

HIERARCHICAL CLUSTERING ALGORITHMS

- ❖ They produce a **hierarchy** of (**hard**) clusterings instead of a **single** clustering.

- ❖ Applications in:
 - Social sciences
 - Biological taxonomy
 - Modern biology
 - Medicine
 - Archaeology
 - Computer science and engineering

- ❖ Let $X = \{\underline{x}_1, \dots, \underline{x}_N\}$, $\underline{x}_i = [x_{i1}, \dots, x_{il}]^T$. Recall that:
 - In hard clustering each vector belongs **exclusively** to a single cluster.
 - An m -(hard) clustering of X , \mathfrak{R} , is a partition of X into m sets (clusters) C_1, \dots, C_m , so that:

$$C_i \neq \emptyset, i = 1, 2, \dots, m$$

$$\bigcup_{i=1}^m C_i = X$$

$$C_i \cap C_j = \emptyset, i \neq j, i, j = 1, 2, \dots, m$$

By the definition: $\mathfrak{R} = \{C_j, j = 1, \dots, m\}$

- **Definition:** A clustering \mathfrak{R}_1 containing k clusters is said to be **nested** in the clustering \mathfrak{R}_2 containing r ($< k$) clusters, if **each** cluster in \mathfrak{R}_1 is a subset of a cluster in \mathfrak{R}_2 .

We write $\mathfrak{R}_1 \sqsubset \mathfrak{R}_2$

► **Example:** Let $\mathcal{R}_1 = \{ \{x_1, x_3\}, \{x_4\}, \{x_2, x_5\} \}$, $\mathcal{R}_2 = \{ \{x_1, x_3, x_4\}, \{x_2, x_5\} \}$,

$$\mathcal{R}_3 = \{ \{x_1, x_4\}, \{x_3\}, \{x_2, x_5\} \}, \mathcal{R}_4 = \{ \{x_1, x_2, x_4\}, \{x_3, x_5\} \}.$$

It is $\mathcal{R}_1 \sqsubset \mathcal{R}_2$, **but not** $\mathcal{R}_1 \sqsubset \mathcal{R}_3$, $\mathcal{R}_1 \sqsubset \mathcal{R}_4$, $\mathcal{R}_1 \sqsubset \mathcal{R}_1$.

► **Remarks:**

- Hierarchical clustering algorithms produce a **hierarchy of nested clusterings**.
- They involve N steps at the most.
- At each step t , the clustering \mathcal{R}_t is produced by \mathcal{R}_{t-1} .

► **Main categories:**

- **Agglomerative** clustering algorithms: Here $\mathcal{R}_0 = \{ \{x_1\}, \dots, \{x_N\} \}$, $\mathcal{R}_{N-1} = \{ \{x_1, \dots, x_N\} \}$ and $\mathcal{R}_0 \sqsubset \dots \sqsubset \mathcal{R}_{N-1}$.
- **Divisive** clustering algorithms: Here $\mathcal{R}_0 = \{ \{x_1, \dots, x_N\} \}$, $\mathcal{R}_{N-1} = \{ \{x_1\}, \dots, \{x_N\} \}$ and $\mathcal{R}_{N-1} \sqsubset \dots \sqsubset \mathcal{R}_0$.

AGGLOMERATIVE ALGORITHMS

❖ Let $g(C_i, C_j)$ a proximity function between two clusters of X .

❖ Generalized Agglomerative Scheme (GAS)

► Initialization

- Choose $\mathcal{R}_0 = \{ \{ \underline{x}_1 \}, \dots, \{ \underline{x}_N \} \}$
- $t = 0$

► Repeat

- $t = t + 1$
- Choose (C_i, C_j) in \mathcal{R}_{t-1} such that

$$g(C_i, C_j) = \begin{cases} \min_{r,s} g(C_r, C_s), & \text{if } g \text{ is a dissimilarity function} \\ \max_{r,s} g(C_r, C_s), & \text{if } g \text{ is a similarity function} \end{cases}$$

- Define $C_q = C_i \cup C_j$ and produce $\mathcal{R}_t = (\mathcal{R}_{t-1} - \{C_i, C_j\}) \cup \{C_q\}$

► Until all vectors lie in a single cluster.

► Remarks:

- If two vectors come together into a single cluster at level t of the hierarchy, they will remain in the same cluster for all subsequent clusterings. As a consequence, there **is no way** to recover a “poor” clustering that may have occurred in an earlier level of hierarchy.
- Number of operations: $O(N^3)$

At each level t , there are $N - t$ clusters. Thus, in order to determine the pair of clusters that is going to be merged at the $t + 1$ level, $\binom{N-t}{2} \equiv \frac{(N-t)(N-t-1)}{2}$ pairs of clusters have to be considered. Thus, the total number of pairs that have to be examined throughout the whole clustering process is

$$\sum_{t=0}^{N-1} \binom{N-t}{2} = \sum_{k=1}^N \binom{k}{2} = \frac{(N-1)N(N+1)}{6}$$

that is, the total number of operations required by an agglomerative scheme is proportional to N^3 . However, the exact complexity of the algorithm depends on the definition of g .

❖ Definitions of some useful quantities:

Let $X=\{\underline{x}_1,\underline{x}_2,\dots,\underline{x}_N\}$, with $\underline{x}_i=[x_{i1},x_{i2},\dots,x_{il}]^T$.

- **Pattern matrix** ($D(X)$): An $N\times l$ matrix whose i -th row is \underline{x}_i (transposed).
- **Proximity (similarity or dissimilarity) matrix** ($P(X)$): An $N\times N$ matrix whose (i,j) element equals the proximity $\wp(\underline{x}_i,\underline{x}_j)$ (similarity $s(\underline{x}_i,\underline{x}_j)$, dissimilarity $d(\underline{x}_i,\underline{x}_j)$).

➤ **Example 1:** Let $X=\{\underline{x}_1, \underline{x}_2, \underline{x}_3, \underline{x}_4, \underline{x}_5\}$, with $\underline{x}_1=[1, 1]^T, \underline{x}_2=[2, 1]^T, \underline{x}_3=[5, 4]^T, \underline{x}_4=[6, 5]^T, \underline{x}_5=[6.5, 6]^T$.

	Euclidean distance	Tanimoto distance
$D(X)=\begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 5 & 4 \\ 6 & 5 \\ 6.5 & 6 \end{bmatrix}$	$P(X)=\begin{bmatrix} 0 & 1 & 5 & 6.4 & 7.4 \\ 1 & 0 & 4.2 & 5.7 & 6.7 \\ 5 & 4.2 & 0 & 1.4 & 2.5 \\ 6.4 & 5.7 & 1.4 & 0 & 1.1 \\ 7.4 & 6.7 & 2.5 & 1.1 & 0 \end{bmatrix}$	$P'(X)=\begin{bmatrix} 1 & 0.75 & 0.26 & 0.21 & 0.18 \\ 0.75 & 1 & 0.44 & 0.35 & 0.20 \\ 0.26 & 0.44 & 1 & 0.96 & 0.90 \\ 0.21 & 0.35 & 0.96 & 1 & 0.98 \\ 0.18 & 0.20 & 0.90 & 0.98 & 1 \end{bmatrix}$

- **Threshold dendrogram** (or **dendrogram**): It is an effective way of representing the sequence of clusterings which are produced by an agglomerative algorithm.

