Echo Chamber Detection in Twitter Networks

A Graph-Theoretic Approach via Weighted Undirected Subgraph

Muhammad Haris Putra Sulastianto - 13524053
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: mharisputras.work@gmail.com , 13524053@std.stei.itb.ac.id

***Abstract*—Social media platforms like X (formerly known as Twitter) indirectly support the formation of echo chambers through their algorithms. An echo chamber refers to a group of individuals who mostly interact only within their own group, typically with others who share the same opinions. As a result, their beliefs are reinforced, and they rarely receive input from outside perspectives. This paper presents an approach to detecting echo chambers on the X platform using weighted undirected graph theory, by identifying densely connected subgraphs with high interaction weights from interaction data. In this context, each node represents an individual account, each edge represents an interaction between users, and each edge weight represents the frequency of interactions between the two nodes. Echo chambers are modeled as weighted undirected subgraphs with high internal edge density and low external edge density. The result of this paper is the application of a community detection algorithm on synthetic interaction data to demonstrate the concept on the X platform.**

Keywords—echo chamber; graph theory; community detection algorithm; weighted undirected subgraph

I. Introduction

Social media is a discussion platform that almost everyone in the world uses. Most social media algorithms show users content (such as FYP or recommended posts) based on what they have previously followed, liked, mentioned, or otherwise interacted with. This can lead to the formation of echo chambers, especially on platforms often used for discussion like X.

An echo chamber is a group where people mostly interact with others who think the same way they do. Over time, this can strengthen their beliefs and reduce open discussion. Understanding and identifying echo chambers is important to see how conversations on social media grow, how strong opinions might become more extreme, and to prevent polarization that could lead to conflict.

In this paper, we try to detect echo chambers using a weighted undirected subgraph approach. We represent social media as a weighted graph where each vertex is a user account, each edge represents an interaction such as a reply, mention, or retweet, and each edge weight represents the frequency of interactions between the two nodes (accounts), where the frequency increases by 1 each time node A and node B interact mutually.

Echo chambers appear as tightly connected user groups in a weighted undirected subgraph, where internal connections have significantly higher interaction frequencies compared to connections outside the group. This reflects how users tend to engage more often with others who share similar views rather than with people outside their group who hold different opinions, forming isolated clusters. To demonstrate this concept, we apply a community detection algorithm approach on synthetic interaction data, finding subgraphs with high internal interaction density and analyzing the distribution of edge weights directed inward to the group compared to those extending outward from the group.

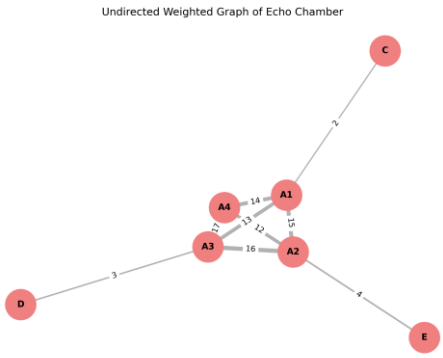

Figure 1. Echo chamber representation in a graph

This introduction explains the basic idea of how we use graph-based detection to find possible echo chambers in online social networks, specifically in the context of X (Twitter).

II. Graph Theory

Graph is a mathematical structure used to model pairwise relations between objects. It consists of two primary components: vertices (also called nodes) and edges (also called links). A graph G can be defined as an ordered pair (V, E) where V is a set of vertices and E is a set of edges, where each edge is a pair of vertices from V.

A. Basic Concepts in Graph Theory

Some basic ideas in graph theory are vertices, edges, and loops.

- Vertex or Node

A vertex (also called a node or junction) is a point in a graph where edges meet or connect. In graph theory, it is one of the

main parts that make up a graph. Vertices can be connected to each other by edges, and they are usually labeled with letters, numbers, or a mix of both.

