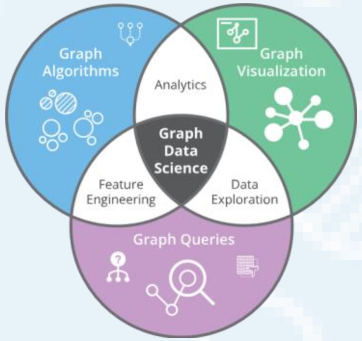


Habib University
shaping futures

CS343 Graph Data Science

Spring 2024

Community Detection

Chapter #6, Mark

Muhammad Qasim Pasta

qasim.pasta@sse.habib.edu.pk

Slides are intended to be filled during the lectures. Certain details are intentionally omitted for in-class discussions. These slides are not meant to be used as reading material.

Community Detection

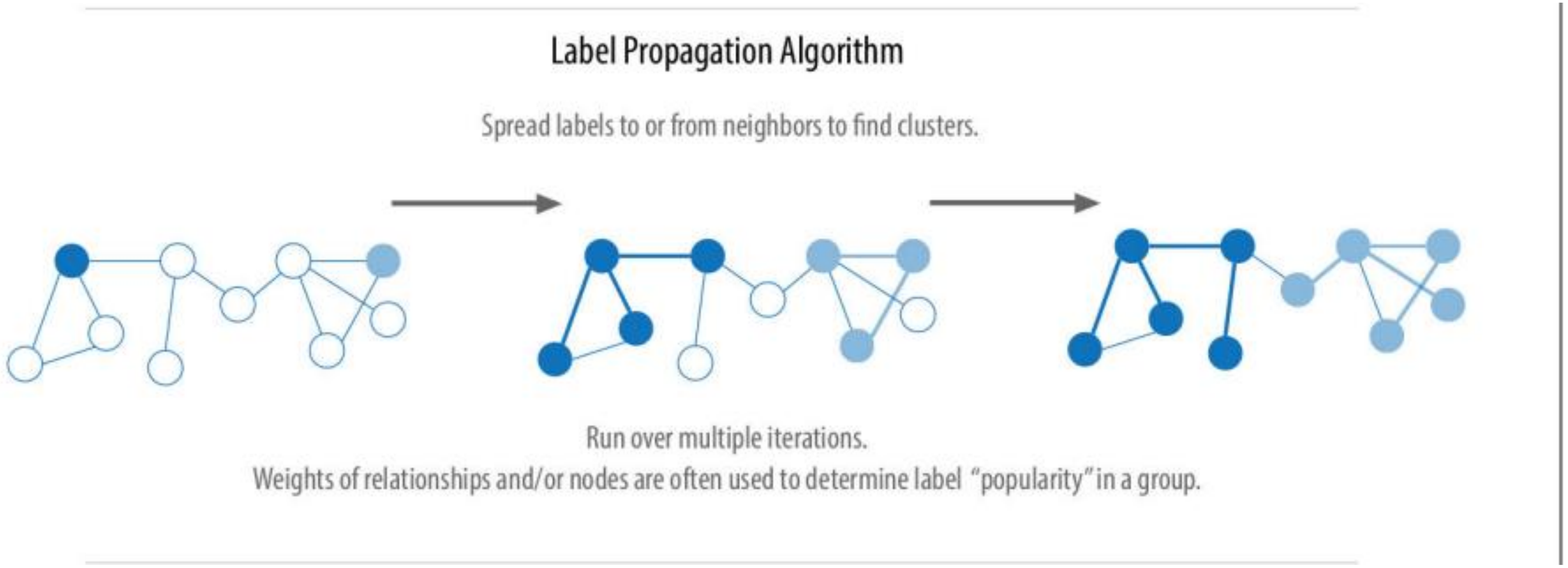
- Identify groups of nodes in a network that are more densely connected internally than with the rest of the network.
- Use underlying structure or organization within a network by partitioning it into cohesive groups or communities.
- Different than Clustering in Data Mining
  - Community detection algorithms can identify fraud rings or networks by detecting clusters of accounts with suspicious transactions or shared identifiers.
  - By clustering customer interactions and identifying common patterns, organizations can create comprehensive customer profiles that aggregate data from various touchpoints.
  - dividing a target market into distinct subgroups or segments based on shared characteristics.

