VIETNAM NATIONAL UNIVERSITY - HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

ASSIGNMENT REPORT

MATHS FOUNDATION for COMPUTER SCIENCE (055263)

Community Structure Identification

Instructors: Dr. Nguyen An Khuong, CSE-HCMUT
Dr. Tran Tuan Anh, CSE-HCMUT
Dr. Nguyen Tien Thinh, CSE-HCMUT
Teaching Assistants: Le Thanh Son, KTH Royal Institute of Technology
Tran Dinh Vinh Thuy, Ecole Centrale de Lyon
Students: Nguyen Quang Khanh - 2010330
Huynh Tan Loc - 2010391
Phan Phuoc Minh - 2010418
Luu Quoc Hung Thinh - 2010651
Nguyen Duc An - 2010102

Ho Chi Minh City, May 2024

Mục lục

List of figures

TASK I

INTRODUCTION

1 Motivation

Graph theory is one of the most classical and widely studied theories in mathematics. There have been numerous research efforts focused on graphs and their properties. The application of graphs across various fields, particularly in computer science.

With the emergence of social networks, there is an increasing number of communities being formed. This leads to a trend of exploring these communities to discover new insights and values. One way to represent these communities is through graphs.

Using graph-based techniques, scientists have proposed many algorithms to identify communities within social networks, such as the Girvan-Newman algorithm, Louvain algorithm, etc. In this report, the group will present their understanding and application of these two algorithms to the problem of community detection.

2 Introduction to Problem: Jazz Musicians

Dataset: The Jazz Musicians dataset is a network representing collaborations between Jazz musicians. Each node in the network represents a Jazz musician, and an edge between two nodes indicates that the two musicians have played together in a band. This dataset was collected in 2003 and provides insights into the interconnectedness of Jazz musicians through their collaborative efforts.

Input of the dataset is a file named **out.arenas-jazz**. Each line includes two numbers that represent two Jazz musicians along with their IDs. If the IDs of two musicians appear on the same line, this indicates that they have performed Jazz music together.

The dataset contains 198 IDs of Jazz musicians with 2742 lines.

Problem: Our task is doing the clustering on this dataset to identify the communities in the Jazz musician, visualize in the graph.

The use-case scenario is: The company wants to organize an anniversary celebration for its establishment and invite Jazz musicians to perform. To ensure quality, they want to ensure that the invited members can perform well and harmonize with each other (meaning they have performed together before). The input to the problem will be a list of musicians, and the output will be whether these Jazz musicians can perform well together and provide information on who should play with whom.

TASK II

PRELIMINARIES

1 Introduction to Graph Theory

Graph theory is a fascinating area of mathematics with wide-ranging applications in various fields, including computer science, biology, social sciences, and engineering. It focuses on studying graphs, which are mathematical structures used to model pairwise relationships between objects.

In its simplest form, a graph consists of a set of vertices (also called nodes) and a set of edges connecting pairs of vertices. Graphs can be represented visually, where vertices are depicted as points, and edges are depicted as lines connecting these points.

1.1 Type of graphs

1. **Undirected Graphs**: In these graphs, edges have no direction. The connection between two vertices is mutual.
2. **Directed Graphs**: These graphs have edges with a direction, indicated by an arrow. Each edge points from one vertex to another.
3. **Weighted Graphs**: Here, edges carry weights, representing the cost, distance, or any other value associated with the connection.
4. **Unweighted Graphs**: These graphs do not have weights on their edges.

1.2 Graph elements

1. **Adjacency**: Two vertices are adjacent if they are connected by an edge.
2. **Degree**: The degree of a vertex is the number of edges incident to it. In directed graphs, we have in-degrees and out-degrees.
3. **Path**: A path is a sequence of vertices where each adjacent pair is connected by an edge.
4. **Cycle**: A cycle is a path that starts and ends at the same vertex without repeating any edges or vertices.
5. **Connectedness**: A graph is connected if there is a path between any two vertices. In directed graphs, we consider strong and weak connectedness.
6. **Subgraph**: A subgraph is a portion of a graph, consisting of a subset of its vertices and edges.
7. **Tree**: A tree is a connected acyclic graph. It has a hierarchical structure with a root and branches.
8. **Bipartite Graph**: A graph whose vertices can be divided into two disjoint sets such that no two vertices within the same set are adjacent.

2 Community Structure Identification in Graphs

Community Structure Identification in graphs refers to the process of detecting clusters or groups of nodes within a graph that are more densely connected to each other than to the rest of the network. This concept is crucial in understanding the organization and functionality of complex networks, such as social networks, biological networks, and technological networks.

2.1 Definition

Here are some key points to understand Community Structure Identification:

- Definition of Communities: Communities (or clusters) are groups of nodes that have a higher probability of connecting with each other than with nodes outside the group. These groups often represent functional units, such as social circles in a social network or functional modules in a biological network.
- Importance:
  - Understanding Network Structure: Community detection helps to reveal the hierarchical structure of networks and understand how networks are organized.
  - Behavioral Insights: Identifying communities can provide insights into the behavior and interactions of different parts of the network. For example, it can help understand how information spreads in social networks or how diseases propagate.
  - Application-Specific Insights: In marketing, community detection can help identify target customer groups. In biology, it can help in identifying molecular pathways.

2.2 Application

Community Structure Identification is applied broadly across various domains to understand and analyze complex networks.

