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](mailto:mharisputras.work@gmail.com) , [13524053@std.stei.itb.ac.id](mailto: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

# 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.

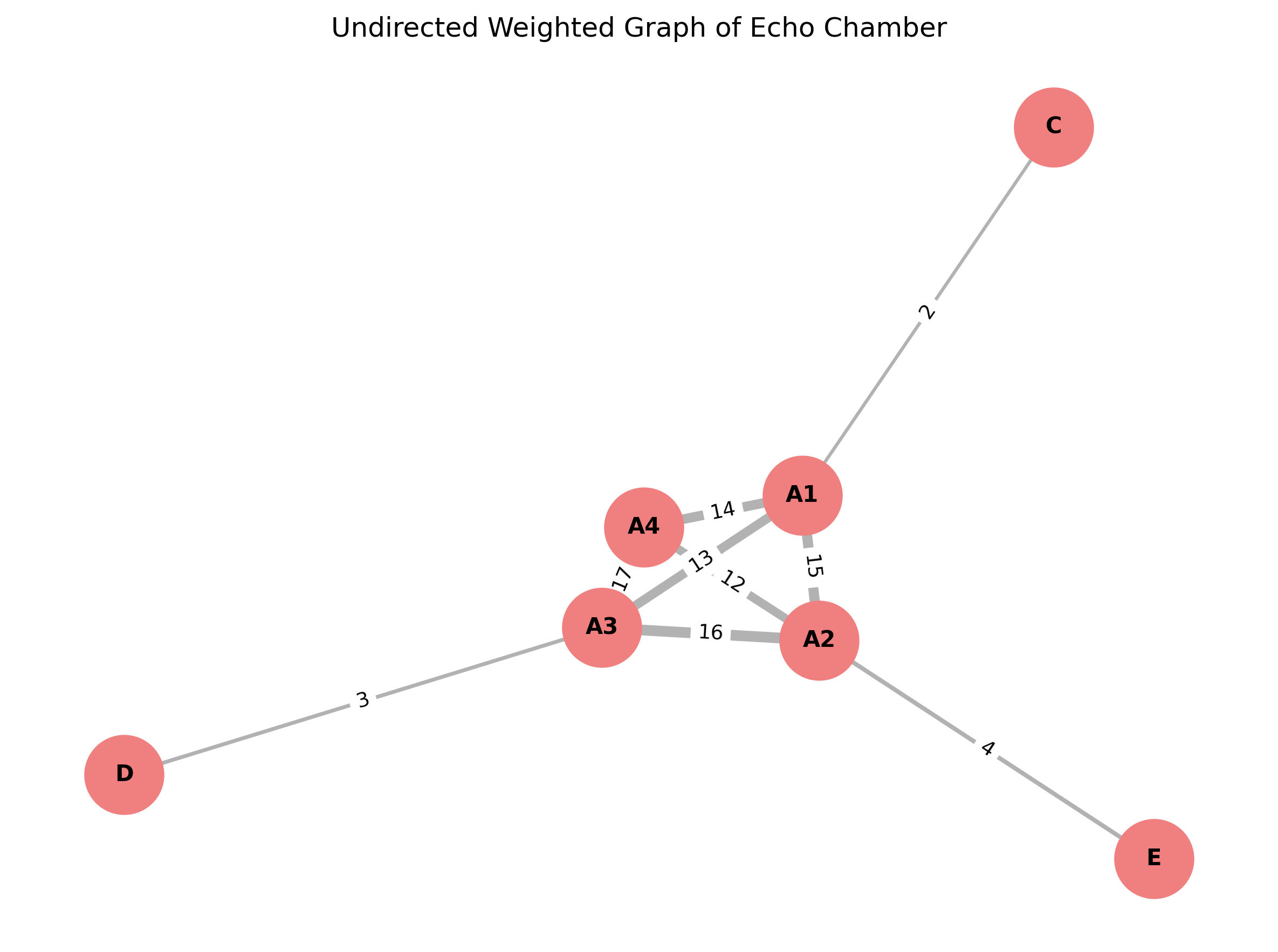


Figure 1. Echo chamber representation in a graph

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).

# 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 can be defined as an ordered pair where is a set of vertices and is a set of edges, where each edge is a pair of vertices from .

## 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.

A diagram of a graph

AI-generated content may be incorrect.

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**.

A black triangle with black lines and dots

AI-generated content may be incorrect.

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.

A diagram of a loop

AI-generated content may be incorrect.