- Edge or Links

In graph theory, an edge is a line that connects two vertices, forming a link between them. A vertex can have multiple edges, but each edge must connect a starting vertex to an ending vertex to be valid. Edges can be **directed**, meaning they have a specific direction, or **undirected**, meaning they do not. They are also commonly called lines, branches, arcs, or links. When two directed edges exist between the same pair of vertices in opposite directions, it is like having one undirected edge. Edges are essential in mathematics for connecting vertices and building relationships in a graph.

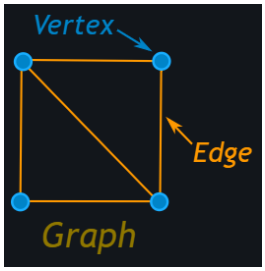

Figure 2. Vertex and edge in a graph

https://www.mathsisfun.com/sets/graph-theory.html

- Multiple Edges

In graph theory, multiple edges — also called parallel edges — are two or more edges that link the same pair of vertices. A graph that permits more than one edge between the same two vertices is known as a **multigraph**.

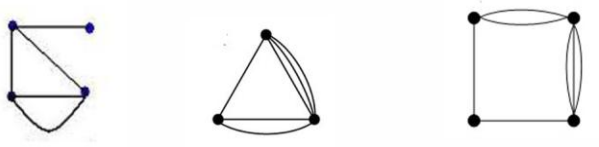

Figure 3. Multi-graph

https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf

- Loop

A loop is a special kind of edge in a graph where both ends connect to the same vertex. In other words, when an edge begins and ends at the same point, it is called a loop.

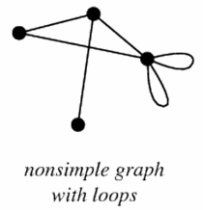

Figure 4. Loop in a graph

https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf

B. Types of Graphs

Graph theory includes several kinds of graphs, such as null graphs, trivial graphs, simple graphs, undirected graphs, directed graphs, weighted graphs, complete graphs, and bipartite graphs.

- Null Graph

A null graph (or empty graph) is a graph that contains one or more vertices but has no edges at all. In other words, while the vertex set V is not empty, the edge set E is completely empty.

For example, a null graph with four vertices would have:

Vertex set V: $\{A, B, C, D\}$, Edge set E: $\{\}$ or $\emptyset$

Since there are no edges, each vertex in a null graph has a degree of zero.

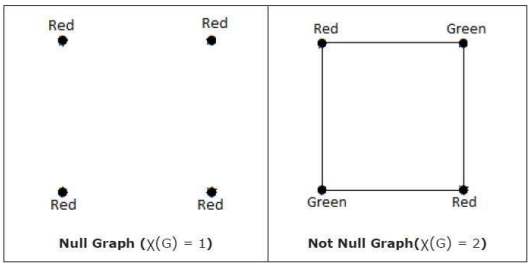

Figure 5. Null graph

https://educativesite.com/line-graph-empty-graph/

- Trivial Graph

A trivial graph is the most basic form of a graph, containing only a single vertex and no edges.

For instance, in a trivial graph:

Vertex set V: $\{A\}$, Edge set E: $\{\}$ or $\emptyset$

- Simple Graph

A simple graph is a graph where each pair of vertices is connected by at most one edge, and no vertex is connected to itself. In other words, it contains no loops or multiple edges between the same pair of vertices.

For example, a simple graph with four vertices may have,

Vertex set V: $\{A, B, C, D\}$, Edge set E:
$\{\{A, B\}, \{A, C\}, \{B, D\}, \{C, D\}\}$

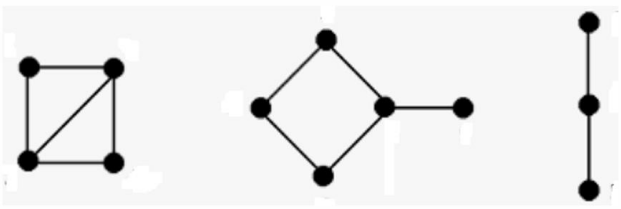

Figure 6. Simple Graph

https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf

- Undirected Graph

An undirected graph is a graph where the edges do not have a specific direction. This implies that the connection between two linked vertices goes both ways. In such graphs, the edge (u, v) is considered the same as (v, u), indicating a mutual relationship between the vertices.