- Social Networks:
  - Friend Groups: Detecting groups of closely connected individuals.
  - Influencer Communities: Identifying key influencers and their followers.
  - Interest Groups: Clustering individuals based on common interests or activities.
- Biological Networks:
  - Protein Complexes: Identifying groups of proteins that interact closely.
  - Functional Modules: Discovering groups of genes or proteins that contribute to specific biological functions.
- Information Networks:
  - Related Documents: Clustering documents or web pages that cover similar topics.
  - Collaborative Filtering: Improving recommendation systems by identifying groups of users with similar preferences.

2.3 Benefits & Drawbacks

: [Cộng đồng] x
There are some notable benefits:

- Understanding Network Structure: Community detection helps reveal the hierarchical structure of networks and how they are organized. This understanding can lead to more efficient designs and strategies for network management.

- Behavioral Insights: By identifying communities, one can gain insights into the behavior and interactions of different parts of the network. For example, understanding how information spreads within and between communities in social networks can be crucial for marketing and public health interventions.
- Targeted Interventions: In various applications, identifying communities allows for targeted interventions. In marketing, this means better-targeted advertising campaigns. In biology, it means focusing on specific protein complexes for drug development.
- Enhanced Recommendation Systems: In information networks, community detection improves collaborative filtering, leading to more accurate recommendations for users based on their interests and preferences.

Beside benefits, there are a lot of challenges:

- Scalability:
  - Large Networks: Community detection algorithms must handle networks with millions of nodes and edges efficiently, which can be computationally intensive.
  - Computational Complexity: Balancing the accuracy of community detection with the computational resources required is a significant challenge. Efficient algorithms are necessary to process large-scale networks within reasonable time frames.
- Overlap:
  - Multiple Membership: Real-world nodes often belong to multiple communities (e.g., a person might be part of multiple social circles). Traditional community detection algorithms struggle with overlapping communities.
  - Detection Algorithms: Developing algorithms that can accurately detect and manage overlapping communities is essential to reflect real-world scenarios better.
- Dynamic Networks:
  - Temporal Changes: Networks evolve over time, with nodes and edges appearing or disappearing. Static community detection algorithms may not perform well on dynamic networks.
  - Adaptive Methods: Algorithms must adapt to changes in network structure to maintain accurate community detection. This requires methods that can incrementally update community structures as the network evolves.

3 Key Concepts and Measures

3.1 Modularity

Modularity is a crucial measure in the context of community detection in graphs. It quantifies the quality of a particular division of a graph into communities (or clusters). The concept was introduced by Newman and Girvan in 2004 and has since become one of the most widely used metrics for evaluating community structures.

Definition and Formula

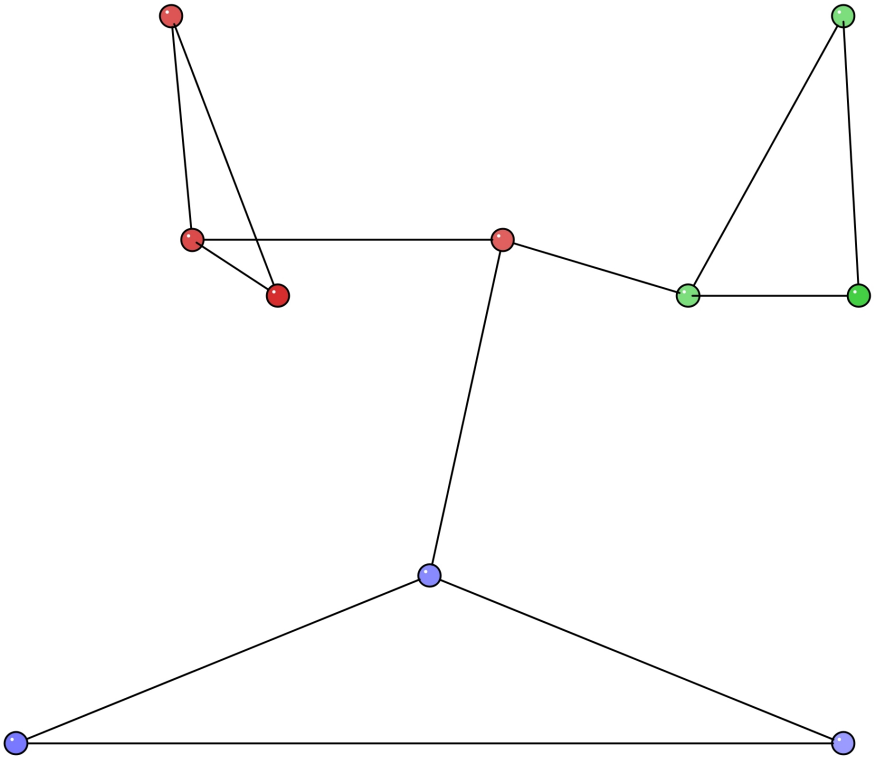
Modularity is defined as the fraction of the edges that fall within the given groups minus the expected fraction if edges were distributed at random. Mathematically, it can be expressed as:

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

where:

- Q is the modularity score.
- A is the adjacency matrix of the graph, where $A_{ij} = 1$ if there is an edge between nodes i and j, and 0 otherwise.
- k_i and k_j are the degrees of nodes i and j, respectively.
- m is the total number of edges in the graph.
- $\delta(c_i, c_j)$ is the Kronecker delta, which is 1 if nodes i and j are in the same community (i.e., $c_i = c_j$) and 0 otherwise.