In the previous example, if $d_{\min}^{ss}(C_i, C_j)$ is employed as the distance measure between two sets and the Euclidean one as the distance measure between two vectors, the following series of clusterings are produced:

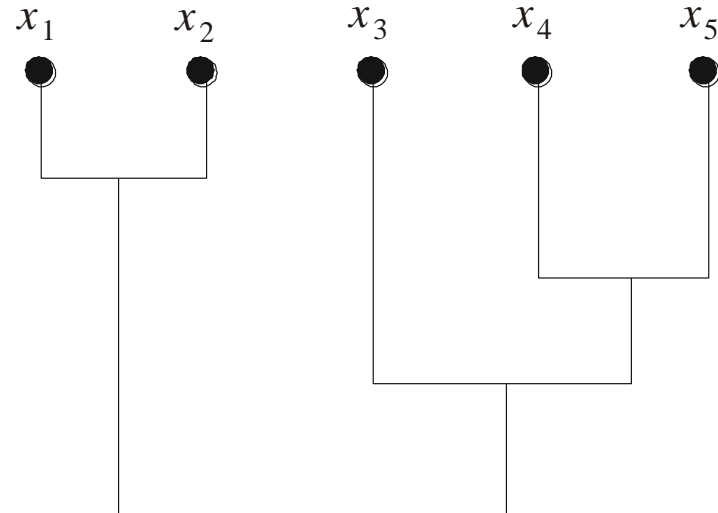
$\{\{x_1\}, \{x_2\}, \{x_3\}, \{x_4\}, \{x_5\}\}$

$\{\{x_1, x_2\}, \{x_3\}, \{x_4\}, \{x_5\}\}$

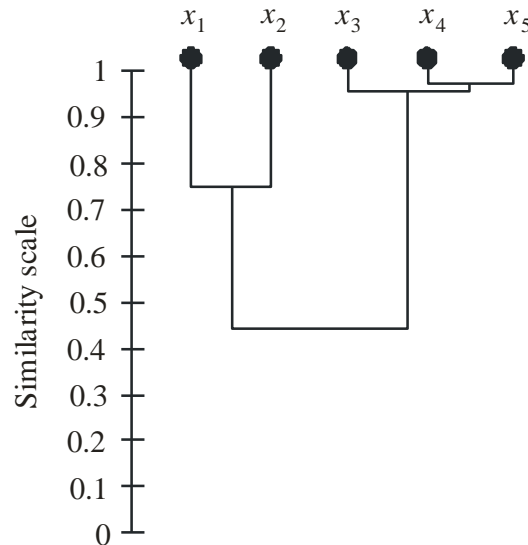
$\{\{x_1, x_2\}, \{x_3\}, \{x_4, x_5\}\}$

$\{\{x_1, x_2\}, \{x_3, x_4, x_5\}\}$

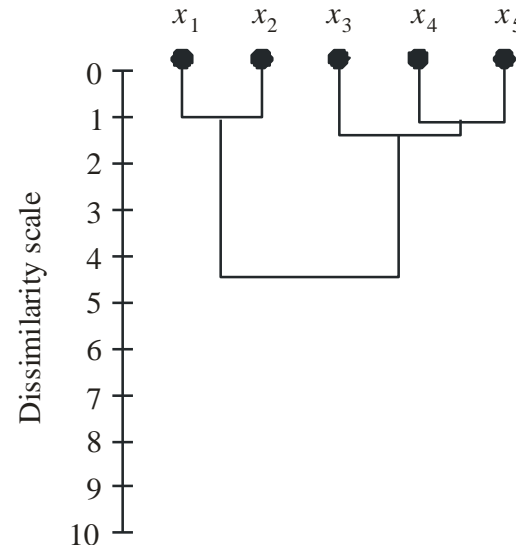
$\{\{x_1, x_2, x_3, x_4, x_5\}\}$



- **Proximity** (dissimilarity or dissimilarity) **dendrogram**: A dendrogram that takes into account the level of proximity (dissimilarity or similarity) where two clusters are **merged** for the first time.
- **Example 2**: In terms of the previous example, the proximity dendrograms that correspond to $P'(X)$ and $P(X)$ are



(a)



(b)

- **Remark**: One can readily observe the level in which a cluster is formed and the level in which it is absorbed in a larger cluster (indication of the natural clustering).

❖ Agglomerative algorithms are divided into:

- Algorithms based on **matrix theory**.
- Algorithms based on **graph theory**.

In the sequel we focus only on **dissimilarity measures**.

➤ **Algorithms based on matrix theory.**

- They take as input the $N \times N$ dissimilarity matrix $P_0 = P(X)$.
- At each level t where two clusters C_i and C_j are merged to C_q , the dissimilarity matrix P_t is extracted from P_{t-1} by:
 - **Deleting** the two rows and columns of P_t that correspond to C_i and C_j .
 - **Adding** a new row and a new column that contain the distances of newly formed $C_q = C_i \cup C_j$ from the remaining clusters C_s , via a relation of the form

$$d(C_q, C_s) = f(d(C_i, C_s), d(C_j, C_s), d(C_i, C_j))$$

- A number of distance functions comply with the following update equation

$$d(C_q, C_s) = a_i d(C_i, C_s) + a_j d(C_j, C_s) + b d(C_i, C_j) + c |d(C_i, C_s) - d(C_j, C_s)|$$

Algorithms that follow the above equation are:

- **Single link (SL) algorithm** ($a_i = 1/2, a_j = 1/2, b = 0, c = -1/2$). In this case

$$d(C_q, C_s) = \min\{d(C_i, C_s), d(C_j, C_s)\}$$

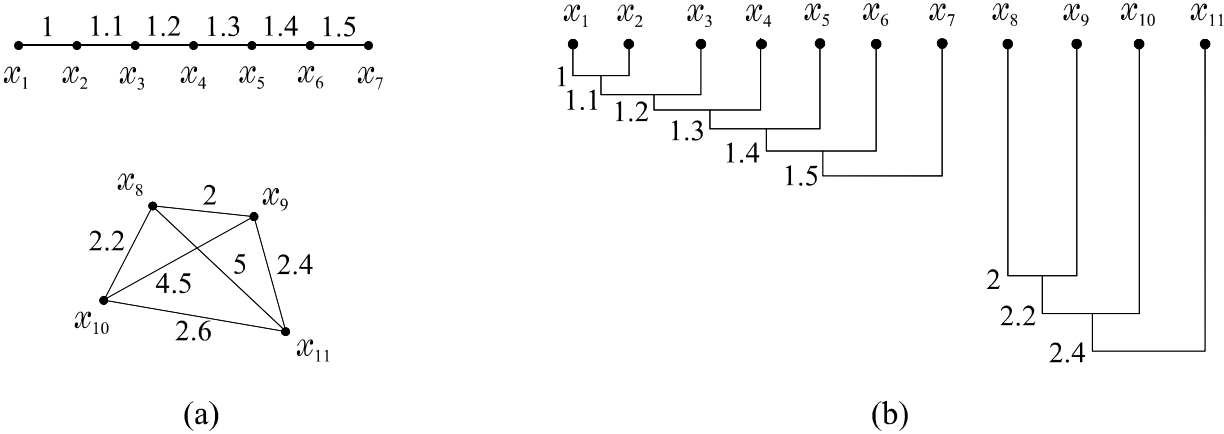
- **Complete link (CL) algorithm** ($a_i = 1/2, a_j = 1/2, b = 0, c = 1/2$). In this case

$$d(C_q, C_s) = \max\{d(C_i, C_s), d(C_j, C_s)\}$$

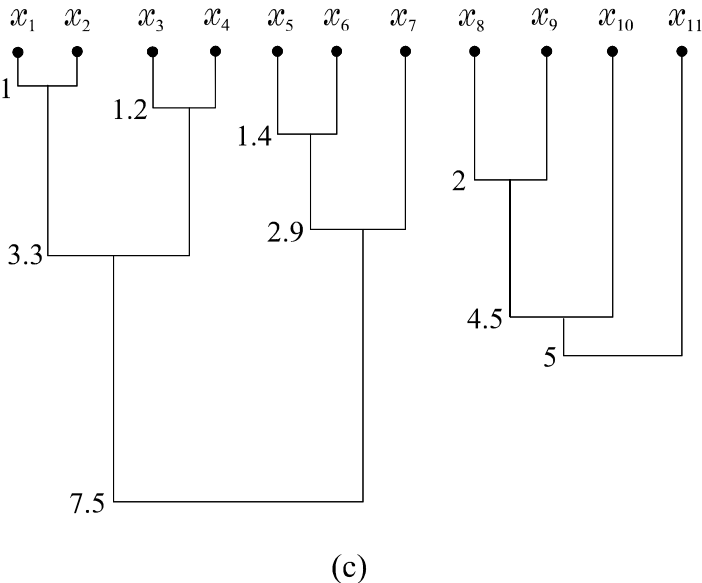
➤ **Remarks:**

- **Single link** forms clusters at **low dissimilarities** while **complete link** forms clusters at **high dissimilarities**.
- **Single link** tends to form **elongated clusters** (*chaining effect*) while **complete link** tends to form **compact clusters**.
- The rest algorithms are compromises between these two extremes.

