CS481/CS583: Bioinformatics Algorithms

Can Alkan
EA509
calkan@cs.bilkent.edu.tr

http://www.cs.bilkent.edu.tr/~calkan/teaching/cs481/

CLUSTERING

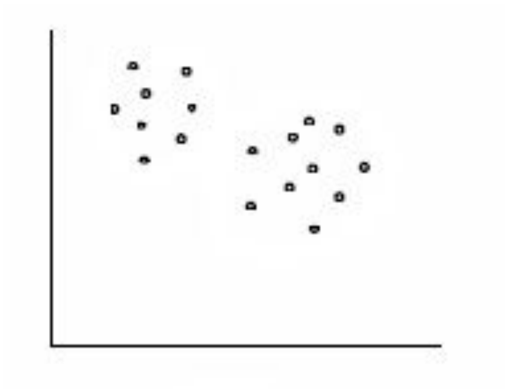
Applications of Clustering

- Viewing and analyzing vast amounts of biological data as a whole set can be infeasible
- It is easier to interpret the data if they are partitioned into clusters combining similar data points.

Homogeneity and Separation Principles

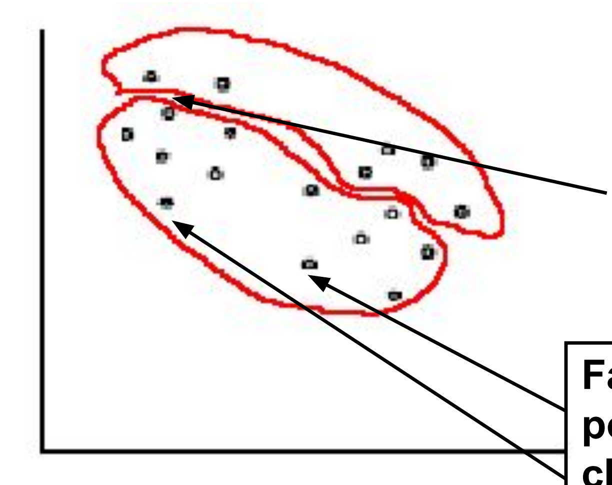
- **Homogeneity:** Elements within a cluster are close to each other
- **Separation:** Elements in different clusters are further apart from each other
- …clustering is not an easy task!

Given these points a clustering algorithm $\longrightarrow$ might make two distinct clusters as follows



Bad Clustering

This clustering violates both Homogeneity and Separation principles

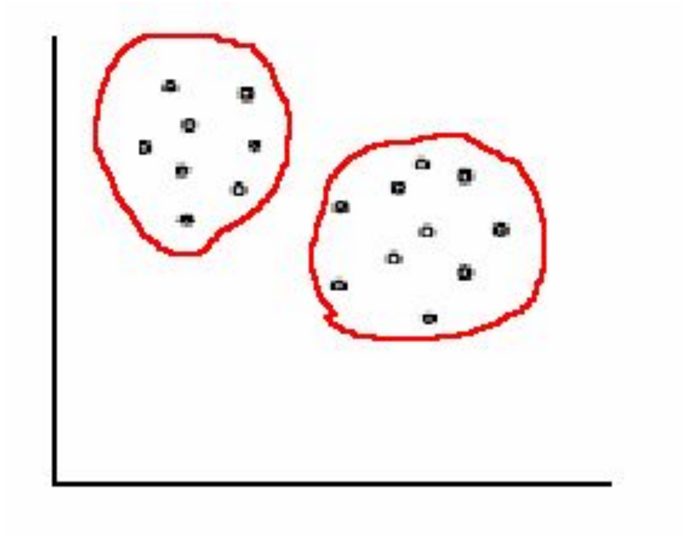


Close distances from points in separate clusters

Far distances from points in the same cluster

Good Clustering

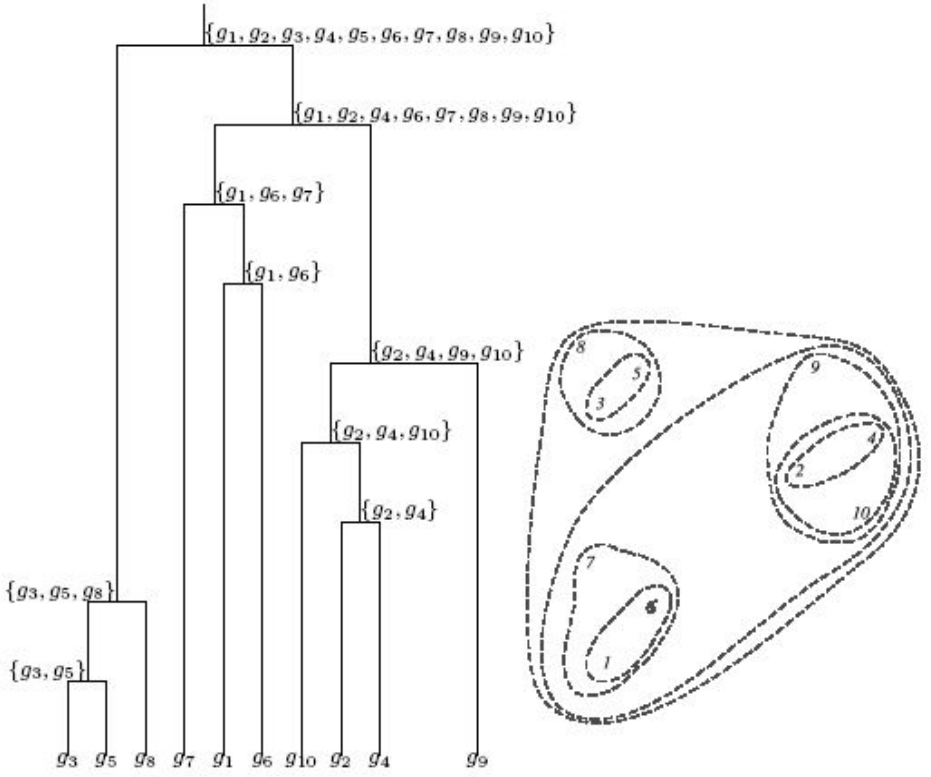
This clustering satisfies both Homogeneity and Separation principles