For example, the following partition maximizes Q with the value of $Q = 0.4896$



Hình 1: *Sample network partition*

Interpretation

- High Modularity: A high modularity score (close to 1) indicates a strong community structure, where nodes within the same community are more densely connected to each other than to nodes in other communities.
- Low Modularity: A low or negative modularity score suggests that the division is not better than a random assignment of nodes into communities.

Significance in Community Detection

Modularity serves as both an objective function for community detection algorithms and a diagnostic tool for the quality of the detected communities. Its importance can be summarized as follows:

- Evaluating Community Structures: Modularity provides a single scalar value that encapsulates the quality of a community structure, making it easier to compare different partitions of the same graph.
- Guiding Algorithm Design: Many community detection algorithms, such as the Louvain method, are designed to maximize modularity. This optimization process iteratively improves the community structure by merging or splitting communities to increase the modularity score.

- Insight into Network Topology: High modularity values often indicate the presence of meaningful and functionally coherent communities, which can provide insights into the underlying structure and function of the network.

Limitations

While modularity is a powerful and popular metric, it has some limitations:

- Resolution Limit: Modularity optimization can sometimes fail to detect small communities, a phenomenon known as the resolution limit. This means that the measure may not be sensitive to community structures at all scales.
- Degeneracy: Different partitions of a graph can yield similar modularity scores, leading to ambiguities in identifying the "best" community structure.

Despite these limitations, modularity remains a fundamental tool in the analysis and identification of community structures in graphs, providing a robust framework for understanding complex networks.

3.2 Intra-Community and inter-Community

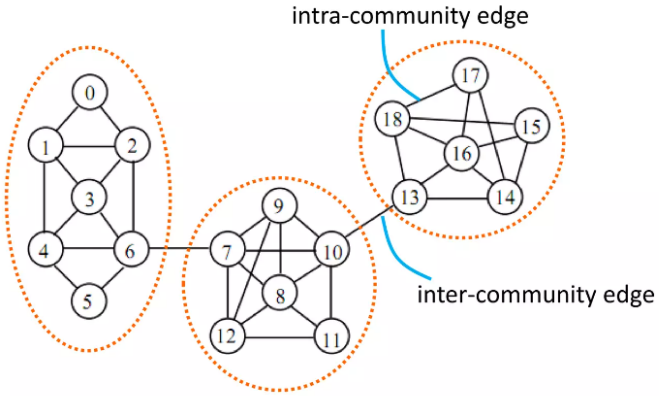
The terms "inter" and "intra" refer to relationships between nodes with respect to the communities they belong to:

Intra-Community

- Intra-Community Edges: These are the edges that connect nodes within the same community. The density and number of intra-community edges are typically higher compared to inter-community edges. In other words, within a well-defined community, nodes are more likely to be connected to each other. This high internal connectivity is a key characteristic of a strong community structure.
- Intra-Community Measures: Metrics that evaluate the cohesion within a community. These can include intra-community density, which is the ratio of the number of edges within a community to the number of possible edges within that community. Another measure could be the average degree of nodes within the community.

Inter-Community

- Inter-Community Edges: These are the edges that connect nodes from different communities. In a well-defined community structure, the number of inter-community edges should be lower compared to intra-community edges. This low external connectivity indicates clear boundaries between different communities.
- Inter-Community Measures: Metrics that evaluate the separation between communities. One such measure is inter-community density, which is the ratio of the number of edges between different communities to the number of possible edges between those communities. Another measure could be the average edge weight of inter-community edges in weighted graphs.



Hình 2: *Inter-Community edge and Intra-Community edge*

Significance in Community Detection

- The balance between intra-community and inter-community edges is crucial for determining the quality of community structures. High intra-community connectivity combined with low inter-community connectivity usually signifies a good community division.
- Modularity inherently incorporates the concepts of inter and intra-community edges. The modularity score is higher when there are many intra-community edges and fewer inter-community edges than expected by chance. The modularity formula essentially rewards configurations where communities have dense internal connections and sparse external connections.

Examples

- Social Networks: In a social network, communities could represent groups of friends. Intra-community edges represent friendships within the same group, while inter-community edges represent connections between friends from different groups.
- Biological Networks: In protein-protein interaction networks, communities might correspond to functional modules or complexes. Intra-community interactions would be the interactions within the same functional module, and inter-community interactions would be the interactions between different modules.

3.3 Centrality Measures

Centrality measures are crucial in network analysis for identifying the most important or influential nodes within a graph. These metrics help understand the roles and significance of individual nodes in the overall network structure.

Degree Centrality

- Definition: The degree centrality of a node is the number of edges connected to it.
- Interpretation: Nodes with high degree centrality are directly connected to many other nodes, indicating their high level of activity or popularity within the network.
- Formula: For a node v, degree centrality $C_D(v)$ is given by $C_D(v) = k_v$, where k_v is the degree of v.

Betweenness Centrality

- Definition: Betweenness centrality quantifies the number of times a node acts as a bridge along the shortest path between two other nodes.
- Interpretation: Nodes with high betweenness centrality have significant control over information flow in the network as they frequently lie on the shortest paths between other nodes.