➤ Example:



- (a) The data set X .
- (b) The single link algorithm dissimilarity dendrogram.
- (c) The complete link algorithm dissimilarity dendrogram



- **Weighted Pair Group Method Average (WPGMA)** ($a_i=1/2, a_j=1/2, b=0, c=0$).
In this case:

$$d(C_q, C_s) = (d(C_i, C_s) + d(C_j, C_s)) / 2$$

- **Unweighted Pair Group Method Average (UPGMA)** ($a_i=n_i/(n_i+n_j), a_j=n_j/(n_i+n_j), b=0, c=0$, where n_i is the cardinality of C_i). In this case:

$$d(C_q, C_s) = (n_i d(C_i, C_s) + n_j d(C_j, C_s)) / (n_i + n_j)$$

- **Unweighted Pair Group Method Centroid (UPGMC)** ($a_i=n_i/(n_i+n_j), a_j=n_j/(n_i+n_j), b=-n_i n_j / (n_i + n_j)^2, c=0$). In this case:

$$d_{qs} = \frac{n_i}{n_i + n_j} d_{is} + \frac{n_j}{n_i + n_j} d_{js} - \frac{n_i n_j}{(n_i + n_j)^2} d_{ij}$$

For the UPGMC, it is true that $d_{qs} = \|\underline{m}_q - \underline{m}_s\|^2$, where \underline{m}_q is the mean of C_q .

- **Weighted Pair Group Method Centroid (WPGMC)** ($a_i=1/2$, $a_j=1/2$, $b=-1/4$, $c=0$). In this case

$$d_{qs} = (d_{is} + d_{js})/2 - d_{ij}/4$$

For WPGMC there are cases where $d_{qs} \leq \max\{d_{is}, d_{js}\}$ (**crossover**)

- **Ward or minimum variance algorithm.** Here the distance d'_{ij} between C_i and C_j is defined as

$$d'_{ij} = (n_i n_j / (n_i + n_j)) \| \underline{m}_i - \underline{m}_j \|^2$$

d'_{qs} can also be written as

$$d'_{qs} = ((n_i + n_j)d'_{is} + (n_i + n_j)d'_{js} - n_s d'_{ij}) / (n_i + n_j + n_s)$$

- **Remark:** Ward's algorithm forms \mathcal{R}_{t+1} by merging the two clusters that **lead to the smallest possible increase of the total variance, i.e.,**

$$E_t = \sum_{r=1}^{N-t} \sum_{\underline{x} \in C_r} \| \underline{x} - \underline{m}_r \|^2$$

➤ **Example 3:** Consider the following dissimilarity matrix (Euclidean distance)

$$P_0 = \begin{bmatrix} 0 & 1 & 2 & 26 & 37 \\ 1 & 0 & 3 & 25 & 36 \\ 2 & 3 & 0 & 16 & 25 \\ 26 & 25 & 16 & 0 & 1.5 \\ 37 & 36 & 25 & 1.5 & 0 \end{bmatrix}$$

$$\begin{aligned} \mathcal{R}_0 &= \{ \{ \underline{x}_1 \}, \{ \underline{x}_2 \}, \{ \underline{x}_3 \}, \{ \underline{x}_4 \}, \{ \underline{x}_5 \} \}, \\ \mathcal{R}_1 &= \{ \{ \underline{x}_1, \underline{x}_2 \}, \{ \underline{x}_3 \}, \{ \underline{x}_4 \}, \{ \underline{x}_5 \} \}, \\ \mathcal{R}_2 &= \{ \{ \underline{x}_1, \underline{x}_2 \}, \{ \underline{x}_3 \}, \{ \underline{x}_4, \underline{x}_5 \} \}, \\ \mathcal{R}_3 &= \{ \{ \underline{x}_1, \underline{x}_2, \underline{x}_3 \}, \{ \underline{x}_4, \underline{x}_5 \} \}, \\ \mathcal{R}_4 &= \{ \{ \underline{x}_1, \underline{x}_2, \underline{x}_3, \underline{x}_4, \underline{x}_5 \} \} \end{aligned}$$

All the algorithms produce the above sequence of clusterings at **different** proximity levels:

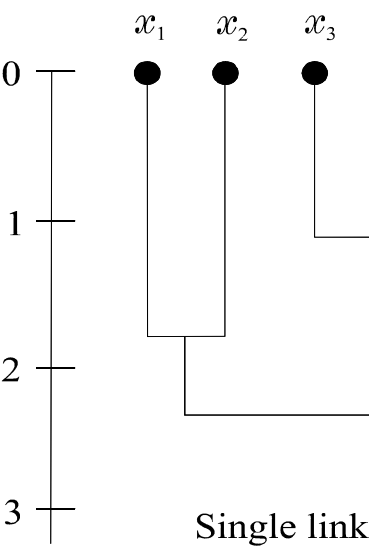
	SL	CL	WPGMA	UPGMA	WPGMC	UPGMC	Ward
\mathcal{R}_0	0	0	0	0	0	0	0
\mathcal{R}_1	1	1	1	1	1	1	0.5
\mathcal{R}_2	1.5	1.5	1.5	1.5	1.5	1.5	0.75
\mathcal{R}_3	2	3	2.5	2.5	2.25	2.25	1.5
\mathcal{R}_4	16	37	25.75	27.5	24.69	26.46	31.75

❖ Monotonicity and crossover:

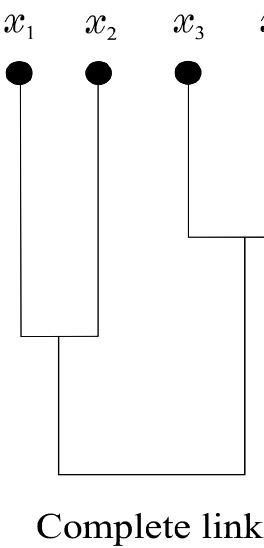
For the following dissimilarity matrix

$$P = \begin{bmatrix} 0 & 1.8 & 2.4 & 2.3 \\ 1.8 & 0 & 2.5 & 2.7 \\ 2.4 & 2.5 & 0 & 1.2 \\ 2.3 & 2.7 & 1.2 & 0 \end{bmatrix}$$

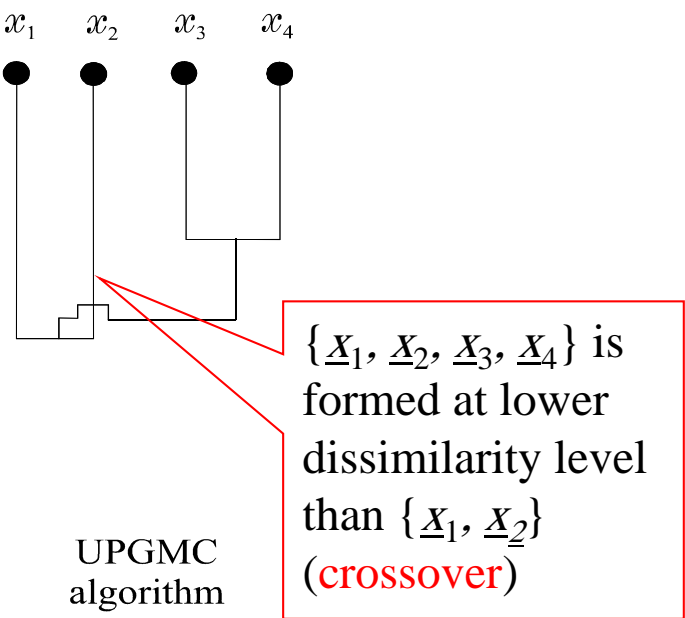
the dissimilarity dendrograms produced by **single link**, **complete link** and **UPGMC** (the same result is produced if WPGMC is employed) are:



(a)



(b)



(c)

► Monotonicity condition:

If clusters C_i and C_j are selected to be merged in cluster C_q , at the t -th level of the hierarchy, the condition

$$d(C_q, C_k) \geq d(C_i, C_j)$$

must hold for all C_k , $k \neq i, j, q$.

In other words, **the monotonicity condition implies that a cluster is formed at higher dissimilarity level than any of its components.**

► Remarks:

- **Monotonicity** is a property that is **exclusively related** to the **clustering algorithm** and not to the (initial) proximity matrix.
- An algorithm that does **not** satisfy the monotonicity condition, does **not necessarily** produce dendrograms with crossovers.
- Single link, complete link, UPGMA, WPGMA and the Ward's algorithm satisfy the monotonicity condition, while UPGMC and WPGMC do not satisfy it.

