Introduction to Graph Cluster Analysis

OAK RIDGE National Laboratory

NC STATE UNIVERSITY
Department of Computer Science

Outline

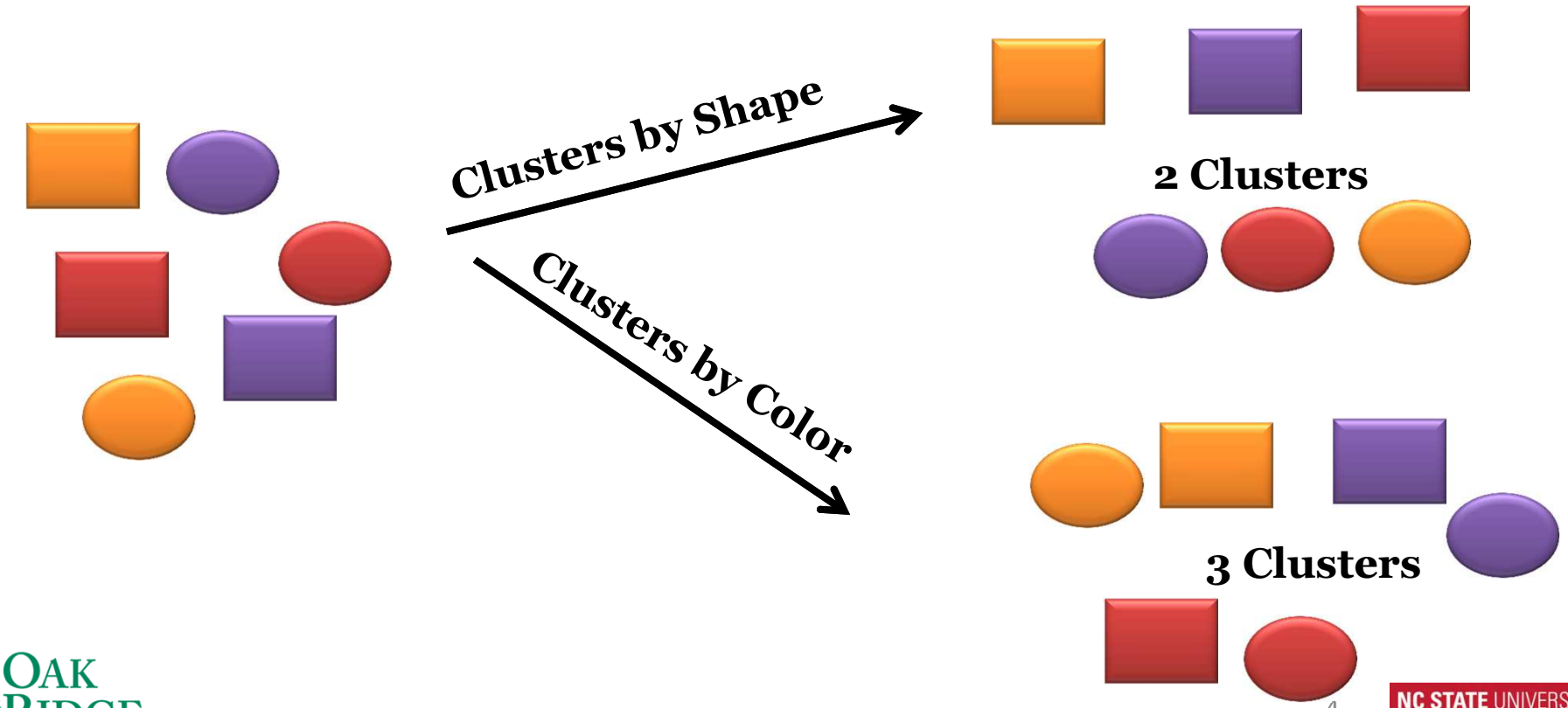
- Introduction to Cluster Analysis
- Types of Graph Cluster Analysis
- Algorithms for Graph Clustering
  - ❑ k-Spanning Tree
  - ❑ Shared Nearest Neighbor
  - ❑ Betweenness Centrality Based
  - ❑ Highly Connected Components
  - ❑ Maximal Clique Enumeration
  - ❑ Kernel k-means
- Application

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

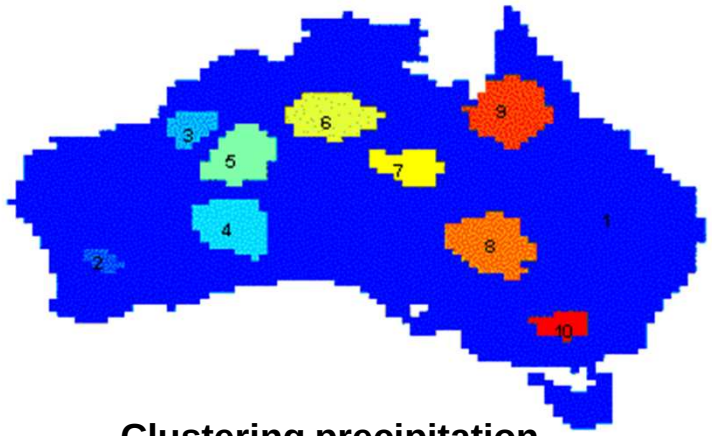
What is Cluster Analysis?

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



Applications of Cluster Analysis

- Summarization
  - Provides a macro-level view of the data-set



Clustering precipitation in Australia

From Tan, Steinbach, Kumar Introduction To Data Mining, Addison-Wesley, Edition 1

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

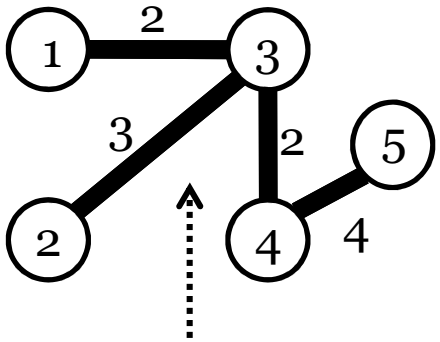
E.g., In a social networking graph, these clusters could represent people with same/similar hobbies

Note: *In this chapter 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

k-Spanning Tree



Minimum Spanning Tree

k-Spanning Tree

k groups of non-overlapping vertices

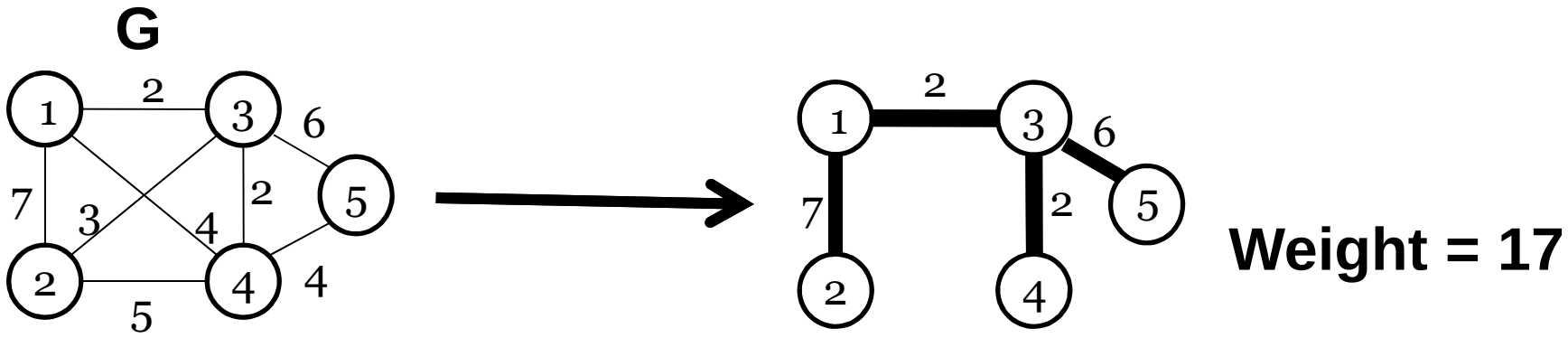
k

STEPS:

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

What is a Spanning Tree?