- Directed Graph

A directed graph, or **digraph**, is a graph in which each edge has a specific direction. This means every edge goes from a source vertex to a destination vertex, showing a one-way connection between the two vertices.

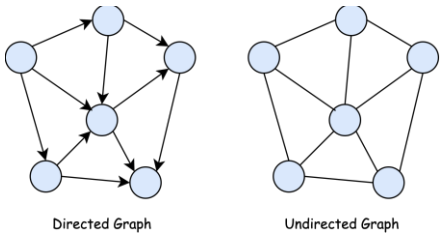

Figure 7. Directed and undirected graph

https://cs226fa21.github.io/notes/26-graph/step05.html

- Weighted Graphs

A weighted graph is a graph where each edge carries a numerical value called a weight or cost. These weights can represent things like distance, expense, capacity, or other measures that describe the strength or significance of the connection between vertices.

For example a weighted graph with four vertices,

Vertex set V: $\{A, B, C, D\}$, Edge set E:
$\{(A, B, 4), (A, C, 10), (A, D, 2), (C, B, 4)\}$

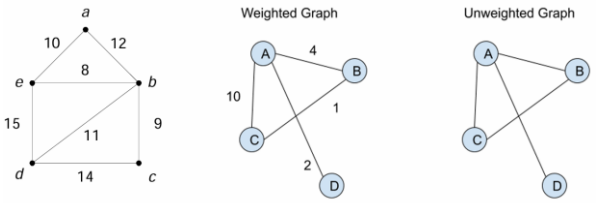

Figure 8. Weighted and unweighted graph

https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf

- Complete Graph

A graph is called complete if every vertex is connected to every other vertex in the graph. In other words, all possible edges between vertices are present. A complete graph with n vertices is usually denoted as K_n.

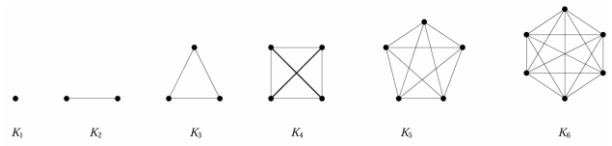

Figure 9. Complete graph

https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf

- Bipartite Graphs

A bipartite graph is a graph in which the vertices can be split into two separate groups, where no vertices within the same group are directly connected. This means that each edge connects a vertex from one group to a vertex in the other group.

For example a bipartite graph with vertex sets,

Vertex set V_1: $\{A, B\}$, Vertex set V_2: $\{C, D, E\}$

Edge set E: $\{\{A, C\}, \{B, C\}, \{A, D\}, \{B, E\}\}$

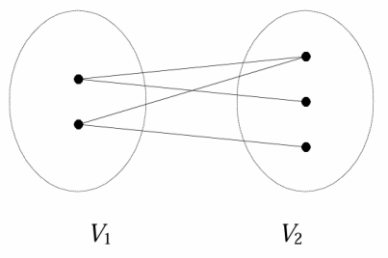

Figure 10. Bipartite graph

https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf

- Cycle Graph

A cycle graph, sometimes called a circular graph, is a graph that forms one continuous loop. In this graph, every vertex is connected to exactly two other vertices, resulting in a closed circular path.

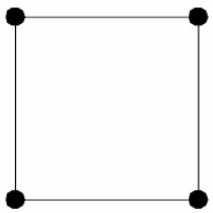

Figure 11. Cycle graph

https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf

C. Representations of Graphs

Besides using graphical form, graphs can also be represented in other important ways, such as adjacency matrix, adjacency list, and incidence matrix.