► Complexity issues:

- GAS requires, in general, $O(N^3)$ operations.
- More efficient implementations require $O(N^2 \log N)$ computational time.
- For a class of widely used algorithms, implementations that require $O(N^2)$ computational time and $O(N^2)$ or $O(N)$ storage have also been proposed.
- Parallel implementations on SIMD machines have also been considered.

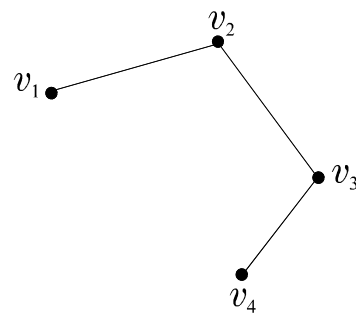
► Algorithms based on graph theory

Some basic definitions from graph theory:

- A **graph**, G , is defined as an **ordered** pair $G=(V,E)$, where $V=\{v_i: i=1,\dots,N\}$ is a set of **vertices** and E is a set of **edges** connecting some pairs of vertices. An edge connecting v_i and v_j is denoted by e_{ij} or (v_i, v_j) .
- A graph is called **undirected graph** if there is no direction assigned to any of its edges. Otherwise, we deal with **directed graphs**.
- A graph is called **unweighted graph** if there is no cost associated with any of its edges. Otherwise, we deal with **weighted graphs**.
- A **path** in G between vertices v_{i_1} and v_{i_n} is a sequence of vertices and edges of the form $v_{i_1} e_{i_1 i_2} v_{i_2} \dots v_{i_{n-1}} e_{i_{n-1} i_n} v_{i_n}$.
- A **loop** in G is a path where v_{i_1} and v_{i_n} coincide.
- A **subgraph** $G'=(V',E')$ of G is a graph with $V' \subseteq V$ and $E' \subseteq E$, where E' is a subset of E containing vertices that connect vertices of V' . **Every graph is a subgraph to itself.**
- A **connected subgraph** $G'=(V',E')$ is a subgraph where there exists at least one path connecting any pair of vertices in V' .

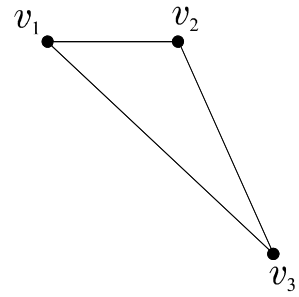
- A **complete subgraph** $G'=(V',E)$ is a subgraph where for **any** pair of vertices in V' there exists an edge in E connecting them.
- A **maximally connected subgraph** of G is a **connected subgraph** G' of G that contains as many vertices of G as possible.
- A **maximally complete subgraph** of G is a **complete subgraph** G' of G that contains as many vertices of G as possible.

Examples for the above, are shown in the following figure.



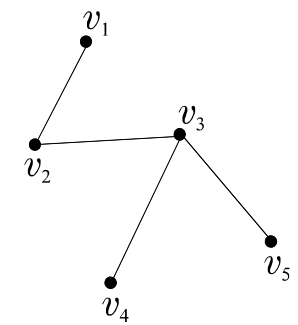
Path

(a)



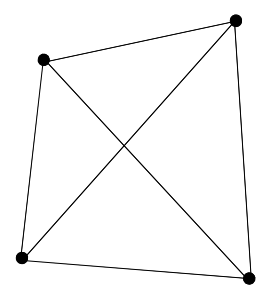
Loop

(b)



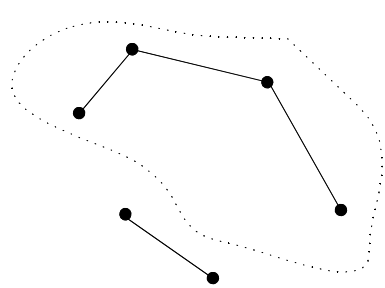
Connected graph

(c)



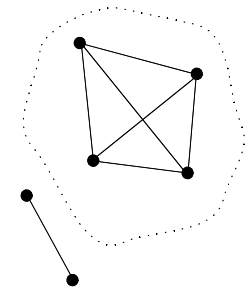
Complete graph

(d)



Maximally connected subgraph

(e)



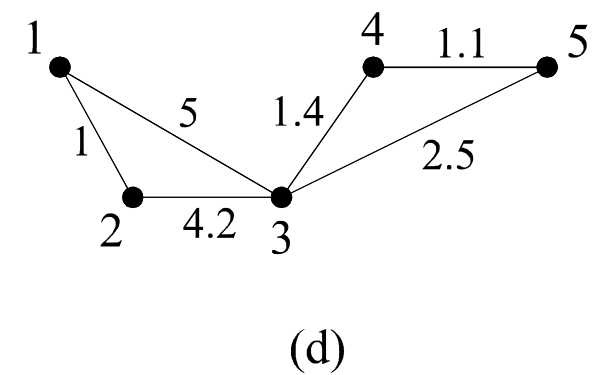
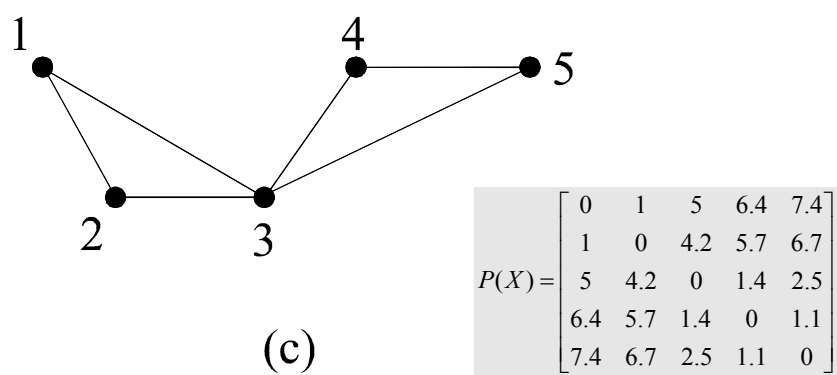
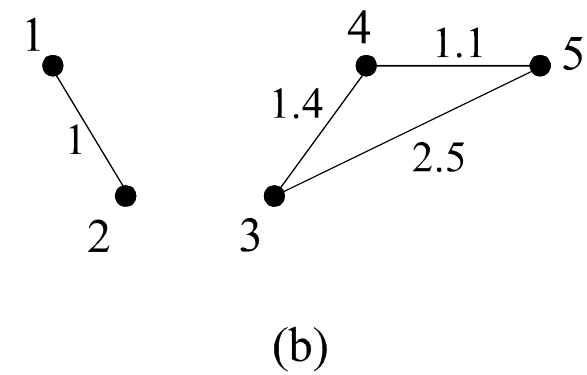
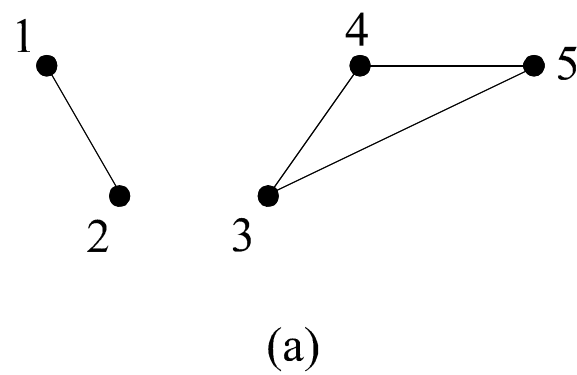
Maximally complete subgraph

(f)

- **NOTE:** In the framework of clustering, **each vertex of a graph corresponds to a feature vector.**

Useful tools for the algorithms based on graph theory are the **threshold graph** and the **proximity graph**.

- A **threshold graph** $G(a)$
 - is an undirected, unweighted graph with N nodes, each one corresponding to a vector of X .
 - No self-loops or multiple edges between any two vertices are encountered.
 - The set of edges of $G(a)$ contains those edges (v_i, v_j) for which the distance $d(\underline{x}_i, \underline{x}_j)$ between the vectors corresponding to v_i and v_j is less than or equal to a .
- A **proximity graph** $G_p(a)$ is a threshold graph $G(a)$, all of whose edges (v_i, v_j) are weighted with the proximity measure $d(\underline{x}_i, \underline{x}_j)$.