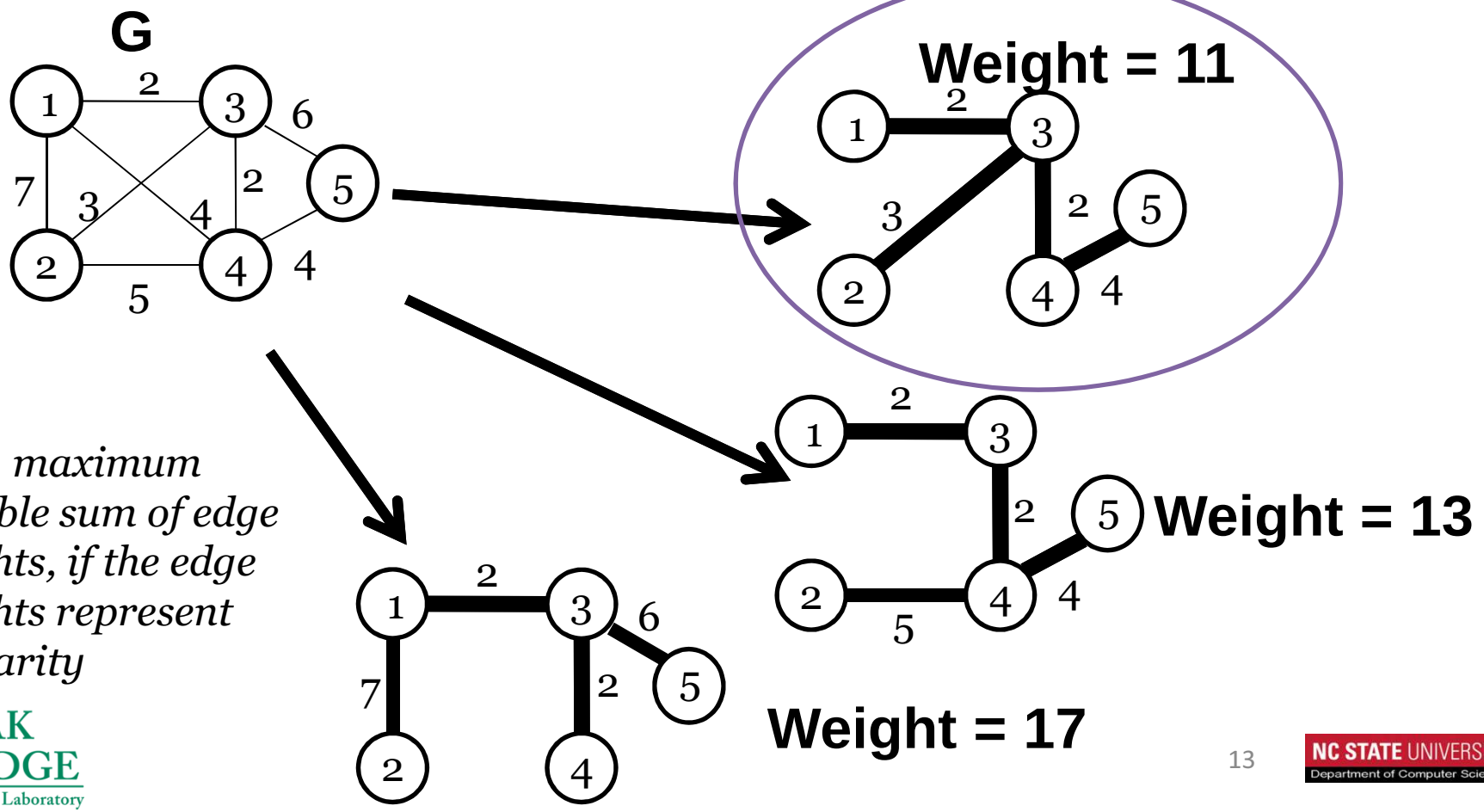
A connected subgraph with no cycles that includes all vertices in the graph



Note: *Weight can represent either distance or similarity between two vertices or similarity of the two vertices*

What is a Minimum Spanning Tree (MST)?

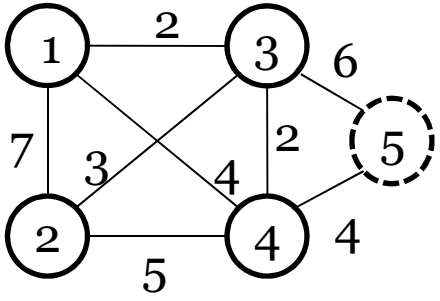
The spanning tree of a graph with the minimum possible sum of edge weights, if the edge weights represent distance



Note: *maximum possible sum of edge weights, if the edge weights represent similarity*

Algorithm to Obtain MST
Prim's Algorithm

Given Input Graph G



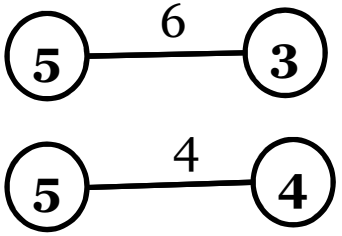
Select Vertex Randomly e.g., Vertex 5

Initialize Empty Graph T with Vertex 5

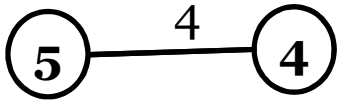
T

Repeat until all vertices are added to T

Select a list of edges L from G such that at most ONE vertex of each edge is in T

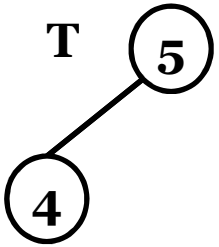


From L select the edge X with minimum weight

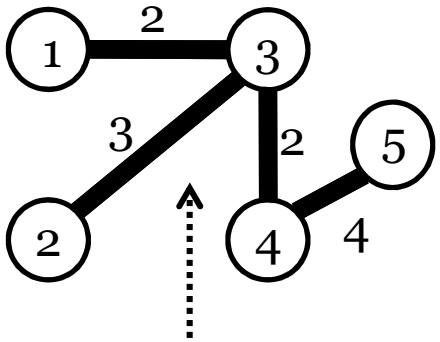


Add X to T

T



k-Spanning Tree

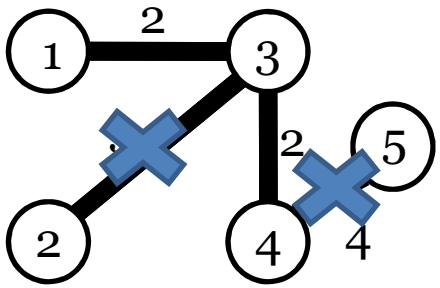


Minimum Spanning Tree

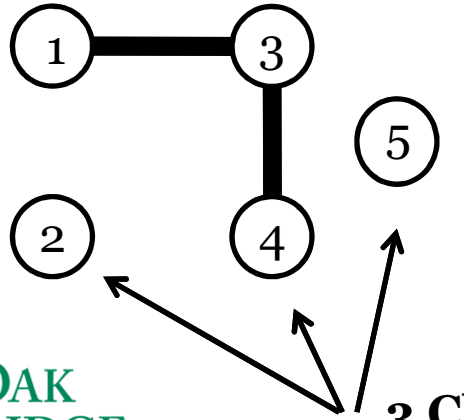
Remove k-1 edges with highest weight

Note: *k – is the number of clusters*

E.g., k=3



E.g., k=3



3 Clusters

OAK RIDGE National Laboratory

NC STATE UNIVERSITY Department of Computer Science

k-Spanning Tree R-code

- library(GraphClusterAnalysis)
- library(RBGL)
- library(igraph)
- library(graph)
- data(MST_Example)
- G = graph.data.frame(MST_Example,directed=FALSE)
- E(G)$weight=E(G)$V3
- MST_PRIM = minimum.spanning.tree(G,weights=G$weight, algorithm = "prim")