- Formula: For a node v, betweenness centrality $C_B(v)$ is given by:

$$C_B(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

where σ_{st} is the total number of shortest paths from node s to node t, and $\sigma_{st}(v)$ is the number of those paths that pass through v.

Closeness Centrality

- Definition: Closeness centrality measures how close a node is to all other nodes in the network.
- Interpretation: Nodes with high closeness centrality can quickly interact with all other nodes, indicating their efficiency in spreading information or influence.
- Formula: For a node v, closeness centrality $C_C(v)$ is given by:

$$C_C(v) = \frac{1}{\sum_u d(v,u)}$$

where $d(v,u)$ is the shortest path distance between nodes v and u.

Eigenvector Centrality

- Definition: Eigenvector centrality assigns relative scores to all nodes in the network based on the principle that connections to high-scoring nodes contribute more to the score of the node in question.
- Interpretation: Nodes with high eigenvector centrality are connected to other nodes that are themselves highly central, indicating influence or importance in the network.
- Formula Eigenvector centrality $C_E(v)$ of a node v is defined as the v-th component of the eigenvector corresponding to the largest eigenvalue of the adjacency matrix A:

$$C_E(v) = \frac{1}{\lambda} \sum_{u \in V} A_{vu} C_E(u)$$

where λ is the largest eigenvalue of A.

Significance in Community Detection

Centrality measures help identify key nodes that may play crucial roles within or between communities. For instance, nodes with high betweenness centrality might act as connectors between different communities, while those with high degree or eigenvector centrality might be central hubs within a community. These insights are valuable for understanding the structure and dynamics of the network.

TASK III

APPROACH

1 Girvan-Newman Algorithm

Girvan-Newman method is one of the classic community clustering techniques, which separates the network based on the betweenness of the edges. By using the algorithm, we are able to separate the network into communities, and the community detection can be used as a good start of data preprocessing.

1.1 Algorithm

The Girvan-Newman algorithm is a popular method in network analysis used for detecting communities within a network. It works by progressively removing edges from the network to uncover the underlying community structure. The basic idea is that edges that are most "between" communities, measured by their betweenness centrality, are removed first, which eventually splits the network into its constituent communities.

Here's a step-by-step outline of the algorithm:

1. **Calculate Edge Betweenness Centrality:** For every edge in the network, calculate its betweenness centrality. Betweenness centrality for an edge is the sum of the fraction of all-pairs shortest paths that pass through this edge.
2. **Remove Edge with Highest Betweenness:** Identify the edge with the highest betweenness centrality and remove it from the network. This edge is considered to be most likely to lie between two communities.
3. **Recompute Betweenness Centrality:** After removing an edge, recompute the betweenness centrality for the remaining edges in the network. This is necessary because the removal of an edge can affect the betweenness centrality of other edges.
4. **Repeat Steps 2 and 3:** Continue removing edges with the highest betweenness centrality and recalculating betweenness until the network splits into distinct communities.

Betweenness centrality for an edge e is given by the formula:

$$C_B(e) = \sum_{s \neq t} \frac{\sigma(s, t \mid e)}{\sigma(s, t)}$$

where $\sigma(s, t)$ is the total number of shortest paths from node s to node t, and $\sigma(s, t \mid e)$ is the number of those paths that pass through edge e.

1.2 Pseudocode

Algorithm 1 Girvan-Newman Algorithm

Input: Network: A graph representing the network.
Output: Communities: A list of detected communities.

```
/* Initialize communities with the entire network */
communities ← {Network}
while Network has edges do
    /* Calculate edge betweenness centrality for all edges */
    CalculateEdgeBetweenness(Network)
    /* Find edge with highest betweenness */
    e ← FindMaxBetweennessEdge(Network)
    /* Remove the edge */
    RemoveEdge(Network, e)
    /* Recompute connected components */
    connected_components ← ConnectedComponents(Network)
    /* Update communities with new components */
    communities ← communities ∪ connected_components
end
Output: communities
```

1.3 Advantages and Disadvantages

Advantages

- No need for pre-defined clusters: Unlike some clustering algorithms, Girvan-Newman doesn't require you to specify the number of communities beforehand. It iteratively breaks down the network based on edge connections, allowing the communities to emerge naturally.
- Effective for large networks: The algorithm has good runtime efficiency (around O(n log n)) making it suitable for analyzing large networks with millions of nodes and billions of edges.
- Intuitive approach: The concept of removing edges with the most inter-community connections is relatively easy to understand.

Disadvantages

- Resolution limit: The algorithm might break down communities too much, resulting in many small clusters instead of the intended larger communities.
- Greedy optimization: It uses a greedy approach, meaning it removes the most central edges at each step. This may not always lead to the optimal community structure.
- Computational cost for dense networks: While efficient for sparse networks, the runtime complexity can become high ($O(m^2n)$) for very dense graphs.
- Difficulty in determining stopping point: There's no clear stopping criteria for the edge removal process. A metric called modularity is used to guide this, but finding the optimal stopping point can be challenging.

Overall, the Girvan-Newman algorithm is a valuable tool for community detection in networks. However, it's important to be aware of its limitations and choose the right algorithm for your specific data and needs.

2 Louvain Algorithm

The Louvain method is an algorithm designed for detecting communities in large networks. The Louvain method for community detection is a method to extract non-overlapping communities from large

networks created by Blondel et al [1] from the University of Louvain (the source of this method's name). It operates by maximizing a modularity score for each community, which measures the quality of the assignment of nodes to communities based on their density compared to a random network.