Clustering Techniques

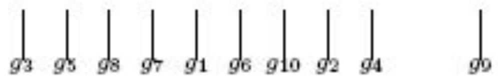
- **Agglomerative:** Start with every element in its own cluster, and iteratively join clusters together
- **Divisive:** Start with one cluster and iteratively divide it into smaller clusters
- **Hierarchical:** Organize elements into a tree, leaves represent data points and the length of the pathes between leaves represents the distances between data points. Similar data points lie within the same subtrees

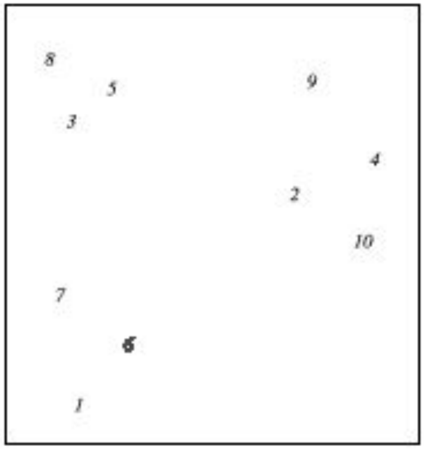
Hierarchical Clustering



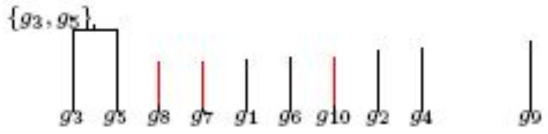
$\{g_1, g_2, g_3, g_4, g_5, g_6, g_7, g_8, g_9, g_{10}\}$
$\{g_1, g_2, g_4, g_6, g_7, g_8, g_9, g_{10}\}$
$\{g_1, g_6, g_7\}$
$\{g_1, g_6\}$
$\{g_2, g_4, g_9, g_{10}\}$
$\{g_2, g_4, g_{10}\}$
$\{g_2, g_4\}$
$\{g_3, g_5, g_8\}$
$\{g_3, g_5\}$
g_3 g_5 g_8 g_7 g_1 g_6 g_{10} g_2 g_4 g_9
8 5 3 9 4 2 10 7 6 1