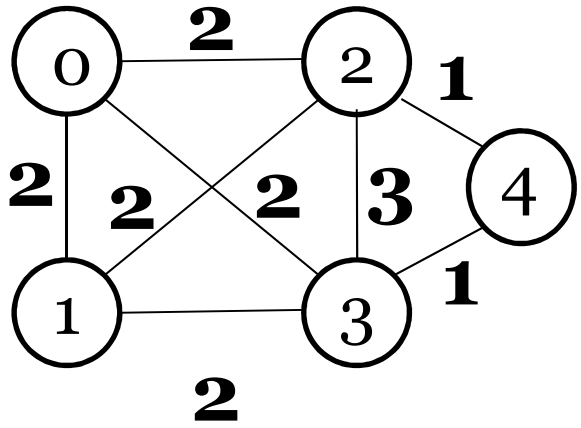
- OutputList = k_clusterSpanningTree(MST_PRIM,3)
- Clusters = OutputList[[1]]
- outputGraph = OutputList[[2]]
- Clusters

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)



Shared Nearest Neighbor Clustering

→ Groups of non-overlapping vertices

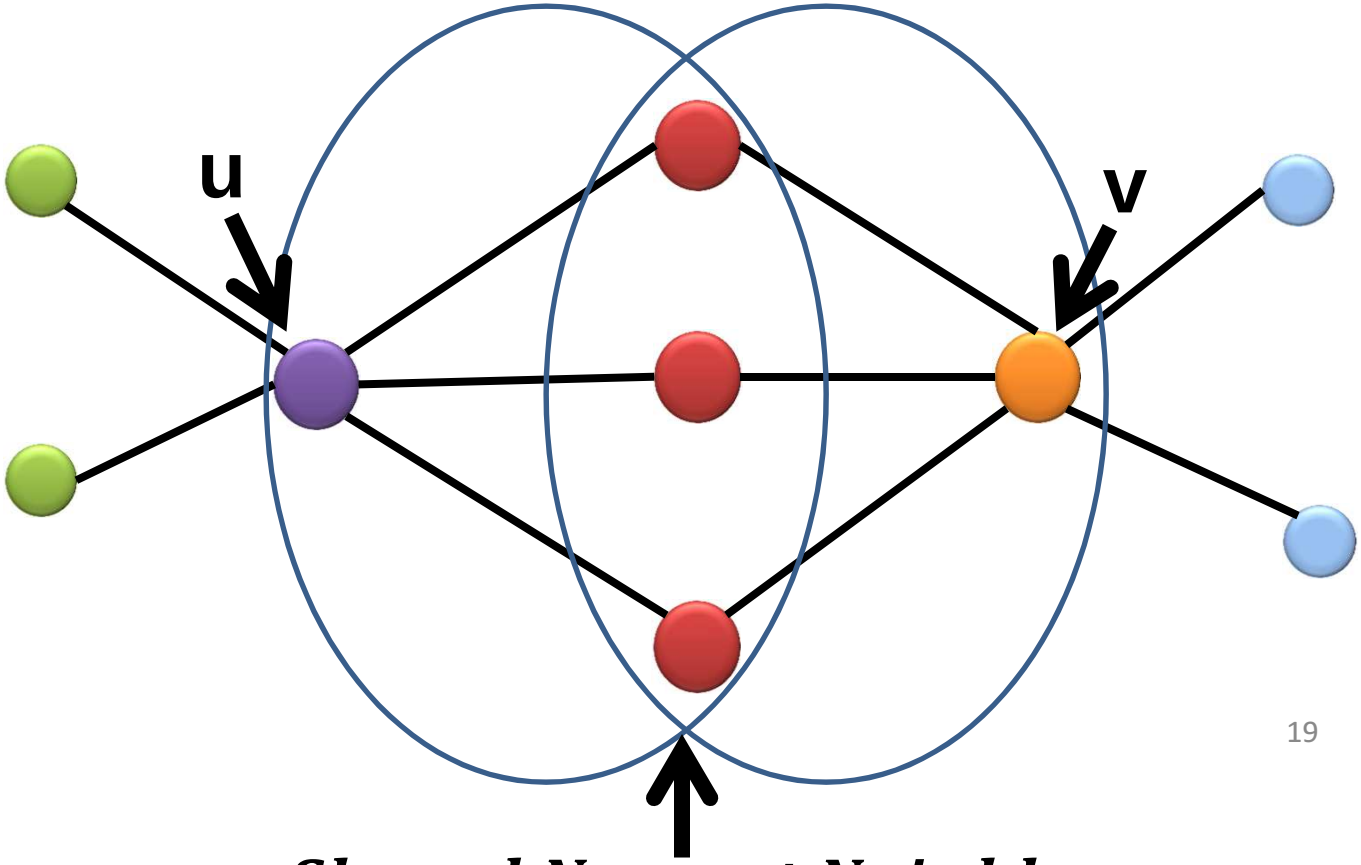
↑ **τ**

STEPS:

- **Obtains the Shared Nearest Neighbor Graph (SNN) of input graph G**
- **Removes edges from the SNN with weight less than τ**

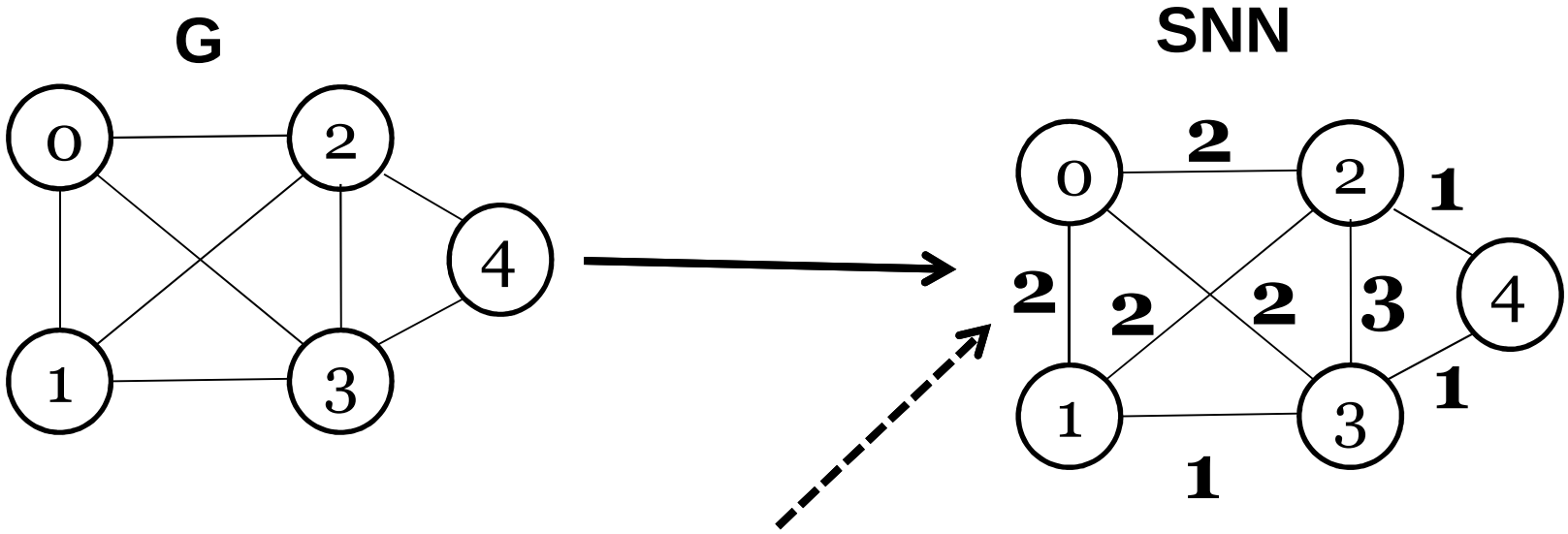
What is Shared Nearest Neighbor? (Refresher from Proximity Chapter)

