Introduction to
Graph Cluster Analysis
Outline

• Algorithms for Graph Clustering
  □ k-Spanning Tree
  □ Shared Nearest Neighbor
  □ Betweenness Centrality Based
  □ Highly Connected Components
  □ Maximal Clique Enumeration
  □ Kernel k-means

• Application
Outline

• Introduction to Clustering
• Introduction to Graph Clustering
• Algorithms for Graph Clustering
  ❑ k-Spanning Tree
  ❑ Shared Nearest Neighbor
  ❑ Betweenness Centrality Based
  ❑ Highly Connected Components
  ❑ Maximal Clique Enumeration
  ❑ Kernel k-means
• Application
What is Cluster Analysis?

The process of dividing a set of input data into possibly overlapping, subsets, where elements in each subset are considered related by some similarity measure.
Outline

• Introduction to Clustering
• Introduction to Graph Clustering
• Algorithms for Graph Clustering
  - k-Spanning Tree
  - Shared Nearest Neighbor
  - Betweenness Centrality Based
  - Highly Connected Components
  - Maximal Clique Enumeration
  - Kernel k-means
• Application
What is Graph Clustering?

• Types
  – Between-graph
    • Clustering a set of graphs
  – Within-graph
    • Clustering the nodes/edges of a single graph
Between-graph Clustering

Between-graph clustering methods divide a set of graphs into different clusters

E.g., A set of graphs representing chemical compounds can be grouped into clusters based on their structural similarity
Within-graph Clustering

Within-graph clustering methods divides the nodes of a graph into clusters

E.g., In a social networking graph, these clusters could represent people with same/similar hobbies

Note: In this lecture we will look at different algorithms to perform within-graph clustering
Outline

• Introduction to Clustering
• Introduction to Graph Clustering
• Algorithms for Within Graph Clustering
  - k-Spanning Tree
  - Shared Nearest Neighbor
  - Betweenness Centrality Based
  - Highly Connected Components
  - Maximal Clique Enumeration
  - Kernel k-means
• Application
Graph-Based Clustering

- Graph-Based clustering uses the proximity graph
  - Start with the proximity matrix
  - Consider each point as a node in a graph
  - Each edge between two nodes has a weight which is the proximity between the two points
  - Initially the proximity graph is fully connected
  - MIN (single-link) and MAX (complete-link) can be viewed as starting with this graph
- In the simplest case, clusters are connected components in the graph.
Graph-Based Clustering: Sparsification

- The amount of data that needs to be processed is drastically reduced
  - Sparsification can eliminate more than 99% of the entries in a proximity matrix
  - The amount of time required to cluster the data is drastically reduced
  - The size of the problems that can be handled is increased
Graph-Based Clustering: Sparsification ...

• **Clustering may work better**
  – Sparsification techniques keep the connections to the most similar (nearest) neighbors of a point while breaking the connections to less similar points.
  – The nearest neighbors of a point tend to belong to the same class as the point itself.
  – This reduces the impact of noise and outliers and sharpens the distinction between clusters.

• **Sparsification facilitates the use of graph partitioning algorithms (or algorithms based on graph partitioning algorithms).**
  – Chameleon and Hypergraph-based Clustering
Sparsification in the Clustering Process
Minimum Spanning Tree based Clustering

STEPS:
- Obtains the Minimum Spanning Tree (MST) of input graph G
- Removes k-1 heaviest edges from the MST
- Results in k clusters

k groups of non-overlapping vertices

Minimum Spanning Tree

k-Spanning Tree
What is a Spanning Tree?

A connected subgraph with no cycles that includes all vertices in the graph

Note: Weight can represent either distance or similarity between two vertices or similarity of the two vertices.
What is a Minimum Spanning Tree (MST)?

The spanning tree of a graph with the minimum possible sum of edge weights, if the edge weights represent distance.