- Adjacency Matrix

An adjacency matrix is a method to represent a graph using a two-dimensional array of size $n \times n$, where n is the number of vertices. Each element $a[i][j]$ in the matrix shows whether an edge exists between vertex i and vertex j. In the case of weighted graphs, the matrix entry can hold the weight of the edge instead of just 0 or 1.

For example an undirected graph with four vertices,

Vertex set $V = \{1, 2, 3, 4\}$

Edge set $E = \{(1, 3), (2, 3), (4, 3), (1, 2), (4, 2)\}$

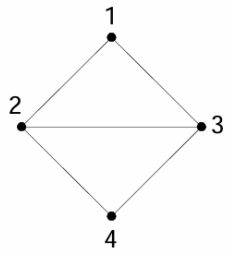

Figure 12. Example four vertices graph for adjacency matrix and adjacency list

https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf

The adjacency matrix for Figure 12 graph is:

Table 1. Adjacency matrix

	$v1$	$v2$	$v3$	$v4$
$v1$	0	1	1	0
$v2$	1	0	1	1
$v3$	1	1	0	1
$v4$	0	1	1	0

- Adjacency List

An adjacency list represents a graph by storing, for each vertex, a list of the other vertices it is connected to. This method is especially efficient for sparse graphs, where the total number of edges is significantly lower than the square of the number of vertices.

The adjacency list representation for Figure 12 graph is:

Vertex	Adjacent Vertices
1	2, 3
2	1, 3, 4
3	1, 2, 4
4	2, 3

- Incidence Matrix

An incidence matrix is a type of graph representation that displays the connection between vertices and edges. It uses a two-dimensional array of size $n \times m$, where n is the number of vertices and m is the number of edges. Each entry in the matrix indicates whether a particular vertex is connected to a specific edge.

The incidence matrix for Figure 12 graph is:

Table 2. Incidence matrix

Vertex / Edge	e_1 (1, 2)	e_2 (1, 3)	e_3 (3, 4)	e_4 (2, 3)	e_5 (2, 4)
1	1	1	0	0	0
2	1	0	0	1	1
3	0	1	1	1	0
4	0	0	1	0	1

D. Community Detection and Modularity Optimization

Community detection in graph theory refers to the task of identifying groups of nodes that are more densely connected to each other than to the rest of the network. In online social networks, these communities often reflect real-world social clusters, such as groups of friends, interest-based communities, or ideological echo chambers.

A common approach for detecting such communities is based on **modularity optimization**. In the **weighted version** of modularity, which is more suitable for social interaction data, edge weights reflect the **strength of relationships** (e.g., frequency or intensity of interaction). The modularity function measures how much the actual distribution of edge weights within communities deviates from a random distribution with the same node strengths.

The **modularity value** Q for a weighted graph is defined as:

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{s_i s_j}{2w} \right] \delta(c_i, c_j)$$

Where:

- A_{ij} is the weight of the edge between node i and node j,
- s_i and s_j are the strengths of nodes i and j, i.e., the sum of their incident edge weights,
- w is the total weight of all edges in the network,
- $\delta(c_i, c_j)$ equals 1 if nodes i and j are in the same community, and 0 otherwise.

A higher modularity score indicates that more edge weight falls within communities than would be expected by chance, suggesting a strong community structure.

One efficient algorithm that uses this approach is the **Fast-Greedy Modularity Optimization algorithm**. It begins by assigning each node to its own community and then iteratively

merges communities in a way that increases the overall modularity score, until no further improvement is possible. This method is computationally efficient and suitable for large-scale social networks.

In this study, we apply this approach to identify groups of users who interact more frequently within their own group than with others, which is a typical pattern observed in echo chambers on social media platforms.

III. Methodology

A. Data Modelling

In this study, a synthetic dataset was constructed to simulate interaction patterns typically found in echo chambers. Each node represents a user, and each edge represents mutual interaction between users. The weight of an edge indicates the frequency or intensity of the interaction. Internal connections within communities are stronger (higher weights) than external connections.

We used two datasets:

- Data 1, strong internal interactions representing a synthetic echo chamber.
- Data 2, randomized connections to simulate a weak or flat social network structure.