Figure 4. Loop in a graph

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

## 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 is not empty, the edge set is completely empty.

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

Vertex set : {}, Edge set : {} or ∅

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

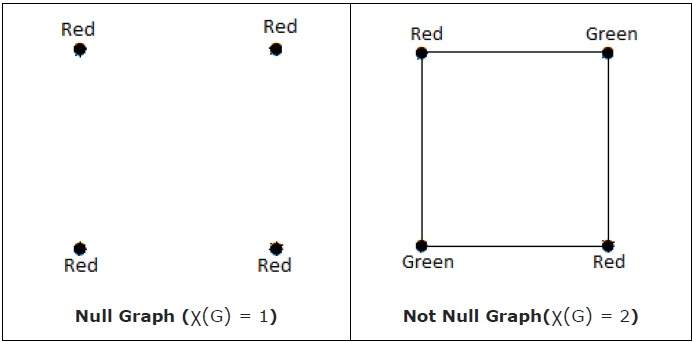


Figure 5. Null graph

<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 : {}, Edge set : {} or ∅

* 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 : {}, Edge set : {}

A black and white image of a diamond

AI-generated content may be incorrect.

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 is considered the same as , 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.

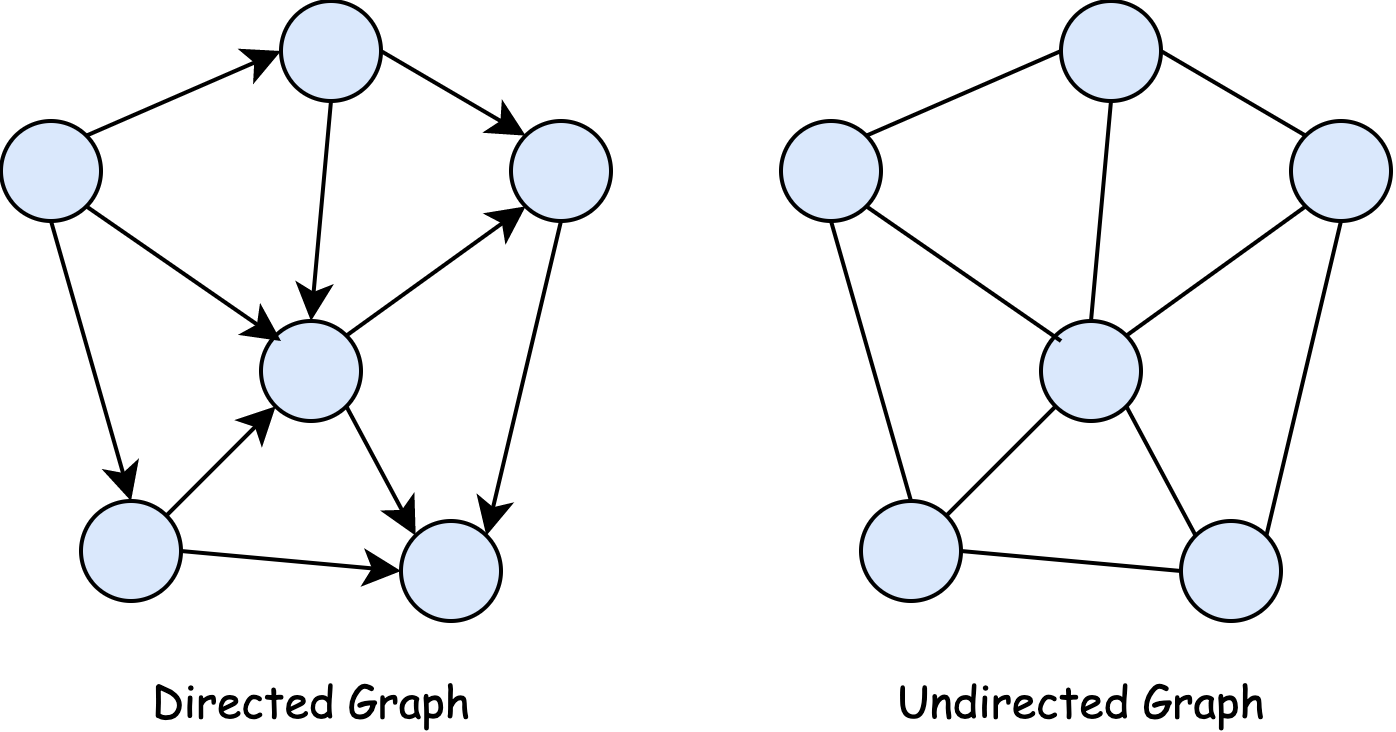


Figure 7. Directed and undirected graph

<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 : {}, Edge set : {}

A diagram of a graph

AI-generated content may be incorrect.

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 vertices is usually denoted as .

A black and white image of a cross and a line

AI-generated content may be incorrect.

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 : {}, Vertex set : {}

Edge set : {}

A diagram of a complex connection

AI-generated content may be incorrect.

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.

A black and white square with black dots

AI-generated content may be incorrect.

Figure 11. Cycle graph

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

## 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 , where is the number of vertices. Each element in the matrix shows whether an edge exists between vertex and vertex . 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 {}

Edge set {}

A diagram of a diamond with black dots and numbers

AI-generated content may be incorrect.