Note: maximum possible sum of edge weights, if the edge weights represent similarity.
k-Spanning Tree

Minimum Spanning Tree

Remove k-1 edges with highest weight

E.g., k=3

Note: k – is the number of clusters

E.g., k=3

3 Clusters
Outline

• Introduction to Clustering
• Introduction to Graph Clustering
• Algorithms for Within Graph Clustering
  - k-Spanning Tree
  - Shared Nearest Neighbor Clustering
  - Betweenness Centrality Based
  - Highly Connected Components
  - Maximal Clique Enumeration
  - Kernel k-means
• Application
Shared Nearest Neighbor Clustering

Shared Nearest Neighbor Graph (SNN)

Groups of non-overlapping vertices

STEPS:
• Obtains the Shared Nearest Neighbor Graph (SNN) of input graph G
• Removes edges from the SNN with weight less than $\tau$
What is Shared Nearest Neighbor?

Shared Nearest Neighbor is a proximity measure and denotes the number of neighbor nodes common between any given pair of nodes.
Shared Nearest Neighbor (SNN) Graph

Given input graph G, weight each edge \((u,v)\) with the number of shared nearest neighbors between \(u\) and \(v\)

Node 0 and Node 1 have 2 neighbors in common: Node 2 and Node 3
Shared Nearest Neighbor Clustering

*Jarvis-Patrick Algorithm*

SNN graph of input graph G

If u and v share more than $\tau$ neighbors
Place them in the same cluster

E.g., $\tau = 3$
Outline

• Introduction to Clustering
• Introduction to Graph Clustering
• Algorithms for Within Graph Clustering
  - k-Spanning Tree
  - Shared Nearest Neighbor Clustering
  - Betweenness Centrality Based
  - Highly Connected Components
  - Maximal Clique Enumeration
  - Kernel k-means
• Application
What is Betweenness Centrality?

Betweenness centrality quantifies the degree to which a vertex (or edge) occurs on the shortest path between all the other pairs of nodes.

Two types:
- Vertex Betweenness
- Edge Betweenness
Vertex Betweenness

The number of **shortest paths in the graph G** that pass through a given node $S$.

E.g., Sharon is likely a liaison between NCSU and DUKE and hence many connections between DUKE and NCSU pass through Sharon.
Edge Betweenness

The number of **shortest paths in the graph G** that pass through given edge \((S, B)\)

E.g., Sharon and Bob both study at NCSU and they are the only link between NY DANCE and CISCO groups.

Vertices and Edges with high Betweenness form good starting points to identify clusters.
Given Input graph $G$

1. Disconnect graph at selected vertex (e.g., vertex 3)
2. Copy vertex to both Components

Repeat until highest vertex betweenness $\leq \mu$

Select vertex $v$ with the highest betweenness
E.g., Vertex 3 with value 0.67
Edge-Betweenness Clustering

Girvan and Newman Algorithm

Given Input Graph G

Repeat until highest edge betweenness ≤ \( \mu \)

Betweenness for each edge

Connect graph at selected edge (E.g., (3,4))

Select edge with Highest Betweenness E.g., edge (3,4) with value 0.571
Outline

• Introduction to Clustering
• Introduction to Graph Clustering
• Algorithms for Within Graph Clustering
  ❑ k-Spanning Tree
  ❑ Shared Nearest Neighbor Clustering
  ❑ Betweenness Centrality Based
  ❑ Highly Connected Components
  ❑ Maximal Clique Enumeration
  ❑ Kernel k-means
• Application
What is a Highly Connected Subgraph?

• Requires the following definitions
  – Cut
  – Minimum Edge Cut (MinCut)
  – Edge Connectivity (EC)
Cut

- The set of edges whose removal disconnects a graph

Cut = \{ (0,1), (1,2), (1,3) \}

Cut = \{ (3,5), (4,2) \}
Minimum Cut

The minimum set of edges whose removal disconnects a graph

MinCut = {(3,5),(4,2)}
Edge Connectivity (EC)

- Minimum **NUMBER** of edges that will disconnect a graph

MinCut = {(3,5),(4,2)}

\[
\text{EC} = | \text{MinCut} | \\
= | \{(3,5),(4,2)\} | \\
= 2
\]
Highly Connected Subgraph (HCS)

A graph $G = (V,E)$ is highly connected if $EC(G) > V/2$

$G$ is NOT a highly connected subgraph
HCS Clustering

Given Input graph G

Process Graph G1

Find the Minimum Cut MinCut (G) (3,5),(4,2)

Process Graph G2

Is EC(G)> V/2

YES

Return G

NO

Divide G using MinCut
Outline

• Introduction to Clustering
• Introduction to Graph Clustering
• Algorithms for Within Graph Clustering
  - k-Spanning Tree
  - Shared Nearest Neighbor Clustering
  - Betweenness Centrality Based
  - Highly Connected Components
  - Maximal Clique Enumeration
  - Kernel k-means
• Application
What is a Clique?

