Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Department of Computer Science and Engineering (Data Science)

**Subject: Social Network Analysis Laboratory (DJ19DSL8014)**

**AY: 2024-25**

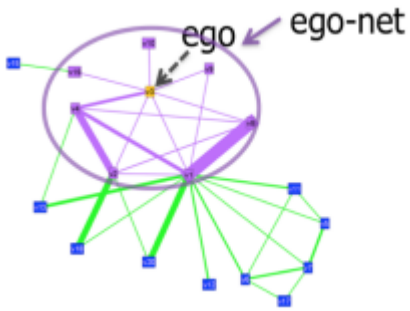
*Bhuvi Ghosh*
*60009210191*

**Experiment 8**

**Aim:** Anomaly detection of static graph using oddball technique.

**Theory:**

1. Introduction




A feature-based anomaly detection technique called ODDBALL is proposed by [Akoglu et al.,], which extracts egonet-based features and finds patterns that most of the egonets of the graph follow with respect to those features. As such, this method can spot anomalous egonets (and hence anomalous nodes), as those that do not follow the observed patterns. An egonet is defined as the 1-step neighborhood around a node; including the node, its direct neighbors, and all the connections among these nodes (an example is shown on the right figure). More formally an egonet is the induced 1-step sub-graph for each node. Given the egonets, the main question and challenge is which features to look at, as there is a long list of possible graph-based measures that can be extracted as egonet features. The paper proposes a carefully chosen subset of features (e.g. number of triangles, total weight of edges, etc.) that are (1) observed to yield patterns across a wide range of real-world graphs, and fast to compute and easy to interpret. The egonet features are then studied in pairs and several patterns in the form of power-laws are observed among strongly related features (e.g. number of neighbors and number of triangles). For a given egonet, its deviation from a particular pattern is computed based on its "distance" to the relevant power-law distribution. Each egonet then receives a separate deviation, or outlierness, score with respect to each pattern.

Following are the steps:

Step 1: Define Egonets for Each Node

- For each node in the network, define its egonet, which includes the node, its direct neighbors, and all the edges among them.

Step 2: Feature Extraction for Each Egonet

- Ni: Calculate the number of neighbors for each node.
- Ei: Count the number of edges within the egonet.
- Wi: Compute the total weight of the egonet, which could represent the strength or frequency of interactions between nodes within the egonet.
- $\lambda_{w,i}$: Determine the principal eigenvalue of the weighted adjacency matrix of the egonet.

Step 3: Identify Patterns and Compute Deviations

- E vs N (CliqueStar): Detect patterns indicative of cliques or star structures by analyzing the relationship between Ei and Ni.
- W vs E (HeavyVicinity): Identify egonets with unusually strong or frequent interactions by comparing Wi against Ei.
- $\lambda_w$ vs W (DominantPair): Look for egonets where one connection is significantly stronger than others by analyzing the relationship between $\lambda_{w,i}$ and Wi.

Step 4: Fit Power-Laws and Measure Deviation

- Using the identified patterns, fit power-law distributions to these feature pairs across the network.
- For each egonet, measure how much it deviates from the expected power-law distribution for each feature pair.

Step 5: Assign Outlierness Scores

- Assign a score that quantifies the deviation for each egonet, with respect to each pattern.
- Egonets with high deviation scores are those that significantly differ from the typical pattern and may be considered anomalous.

Step 6: Anomaly Detection

- Establish a threshold or criteria to decide which egonets are anomalous based on their outlierness scores.
- Flag egonets (and consequently nodes) that exceed this threshold as anomalies.

Step 7: Post-Analysis

- Investigate the flagged anomalies to understand their context and implications.
- Depending on the type of network and the domain (social, biological, informational, etc.), determine the appropriate response to these anomalies.

Step 8: Refinement and Iteration

- Refine the process by potentially updating feature selection, pattern identification, and threshold definition based on the insights gained from the anomalies detected.
- Iterate the process as the network evolves or as new data becomes available to continually monitor for anomalies.

**Lab Assignment**

Apply the Oddball technique on any network, constructed at random or any other network, to detect graph anomalies.

## Department of Computer Science and Engineering (Data Science)

[1] 2s
```
import networkx as nx
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as stats
```

[2] 0s
```
G = nx.barabasi_albert_graph(n=100, m=3)
```

0s
```
node_ids = []
ego_sizes = []
ego_edges = []

for node in G.nodes():
    ego = nx.ego_graph(G, node)
    n = ego.number_of_nodes()
    e = ego.number_of_edges()

    if n > 1:
      node_ids.append(node)
      ego_sizes.append(n)
      ego_edges.append(e)
```

[4] 0s
```
x = np.log(ego_sizes)
y = np.log(ego_edges)
slope, intercept, r_value, p_value, std_err = stats.linregress(x, y)
```