(a) The threshold graph $G(3)$, (b) the proximity (dissimilarity) graph $G_p(3)$, (c) the threshold graph $G(5)$, (d) the dissimilarity graph $G_p(5)$, for the dissimilarity matrix $P(X)$ given in example 1.

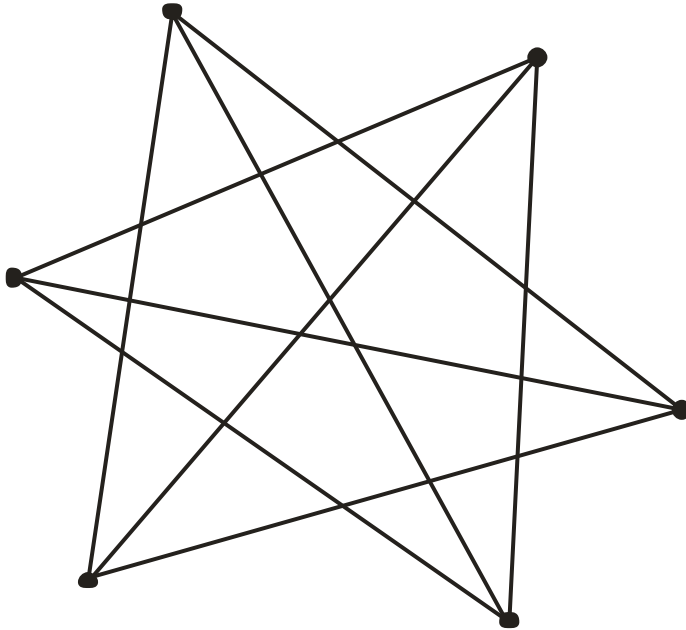
► More definitions:

In this framework, we consider graphs G , of N nodes, where each node corresponds to a vector of X .

Valid clusters are **connected components** of G that satisfy an **additional** graph property $h(k)$.

Typical graph properties for a connected subgraph G' of G are:

- **Node connectivity**: The largest integer k such that all pairs of nodes of G' are joined by **at least k paths having no nodes in common**.
- **Edge connectivity**: The largest integer k such that all pairs of nodes are joined by **at least k paths having no edges in common**.
- **Node degree**: The largest integer k such that each node has **at least k incident edges**.



Node connectivity : 3

Edge connectivity : 3

Node degree : 3

- The proximity function $g_{h(k)}(C_r, C_s)$ between two clusters is defined in terms of
 - a proximity measure between vectors (nodes)
 - certain constraints imposed by property $h(k)$ on the subgraphs that are formed.

In mathematical language:

$$g_{h(k)}(C_r, C_s) = \min_{\underline{x}_u \in C_r, \underline{x}_v \in C_s} \{d(\underline{x}_u, \underline{x}_v) \equiv a: \text{the } G(a) \text{ subgraph defined by } C_r \cup C_s \text{ is} \\ \text{(a) connected and either (b1) has the property } h(k) \text{ or (b2) is complete}\}$$

- **Graph theory-based algorithmic scheme (GTAS):** It is the **GAS** in the context of **graph theory**. In the context of GTAS, a pair of clusters (C_i, C_j) is selected to be merged according to:

$$g_{h(k)}(C_i, C_j) = \begin{cases} \min_{r,s} g_{h(k)}(C_r, C_s), & \text{for dissimilarity functions} \\ \max_{r,s} g_{h(k)}(C_r, C_s), & \text{for similarity functions} \end{cases}$$

- **Single link (SL)** algorithm. Here

$$g_{h(k)}(C_r, C_s) = \min_{\underline{x}_u \in C_r, \underline{x}_v \in C_s} \{d(\underline{x}_u, \underline{x}_v) \equiv a: \text{the } G(a) \text{ subgraph defined by } C_r \cup C_s \text{ is connected}\} \equiv \min_{\underline{x} \in C_r, \underline{y} \in C_s} d(\underline{x}, \underline{y}) \text{ (why?)}$$

- **Remarks:**

- No property $h(k)$ or completeness is required.
- The SL stemming from the graph theory is exactly the same with the SL stemming from the matrix theory.

- **Complete link (CL)** algorithm. Here

$$g_{h(k)}(C_r, C_s) = \min_{\underline{x}_u \in C_r, \underline{x}_v \in C_s} \{d(\underline{x}_u, \underline{x}_v) \equiv a: \text{the } G(a) \text{ subgraph defined by } C_r \cup C_s \text{ is complete}\} \equiv \max_{\underline{x} \in C_r, \underline{y} \in C_s} d(\underline{x}, \underline{y}) \text{ (why?)}$$

- **Remarks:**

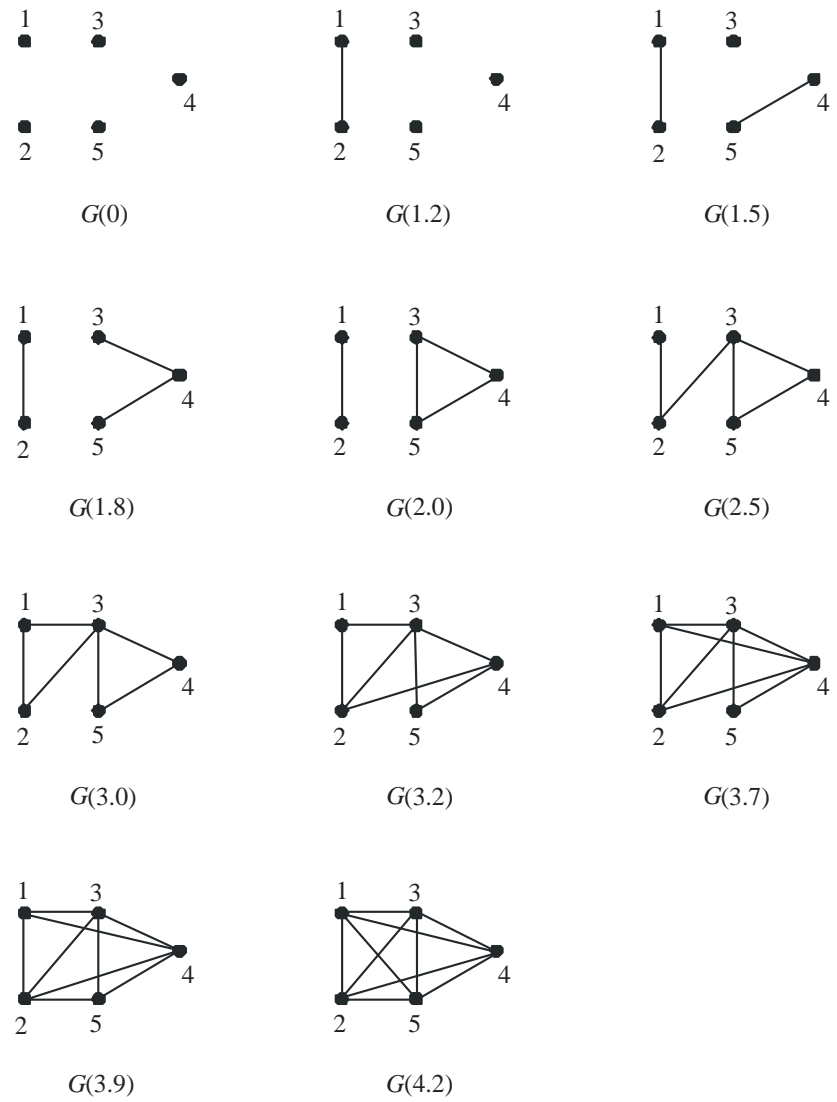
- No property $h(k)$ is required.
- The CL stemming from graph theory is exactly the same with the CL stemming from matrix theory.

➤ **Example 5:** For the dissimilarity matrix,

$$P = \begin{bmatrix} 0 & 1.2 & 3 & 3.7 & 4.2 \\ 1.2 & 0 & 2.5 & 3.2 & 3.9 \\ 3 & 2.5 & 0 & 1.8 & 2.0 \\ 3.7 & 3.2 & 1.8 & 0 & 1.5 \\ 4.2 & 3.9 & 2.0 & 1.5 & 0 \end{bmatrix}$$

SL and CL produce the same hierarchy of clusterings at the levels given in the table.