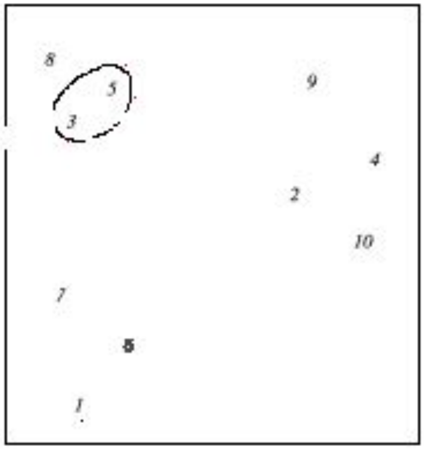
Hierarchical Clustering: Example



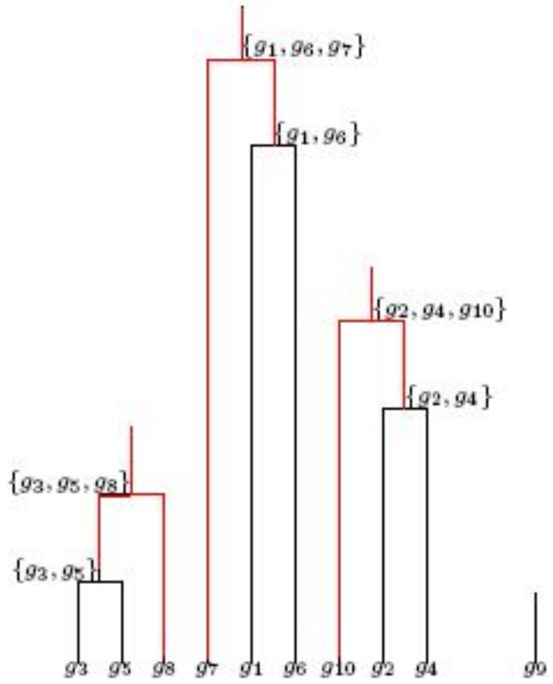


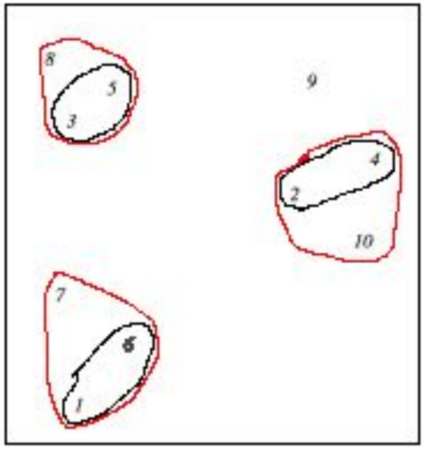
Hierarchical Clustering: Example


$\{g_3, g_5\}$
g_3 g_5 g_8 g_7 g_1 g_6 g_{10} g_2 g_4 g_9

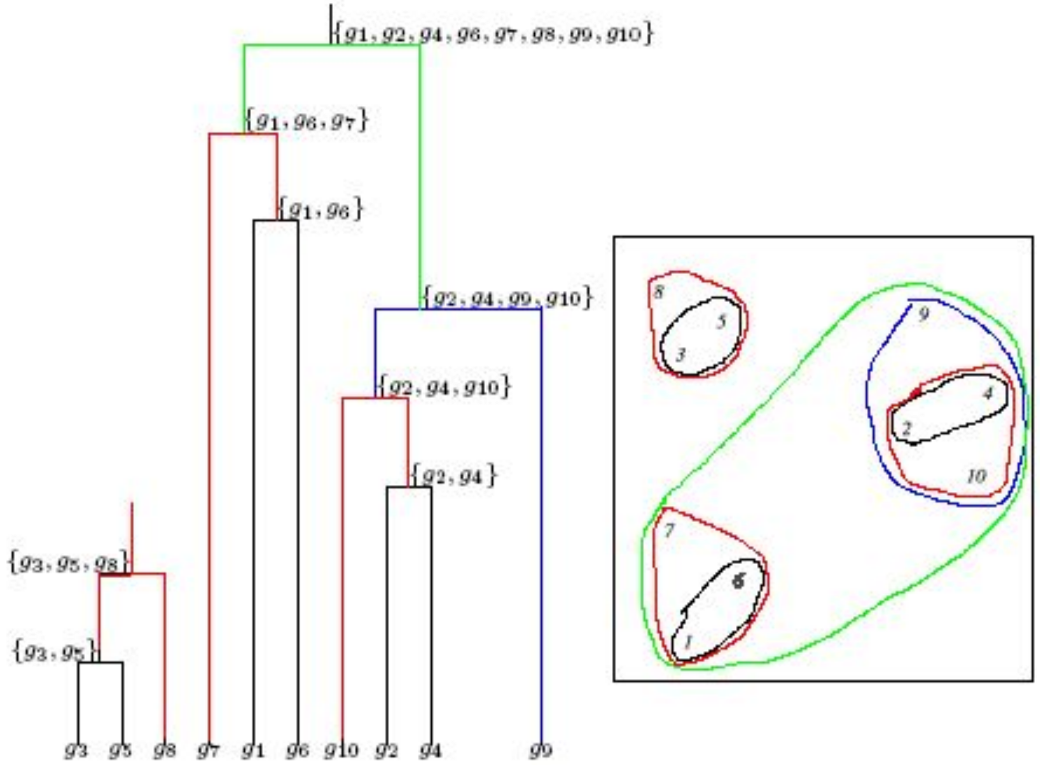


Hierarchical Clustering: Example

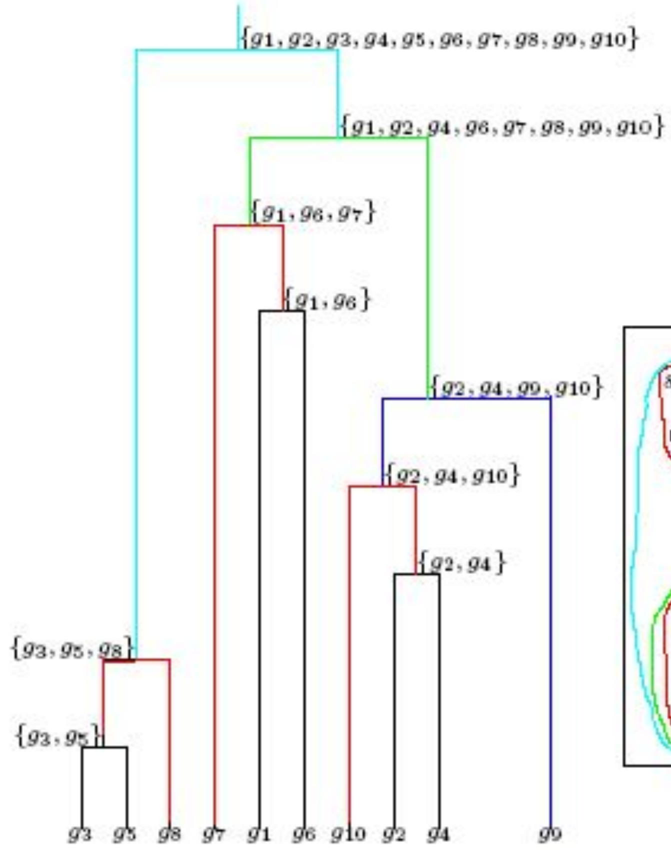


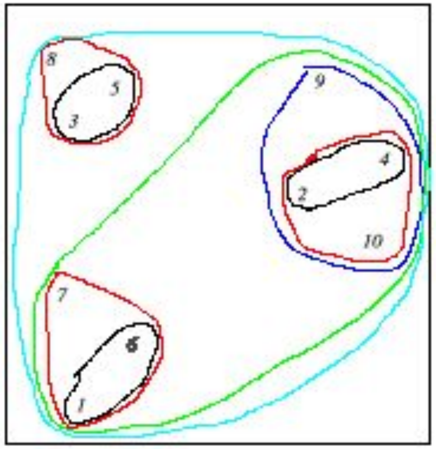


Hierarchical Clustering: Example



Hierarchical Clustering: Example





Hierarchical Clustering Algorithm

```
1.  Hierarchical Clustering ($\boldsymbol{d}$ , $n$)
2.    Form $n$ clusters each with one element
3.    Construct a graph $\boldsymbol{T}$ by assigning one vertex to each cluster
4.    while there is more than one cluster
5.      Find the two closest clusters $C_1$ and $C_2$
6.      Merge $C_1$ and $C_2$ into new cluster $C$ with $|C_1| + |C_2|$ elements
7.      Compute distance from $C$ to all other clusters
8.      Add a new vertex $\boldsymbol{C}$ to $\boldsymbol{T}$ and connect to vertices $C_1$ and $C_2$
9.      Remove rows and columns of $\boldsymbol{d}$ corresponding to $C_1$ and $C_2$
10.     Add a row and column to $\boldsymbol{d}$ corrsponding to the new cluster $\boldsymbol{C}$
11.   return $\boldsymbol{T}$
```

The algorithm takes a *nxn* distance matrix *d* of pairwise distances between points as an input.

Hierarchical Clustering Algorithm

```
1.  Hierarchical Clustering (d , n)
2.    Form n clusters each with one element
3.    Construct a graph T by assigning one vertex to each cluster
4.    while there is more than one cluster
5.      Find the two closest clusters $C_1$ and $C_2$
6.      Merge $C_1$ and $C_2$ into new cluster $C$ with $|C_1| + |C_2|$ elements
7.      Compute distance from C to all other clusters
8.      Add a new vertex C to T and connect to vertices $C_1$ and $C_2$
9.      Remove rows and columns of d corresponding to $C_1$ and $C_2$
10.     Add a row and column to d corrsponding to the new cluster C
11.   return T
```

Different ways to define distances between clusters may lead to different clusterings

Hierarchical Clustering: Recomputing Distances

- $$d_{min}(C, C^{*}) = \min\ d(x,y)$$
  for all elements x in C and y in C^{*}

  - Distance between two clusters is the **smallest** distance between any pair of their elements

- $$d_{avg}(C, C^{*}) = (1 / |C^{*}||C|) \sum d(x,y)$$
  for all elements x in C and y in C^{*}

  - Distance between two clusters is the **average** distance between all pairs of their elements

Squared Error Distortion

- Given a data point ***v*** and a set of points ***X***, define the **distance** from ***v*** to ***X***

  $$d(v, \boldsymbol{X})$$

  as the (Eucledian) distance from ***v*** to the ***closest*** point from ***X***.