B. Graph Construction

Let's say we have a set of interaction data defined as a list of weighted edges, where each edge is represented by a tuple (u, v, w) with u and v being the interacting users and w the interaction weight. An example of such strong echo chamber is shown in Figure 13 and weak echo chamber is shown in Figure 14.

```
edges = [
    # 🔵 Community A (Echo Chamber A)
    ('Luna', 'Rex', 6), ('Luna', 'Zane', 5), ('Luna', 'Mira', 4),
    ('Rex', 'Zane', 5), ('Rex', 'Mira', 6), ('Zane', 'Mira', 5),
    ('Zane', 'Kira', 4), ('Mira', 'Kira', 5),

    # 🔴 Community B (Echo Chamber B)
    ('Nova', 'Axel', 6), ('Nova', 'Sage', 4), ('Nova', 'Lex', 5),
    ('Axel', 'Sage', 4), ('Axel', 'Lex', 5), ('Sage', 'Lex', 6),
    ('Lex', 'Tara', 3), ('Sage', 'Tara', 4),

    # 🟢 Community C (Echo Chamber C)
    ('Yuna', 'Vega', 6), ('Yuna', 'Kai', 5), ('Yuna', 'Niko', 5),
    ('Vega', 'Kai', 4), ('Kai', 'Niko', 5), ('Niko', 'Orin', 4),

    # 🌐 Cross-community interactions (weaker connections)
    ('Rex', 'Axel', 1), ('Zane', 'Yuna', 1), ('Nova', 'Orin', 2),
    ('Luna', 'Kai', 1), ('Tara', 'Kira', 2),
]
```

Figure 13. Synthetic data 1 strong community data using python

```
edges = [
    ('Luna', 'Nova', 6), ('Luna', 'Axel', 5), ('Luna', 'Yuna', 4),
    ('Rex', 'Kai', 6), ('Rex', 'Lex', 5), ('Rex', 'Mira', 4),
    ('Zane', 'Sage', 5), ('Zane', 'Tara', 6), ('Zane', 'Kira', 4),
    ('Nova', 'Zane', 5), ('Nova', 'Kira', 4), ('Nova', 'Orin', 6),
    ('Kai', 'Vega', 5), ('Kai', 'Mira', 4), ('Kai', 'Tara', 5),
    ('Lex', 'Yuna', 6), ('Lex', 'Niko', 5), ('Lex', 'Axel', 4),
    ('Sage', 'Mira', 5), ('Sage', 'Yuna', 6), ('Sage', 'Rex', 5),
    ('Tara', 'Orin', 4), ('Tara', 'Kira', 5), ('Tara', 'Luna', 4),
    ('Kira', 'Vega', 5), ('Kira', 'Niko', 4), ('Kira', 'Axel', 5),
    ('Niko', 'Zane', 5), ('Niko', 'Yuna', 6), ('Niko', 'Vega', 5),
    ('Mira', 'Orin', 4), ('Mira', 'Lex', 5), ('Mira', 'Vega', 6)
]
```

Figure 14. Synthetic data 2 weak community data using python

In this study, we used Python along with several libraries to construct and visualize the graph. The `networkx` library was used for creating and manipulating the graph structure, `matplotlib.pyplot` for visualizing the network, and `community` (also known as `community_louvain`) to apply the Louvain method for modularity-based community detection.

```
import networkx as nx
import matplotlib.pyplot as plt
import community as community_louvain  # Louvain method = modularity optimization
```

Figure 15. Python libraries used for graph visualization and community detection.

To make echo chamber detection more reliable, each edge between nodes A and B represents a single mutual interaction. For example, in Figure 16, the connection between A and B on platform X means they both replied to each other, and this interaction is counted once as the edge's weight. This also helps filter out one-sided interactions and keeps the analysis focused on actual two-way communication. Additionally, passive interactions such as likes and shares were not included as edge weights, as they do not reflect active reciprocal engagement.