The Louvain method can be applied to various types of graphs, including directed, undirected, heterogeneous nodes, heterogeneous relationships, and weighted relationships.

2.1 Algorithm

The methodology employed for community detection is fundamentally rooted in the optimization of modularity as the algorithm evolves. Modularity serves as a metric that quantifies the degree of modularity within a network, ranging from -0.5, indicative of non-modular clustering, to 1, representing fully modular clustering. This metric evaluates the relative density of edges within communities against those external to communities. Theoretically, the optimization of modularity aims to achieve the optimal partitioning of nodes within a given network. However, due to the computational infeasibility of exhaustively exploring all potential groupings of nodes, heuristic algorithms are employed to approximate this ideal state.

A recall that the modularity of can be calculated as:

$$Q = \frac{1}{2m} \sum_{i=1}^{N} \sum_{j=1}^{N} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta\left(c_i, c_j\right)$$

On the other hand, with a community c, can be calculated as:

$$Q = \frac{\sum_{in}}{2m} - \left(\frac{\sum_{tot}}{2m} \right)^2$$

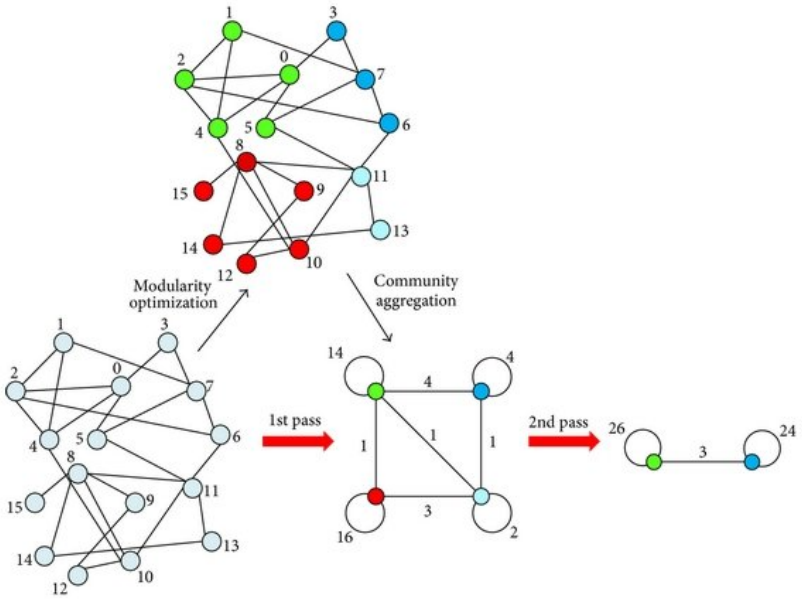
The algorithm has two phases:

- **Phase 1:**
  - First, create a network with the number of community equal to the number of each node and each node belong to a community.
  - Second, for all node i, move it from the current communities to the new communities of its neighbor nodes and calculate the change in the modularity with the formula. When moving an isolate node i into a new community, the formula can be calculate with:

$$\Delta Q = \left[\frac{\sum_{in} k_{i,in}}{2m} - \left(\frac{\sum_{tot} + k_i}{2m} \right)^2 \right] - \left[\frac{\sum_{in}}{2m} - \left(\frac{\sum_{tot}}{2m} \right)^2 - \left(\frac{\sum_{tot}}{2m} \right)^2 \right]$$

  - When all the change in modularity has been calculated, put the node i into the community c_i' that make the change in modularity is highest and positive. If no positive case, node i stays again with the c_i community.
  - Replace this step for all of the nodes
- **Phase 2:** The second phase of the algorithm involves reducing communities to a single node and repeating the steps in Phase 1:
  - Each community is transferred to a single node. The **self-loop** weight is the **intra-community** links and the **weight between two communities** is the **inter-community** link between the two corresponding.
  - Replay the first phase and second phase until there is no possible update solution.

  Here is an intuitive example for Louvain algorithm:



Hình 3: *Example for Louvain algorithm*

2.2 Pseudocode

The pseudo-code for Louvain algorithm is show in figure 4 [2].

```
function LouvainAlgorithm(Graph G)
    G' = G
    C the index of community of each nodes of G'
    Initialize each nodes with its own community
    q = −∞
    while q < Q(G',C) do
        q = Q(G',C)
        C = MoveNodes(G')      //Phase 1
        G' = Aggregate(G', C)  //Phase 2
        C = put each node of G'
                    in its own community
    end while
    return G'
end function

function MoveNodes(Graph G)
    C the index of communities
                    for each nodes of G
    while one or more nodes are moved do
        for random v ∈ V(G) do
            best_q = −∞
            best_c = community of v
            for all neighboring nodes n of v do
                gain_q = ΔQ between v and n
                if best_q < gain_q < then
                    best_q = gain_q
                    best_c = community of n
                end if
            end for
            C = Place v in the best_q
        end for
    end while
    return C
end function

function Aggregate(Graph G, Partition C)
    G' = aggregate nodes which are in
                    same community based on
C
    return G'
end function
```

Hình 4: *Psedu-code of Louvain algorithm*

The time complexity of the Louvain algorithm is approximately $O(n \log n)$, where n represents the number of nodes in the network. This complexity indicates that the algorithm scales logarithmically with the size of the input, making it highly efficient for large-scale network analysis.