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

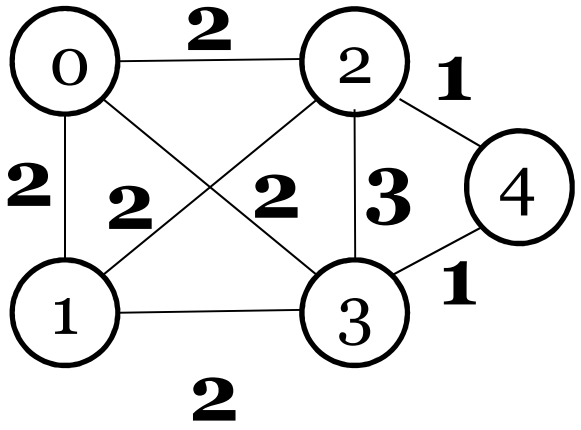
Given input graph G, weight each edge (u,v) with the number of shared nearest neighbors between u and v



Node 0 and Node 1 have 2 neighbors in common: Node 2 and Node 3

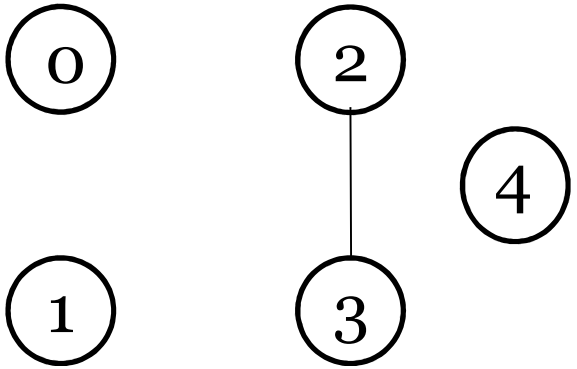
Shared Nearest Neighbor Clustering
Jarvis-Patrick Algorithm

SNN graph of input graph G



If u and v share more than τ neighbors
Place them in the same cluster

E.g., τ =3



SNN-Clustering R code

- library(GraphClusterAnalysis)
- library(RBGL)
- library(igraph)
- library(graph)
- data(SNN_Example)
- G = graph.data.frame(SNN_Example,directed=FALSE)
- tkplot(G)

- Output = SNN_Clustering(G,3)
- 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 Betweenness Centrality? (Refresher from Proximity Chapter)

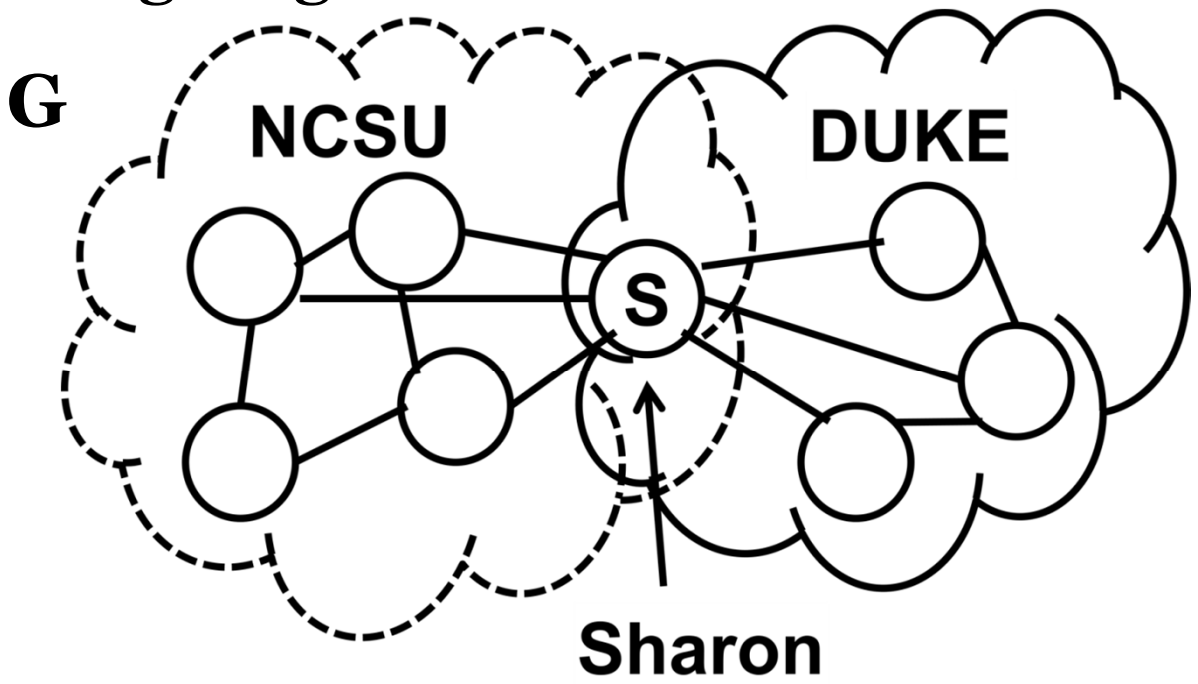
Betweenness centrality quantifies the degree to which a vertex (or edge) occurs on the shortest path between all the other pairs of nodes

Two types:

- **Vertex Betweenness**
- **Edge Betweenness**

Vertex Betweenness

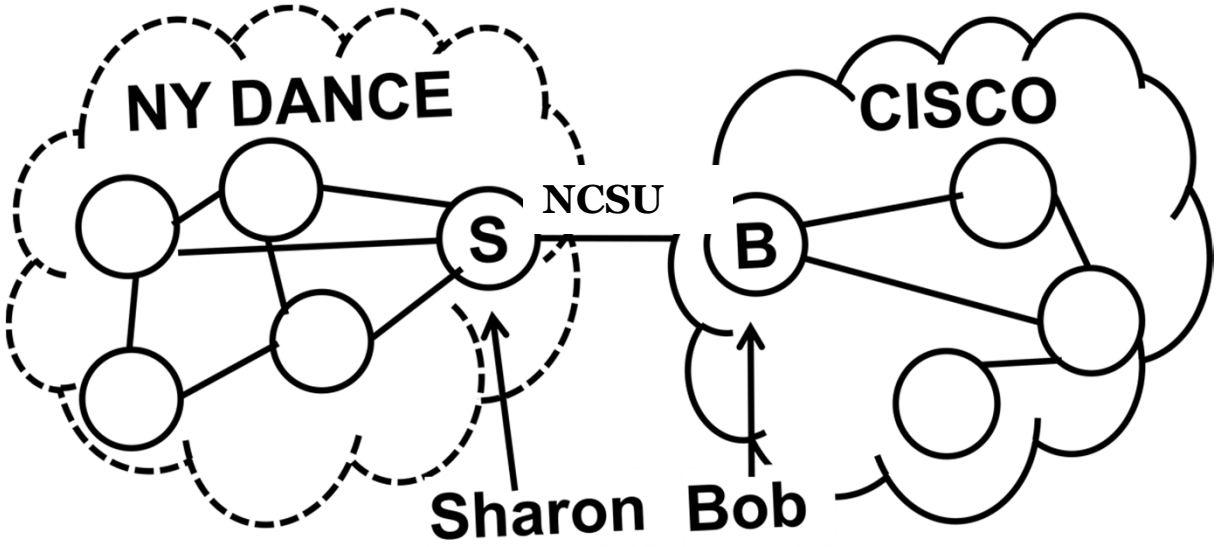
The number of **shortest paths in the graph G** that pass through a given node **S**



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)

