Graph Clustering/Partitioning

Andrea Smith

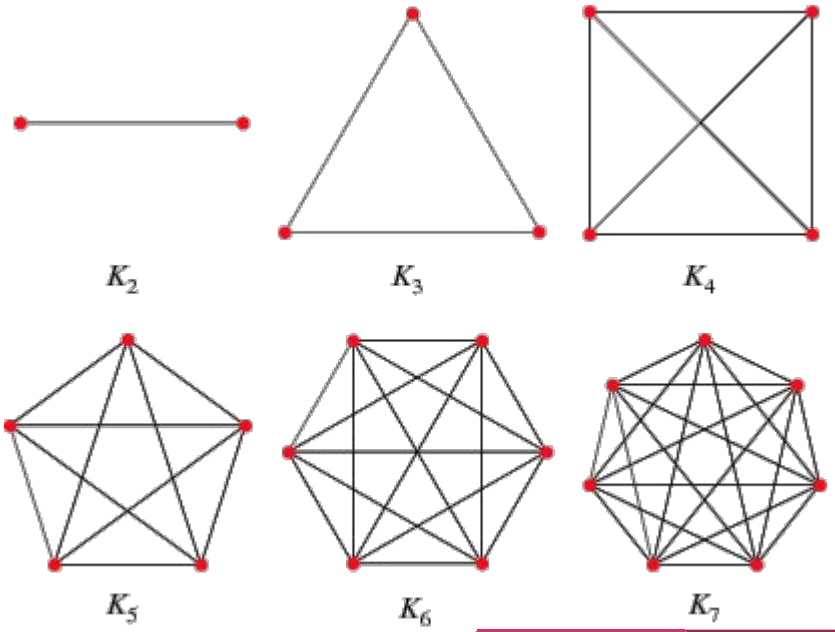
Definitions

Graph G = (V, E), where V is a set of nodes and E is the set of edges between nodes

The degree of a node v ∈ V is the number of edges connected to v

A clique is a set of nodes such that each node has an edge connecting it to all other nodes

Density of G: D(G) = 2|E| / (|V|(|V| - 1))



Definitions Cont’d

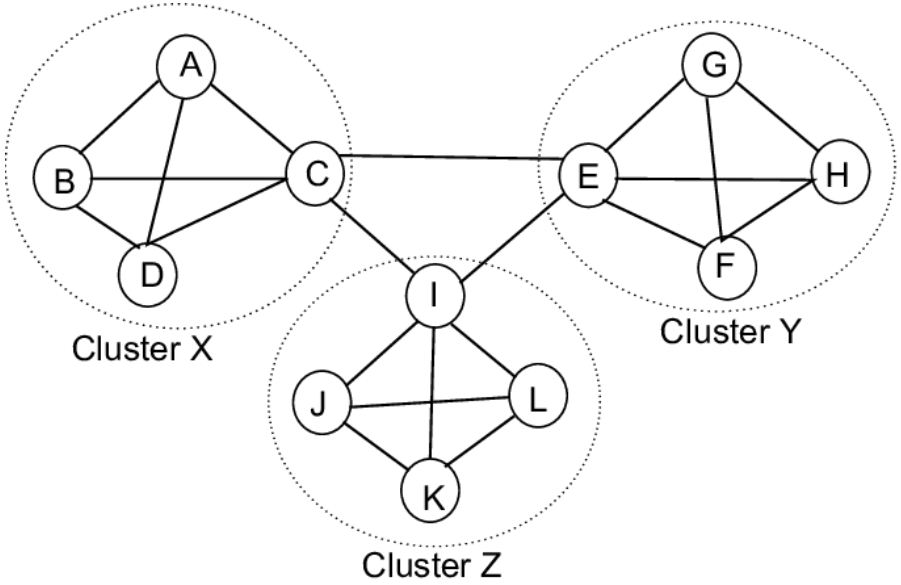
Distance: shortest path between two nodes

Diameter: longest shortest path

Graph Clustering: Partition a graph G such that within each cluster exists relatively high density compared to connections between clusters

Density(X) = 12 /12 = 1

Density(X ∩ Y) = 26 / 56 = 0.46



Distance-k Cliques

1. Create initial clusters around highest degree nodes
2. Create potential cluster by combining two neighbor clusters
    a. If diameter of new cluster is ≤ k, accept cluster
    b. Otherwise, discard combination
3. Continue to grow cluster until it cannot combine with any neighbors
4. Repeat 2-3 with a different base cluster

Modifications exists for growing multiple clusters at once

Multilevel Algorithm

1. Coarsening the graph
   - Combining nodes until the graph is sufficiently smaller
2. Initial clustering
   - Spectral algorithm
3. Refining
   - Remove coarseness and refine clusters