Algorithms

- Weakly Connected Components
  - Finds groups where each node is reachable from every other node in that same group, regardless of the direction of relationships
  - Identify islands
- Strongly Connected Components
  - Finds groups where each node is reachable from every other node in that same group following the direction of relationships
  - Making product recommendations based on group affiliation or similar items
- Label Propagation
- Louvain Modularity

Label Propagation

- Infers clusters by spreading labels based on neighbourhood majorities
- Understanding consensus in social communities or finding dangerous combinations of possible co-prescribed drugs



Label Propagation Algorithm

Spread labels to or from neighbors to find clusters.

Run over multiple iterations.

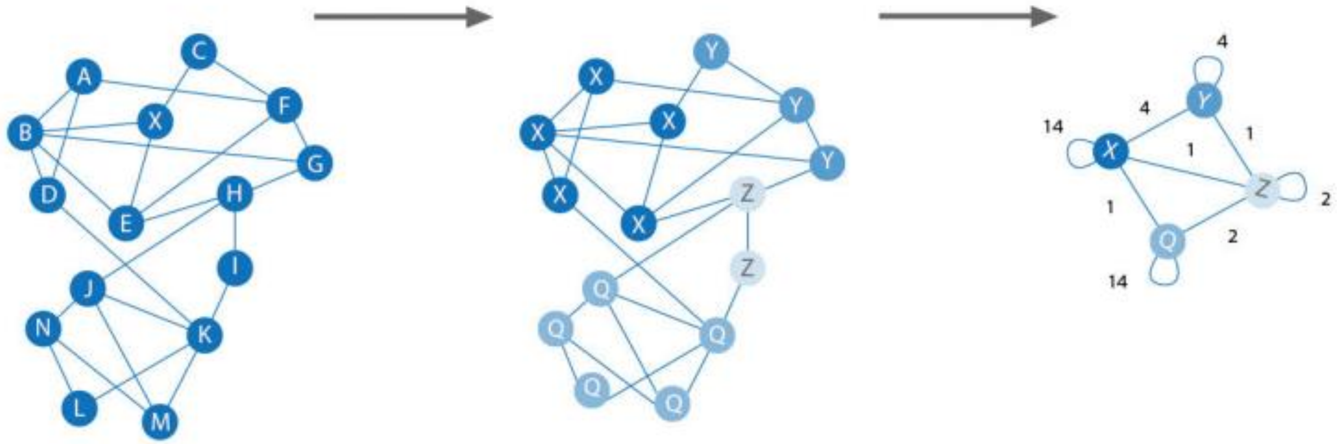
Weights of relationships and/or nodes are often used to determine label “popularity” in a group.

Louvain Modularity

- Maximizes the presumed accuracy of groupings by comparing relationship weights and densities to a defined estimate or average
- n fraud analysis, evaluating whether a group has just a few discrete bad behaviours or is acting as a fraud ring

Louvain Modularity Algorithm

Find clusters by moving nodes into higher relationship density groups and aggregating into supercommunities.



Run over multiple iterations.

Relationship weights and totals are used to determine grouping.

Syntax

- Weakly Connected Component

```
CALL gds.wcc.stream('myGraph’)
YIELD nodeId, componentId
RETURN gds.util.asNode(nodeId).name AS name, componentId
ORDER BY componentId, name
```

- Strongly Connected Component

```
CALL gds.scc.stream('graph', {})
YIELD nodeId, componentId
RETURN gds.util.asNode(nodeId).name AS Name, componentId AS Component
ORDER BY Component DESC
```

Syntax

- Louvain

```
CALL gds.louvain.stream('myGraph')
YIELD nodeId, communityId, intermediateCommunityIds
RETURN gds.util.asNode(nodeId).name AS name, communityId
ORDER BY name ASC
```

- Label Propagation

```
CALL gds.labelPropagation.stream('myGraph')
YIELD nodeId, communityId AS Community
RETURN gds.util.asNode(nodeId).name AS Name, Community
ORDER BY Community, Name
```

Combining

- Mutate property to write back to projection

```
CALL gds.louvain.mutate('test', {mutateProperty:'communityId’})
```

- Finding nodes and their communities

```
CALL gds.graph.nodeProperty.stream('test','communityId', ['Person'])
YIELD nodeId, propertyValue
WITH gds.util.asNode(nodeId) AS n, propertyValue AS communityId
WHERE n:Person
RETURN n.name, communityId LIMIT 10
```