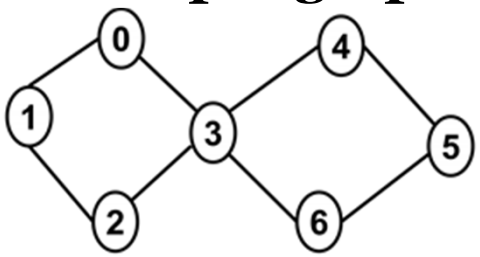


E.g., Sharon and Bob both study at NCSU and they are the only link between NY DANCE and CISCO groups

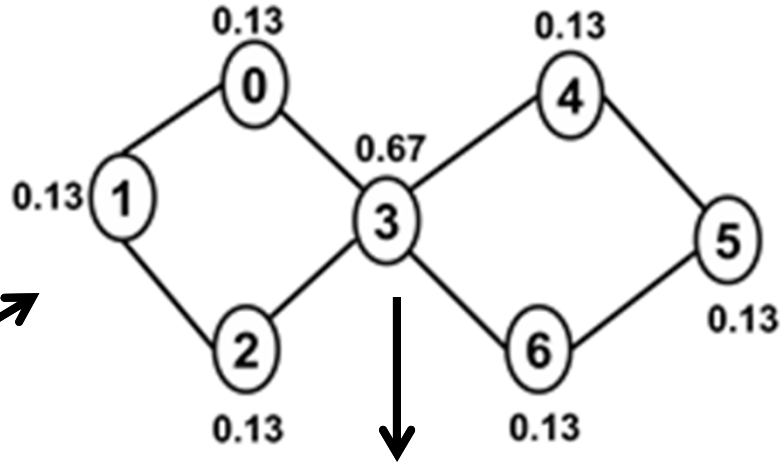
Vertices and Edges with high Betweenness form good starting points to identify clusters

Vertex Betweenness Clustering

Given Input graph G

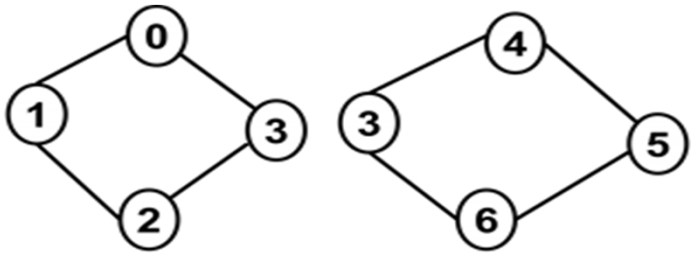


Betweenness for each vertex



Select vertex v with the highest betweenness
E.g., Vertex 3 with value 0.67

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



Repeat until highest vertex betweenness ≤ μ

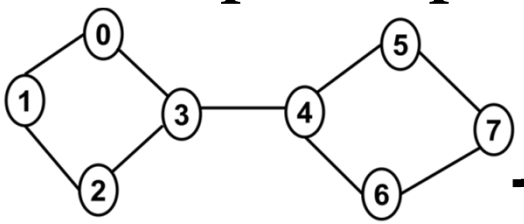
OAK RIDGE National Laboratory

Vertex Betweenness Clustering R code

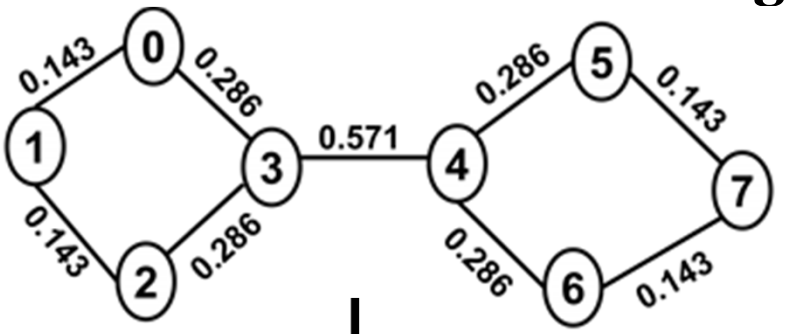
- library(GraphClusterAnalysis)
- library(RBGL)
- library(igraph)
- library(graph)
- data(Betweenness_Vertex_Example)
- G = graph.data.frame(Betweenness_Vertex_Example,directed=FALSE)
- betweennessBasedClustering(G,mode="vertex",threshold=0.2)

Edge-Betweenness Clustering
Girvan and Newman Algorithm

Given Input Graph G

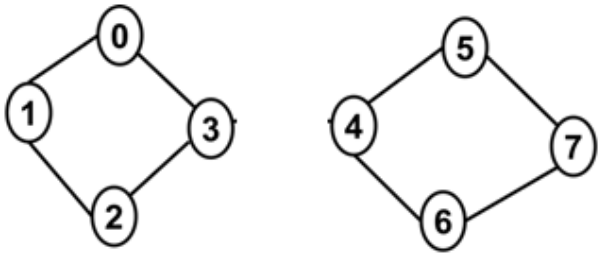


Betweenness for each edge



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

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



Repeat until highest edge betweenness ≤ μ

OAK RIDGE National Laboratory

Edge Betweenness Clustering R code

- library(GraphClusterAnalysis)
- library(RBGL)
- library(igraph)
- library(graph)
- data(Betweenness_Edge_Example)
- G = graph.data.frame(Betweenness_Edge_Example,directed=FALSE)
- betweennessBasedClustering(G,mode="edge",threshold=0.2)

Outline

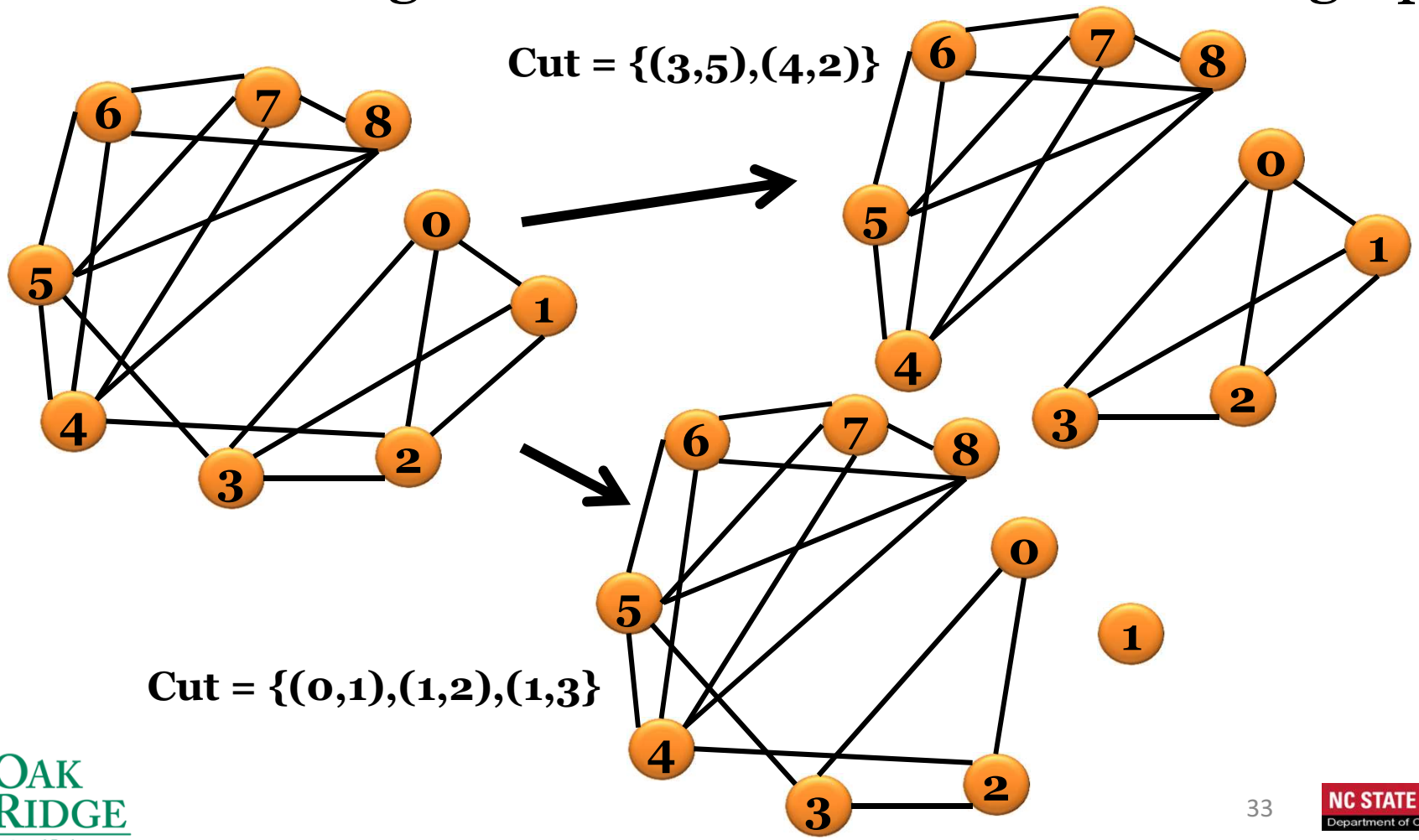
- Introduction to Clustering
- Introduction to Graph Clustering
- Algorithms for Within Graph Clustering
  - ❑ k-Spanning Tree
  - ❑ Shared Nearest Neighbor Clustering
  - ❑ Betweenness Centrality Based
  - ❑ Highly Connected Components
  - ❑ Maximal Clique Enumeration
  - ❑ Kernel k-means
- Application