- Given a set of *n* data points $V=\{v_1 \ldots v_n\}$ and a set of *k* points ***X***, define the **Squared Error Distortion**

  $$d(\boldsymbol{V},\boldsymbol{X}) = \sum d(v_i, \boldsymbol{X})^2 \;/\; n \qquad 1 \leq i \leq n$$

K-Means Clustering Problem: Formulation

- **Input**: A set, ***V***, consisting of *n* points and a parameter *k*
- **Output**: A set ***X*** consisting of *k* points (*cluster centers*) that minimizes the squared error distortion *d(**V**,**X**)* over all possible choices of ***X***

1-Means Clustering Problem: an Easy Case

- **Input**: A set, ***V***, consisting of *n* points
- **Output**: A single point ***x*** (*cluster center*) that minimizes the squared error distortion *d(**V**,**x**)* over all possible choices of ***x***

1-Means Clustering Problem: an Easy Case

- **Input**: A set, ***V***, consisting of *n* points
- **Output**: A single point ***x*** (cluster center) that minimizes the squared error distortion *d(**V**,**x**)* over all possible choices of ***x***

1-Means Clustering problem is easy.

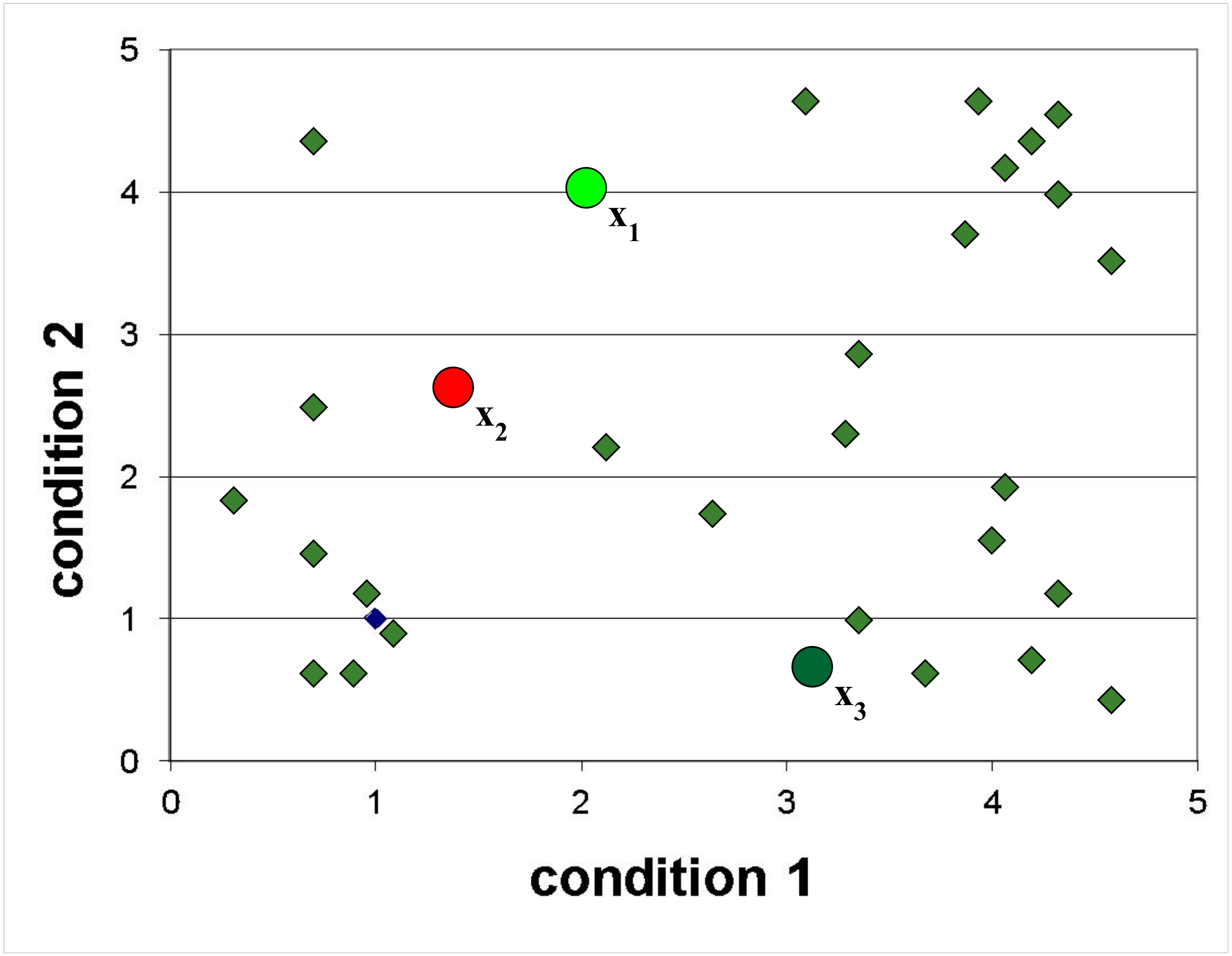
However, it becomes very difficult (NP-complete) for more than one center.

