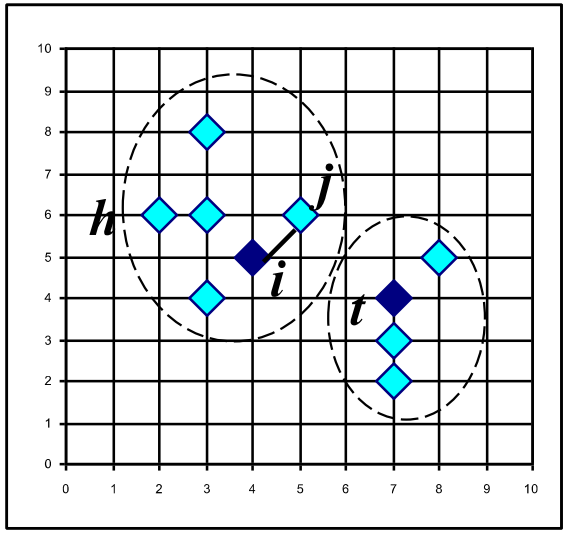
PAM Clustering: Total swapping cost $TC_{ih}=\sum_j C_{jih}$



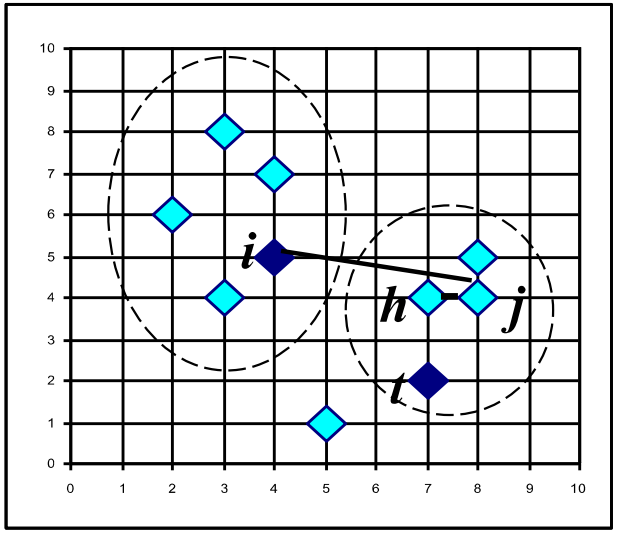
$C_{jih} = d(j, h) - d(j, i)$



$C_{jih} = 0$



$C_{jih} = d(j, t) - d(j, i)$



$C_{jih} = d(j, h) - d(j, t)$

Explanation on whiteboard

Properties of PAM

- Pam is more robust than k-means in the presence of noise and outliers because a medoid is less influenced by outliers or other extreme values than a mean
- Pam works efficiently for small data sets but does not **scale well** for large data sets.
  - $O(k(n-k)^2)$ for each iteration

    where n is # of data,k is # of clusters

➔ Sampling based method,

CLARA(Clustering LARge Applications)

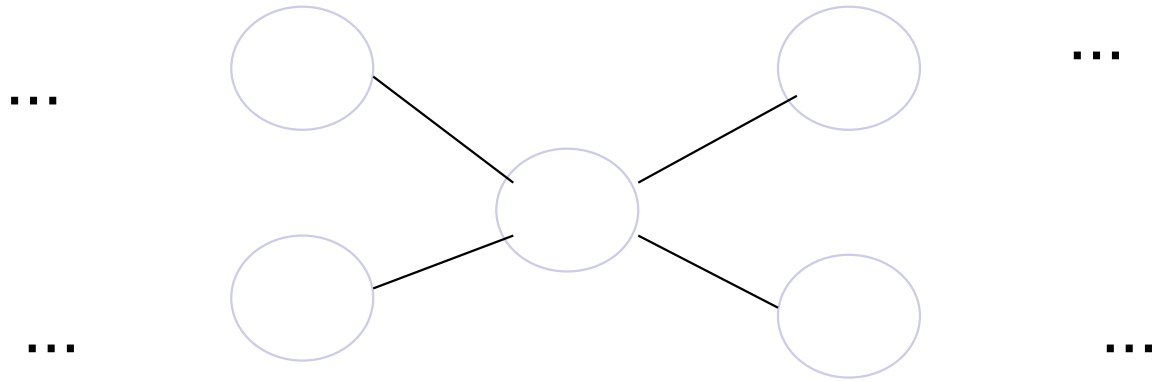
CLARA (Clustering Large Applications)

- *CLARA* (Kaufmann and Rousseeuw,1990)
  - Built in statistical analysis packages, such as S+
- It draws *multiple samples* of the data set, applies *PAM* on each sample, and gives the best clustering as the output
- Strength: deals with larger data sets than *PAM*
- Weakness:
  - Efficiency depends on the sample size
  - A good clustering based on samples will not necessarily represent a good clustering of the whole data set if the sample is biased

CLARA (Clustering Large Applications)

- Algorithm (n=5, s = 40+2k)
  - Repeat n times:
    - Draw sample of s objects from the entire data set and perform PAM to find k mediods of the sample
    - Assign each non-selected object in the entire data set to the most similar mediod
    - Calculate average dissimilarity of the clustering. If the value is smaller than the current minimum, use this value as current minimum and retain the k medoids as best so far.

Graph abstraction



Node represents k objects (medoids), a potential solution for the clustering.

Nodes are neighbors if the sets of objects differ by one object.

Each node has k(n-k) neighbors.

Cost differential between two neighbors is *TCih*

(with Oi and Oh are the differing nodes in the mediod sets)

Example on whiteboard

Example on whiteboard

Example on whiteboard

Graph abstraction

- PAM searches for node in the graph with minimum cost
- CLARA searches in smaller graphs (as it uses PAM on samples of the entire data set)
- CLARANS
  - Searches in the original graph
  - Searches part of the graph
  - Uses the neighbors to guide the search

CLARANS (“Randomized” CLARA)

- *CLARANS* (A Clustering Algorithm based on Randomized Search) (Ng and Han,1994)

- It is more efficient and scalable than both *PAM* and *CLARA*

CLARANS ("Randomized" CLARA)

- Algorithm
  - *Numlocal*: number of local minima to be found
  - *Maxneighbor*: maximum number of neighbors to compare
  - Repeat *numlocal* times: (find local minimum)
    - Take arbitrary node in the graph
    - Consider random neighbor S of the current node and calculate the cost differential. If S has lower cost, then set S to current and repeat this step. If S does not have lower cost, repeat this step (check at most *Maxneighbor* neighbors)
    - Compare the cost of current node with minimum cost so far. If the cost is lower, set minumum cost to cost of the current node, and bestnode to the current node.

Example on whiteboard