Figure 16. Example of a mutual interaction between two users, counted one as the edge's weight.

C. Community Detection

To identify echo chambers, we applied modularity-based community detection using the Louvain method, implemented in the `community` package (`community-louvain`). This method iteratively optimizes modularity by grouping nodes into communities with dense internal connections.

```
# 2. Apply modularity-based community detection
partition = community_louvain.best_partition(G, weight='weight')
```

Figure 17. Community detection using python library

```
modularity = community_louvain.modularity(partition, G, weight='weight')
print("Modularity score:", modularity)
```

Figure 18. Modularity score

D. *Visualization*

We visualized the resulting graph and its detected communities using a **spring layout**, where node positions are influenced by interaction strengths. Node colors represent the communities assigned through modularity optimization.

In addition, we also created a separate visualization of the graph **without applying community detection**, using a **random layout** to display the raw structure of the network. This provides a baseline view of the graph before modularity-based clustering is applied.

E. *Assumptions*

- Only mutual (two-way) interactions were considered meaningful and assigned as one edge.
- Edge weights were set higher for intra-community interactions and lower for cross-community interactions.
- Self-loops and isolated nodes were excluded from the graph.
- Passive interactions (e.g., likes and shares) were not included in the edge weights.

IV. Result and Discussion

A. *Graph Visualization Without Community Detection*

To begin, we visualized both graphs using a random layout without applying any community detection algorithm. In this form, all nodes and edges are displayed without grouping or color differentiation, making it difficult to identify any distinct communities or interaction patterns by eye. This visualization is shown in Figure 19 (data 1 strong echo chamber) and Figure 20 (data 2 strong echo chamber).

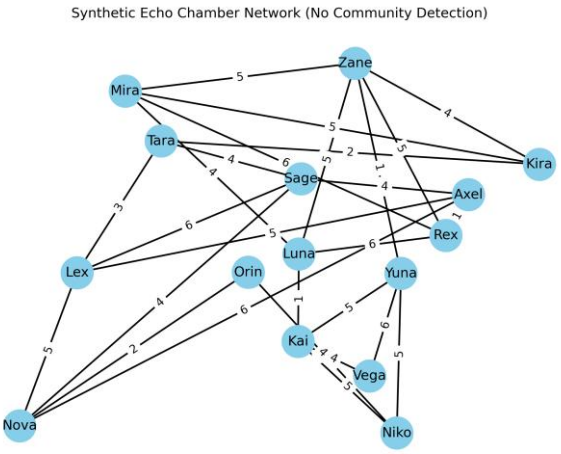

Figure 19. Synthetic Data 1 Strong Echo Chamber Network (No Community Detection)

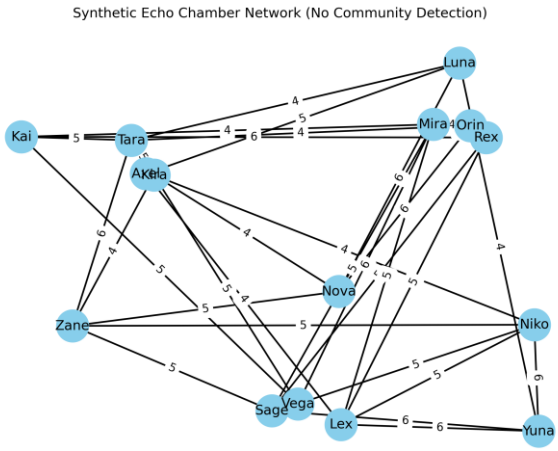

Figure 20. Synthetic Data 2 Weak Echo Chamber Network (No Community Detection)

B. *Community Detection Result*

After applying the **Louvain community detection algorithm** to both graphs, the differences became evident. The first dataset (echo chamber) produced **well-defined communities**, where nodes interacted mostly within their own group. This is illustrated in Figure 21, where distinct color-coded clusters emerge, each representing a potential echo chamber.