An efficient ***heuristic*** method for K-Means clustering is the Lloyd algorithm

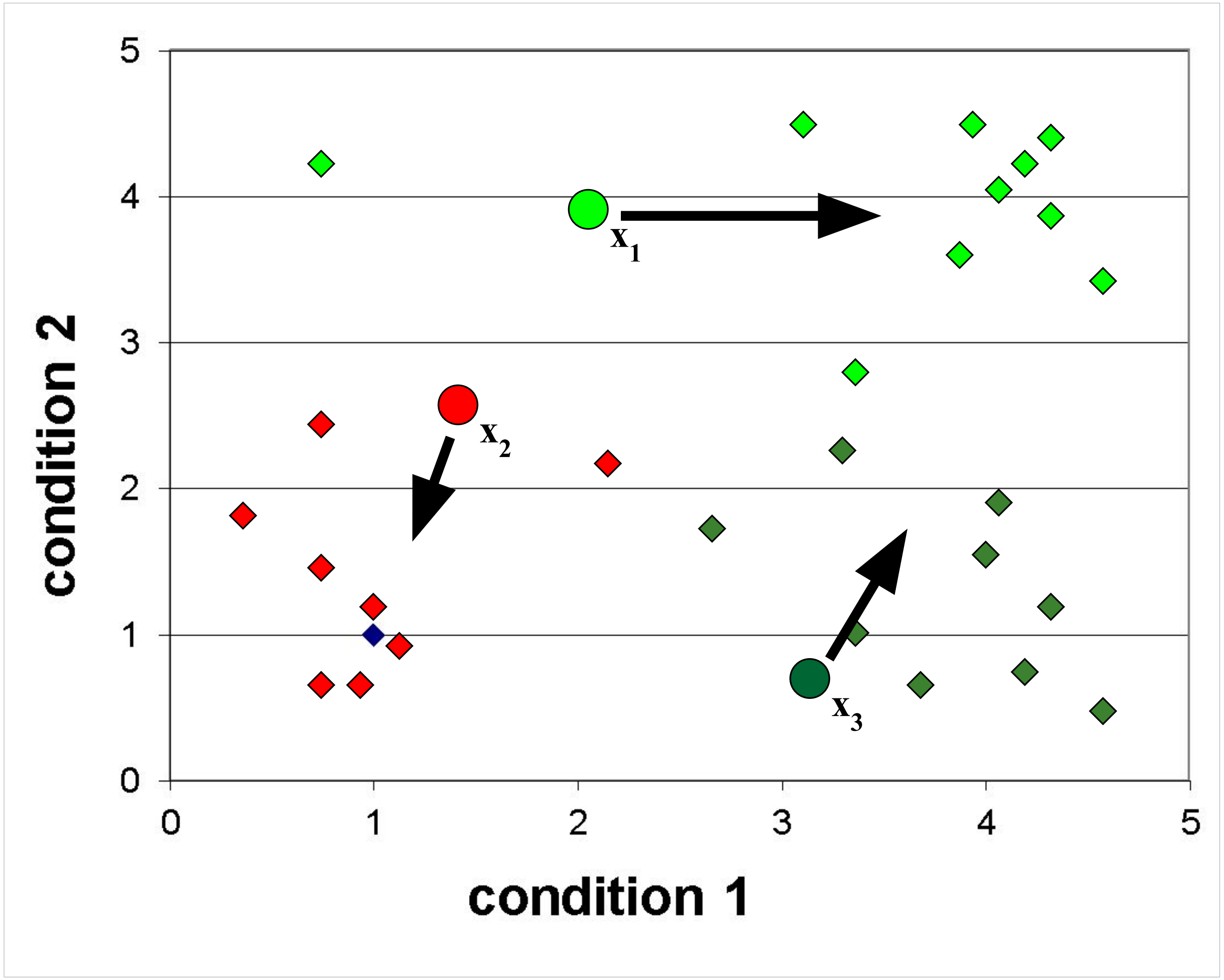
K-Means Clustering: Lloyd Algorithm

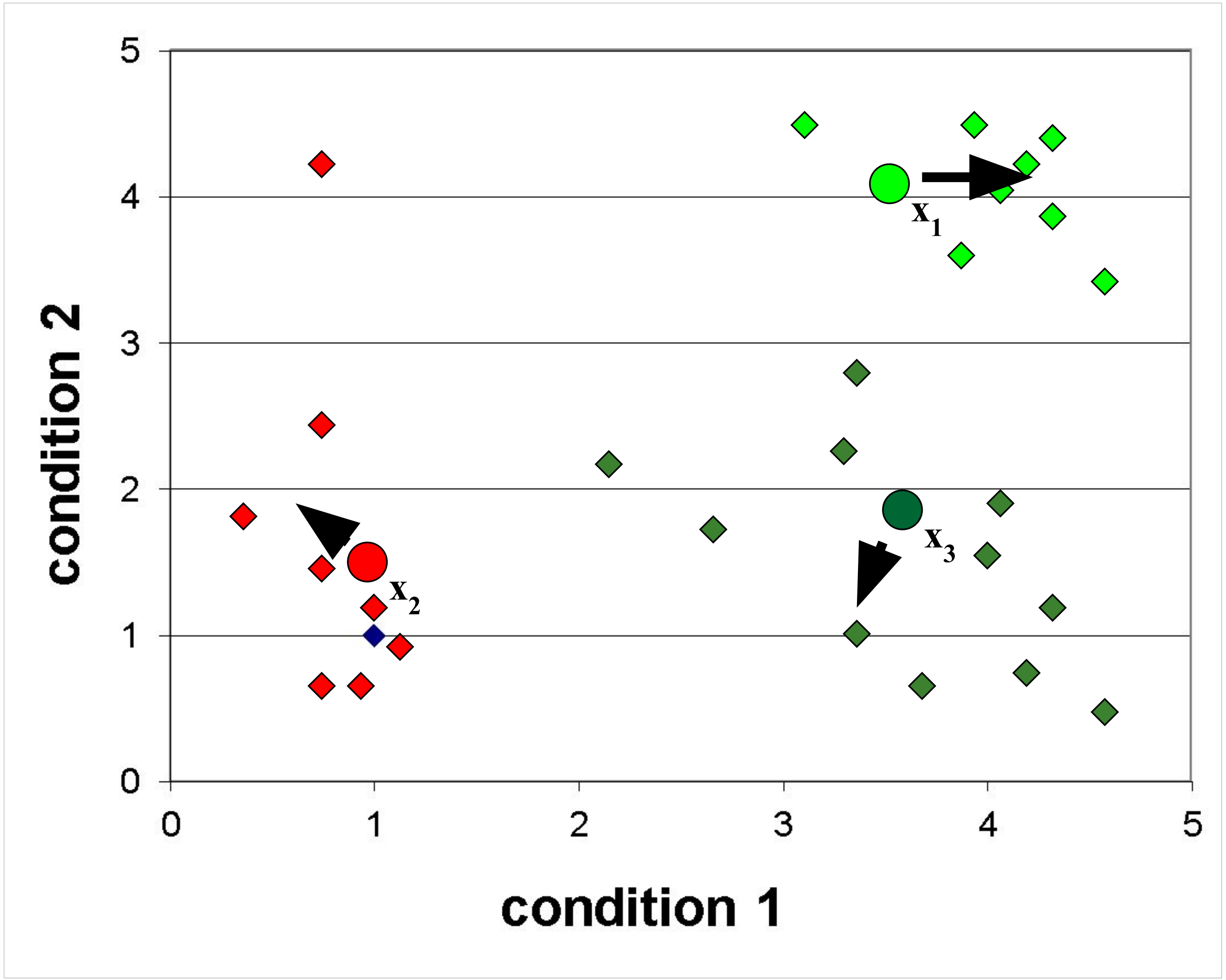
```
1. Lloyd Algorithm
2.   Arbitrarily assign the $k$ cluster centers
3.   **while** the cluster centers keep changing
4.     Assign each data point to the cluster $C_i$ corresponding to the closest cluster representative (center) $(1 \le i \le k)$
5.     After the assignment of all data points, compute new cluster representatives according to the center of gravity of each cluster, that is, the new cluster representative is
           $\Sigma v / |C|$ for all $v$ in $C$   for every cluster $C$
```