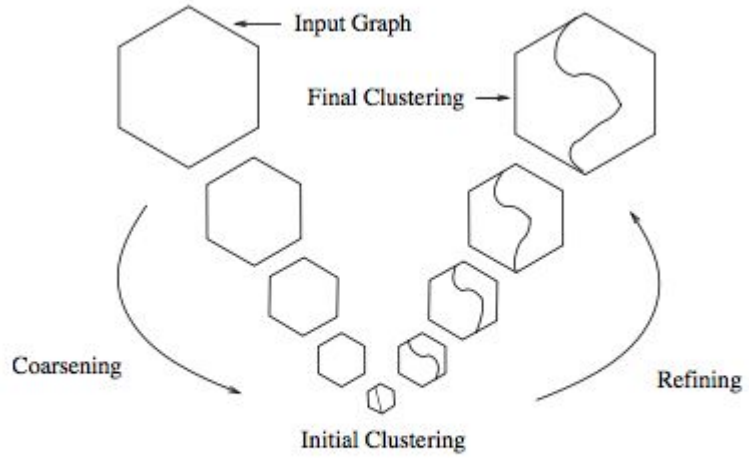


Figure 1: Overview of the multilevel algorithm (for $k = 2$).

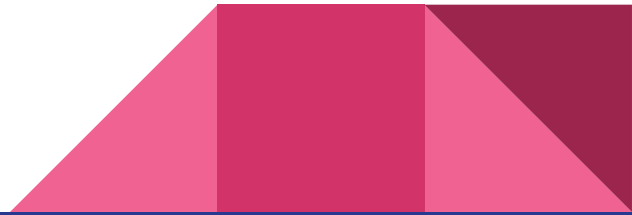
Coarsening

Repeat until the graph has ≤ 20k nodes remaining (k is the desired number of clusters)

1. Start with all nodes unmarked
2. Select unmarked node randomly
   a. If node has no unmarked neighbors, mark it
   b. Else, combine with an unmarked neighbor and mark both nodes, maximizing

$$(e(x,y) / w(x)) + (e(x,y) / w(y))$$

3. Repeat until all nodes marked

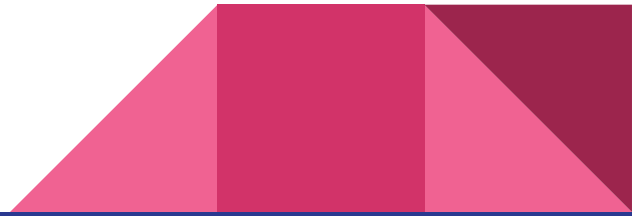


Initial Clustering: Spectral Algorithms

Clusters are based on eigenvectors formed from an adjacency matrix

- Allows for dealing with intertwined spirals

1. Create the nearest neighbor matrix (W)
2. Create the Laplacian matrix (L = D - W)
    a. D is the diagonal matrix of node degrees
3. Calculate the first k eigenvectors
    a. k is the number of desired clusters
4. Create a matrix of the eigenvectors and apply k-means



Refinement

1. Step backwards through graphs created during coarsening
2. At each level, set initial clusters such that if a supernode is in cluster c, then all nodes that form that supernode are in cluster c
3. Refine clusters
   a. Since clusters are close to what they should be, algorithm should resolve quickly
   b. Suggested algorithm is a combination of k-means and spectral algorithm

ALGORITHM 1: Refinement Algorithm.

WEIGHTED_KERNEL_KMEANS(K, k, w, t_{max}, $\{\pi_c^{(0)}\}_{c=1}^k$, $\{\pi_c\}_{c=1}^k$)

Input: K: kernel matrix, k: number of clusters, w: weights for each point, t_{max}: optional maximum number of iterations, $\{\pi_c^{(0)}\}_{c=1}^k$: optional initial clustering

Output: $\{\pi_c\}_{c=1}^k$: final clustering of the points

1. If no initial clustering is given, initialize the k clusters $\pi_1^{(0)}, \ldots, \pi_k^{(0)}$ randomly. Set $t = 0$.
2. For each row i of K and every cluster c, compute

$$d(i, \mathbf{m}_c) = K_{ii} - \frac{2\sum_{j \in \pi_c^{(t)}} w_j K_{ij}}{\sum_{j \in \pi_c^{(t)}} w_j} + \frac{\sum_{j,l \in \pi_c^{(t)}} w_j w_l K_{jl}}{(\sum_{j \in \pi_c^{(t)}} w_j)^2}.$$

3. Find $c^*(i) = \text{argmin}_c d(i, \mathbf{m}_c)$, resolving ties arbitrarily. Compute the updated clusters as

$$\pi_c^{(t+1)} = \{i : c^*(i) = c\}.$$

4. If not converged or $t_{max} > t$, set $t = t + 1$ and go to Step 3; Otherwise, stop and output final clusters $\{\pi_c^{(t+1)}\}_{c=1}^k$.

References

[1] “A Fast Kernel-based Multilevel Algorithm for Graph Clustering”, Dhillon I., Guan Y., Kulis B.. 2005. University of Texas at Austin.

[2] “Spectral Graph Clustering”, Auffarth B.. 2007. University of Barcelona.

[3] “Graph Clustering using Distance-k Cliques”, Edachery J., Sen A., Brandenburg F.. 1999. Arizona State University.