What is a Highly Connected Subgraph?

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

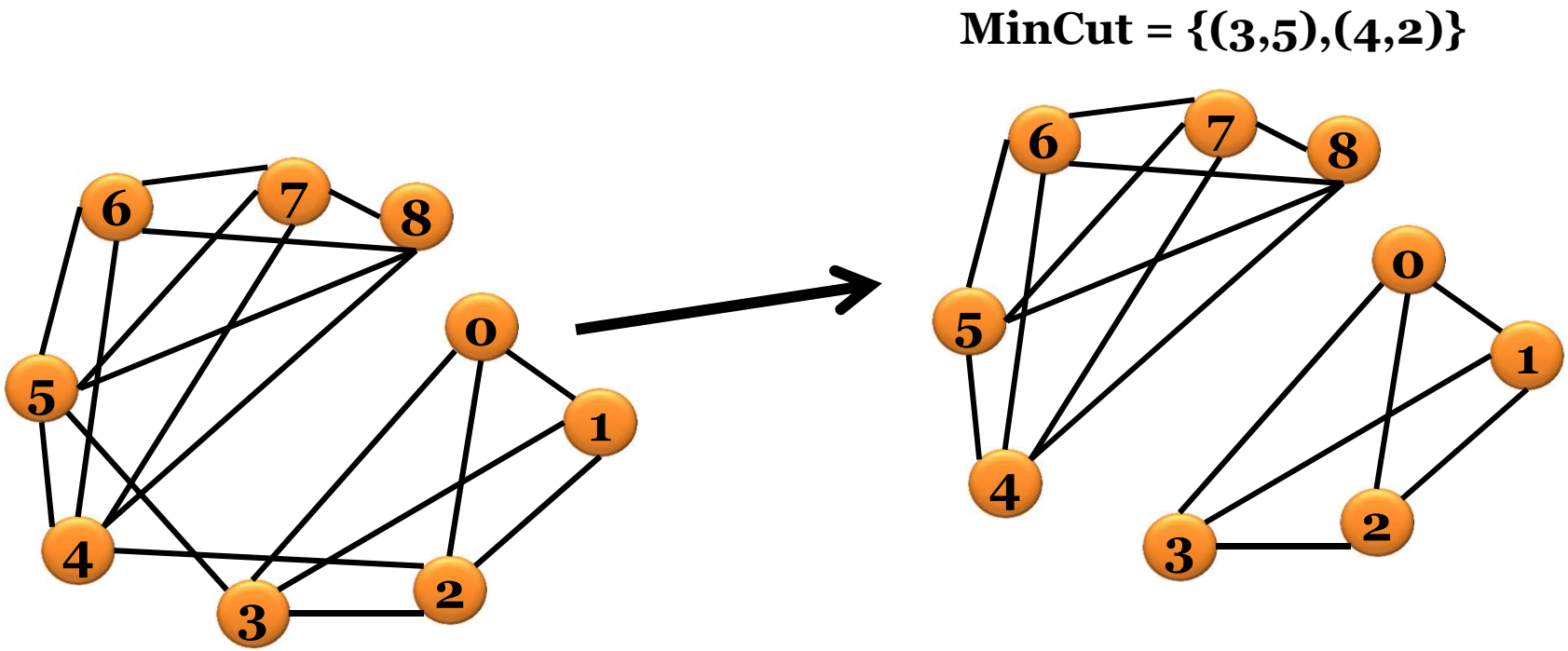
Cut

- The set of edges whose removal disconnects a graph



Minimum Cut

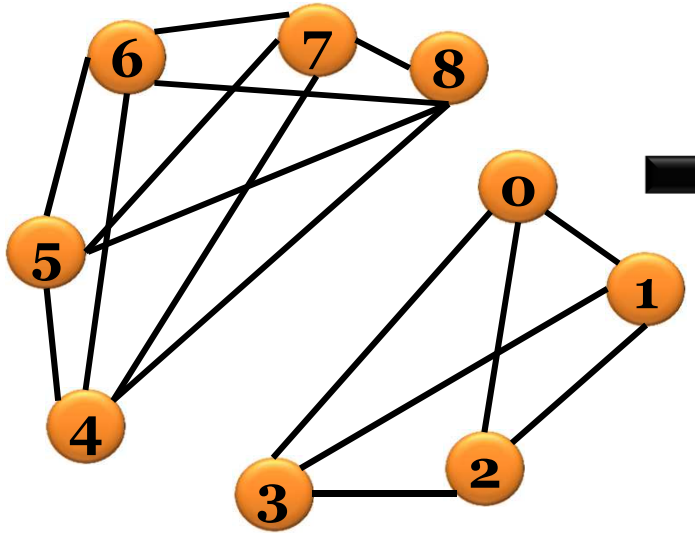
The minimum set of edges whose removal disconnects a graph



Edge Connectivity (EC)

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

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

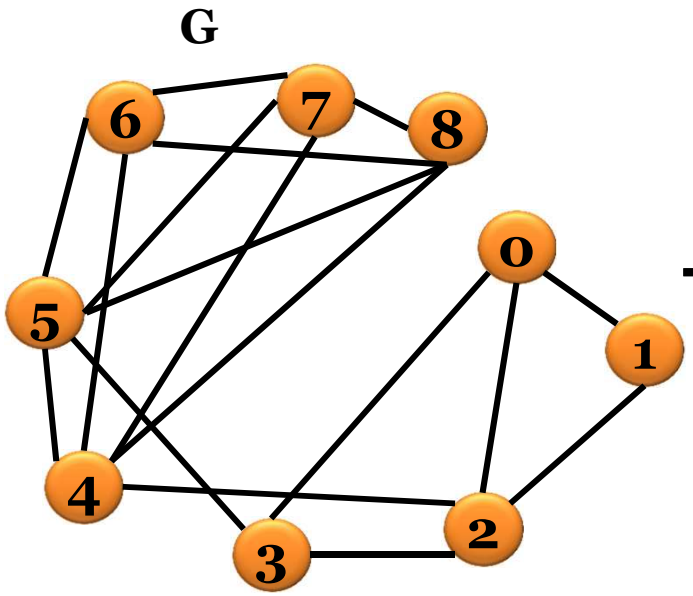


Edge Connectivity

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

Highly Connected Subgraph (HCS)