*This may lead to merely a locally optimal clustering.

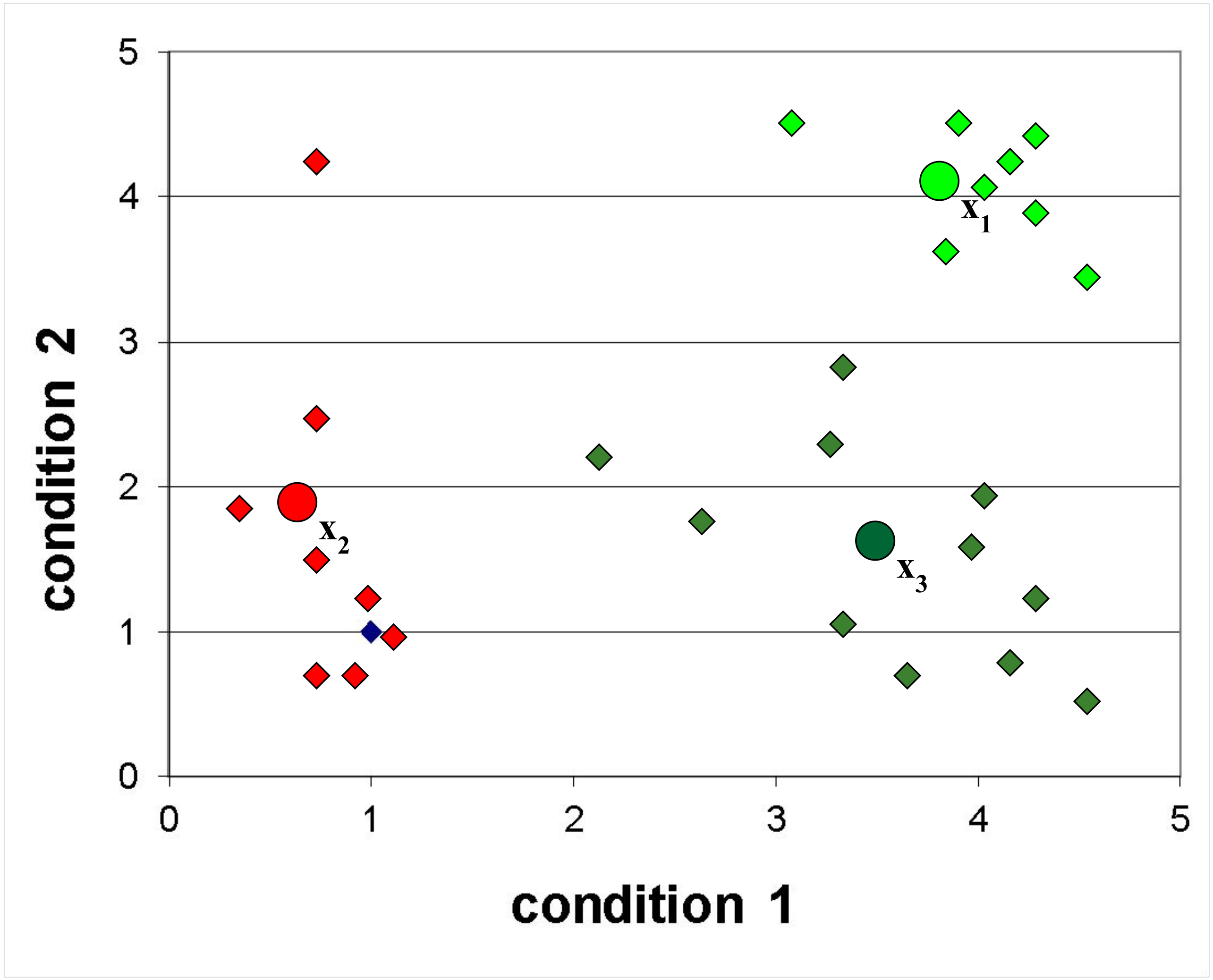


condition 2
condition 1
x_1
x_2
x_3
0
1
2
3
4
5


condition 1
condition 2
0
1
2
3
4
5
x_1
x_2
x_3



5
4
3
2
1
0
0
1
2
3
4
5
condition 2
condition 1
x_1
x_2
x_3



condition 2
condition 1
0
1
2
3
4
5
x_1
x_2
x_3

Conservative K-Means Algorithm

- Lloyd algorithm is fast but in each iteration it moves many data points, not necessarily causing better convergence.
- A more conservative method would be to move one point at a time only if it improves the overall **clustering cost**
  - The smaller the clustering cost of a partition of data points is the better that clustering is
  - Different methods (e.g., the squared error distortion) can be used to measure this clustering cost