In contrast, the second dataset produced **blurry or overlapping community structures**, reflecting a lower modularity score and less clearly separated clusters (see Figure 22). This confirms that **dense intra-group connections** are a strong indicator of echo chamber formation, whereas **more distributed** or **random interactions** lead to weaker community structures.

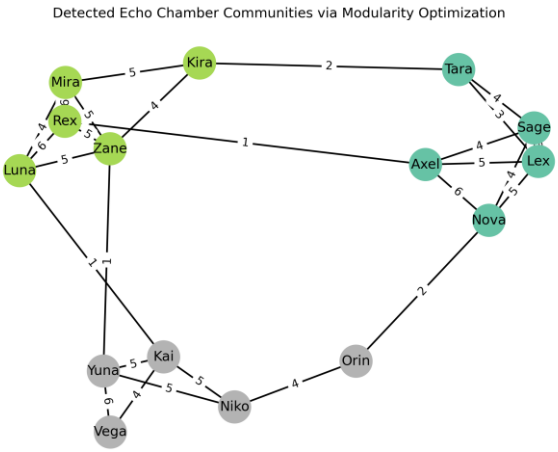

Figure 21. Detecting Echo Chamber Communities via Modularity Optimization (Strong Echo Chamber Dataset)

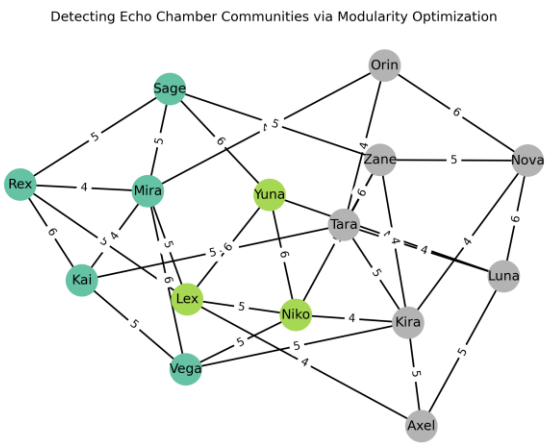

Figure 22. Detecting Echo Chamber Communities via Modularity Optimization (Weak Echo Chamber Dataset)

C. *Modularity Score*

To quantitatively evaluate the strength of community structure in both graphs, we calculated their modularity scores using the Louvain method. The modularity score reflects how well the graph is divided into communities, where higher values indicate denser intra-community connections and sparser inter-community links.

- Data 1 Echo Chamber Graph

```
Modularity score: 0.5991463701151226
```

Figure 23. Modularity score data 1 (strong echo chamber)

Modularity Score = 0.5991.

This high score suggests a strong community structure, consistent with the graph's design—each group has tight internal interactions and only a few weak connections to other groups.

- Data 2 Weak Community Graph

```
Modularity score: 0.29255899732771273
```

Figure 24. Modularity score data 1 (weak echo chamber)

Modularity Score = 0.2926.

This lower score indicates weaker community boundaries. The interactions are more evenly spread, and the detected clusters are less distinct, meaning the network lacks clear separation into echo chambers.

These results support the hypothesis that echo chambers can be identified through high modularity values, representing socially clustered structures with limited external communication.

V. CONCLUSION

A. *Research Conclusion*

This study aimed to detect echo chambers in a social network using a graph-based approach, specifically modularity optimization via the Louvain algorithm. By constructing two synthetic datasets — one with strong intra-community interactions and another with weak or random connections — we were able to compare how graph structure affects community detection.

The results showed that the graph with stronger internal connections yielded a **higher modularity score** (0.5991) and **clearer community separation**, which aligns with the characteristics of an echo chamber. Meanwhile, the graph with weaker or distributed interactions had a **lower modularity score** (0.2926), indicating **less defined communities**. These findings demonstrate that modularity is a useful metric in identifying echo chamber-like structures within social graphs.

Visualizations further supported these findings by showing compact, well-separated clusters in the strong community graph and loosely connected groups in the weak community graph.