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

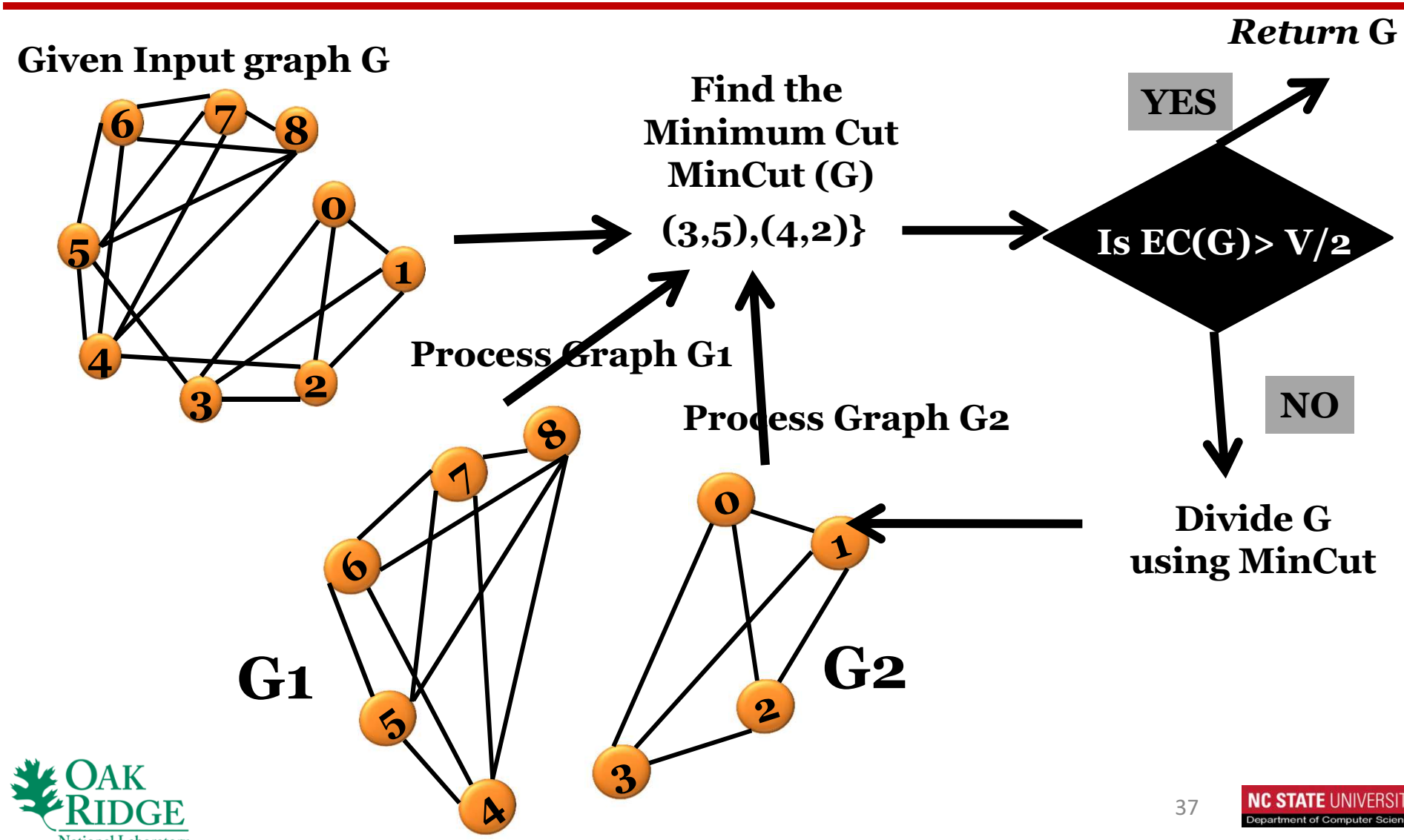


$EC(G) \not> V/2$

$2 \not> 9/2$

G is NOT a highly connected subgraph

HCS Clustering



HCS Clustering R code

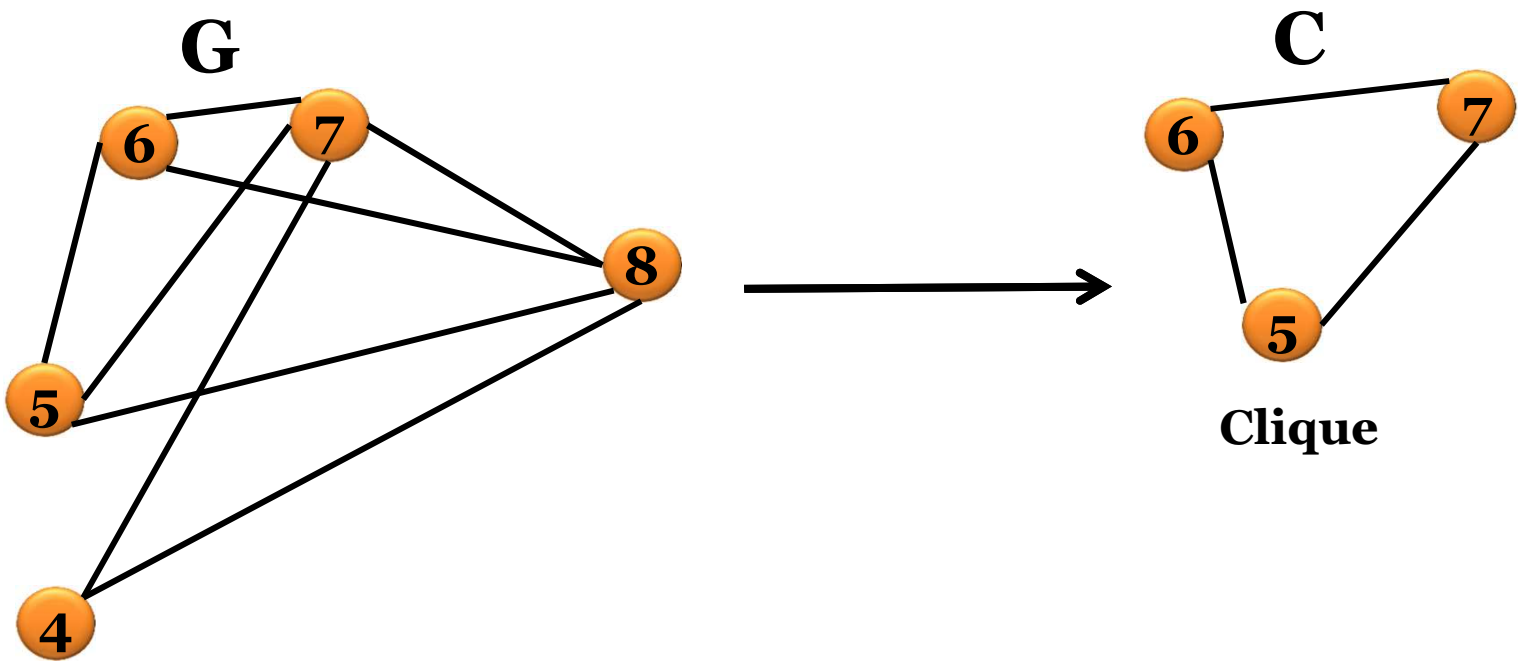
- library(GraphClusterAnalysis)
- library(RBGL)
- library(igraph)
- library(graph)
- data(HCS_Example)
- G = graph.data.frame(HCS_Example,directed=FALSE)
- HCSClustering(G,kappa=2)