2.3 Advantages and Disadvantages

The Louvain algorithm is a popular method for community detection in large networks, offering several advantages and disadvantages:

Advantages:

- Optimize the Modularity: Louvain is designed to use heuristic to optimize the modularity, so the result for this algorithm is always quite good.
- Speed and scale: The complexity of Louvain is $O(n \log n)$ with n as the number of nodes, so its runtime is good for the context with the number of nodes is large.
- Hierarchical Clustering: With the recursive in the algorithm, Louvain provides the good view in understanding the multi-level community.

Disadvantages:

- Can not solve the overlap communities: In the Louvain method, one node can only belong to a community. In the real world, there are many cases that a person can belong to many communities.
- Unstable outcome: The output of the algorithm depends on phase 1. But in phase 1, the heuristic is used. If we change the initial such as change the order of the nodes, the result can be different. It leads to various answers in spite of using the same technique on the same dataset.

TASK IV

EXPERIMENTS

In this section, we conduct the experiments when using Girvan-Newman algorithm and Louvain algorithm on the Jazz musicians dataset

1 Dataset

This dataset which was found at http://konect.cc/networks/arenas-jazz/ is about the collaboration network between Jazz musicians. Each node is a Jazz musician and an edge denotes that two musicians have played together in a band and the dataset was collected in 2003. We can download this dataset at here

This dataset include a file with TSV format which define Jazz musician network:

- First column: ID of from node
- Second column: ID of to node

This graph have no weight and no direction.

2 Graph building

From the input file TSV, we construct a graph in the follow steps:

- First of all, we create a graph with 198 nodes where each node represents for a Jazz musicians
- Iteration through each line of the input file, if two Jazz musicians have performed together, their corresponding vertex is connected

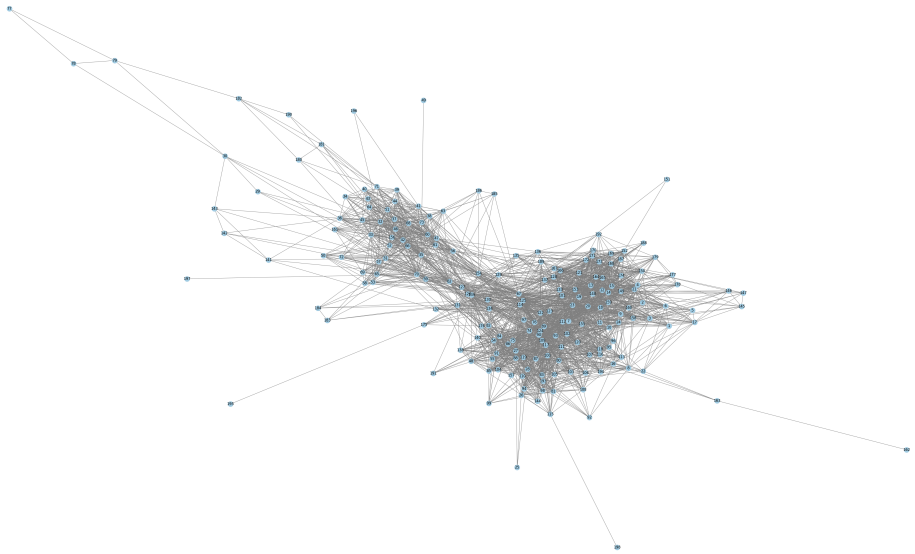
Because the dataset is unweighted, the graph in the un-directed graph and all of its edges has weight 1.

3 Exploratory Data Analysis

Some statistic about this graph:

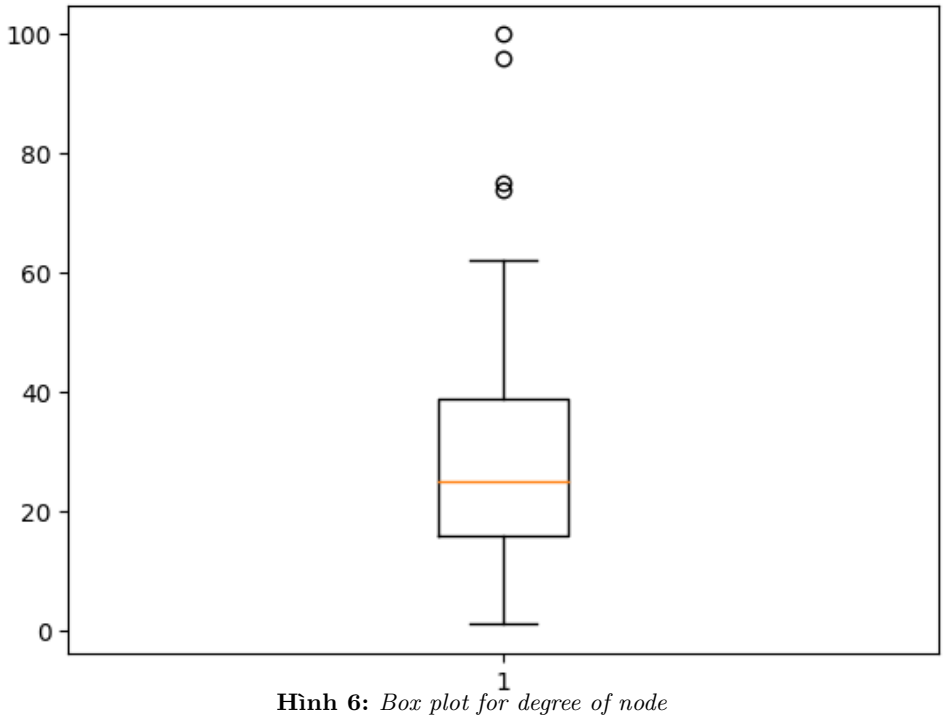
- Number node: 198
- Number edge: 2742
- Minimum degree: 1
- Maximum degree: 100
- Mean degree: 27.69

Figure 5 show connection of node in graph.