	SL	CL
$\mathcal{R}_0 = \{\{\underline{x}_1\}, \{\underline{x}_2\}, \{\underline{x}_3\}, \{\underline{x}_4\}, \{\underline{x}_5\}\}$	0	0
$\mathcal{R}_1 = \{\{\underline{x}_1, \underline{x}_2\}, \{\underline{x}_3\}, \{\underline{x}_4\}, \{\underline{x}_5\}\}$	1.2	1.2
$\mathcal{R}_2 = \{\{\underline{x}_1, \underline{x}_2\}, \{\underline{x}_3\}, \{\underline{x}_4, \underline{x}_5\}\}$	1.5	1.5
$\mathcal{R}_3 = \{\{\underline{x}_1, \underline{x}_2\}, \{\underline{x}_3, \underline{x}_4, \underline{x}_5\}\}$	1.8	2.0
$\mathcal{R}_4 = \{\{\underline{x}_1, \underline{x}_2, \underline{x}_3, \underline{x}_4, \underline{x}_5\}\}$	2.5	4.2



► Remarks:

- SL poses the **weakest** possible graph condition (**connectivity**) for the formation of a cluster, while CL poses the **strongest** possible graph condition (**completeness**) for the formation of a cluster.
- A variety of graph theory-based algorithms, that lie between these two extremes result for various choices of $h(k)$.
 - For $k=1$ all these algorithms collapse to the single link algorithm.
 - As k increases, the resulting subgraphs approach completeness.

❖ Clustering algorithms based on the Minimum Spanning Tree (MST)

➤ Definitions:

- **Spanning Tree:** It is a connected graph (containing all the vertices of the graph), with no loops (**only one path connects any two vertices**).
- **Weight of a Spanning Tree:** The sum of the weights of its edges (provided that they have been assigned with a weight).
- **Minimum Spanning Tree (MST):** A spanning tree with the **smallest** weight among the spanning trees connecting all the vertices of the graph.

➤ Remarks:

- The MST has $N-1$ edges.
- When all the **weights are different** from each other, the **MST is unique**. Otherwise, it may not be unique.

➤ Employing the GTAS and substituting $g_{h(k)}(C_r, C_s)$ with

$$g(C_r, C_s) = \min_{ij} \{w_{ij} : \underline{x}_i \in C_r, \underline{x}_j \in C_s\}$$

where $w_{ij} = d(\underline{x}_i, \underline{x}_j)$, **we can determine the MST**.

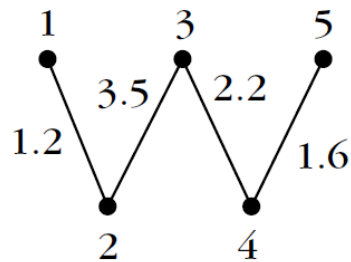
➤ Alternatively, a hierarchy of clusterings may be obtained by the MST as follows:

The clustering \mathfrak{R}_t at the t th level is the set of **connected** components of the MST, when only **its t smallest** weights are considered.

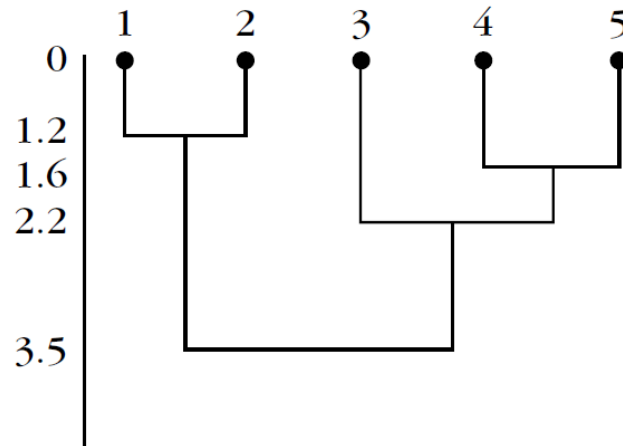
➤ Remark:

- The hierarchy produced by **MST** is the same with that produced by the **single link algorithm**, at least when all w_{ij} 's are different from each other.

$$P = \begin{bmatrix} 0 & 1.2 & 4.0 & 4.6 & 5.1 \\ 1.2 & 0 & 3.5 & 4.2 & 4.7 \\ 4.0 & 3.5 & 0 & 2.2 & 2.8 \\ 4.6 & 4.2 & 2.2 & 0 & 1.6 \\ 5.1 & 4.7 & 2.8 & 1.6 & 0 \end{bmatrix}$$



(a)



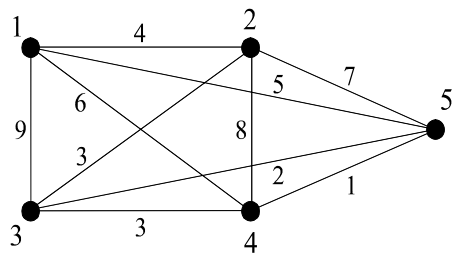
(b)

FIGURE 13.12

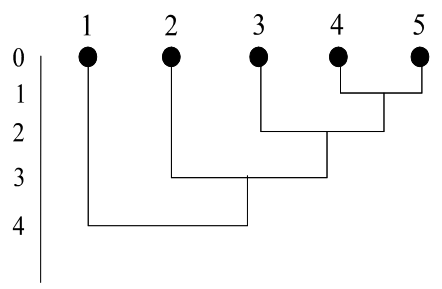
(a) The minimum spanning tree derived with the dissimilarity matrix given in Example 13.7.
 (b) The dissimilarity dendrogram obtained with the algorithm based on the MST.

► Ties in the proximity matrix

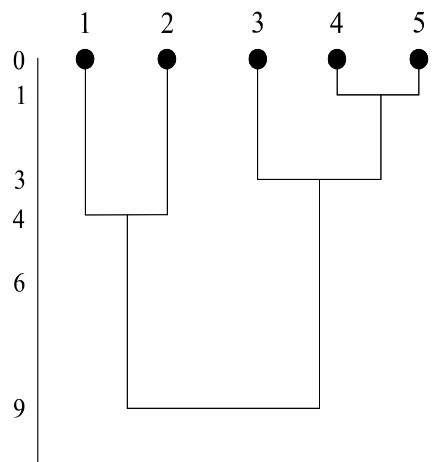
- SL produces the **same hierarchy of clusterings**, independent of the order of consideration of edges with equal weights.
- CL may produce **different hierarchies**, depending on the order of consideration of edges with equal weights.
- The other graph theory-based algorithms behave as the CL.
- The **same trend** appears in the **matrix-based algorithms**. In this case, *ties may appear at a later stage of the algorithm*.



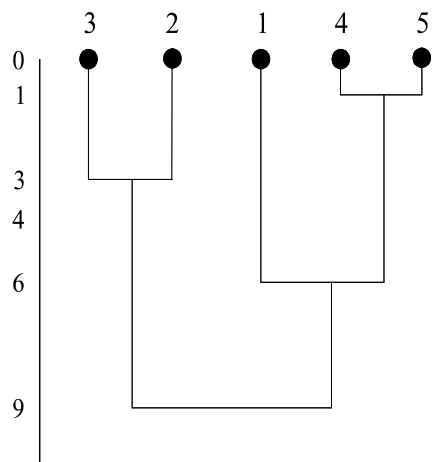
(a)



(b) (SL)



(c)
(CL(a))



(d)
(CL(b))

► Example 6: Let

$$P = \begin{bmatrix} 0 & 4 & 9 & 6 & 5 \\ 4 & 0 & 3 & 8 & 7 \\ 9 & 3 & 0 & 3 & 2 \\ 6 & 8 & 3 & 0 & 1 \\ 5 & 7 & 2 & 1 & 0 \end{bmatrix}$$

Note that $P(2,3)=P(3,4)$.

DIVISIVE ALGORITHMS

- Let $g(C_i, C_j)$ be a dissimilarity function between two clusters.
- Let C_{tj} denote the j th cluster of the t th clustering \mathfrak{R}_t ,
 $t = 0, \dots, N-1, j = 1, \dots, t+1$.