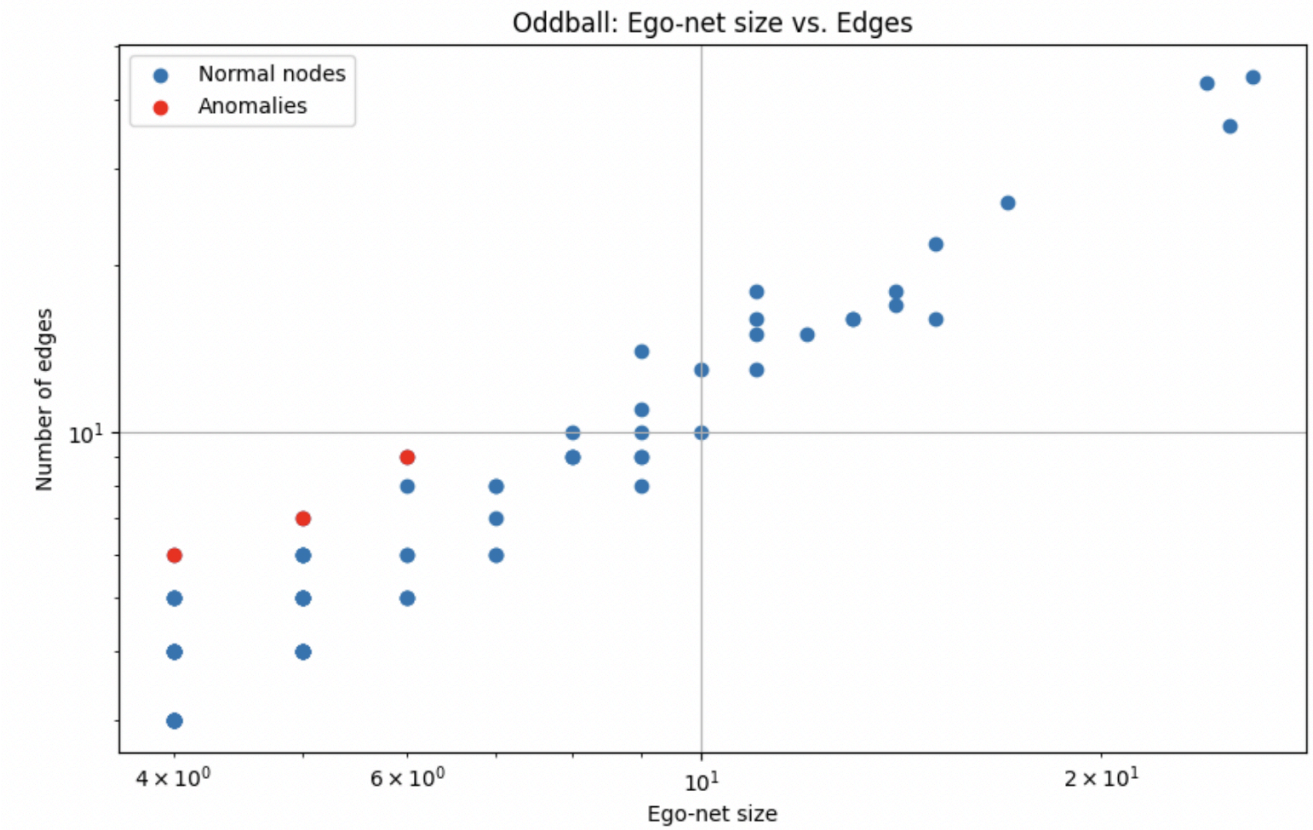
[4] 0s
```
x = np.log(ego_sizes)
y = np.log(ego_edges)
slope, intercept, r_value, p_value, std_err = stats.linregress(x, y)
```

[5] 0s
```
y_pred = slope * x + intercept
residuals = y - y_pred
```

[6] 0s
```
threshold = 2 * np.std(residuals)
anomalies = [node_ids[i] for i in range(len(residuals)) if abs(residuals[i]) > threshold]
```

0s
```
plt.figure(figsize=(10,6))
plt.scatter(ego_sizes, ego_edges, label="Normal nodes")
plt.scatter([ego_sizes[i] for i in range(len(ego_sizes)) if node_ids[i] in anomalies],
            [ego_edges[i] for i in range(len(ego_edges)) if node_ids[i] in anomalies],
            color='r', label='Anomalies')
plt.xlabel("Ego-net size")
plt.ylabel("Number of edges")
plt.title("Oddball: Ego-net size vs. Edges")
plt.legend()
plt.xscale('log')
plt.yscale('log')
plt.grid(True)
plt.show()

print(f"Anomalous nodes detected: {anomalies}")
```




```
Anomalous nodes detected: [14, 20, 89]
```

**Conclusion:** The ODDBALL technique was effectively applied to detect anomalies in a static graph by analyzing egonet-based features and their power-law relationships. Nodes with egonets that significantly deviated from expected patterns were flagged as anomalies. This method proved efficient, interpretable, and suitable for unsupervised anomaly detection in network data.