Hình 5: *Graph of Jazz musician*

Figure 6 show box plot of degree node in the graph.



Hình 6: *Box plot for degree of node*

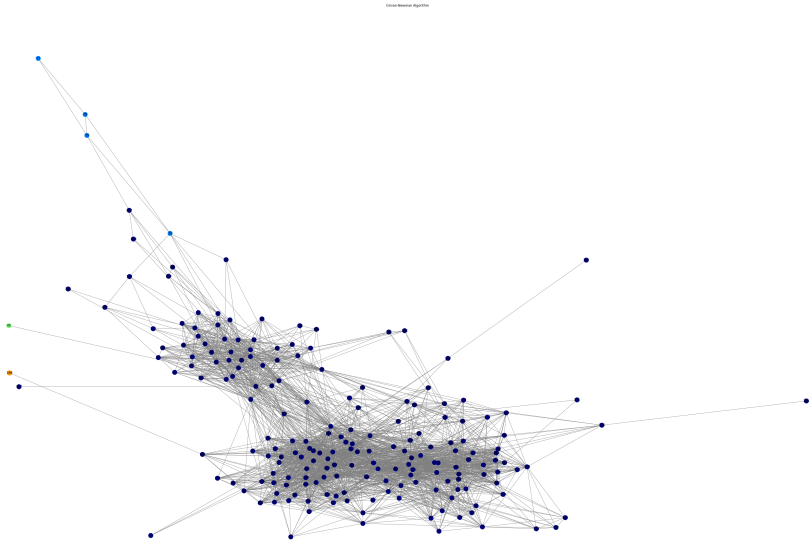
We can see that over 60% node in graph having degree greater then 20 in a graph having 198 node. This mean having 60% musicians who have played with at least other twenty musicians.

4 Identify communities in Jazz musicians

Running environment: Google Colab (free-tier). After running the experiments, the results are:

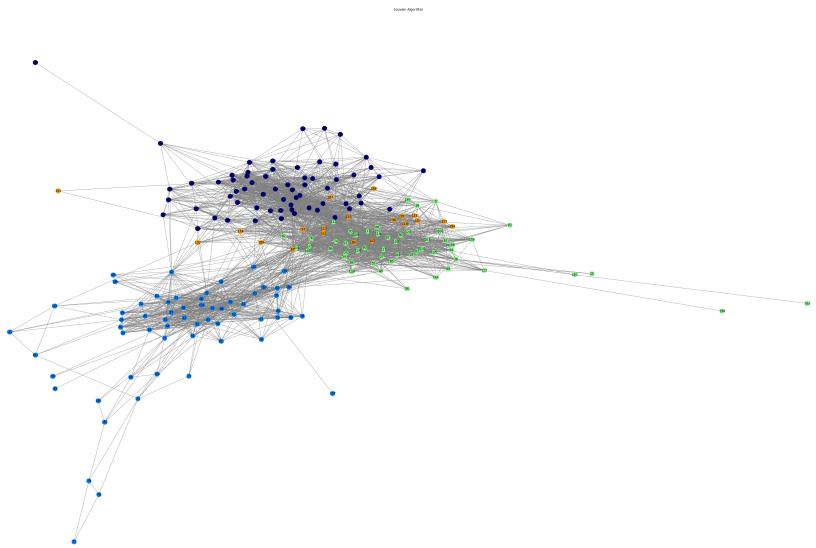
- Graph build time: 0.02. The graph has 198 nodes and 2742 edges.

- Communities identification:
  - **Girvan-Newman algorithm**:
    - Run time for community identification is 7s.
    - The result is set to 4 communities.
    - Modularity metrics: 0.0036
    - Statistic:
      - Community 1: 192
      - Community 2: 4
      - Community 3: 1
      - Community 4: 1
    - The plot for 4 communities is shown in the picture below.



Hình 7: *Girvan-Newman communities results*

  - **Louvain algorithm**:
    - Run time for community identification is 0.09s.
    - The result is to identify 4 communities.
    - Modularity metrics: 0.45
    - Statistic:
      - Community 1: 64
      - Community 2: 61
      - Community 3: 53
      - Community 4: 20
    - The plot for 4 communities is shown in the picture below.



Hình 8: *Louvain communities results*

5 Discuss about two approaches result

With the result of the experiment, the running time for Girvan-Newman algorithm is 80 times larger than others because the Girvan-Newman algorithm is computationally expensive, especially for large networks, as it requires calculating betweenness centrality for all edges. The distribution of nodes into communities of Louvian algorithm is so fair and with Girvan-Newman algorithm, number node in every community is imbalance, this is cause by it focuses on the concept of betweenness centrality, which measures the number of shortest paths passing through a given edge.