➤ Generalized Divisive Scheme (GDS)

- Initialization
 - Choose $\mathfrak{R}_0 = \{X\}$ as the initial clustering
 - $t = 0$
- Repeat
 - $t = t + 1$
 - **For** $i = 1$ to t
 - Among all possible pairs of clusters (C_r, C_s) that form a partition of $C_{t-1,i}$, find the pair $(C_{t-1,i}^1, C_{t-1,i}^2)$ that gives the maximum value for g .
 - **End for**
 - From the t pairs defined in the previous step, choose the one that maximizes g . Suppose that this is $(C_{t-1,j}^1, C_{t-1,j}^2)$.
 - The new clustering is:

$$\mathfrak{R}_t = (\mathfrak{R}_{t-1} - \{C_{t-1,j}\}) \cup \{C_{t-1,j}^1, C_{t-1,j}^2\}$$
 - Relabel the clusters of \mathfrak{R}_t
- **Until** each vector lies in a single cluster.

► Remarks:

- Different choices of g give rise to different algorithms.
- The GDS is computationally very demanding even for small N .
- Algorithms that rule out many partitions as not “reasonable”, under a prespecified criterion, have also been proposed.
- Algorithms where the splitting of the clusters is based on all features of the feature vectors are called **polythetic algorithms**. Otherwise, if the splitting is based on a single feature at each step, the algorithms are called **monothetic algorithms**.

HIERARCHICAL ALGORITHMS FOR LARGE DATA SETS

- ❖ Since the number of operations required by GAS is greater than $O(N^2)$, algorithms resulting by GAS are **prohibitive** for very large data sets encountered, for example, in web mining and bioinformatics. To overcome this drawback, various hierarchical algorithms of special type have been developed that are tailored to handle large data sets.

Typical examples are:

- The CURE algorithm.
- The ROCK algorithm.
- The Chameleon algorithm.

❖ The CURE (Clustering Using Representatives) algorithm

In CURE:

- Each cluster C is represented by a set of $k > 1$ representatives, denoted by R_C .
- These representatives try to “capture” the shape of the cluster.
- They are chosen at the “border” of the cluster and then, they are pushed toward its mean, in order to discard the irregularities of the border.

More specifically, R_C is determined as follows:

- Select $\underline{x} \in C$, with the maximum distance from the mean \underline{m}_C of C and set $R_C = \{\underline{x}\}$
- For $i = 2$ to $\min\{k, n_C\}$ (n_C is the number of points in C)
 - Determine $\underline{y} \in C - R_C$ that lies farthest from the points of R_C and set $R_C = R_C \cup \{\underline{y}\}$.
- Shrink the points $\underline{x} \in R_C$ toward the mean \underline{m}_C in C by a factor a . That is $\underline{x} = (1-a)\underline{x} + a\underline{m}_C, \forall \underline{x} \in R_C$.

CURE is a special case of GAS where the distance between two clusters is defined as:

$$d(C_i, C_j) = \min_{\underline{x} \in R_{C_i}, \underline{y} \in R_{C_j}} d(\underline{x}, \underline{y})$$

- Worst case time complexity for CURE: $O(N^2 \log_2 N)$.
- This is prohibitive for very large data sets.
- Solution: Adoption of the random sampling technique.

The size N' of a sample data set X' , created from X , via random sampling, should be sufficiently large in order to ensure that the probability of missing a cluster due to sampling is low.

❖ CURE utilizing random sampling

➤ Identification of clusters

- Partition randomly X into $p = N/N'$ sample data sets.
- For each one of the p sample data sets.
 - Apply the original version of CURE, until N'/q clusters (at the most) are formed (q is user-defined).
- Consider all the above $p(N'/q)$ clusters (at the most) and apply the original CURE until the required number of clusters, m , is formed.

➤ Assignment of points to clusters

- For each of the m clusters select a random sample of representative points.
- Assign each point \underline{x} that is not cluster representative to the cluster that contains the representative closest to it.

► Remarks:

- CURE is sensitive to the parameters k , N' , a . Specifically:
 - k must be large enough to capture the geometry of each cluster.
 - N' must be higher than a certain percentage of N (typically $N' \geq 2.5\% N$)
 - For small a CURE behaves like the MST algorithm, while for large a it resembles the algorithms that use a single point representative for each cluster.
- Worst case time complexity for CURE using random sampling: $O(N^2 \log_2 N')$
- The algorithm exhibits low sensitivity with respect to outliers within the clusters.
- A few stages before its termination, CURE checks for clusters containing very few data points and removes them (since they are likely to be outliers).
- If N'/q is sufficiently large, compared to m , it is expected that the partition of X will not affect significantly the final clustering obtained by CURE.
- CURE can, in principle, reveal clusters of non-spherical or elongated shapes, as well as clusters of wide variance in size.
- CURE can be implemented efficiently using the heap and the *k-d tree* data structures.

❖ The ROCK (RObust Clustering using linKs) algorithm

It is best suited for nominal (categorical) features.

➤ Some preliminaries

- Two points $\underline{x}, \underline{y} \in X$ are considered **neighbors** if $s(\underline{x}, \underline{y}) \geq \theta$, where $s(\cdot)$ is a similarity function and θ a user-defined threshold of similarity between two vectors.
- $link(\underline{x}, \underline{y})$ is the number of **common neighbors** between \underline{x} and \underline{y} .

$$link(C_i, C_j) = \sum_{\underline{x} \in C_i} \sum_{\underline{y} \in C_j} link(\underline{x}, \underline{y})$$

➤ **Assumption:** There exists a function $f(\theta)$ (<1) such that:

“Each point assigned to a cluster C_i has approximately $n_i^{f(\theta)}$ neighbors in C_i (n_i is the number of points in C_i)”

It can be proved that the expected total number of links among all pairs in C_i is $n_i^{1+2f(\theta)}$.

➤ ROCK is a special case of GAS where

- The **closeness** between two clusters is defined as

$$g(C_i, C_j) = \frac{\text{link}(C_i, C_j)}{(n_i + n_j)^{1+2f(\theta)} - n_i^{1+2f(\theta)} - n_j^{1+2f(\theta)}}$$

The denominator is the expected total number of links *between* the two clusters.

The larger the $g(\cdot)$, the more similar the clusters C_i and C_j are.

➤ The stopping criterion is:

- the number of clusters become equal to a predefined number m or
- $\text{link}(C_i, C_j) = 0$ for every pair in a clustering \mathcal{R}_t .

➤ Time complexity for ROCK: Similar to CURE for large N .

➤ **Prohibitive** for very large data sets.

➤ Solution: Adoption of random sampling techniques.

► ROCK utilizing Random Sampling

- Identification of clusters
 - Select a subset X' of X via random sampling
 - Run the original ROCK algorithm on X'
- Assignment of points to clusters
 - For each cluster C_i select a set L_i of n_{L_i} points
 - For each $\underline{z} \in X - X'$
 - o Compute $t_i = N_i / (n_{L_i} + 1)^{f(\theta)}$,
where N_i is the number of neighbors of \underline{z} in L_i .
 - o Assign \underline{z} to the cluster with the maximum t_i .

► Remarks:

- A choice for $f(\theta)$ is $f(\theta) = (1 - \theta) / (1 + \theta)$, with $(\theta < 1)$.
- $f(\theta)$ depends on the data set and the type of clusters we are interested in.
- The hypothesis about the existence of $f(\theta)$ is very strong. It may lead to poor results if the data do not satisfy it.

❖ The Chameleon algorithm

- This algorithm is not based on a “static” modeling of clusters like CURE (where each cluster is represented by the same number of representatives) and ROCK (where constraints are posed through the function $f(\theta)$).
- It enjoys both **divisive** and **agglomerative** features.

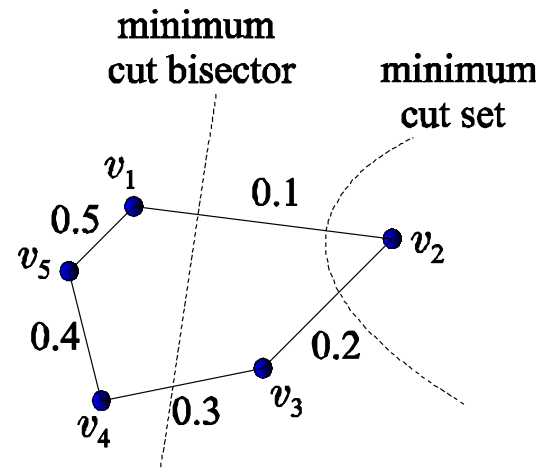
➤ Some preliminaries:

Let $G=(V,E)$ be a graph where:

- each vertex of V corresponds to a data point in X .
- E is a set of edges connecting pairs of vertices in V . Each vertex is weighted by the similarity of the corresponding points.