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | 0 | 1 | 1 | 0 |
|  | 1 | 0 | 1 | 1 |
|  | 1 | 1 | 0 | 1 |
|  | 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 |
|  | 2, 3 |
|  | 1, 3, 4 |
|  | 1, 2, 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 , where is the number of vertices and 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 | (1, 2) | (1, 3) | (3, 4) | (2, 3) | (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 |

## 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** for a weighted graph is defined as:

Where:

* is the weight of the edge between node and node ,
* and are the strengths of nodes and , i.e., the sum of their incident edge weights,
* is the total weight of all edges in the network,
* equals 1 if nodes and 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.

# Methodology

## Data Preparation

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.

## 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 with and being the interacting users and the interaction weight. An example of such data is shown in Figure 13.

A computer code with many letters

AI-generated content may be incorrect.

Figure 13. Synthetic 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.

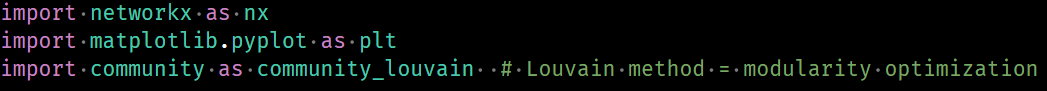


Figure 14. Python libraries used for graph visualization and community detection.

To make echo chamber detection more reliable, each edge between nodes A and B represents a single mutual interaction. For example, in Figure 15, 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.

A screenshot of a black screen

AI-generated content may be incorrect.

Figure 15. Example of a mutual interaction between two users, counted as a single edge.

## 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.

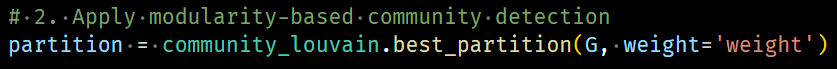


Figure 16. Community detection using python library

## 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.

## 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.

# Result and Discussion

## Graph Visualization Without Community Detection

Before applying any detection algorithm, the graph was visualized using a random layout (see Figure 17). In this version, all nodes and edges are shown without grouping, making it difficult to identify patterns or clusters manually.

A network diagram with blue circles and white text

AI-generated content may be incorrect.

Figure 17. Synthetic Echo Chamber Network (No Community Detection)

## Community Detection Result

After running the **Louvain method** on the weighted graph, the algorithm successfully grouped nodes into communities with high internal connectivity (see Figure 18). Each community corresponds to a potential echo chamber, where users interact more with others inside the group than with those outside.

A diagram of a network

AI-generated content may be incorrect.

Figure 18. Detected Echo Chamber Communities via Modularity Optimization

## Modularity Score

### Positioning Figures and Tables: Place figures and tables at the top and bottom of columns. Avoid placing them in the middle of columns. Large figures and tables may span across both columns. Figure captions should be below the figures; table heads should appear above the tables. Insert figures and tables after they are cited in the text. Use the abbreviation “Fig. 1,” even at the beginning of a sentence.

1. Table Styles

| Table Head | Table Column Head | | |
| --- | --- | --- | --- |
| Table column subhead | Subhead | Subhead |
| copy | More table copya |  |  |

1. Sample of a Table footnote. *(Table footnote)*
2. Example of a figure caption. *(figure caption)*

Figure Labels: Use 8 point Times New Roman for Figure labels. Use words rather than symbols or abbreviations when writing Figure axis labels to avoid confusing the reader. As an example, write the quantity “Magnetization,” or “Magnetization, M,” not just “M.” If including units in the label, present them within parentheses. Do not label axes only with units. In the example, write “Magnetization (A/m)” or “Magnetization (A ( m(1),” not just “A/m.” Do not label axes with a ratio of quantities and units. For example, write “Temperature (K),” not “Temperature/K.”

##### Video Link at Youtube *(Heading 5)*

Include link of your video on YouTube in this section.

##### Acknowledgment *(Heading 5)*

The preferred spelling of the word “acknowledgment” in America is without an “e” after the “g.” Avoid the stilted expression “one of us (R. B. G.) thanks ...”. Instead, try “R. B. G. thanks...”. Put sponsor acknowledgments in the unnumbered footnote on the first page.

##### References

The template will number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use “Ref. [3]” or “reference [3]” except at the beginning of a sentence: “Reference [3] was the first ...”

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors’ names; do not use “et al.”. Papers that have not been published, even if they have been submitted for publication, should be cited as “unpublished” [4]. Papers that have been accepted for publication should be cited as “in press” [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

1. G. Eason, B. Noble, and I.N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529-551, April 1955. (*references*)

We suggest that you use a text box to insert a graphic (which is ideally a 300 dpi resolution TIFF or EPS file with all fonts embedded) because this method is somewhat more stable than directly inserting a picture.

To have non-visible rules on your frame, use the MSWord “Format” pull-down menu, select Text Box > Colors and Lines to choose No Fill and No Line.

1. J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.
2. I.S. Jacobs and C.P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G.T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271-350.
3. K. Elissa, “Title of paper if known,” unpublished.
4. R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.
5. Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740-741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 1 Juni 2025

Ttd

Nama dan NIM