B. *Limitiations*

While the results support the effectiveness of modularity-based community detection for identifying potential echo chambers, several limitations remain:

- **Synthetic Dataset**: The networks used in this study were artificially constructed to simulate echo chamber behavior. Although useful for illustrating the method, they may not fully capture the complexity and unpredictability of real-world social interactions.
- **Interaction Assumptions**: Only mutual (two-way) interactions were considered meaningful. Passive actions such as likes or shares were excluded from edge weights. As a result, some subtle influence patterns might be overlooked.
- **Simplified Edge Weights**: Edge weights were assigned based on assumed interaction intensity, not from actual frequency or content of interaction.
- **Echo Chamber ≠ Closed Mindset**: High internal interaction within a community does not always imply a closed mindset. Individuals might frequently communicate within a group simply due to shared interests or friendships, without necessarily rejecting outside perspectives. Therefore, **structural patterns alone cannot fully capture ideological rigidity**, and further qualitative analysis would be needed to confirm the presence of echo chamber behavior.

C. *Future Research Opportunities*

Several directions can be pursued to enhance and expand this research:

- **Real-World Dataset Integration**: Future studies can apply this method to large-scale social media datasets

(e.g., Twitter, Reddit, or Facebook public data) to validate its effectiveness in real environments.

- **Temporal Graph Analysis**: Including the dimension of time can help observe how echo chambers evolve over time.
- **Interaction Type Classification**: Differentiating between types of interactions (e.g., agreement, disagreement, sarcasm) can offer deeper insight into community dynamics.
- **Comparison with Other Algorithms**: Testing and comparing other community detection methods, such as Girvan–Newman or label propagation, may provide complementary perspectives.

VIDEO LINK AT YOUTUBE

https://youtu.be/fXtCmtyZ4hI

SOURCE CODE AND GRAPH IMAGES

https://github.com/MHarisPutraS/Makalah-Matdis.git

ACKNOWLEDGMENT

First and foremost, the author expresses deep gratitude to Allah SWT for the strength, health, and guidance that made this work possible.

The author would also like to sincerely thank Mr. Rinaldi Munir, lecturer of Discrete Mathematics K01 at the School of Electrical Engineering and Informatics (STEI) ITB, for his valuable teaching and inspiration throughout the course.

Lastly, heartfelt appreciation is extended to all friends and the surrounding environment who have consistently supported, encouraged, and provided space for growth during the completion of this paper.

REFERENCES

[1] GeeksforGeeks, "Fundamentals of Graph Theory," GeeksforGeeks, [Online]. Available: https://www.geeksforgeeks.org/maths/fundamentals-of-graph-theory/. [Accessed: 17-Jun-2025].

[2] D. Guichard, "5.01: The Basics of Graph Theory," in *Combinatorics and Graph Theory*, LibreTexts, [Online]. Available: https://math.libretexts.org/Bookshelves/Combinatorics_and_Discrete_Mathematics/Combinatorics_and_Graph_Theory_(Guichard)/05%3A_Graph_Theory/5.01%3A_The_Basics_of_Graph_Theory. [Accessed: 17-Jun-2025].

[3] Educative Site, "Line Graph, Empty Graph," *educativesite.com*, [Online]. Available: https://educativesite.com/line-graph-empty-graph/. [Accessed: 17-Jun-2025].

[4] Stanford University, "Step 5: Graph Communities," CS226 Notes, Fall 2021, [Online]. Available: https://cs226fa21.github.io/notes/26-graph/step05.html. [Accessed: 17-Jun-2025].

[5] R. Munir, "Materi Kuliah Matematika Diskrit," *STEI ITB*, [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis. [Accessed: 17-Jun-2025].

STATEMENT

I hereby declare that the paper I have written is entirely my own work. It is not an adaptation, translation, or plagiarism of someone else's work.

Bandung, 1 June 2025

Muhammad Haris Putra Sulastianto 13524053