A subgraph C of graph G with edges between all pairs of nodes
What is a Maximal Clique?

A maximal clique is a clique that is not part of a larger clique.
Maximal Clique Enumeration

**Bron and Kerbosch Algorithm**

Input Graph $G$

- **$BK(C,P,N)$**
  - $C$ - vertices in current clique
  - $P$ – vertices that can be added to $C$
  - $N$ – vertices that cannot be added to $C$

**Condition:**

*If both $P$ and $N$ are empty – output $C$ as maximal clique*

---

**Example:**

- $BK(\emptyset,\{1,2,3,4\},\emptyset)$
- $BK(\emptyset,\emptyset,\emptyset)$
- $BK(\emptyset,\emptyset,\emptyset)$
- $BK(\{1,2\},\emptyset,\emptyset)$
- $BK(\{1,2\},\{3\},\emptyset)$
- $BK(\{1,2\},\{3\},\emptyset)$
- $BK(\{1,2,3\},\emptyset,\emptyset)$
- $BK(\{1,3\},\emptyset,\{2\})$
- $BK(\{1,3\},\emptyset,\{2\})$

- $P = \emptyset, N = \emptyset$
  - Output: $C = \{1,2,3\}$ as maximal clique

- $P = \emptyset, N \neq \emptyset$
  - No Output
Outline

• Introduction to Clustering
• Introduction to Graph Clustering
• Algorithms for Within Graph Clustering
  - k-Spanning Tree
  - Shared Nearest Neighbor Clustering
  - Betweenness Centrality Based
  - Highly Connected Components
  - Maximal Clique Enumeration
  - Kernel k-means
• Application
What is k-means?

• k-means is a clustering algorithm applied to vector data points
• k-means recap:
  – Select k data points from input as centroids
  1. Assign other data points to the nearest centroid
  2. Recompute centroid for each cluster
  3. Repeat Steps 1 and 2 until centroids don’t change
k-means on Graphs

*Kernel K-means*

- Basic algorithm is the same as k-means on Vector data
- We utilize the “kernel trick”
- “kernel trick” recap
  - We know that we can use *within-graph kernel* functions to calculate the inner product of a pair of vertices in a user-defined feature space.
  - We replace the standard distance/proximity measures used in k-means with this *within-graph kernel* function
Application

- **Functional modules** in protein-protein interaction networks
- Subgraphs with pair-wise interacting nodes => Maximal cliques
Chameleon: Clustering Using Dynamic Modeling

• Adapt to the characteristics of the data set to find the natural clusters
• Use a dynamic model to measure the similarity between clusters
  – Main property is the relative closeness and relative inter-connectivity of the cluster
  – Two clusters are combined if the resulting cluster shares certain \textit{properties} with the constituent clusters
  – The merging scheme preserves \textit{self-similarity}
• One of the areas of application is \textit{spatial data}
Characteristics of Spatial Data Sets

- Clusters are defined as densely populated regions of the space
- Clusters have arbitrary shapes, orientation, and non-uniform sizes
- Difference in densities across clusters and variation in density within clusters
- Existence of special artifacts (streaks) and noise

The clustering algorithm must address the above characteristics and also require minimal supervision.
Chameleon: Steps

• **Preprocessing Step:**
  Represent the Data by a Graph
  – Given a set of points, construct the k-nearest-neighbor (k-NN) graph to capture the relationship between a point and its k nearest neighbors
  – Concept of neighborhood is captured dynamically (even if region is sparse)

• **Phase 1:** Use a multilevel graph partitioning algorithm on the graph to find a large number of clusters of well-connected vertices
  – Each cluster should contain mostly points from
Chameleon: Steps ...