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

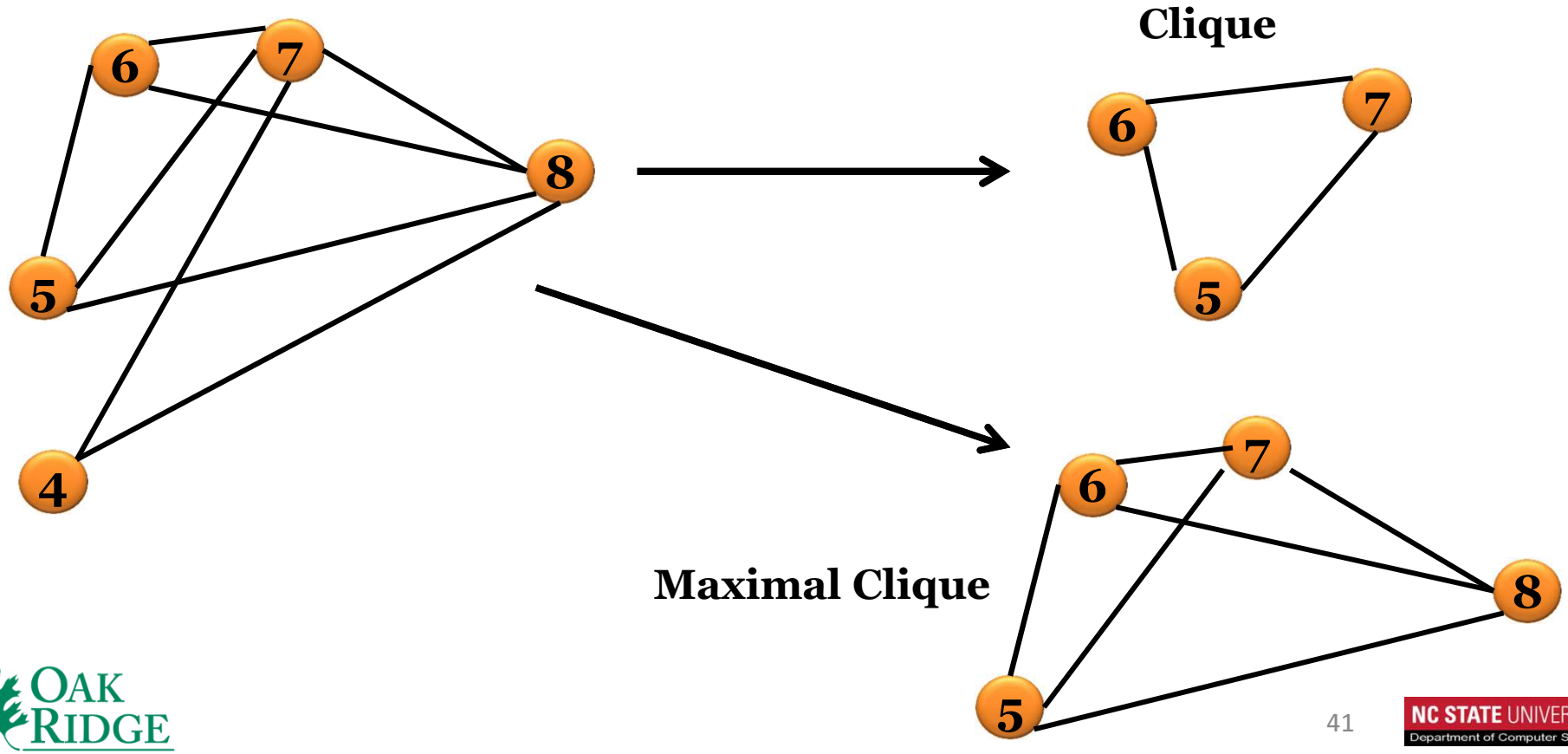
What is a Clique?

A subgraph C of graph G with edges between all pairs of nodes



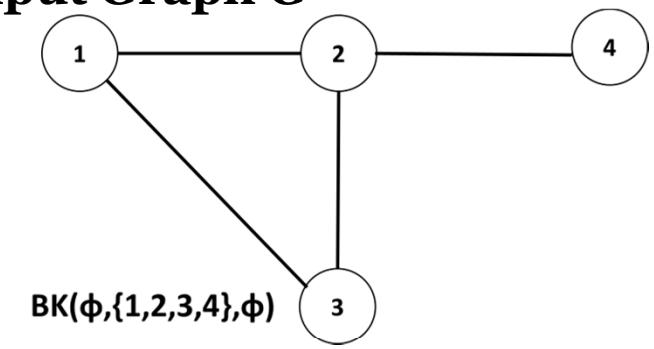
What is a Maximal Clique?

A maximal clique is a clique that is not part of a larger clique.



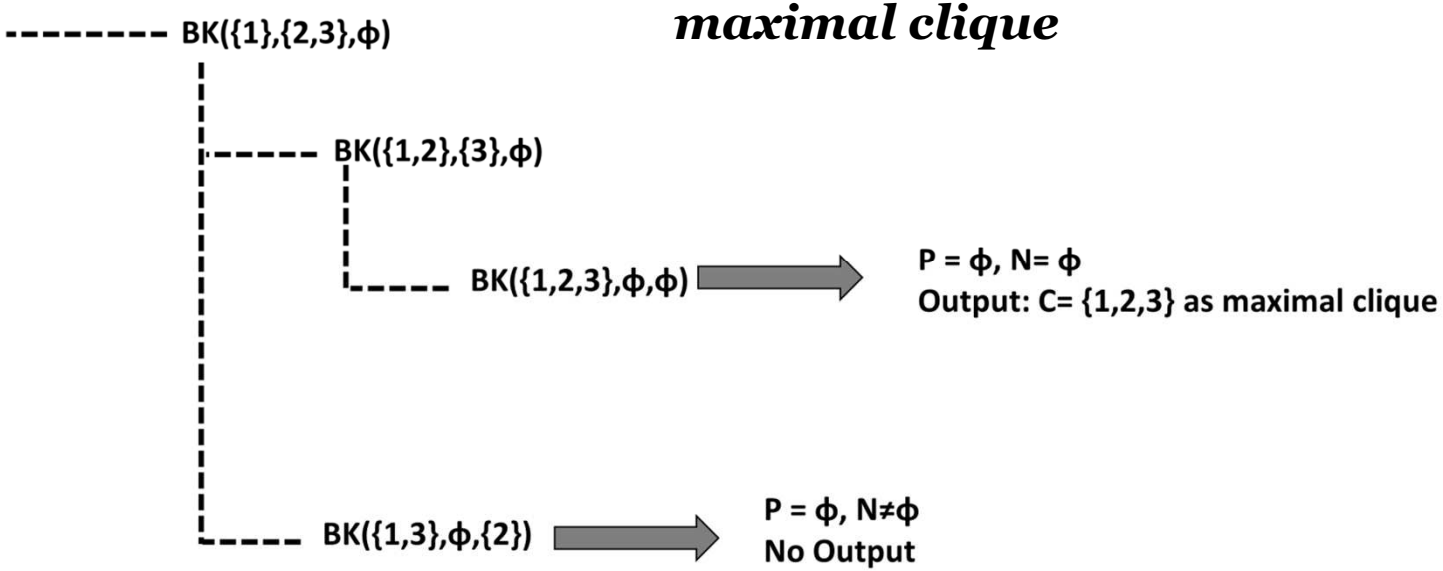
Maximal Clique Enumeration
Bron and Kerbosch Algorithm

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



Maximal Clique R code

- library(GraphClusterAnalysis)
- library(RBGL)
- library(igraph)
- library(graph)
- data(CliqueData)
- G = graph.data.frame(CliqueData,directed=FALSE)
- tkplot(G)
- maximalCliqueEnumerator (G)

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" (recall Kernel Chapter)
- "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

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

Application

- **Functional modules** in protein-protein interaction networks
- Subgraphs with pair-wise interacting nodes => Maximal cliques

R-code

- library(GraphClusterAnalysis)
- library(RBGL)
- library(igraph)
- library(graph)
- data(YeasPPI)
- G = graph.data.frame(YeasPPI,directed=FALSE)
- Potential_Protein_Complexes = maximalCliqueEnumerator (G)
- Potential_Protein_Complexes