In the Girvan-Newman's result, there are 2 isolated communities, for example, the musician with ID 198 belongs to itself. But in the relationship, the musician 195 can play with musician 118. On the other hands, the Louvain algorithm is so fair but it leads to the case that a musician only perform with one other musician in the community. It makes the intra-density of this community decrease.

6 Application in choosing the band

With communities which were detected with two algorithm, we can apply for the problem inviting musician who can perform well and harmonize with each other (meaning they have performed together before).

- **Input**: A list of musician ID
- **Output**: Answer the question: All musician in the list can play together or not. If not, patition them into clusters that can play together.

With input: $[1, 2, 3, 122, 151]$. Result for each communities which be detected by two approach is:

- **Girvan-Newman algorithm**:

```
{
    'ability_working_together': True,
    'partition': [[1, 2, 3, 151, 122]]
}
```

- **Louvain algorithm**:

```
{
    'ability_working_together': False,
```

```
    'partition': [
        [1, 2, 3],
        [122, 151]
    ]
}
```

TASK V

CONCLUSION

Uncovering the hidden structure within networks is crucial across various fields, such as social sciences and biology. Two prominent algorithms, Girvan-Newman and Louvain, provide powerful tools for identifying communities (or clusters) of interconnected nodes, each with distinct approaches and applications.

The Girvan-Newman algorithm works hierarchically, akin to progressively removing bridges between islands. It calculates the "betweenness centrality" of edges, measuring how often an edge lies on the shortest path between communities. The algorithm iteratively removes the edge with the highest betweenness, causing the network to fragment into isolated nodes and revealing a hierarchy of communities.

Girvan-Newman excels when the number of communities is unknown and is efficient for large, sparse networks due to its runtime complexity. Its intuitive approach of removing edges that bridge communities is advantageous. However, it can over-partition the network, creating many small clusters instead of larger communities. Additionally, it may not always find the most optimal community structure, struggles with dense networks, and lacks a clear stopping point for edge removal.

In contrast, the Louvain algorithm optimizes a metric called "modularity" to evaluate the quality of community divisions. It iteratively moves nodes between communities to maximize modularity, resulting in a single-level community structure based on the strongest connections.

Louvain is generally faster than Girvan-Newman, especially for large networks, and often identifies high-quality communities. Some implementations can handle overlapping communities. However, it may require specifying the number of communities beforehand, can get stuck in local maxima, and does not provide a hierarchical community structure.

Choosing between Girvan-Newman and Louvain depends on specific needs. Girvan-Newman is suitable when the number of communities is unknown and a hierarchical structure is desired. Louvain is preferred for speed and efficiency, especially with large networks, and when overlapping communities are suspected. Both perform better on sparse networks, but for very dense networks, other algorithms might be more suitable. Evaluating community quality using metrics like modularity or conductance is crucial for informed decision-making.

The Jazz Musicians dataset illustrates how these algorithms reveal communities of collaborating artists. Girvan-Newman identifies communities hierarchically by progressively removing edges, useful when the number of communities is unknown. Louvain optimizes modularity to find high-quality communities at a single level and can handle overlapping communities in some implementations.

Experimental results show that the Girvan-Newman algorithm is computationally expensive, taking 80 times longer than Louvain due to the need to calculate betweenness centrality for all edges. Girvan-Newman tends to produce imbalanced communities due to its focus on betweenness centrality. In contrast, Louvain distributes nodes more evenly across communities but may lead to reduced intra-community density.

While both Girvan-Newman and Louvain are powerful tools for uncovering communities in networks, they do have limitations. Both algorithms tend to struggle with dense networks, where the high number of connections can hinder their performance. Girvan-Newman, though not designed for overlaps, can inadvertently split them during its edge-removal process. This can lead to isolated nodes that actually belong to multiple communities, misrepresenting the network's true structure. Louvain, on the other hand, has limitations depending on the specific implementation. Some versions don't handle overlaps at all, forcing nodes into single communities even if they have connections in multiple ones. This can cause a loss of information about the network. Additionally, accounting for overlaps can significantly increase the algorithm's complexity, impacting processing time for large networks.

Future research in community detection algorithms could focus on addressing these limitations. Developing algorithms that can effectively handle dense networks would be a significant advancement. For

Girvan-Newman, improvements could include better methods to find the optimal community structure and establish a clear stopping point for edge removal. Additionally, exploring ways to handle overlapping communities within the algorithm would be beneficial. With Louvain, research could focus on methods to automatically determine the ideal number of communities, eliminating the need for user input. By addressing these limitations, future community detection algorithms will be even more powerful tools for understanding the hidden structures within networks. One promising avenue lies in utilizing matrix factorization. This technique decomposes a network representation matrix into lower-dimensional matrices, potentially revealing hidden community structures within the data. Additionally, deep learning and neural networks hold immense potential for community detection. Their ability to learn complex patterns from network data could lead to algorithms that outperform traditional methods, particularly in identifying overlapping communities or handling very dense networks.

Tài liệu

[1] Blondel, Vincent D; Guillaume, Jean-Loup; Lambiotte, Renaud; Lefebvre, Etienne (9 October 2008). "Fast unfolding of communities in large networks". Journal of Statistical Mechanics: Theory and Experiment. 2008 (10): P10008

[2] Ozaki, N., Tezuka, H., & Inaba, M. (2016). A Simple Acceleration Method for the Louvain Algorithm. International Journal of Computer and Electrical Engineering, 8, 207-218.