- **Edge cut set:** Let C be a set of points corresponding to a subset of V . Assume that C is partitioned into two nonempty sets C_i and C_j . The subset E'_{ij} of E that connect C_i and C_j is called **edge cut set**.

- **Minimum cut set:** Let $|E'_{ij}|$ be the sum of weights of the edges in E'_{ij} . If $|E'_{ij}| = \min_{(C_u, C_v)} |E_{uv}|$, then (C_i, C_j) is the **minimum cut set** of C .
- **Minimum cut bisector:** If C_i, C_j are constrained to be of approximate equal size, the minimum cut set (over all possible partitions of approximately equal size) is known as the **minimum cut bisector**.
- **Example 7:** The graph in the following figure consists of **5** vertices and the edges shown, each one weighted by the similarity of the points that correspond to the vertices it connects. The minimum cut set and the minimum cut bisector are shown.



► Measuring the similarity between clusters

• Relative interconnectivity:

- Let E_{ij} be the set of edges connecting points in C_i with points in C_j .
- Let E_i be the set of edges corresponding to the minimum cut bisector of C_i .
- Let $|E_i|$, $|E_{ij}|$ be the sum of the weights of the edges of E_i , E_{ij} , respectively.
- **Absolute interconnectivity** between C_i , $C_j = |E_{ij}|$
- **Internal interconnectivity** of $C_i = |E_i|$
- **Relative interconnectivity** between C_i , C_j :

$$RI_{ij} = \frac{|E_{ij}|}{\frac{|E_i| + |E_j|}{2}}$$

• Relative closeness:

- Let S_{ij} be the average weight of the edges in E_{ij} .
- Let S_i be the average weight of the edges in E_i .
- **Relative closeness** between C_i and C_j :

$$RC_{ij} = \frac{S_{ij}}{\frac{n_i}{n_i + n_j} S_i + \frac{n_j}{n_i + n_j} S_j}$$

► The Chameleon algorithm

Preliminary phase

Create a k -nearest neighbor graph $G=(V,E)$ such that:

- Each vertex of V corresponds to a data point.
- The edge between two vertices v_i and v_j is added to E if v_i is one of the k -nearest neighbors of v_j or vice versa.

Divisive phase

Set $\mathcal{R}_0 = \{X\}$

$t=0$

Repeat

- $t=t+1$
- Select the largest cluster C in \mathcal{R}_{t-1} .
- Referring to E , partition C into two sets so that:
 - the sum of the weights of the edge cut set between the resulting clusters is minimized.
 - each cluster contains at least 25% of the vertices of C .

Until each cluster in \mathcal{R}_t contains fewer than q points.

The Chameleon algorithm (cont.)

Agglomerative phase

Set $\mathfrak{R}'_0 = \mathfrak{R}_t$

$t = 0$

Repeat

- $t = t + 1$
- Merge C_i, C_j in \mathfrak{R}'_{t-1} to a single cluster if

$$RI_{ij} \geq T_{RI} \text{ and } RC_{ij} \geq T_{RC} \quad (\text{A})$$
 (if more than one C_j satisfy the conditions for a given C_i , the C_j with the highest $|E_{ij}|$ is selected).

Until (A) does not hold for any pair of clusters in \mathfrak{R}'_{t-1} .

Return \mathfrak{R}'_{t-1}

NOTE: The internal structure of two clusters to be merged is of significant importance. **The more similar** the elements within each cluster the higher “their resistance” in merging with another cluster.

► Remarks:

- Condition (A) can be replaced by $\max_{(C_i, C_j)} RI_{ij} RC_{ij}^a$
- Chameleon is not very sensitive to the choice of the user-defined parameters k (typically it is selected between 5 and 20), q (typically chosen in the range 1 to 5% of the total number of data points), T_{RI} , T_{RC} and/or a .
- Chameleon is well suited for large data sets (better estimation of $|E_{ij}|$, $|E_i|$, S_{ij} , S_i)
- For large N , the worst-case time complexity of the algorithm is $O(N(\log_2 N + m))$, where m is the number of clusters formed by the divisive phase.

➤ **Example 8:** For the clusters shown in the figure we have:

$$|E_1|=0.48, |E_2|=0.48, |E_3|=1.45, |E_4|=1.45,$$

$$|S_1|=0.48, |S_2|=0.48, |S_3|=0.725, |S_4|=0.725,$$

$$|E_{12}|=0.4, |E_{34}|=0.6, |S_{12}|=0.4, |S_{34}|=0.6.$$

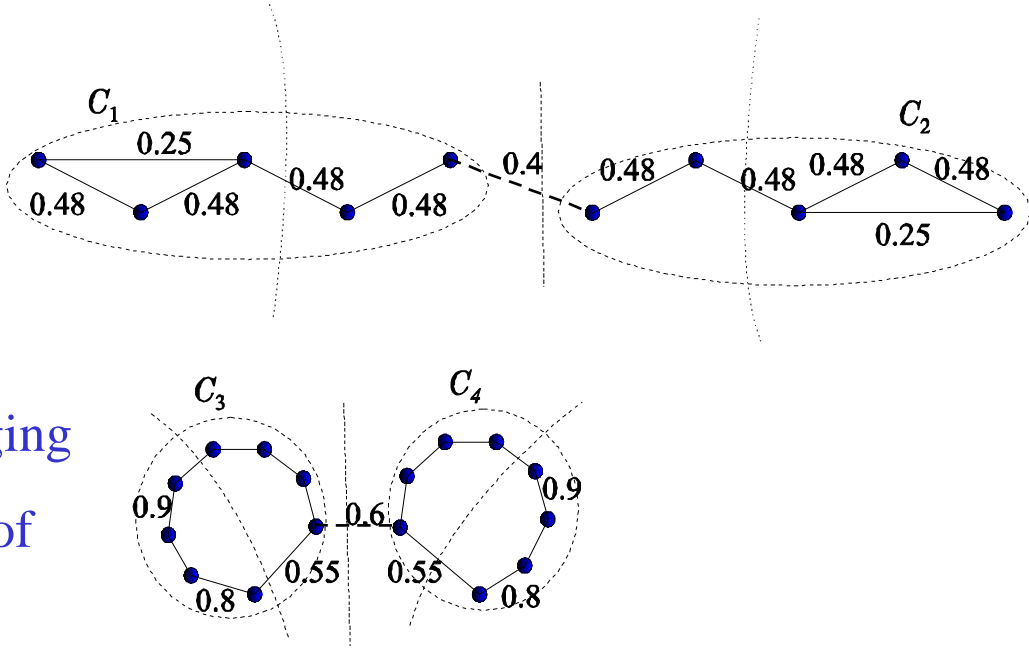
Thus,

$$RI_{12}=0.833, RI_{34}=0.414$$

$$RC_{12}=0.833, RC_{34}=0.828$$

In conclusion:

Both *RI* and *RC* favor the merging
*C*₁ and *C*₂ against the merging of
*C*₃ and *C*₄.



Note that MST would merge C_3 and C_4 instead of C_1 and C_2 .

CHOICE OF THE BEST NUMBER OF CLUSTERS

❖ A major issue associated with hierarchical algorithms is:

“How the clustering that best fits the data is extracted from a hierarchy of clusterings?”

Some approaches:

- Search in the proximity dendrogram for clusters that have a large **lifetime** (the difference between the proximity level at which a cluster is created and the proximity level at which it is absorbed into a larger cluster (however, this method involves human subjectivity)).
- Define a function $h(C)$ that measures the dissimilarity between the vectors of the same cluster C . Let \mathcal{G} be an appropriate threshold for $h(C)$. Then \mathcal{R}_t is the final clustering if there exists a cluster C in \mathcal{R}_{t+1} with dissimilarity between its vectors ($h(C)$) greater than \mathcal{G} (**extrinsic method**). The final clustering \mathcal{R}_t must satisfy the following condition:

$$d_{min}^{ss}(C_i, C_j) > \max \{h(C_i), h(C_j)\}, \quad \forall C_i, C_j \in \mathcal{R}_t$$

In words, in the final clustering, the dissimilarity between every pair of clusters is larger than the “self-similarity” of each one of them (**intrinsic method**).