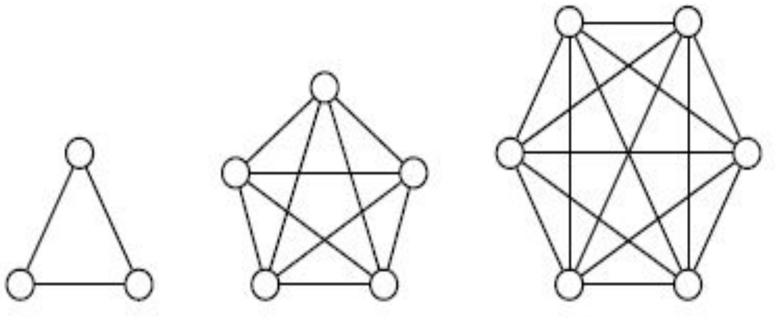
K-Means Greedy Algorithm

```
ProgressiveGreedyK-Means($k$)
Select an arbitrary partition $P$ into $k$ clusters
while forever
  $bestChange$ □ 0
  for every cluster $C$
    for every element $i$ not in $C$
      if moving $i$ to cluster $C$ reduces its clustering cost
        if (cost($P$) – cost($P_{i □ C}$) > $bestChange$
          $bestChange$ □ cost($P$) – cost($P_{i □ C}$)
          $i^*$ □ $I$
          $C^*$ □ $C$
  if $bestChange$ > 0
    Change partition $P$ by moving $i^*$ to $C^*$
  else
    return $P$
```

CLUSTERING USING GRAPHS

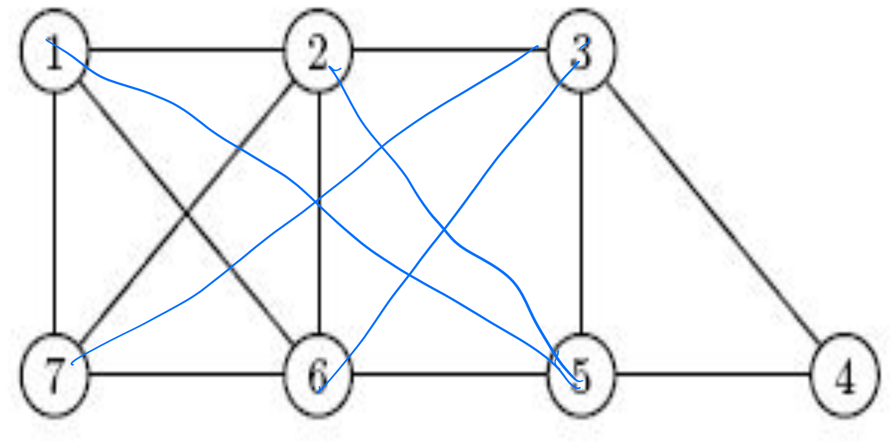
Clique Graphs

- A **clique** is a graph with every vertex connected to every other vertex
- A **clique graph** is a graph where each connected component is a clique



Transforming an Arbitrary Graph into a Clique Graphs

- **A graph can be transformed into a clique graph by adding or removing edges**



Corrupted Cliques Problem

Input: A graph *G*

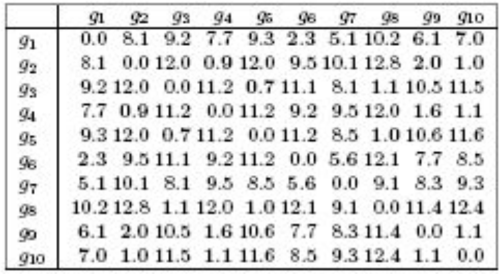
Output: The smallest number of additions and removals of edges that will transform *G* into a clique graph

Distance Graphs

- Turn the distance matrix into a distance graph
  - Genes are represented as vertices in the graph
  - Choose a distance threshold θ
  - If the distance between two vertices is below θ, draw an edge between them
  - The resulting graph may contain cliques
  - These cliques represent clusters of closely located data points

Transforming Distance Graph into Clique Graph

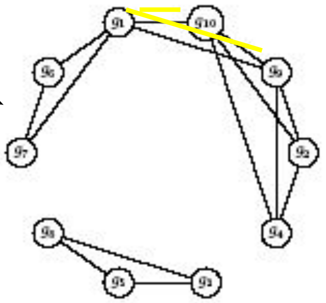
The distance graph (threshold *θ*=7) is transformed into a clique graph after removing the two highlighted edges



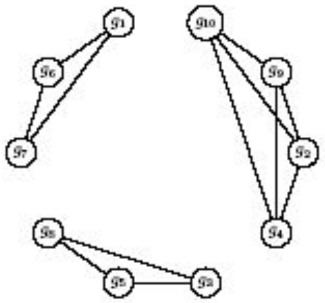
	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_{10}
g_1	0.0	8.1	9.2	7.7	9.3	2.3	5.1	10.2	6.1	7.0
g_2	8.1	0.0	12.0	0.9	12.0	9.5	10.1	12.8	2.0	1.0
g_3	9.2	12.0	0.0	11.2	0.7	11.1	8.1	1.1	10.5	11.5
g_4	7.7	0.9	11.2	0.0	11.2	9.2	9.5	12.0	1.6	1.1
g_5	9.3	12.0	0.7	11.2	0.0	11.2	8.5	1.0	10.6	11.6
g_6	2.3	9.5	11.1	9.2	11.2	0.0	5.6	12.1	7.7	8.5
g_7	5.1	10.1	8.1	9.5	8.5	5.6	0.0	9.1	8.3	9.3
g_8	10.2	12.8	1.1	12.0	1.0	12.1	9.1	0.0	11.4	12.4
g_9	6.1	2.0	10.5	1.6	10.6	7.7	8.3	11.4	0.0	1.1
g_{10}	7.0	1.0	11.5	1.1	11.6	8.5	9.3	12.4	1.1	0.0

(a) Distance matrix, d (distances shorter than 7 are shown in bold).

After transforming the distance graph into the clique graph, the dataset is partitioned into three clusters



(b) Distance graph for $\theta = 7$.



(c) Clique graph.

Figure 10.6 The distance graph (b) for $\theta = 7$ is not quite a clique graph. However, it can be transformed into a clique graph (c) by removing edges (g_1, g_{10}) and (g_1, g_9).