- **Phase 2**: Use Hierarchical Agglomerative Clustering to merge sub-clusters
  - Two clusters are combined if the resulting cluster shares certain properties with the constituent clusters
  - Two key properties used to model cluster similarity:
    - **Relative Interconnectivity**: Absolute interconnectivity of two clusters normalized by the internal connectivity of the clusters
    - **Relative Closeness**: Absolute closeness of two clusters normalized by the internal closeness of the clusters
Experimental Results:

CHAMELEON
Experimental Results:

CHAMELEON
Experimental Results: CURE (10 clusters)
Experimental Results: CURE (75 clusters)
Experimental Results: CHAMELEON
Shared Near Neighbor Approach

**SNN graph**: the weight of an edge is the number of shared neighbors between vertices given that the vertices are connected.
Creating the SNN Graph

Sparse Graph

Link weights are similarities between neighboring points

Shared Near Neighbor Graph

Link weights are number of Shared Nearest Neighbors
ROCK (RObust Clustering using links)

- Clustering algorithm for data with categorical and Boolean attributes
  - A pair of points is defined to be neighbors if their similarity is greater than some threshold
  - Use a hierarchical clustering scheme to cluster the data.

1. Obtain a sample of points from the data set
2. Compute the link value for each set of points, i.e., transform the original similarities (computed by Jaccard coefficient) into similarities that reflect the number of shared neighbors between points
3. Perform an agglomerative hierarchical clustering on the data using the “number of shared neighbors” as similarity measure and maximizing “the shared neighbors” objective function
4. Assign the remaining points to the clusters that have been found
Jarvis-Patrick Clustering

• First, the k-nearest neighbors of all points are found
  – In graph terms this can be regarded as breaking all but the k strongest
    links from a point to other points in the proximity graph

• A pair of points is put in the same cluster if
  – any two points share more than T neighbors and
  – the two points are in each others k nearest neighbor list

• For instance, we might choose a nearest neighbor list of size
  20 and put points in the same cluster if they share more than
  10 near neighbors

• Jarvis-Patrick clustering is too brittle
When Jarvis-Patrick Works Reasonably Well

Original Points

Jarvis Patrick Clustering
6 shared neighbors out of 20
When Jarvis-Patrick Does NOT Work Well

Smallest threshold, $T$, that does not merge clusters.

Threshold of $T - 1$
SNN Clustering Algorithm

1. **Compute the similarity matrix**
   This corresponds to a similarity graph with data points for nodes and edges whose weights are the similarities between data points.

2. **Sparsify the similarity matrix by keeping only the $k$ most similar neighbors**
   This corresponds to only keeping the $k$ strongest links of the similarity graph.

3. **Construct the shared nearest neighbor graph from the sparsified similarity matrix.**
   At this point, we could apply a similarity threshold and find the connected components to obtain the clusters (Jarvis-Patrick algorithm).

4. **Find the SNN density of each Point.**
   Using a user specified parameters, $Eps$, find the number points that have an SNN similarity of $Eps$ or greater to each point. This is the SNN density of the point.
5. **Find the core points**
Using a user specified parameter, $MinPts$, find the core points, i.e., all points that have an SNN density greater than $MinPts$.

6. **Form clusters from the core points**
If two core points are within a radius, $Eps$, of each other they are placed in the same cluster.

7. **Discard all noise points**
All non-core points that are not within a radius of $Eps$ of a core point are discarded.

8. **Assign all non-noise, non-core points to clusters**
This can be done by assigning such points to the nearest core point.

(Note that steps 4-8 are DBSCAN)
SNN Density

a) All Points

b) High SNN Density

c) Medium SNN Density

d) Low SNN Density
SNN Clustering Can Handle Other Difficult Situations
Finding Clusters of Time Series In Spatio-Temporal Data

26 SLP Clusters via Shared Nearest Neighbor Clustering (100 NN, 1982-1994)

SNN Clusters of SLP.

SNN Density of SLP Time Series Data

SNN Density of Points on the Globe.
Features and Limitations of SNN Clustering

• Does not cluster all the points

• Complexity of SNN Clustering is high
  – $O(n \times \text{time to find numbers of neighbor within } Eps)$
  – In worst case, this is $O(n^2)$
  – For lower dimensions, there are more efficient ways to find the nearest neighbors
    • R* Tree
    • k-d Trees