Heuristics for Corrupted Clique Problem

- Corrupted Cliques problem is NP-Hard, some heuristics exist to approximately solve it:
- **CAST** (Cluster Affinity Search Technique): a practical and fast algorithm:
  - **CAST** is based on the notion of genes *close* to cluster *C* or *distant* from cluster *C*
  - Distance between gene *i* and cluster *C*:

d(i,C) = average distance between gene *i* and all genes in *C*

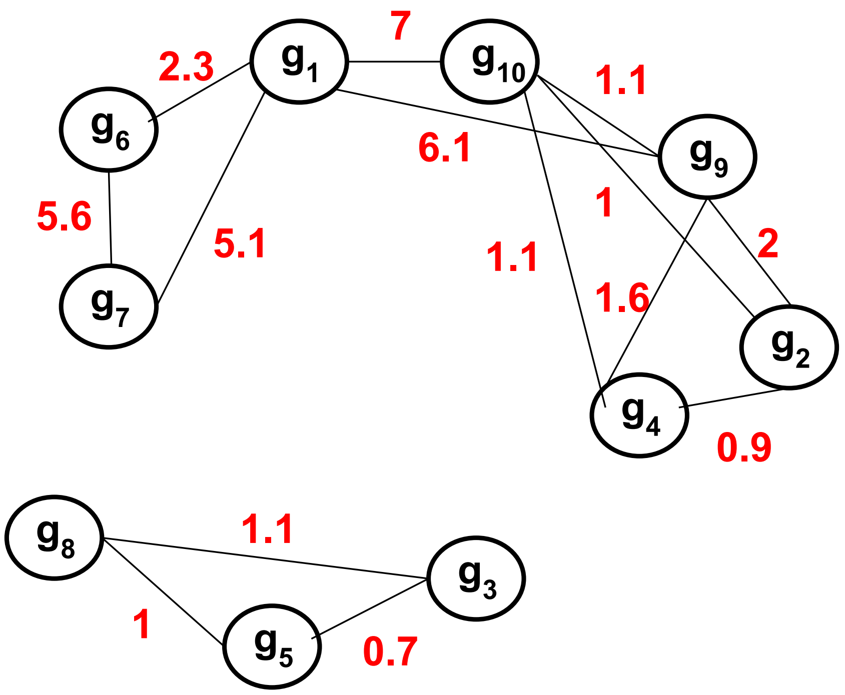
Gene *i* is ***close*** to cluster *C* if $d(i,C) < \theta$ and ***distant*** otherwise

CAST Algorithm

```
CAST($S, G, \theta$)
  $P$ □ Ø
  while $S \neq$ Ø
    $V$ □ vertex of maximal degree in the distance graph $G$
    $C$ □ {$v$}
    while a close gene $i$ not in $C$ or distant gene $i$ in $C$ exists
      Find the nearest close gene $i$ not in $C$ and add it to $C$
      Remove the farthest distant gene $i$ in $C$
    Add cluster $C$ to partition $P$
    $S$ □ $S \setminus C$
    Remove vertices of cluster $C$ from the distance graph $G$
  return $P$
```

***S* – set of elements, *G* – distance graph, θ - distance threshold**

CAST Algorithm



$\Theta = 7$
P = Ø
$S=\{g_1,\ldots,g_{10}\}$
$\text{degree}(g_{10}) = 4$

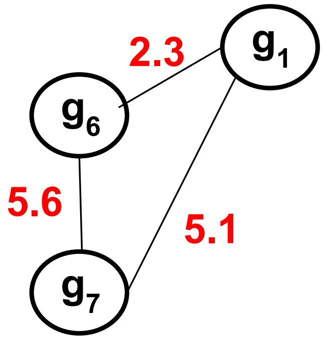
$C_1 = \{g_{10}\}$
$C_1 = \{g_2, g_{10}\}$

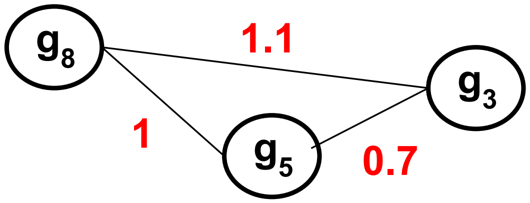
$d(g_1, C_1) = (7+8.1) / 2 = 7.55$
$d(g_4, C_1) = (0.9+1.1) / 2 = 1$
$d(g_9, C_1) = (2+1.1) / 2 = 1.55$

$C_1 = \{g_2, g_4, g_{10}\}$
$d(g_9,C) = (2+1.6+1) / 3 = 1.53$

$C_1 = \{g_2, g_4, g_9, g_{10}\}$
$P = \{C_1\}$

CAST Algorithm





$\Theta = 7$
$P = \{C_1\}$
$C_1 = \{g_2, g_4, g_9, g_{10}\}$
$S=\{g_1,g_3,g_5, g_6,g_7, g_8\}$
$degree(g_1) = 2$

$C_2 = \{g_1\}$
$C_2 = \{g_1, g_6\}$

$d(g_7, C_2) = (5.1+5.6) / 2 = 5.35$

$C_2 = \{g_1, g_6, g_7\}$

$P = \{C_1, C_2\}$

CAST Algorithm

$\Theta = 7$

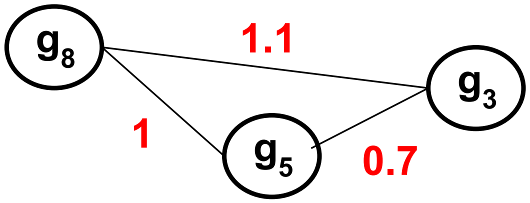
$P = \{C_1, C_2\}$

$C_1 = \{g_2, g_4, g_9, g_{10}\}$

$C_2 = \{g_1, g_6, g_7\}$

$S=\{g_3,g_5, g_8\}$

$\text{degree}(g_3) = 2$



$C_3 = \{g_3\}$

$C_3 = \{g_3, g_5\}$

$d(g_8, C_3) = (1.1+1) / 2 = 1.05$

$C_3 = \{g_3, g_5, g_8\}$

$P = \{C_1, C_2, C_3\}$

CAST Algorithm

$\Theta = 7$
$P = \{C_1, C_2, C_3\}$
$C_1 = \{g_2, g_4, g_9, g_{10}\}$
$C_2 = \{g_1, g_6, g_7\}$
$C_3 = \{g_3, g_5, g_8\}$
$S = \emptyset$

… done