ABSTRACT

This report applies graph analytics methods to analyze the curriculum structures of MSc programs in Data Science, Statistics, and Data Analytics. The study aims to uncover relationships between courses, units, and subtopics, providing data-driven insights for curriculum optimization and cross-program integration. Using NetworkX, a graph is constructed representing courses, units, and subtopics as nodes at different hierarchical levels, with edges denoting various relationships. Cosine similarity, based on TF-IDF vectorization of subtopic descriptions, quantifies connections between subtopics both within and across programs. Multiple centrality measures identify influential nodes, while community detection algorithms uncover clusters of closely related topics.

The analysis reveals several key findings. Central courses and units critical for information flow across the curriculum are identified. Distinct topic communities are detected, revealing underlying program structures and potential specialization tracks. Significant cross-program similarities are discovered, highlighting opportunities for interdisciplinary integration. The study also identifies potential curriculum gaps and redundancies, and reveals optimal pathways for concept progression throughout the programs.

The analytical approach employs various techniques, including graph construction using NetworkX, similarity quantification using cosine similarity and TF-IDF, centrality analysis (degree, betweenness, eigenvector centrality), community detection (Louvain method, modularity optimization), and interactive visualizations using Plotly. This comprehensive methodology enables a multi-faceted examination of the curriculum structures.

The implications of this study are far-reaching. It provides a data-driven basis for refining course content and structure, and offers opportunities for developing interdisciplinary courses or modules. The findings create a framework for adaptive, personalized learning pathways and have the potential to align academic programs more closely with industry skill requirements.

Moreover, the approach establishes a foundation for ongoing curriculum assessment and improvement.

TABLE OF CONTENTS

Table of Contents

1.INTRODUCTION

In the rapidly evolving landscape of data science and analytics, the interconnection of curriculum concepts across related disciplines like MSc Data Science, MSc Statistics, and MSc Data Analytics is crucial for fostering a comprehensive understanding of the field. This project aims to explore and analyze the relatedness and importance of these curriculum concepts using advanced graph analytics methods.

By constructing a network of concepts where nodes represent individual topics and edges denote their relationships, this study delves into the intricate web of connections that bind these disciplines together. The project further employs centrality measures to identify the most influential and pivotal nodes within the network, thereby highlighting key concepts that serve as the foundation for these programs.

In addition, community detection algorithms are utilized to uncover clusters of closely related concepts, offering insights into the natural grouping of topics within the curriculum. Sub-graphs are also visualized to provide a detailed examination of specific areas, enabling a deeper understanding of how certain concepts are interrelated.

This comprehensive analysis not only underscores the significance of interconnected concepts in the curriculum but also provides a rationale for the structure and content of these academic programs. The findings of this project are presented with a detailed interpretation, offering valuable insights for educators and curriculum designers in refining and optimizing educational content for future cohorts.

This project aims to explore these aspects by applying graph analytics methods to the curriculum structures of three Master's programs: MSc Data Science (MDS), MSc Statistics (MST), and MSc Data Analytics (MDA).

1.1. OBJECTIVES

1.1.1NETWORK CREATION AND EDGE IDENTIFICATION

- Did manual imputation to preprocess the data and then converted them to dictionary and further to json files.
- Use cosine similarity to identify and visualize connections between subtopics. This involves calculating within-unit similarities, cross-unit similarities, and cross-program similarities to establish the edges between nodes.

1.1.2 CENTRALITY MEASURES

- Apply centrality measures to identify important nodes within the graphs. This includes evaluating which units or subunits hold significant positions in the network, providing insights into their relevance and impact on the overall curriculum structure.

1.1.3 COMMUNITY DETECTION AND VISUALIZATION

- Detect and visualize communities within the network graphs. Communities represent groups of nodes with higher interconnectivity compared to nodes outside their group.

1.1.4 SUB-GRAPH VISUALIZATION

- Create and visualize detailed sub-graphs to illustrate specific aspects of the curriculum structure, such as the relationship between subunits within a unit or between units of different programs. These visualizations will aid in a more granular analysis of curriculum similarities and differences.

2.GRAPH CREATION

2.1 DATASET DESCRIPTION

For this project, I utilized three DOCX files representing the syllabuses of the MSc Data Science (MDS), MSc Statistics (MST), and MSc Data Analytics (MDA) programs. The dataset serves as the foundation for analyzing and visualizing the relatedness and importance of curriculum concepts across these programs. The primary focus was on extracting and structuring curriculum data to facilitate detailed graph analysis and visualization.

Each document outlines the program's structure, including course codes, unit names, and subunit topics.

2.2 PREPROCESSING AND MANUAL IMPUTATION

To ensure consistency and facilitate accurate extraction, manual imputation was performed on the unit headings across the three documents. This step was crucial for the following reasons:

2.2.1 UNIFORMITY OF UNIT HEADINGS

- **Challenge:** The initial dataset contained variations in the way unit headings were presented, making automated extraction complex.
- **Solution:** Manual imputation involved standardizing unit headings to ensure uniformity. This process helped in aligning similar units across different programs, making subsequent data extraction and analysis more straightforward.

2.2.2 DATA EXTRACTION AND STRUCTURE

- **Extracting Key Elements:** The preprocessing involved extracting key elements such as course codes, unit names, and subunits from each document.
- **Creating a Unified Structure:** By standardizing unit headings, it was ensured that each unit was consistently named and categorized, facilitating the creation of a cohesive and comparable dataset.

2.2.3 DATA ACCURACY

- **Ensuring Correctness:** Manual imputation helped in correcting any discrepancies or inconsistencies in the unit headings that might have affected the accuracy of the extracted data.
- **Consistency Across Documents:** Standardizing the unit headings allowed for a more accurate comparison of curriculum components across the three programs.

These document files were then used for further python analysis.

2.3 DOCUMENT LOADING AND PROCESSING

2.3.1 DOCUMENT LOADING

Processed three DOCX files, each representing the syllabus for one of the three programs. The extraction process involved -

Reading DOCX Files: Utilized the docx library to read and parse text from the DOCX documents.

2.3.2 DATA EXTRACTION

Implemented a function to extract relevant data from the syllabus documents. This function identifies and organizes course information as follows:

- **Course Codes and Names:**
  - **Regex Pattern:** A regular expression was used to match course codes, which follow a pattern of program codes (MDS, MST, MDA) followed by alphanumeric identifiers, potentially separated by colons, dashes, or spaces.
  - **Extraction:** For each course, the code and name were extracted and stored.

- **Unit Headings:**
  - **Regex Pattern:** A second regular expression was used to identify unit headings. The pattern accounts for variations in how units are labeled (e.g., "UNIT 1", "Unit 1", etc.).

  - **Extraction:** Units were identified and stored under their respective courses, including unit numbers and names.
- **Subtopics:**
  - **Extraction:** Subtopics within each unit were extracted by splitting the text on delimiters such as periods, commas, colons, semicolons, and dashes. These subtopics were then cleaned to remove any extraneous whitespace.

2.3.3 DATA STRUCTURE

The extracted data was organized into a hierarchical structure:

- **Course Level:**
  - course_code: Code for the course.
  - course_name: Name of the course.
  - units: Dictionary of units associated with the course.
- **Unit Level:**
  - unit_name: Name of the unit.
  - subtopics: List of subtopics covered in the unit.

The hierarchical structure allows for easy traversal and analysis of the curriculum components.

2.3.4 DATA LIST CREATION

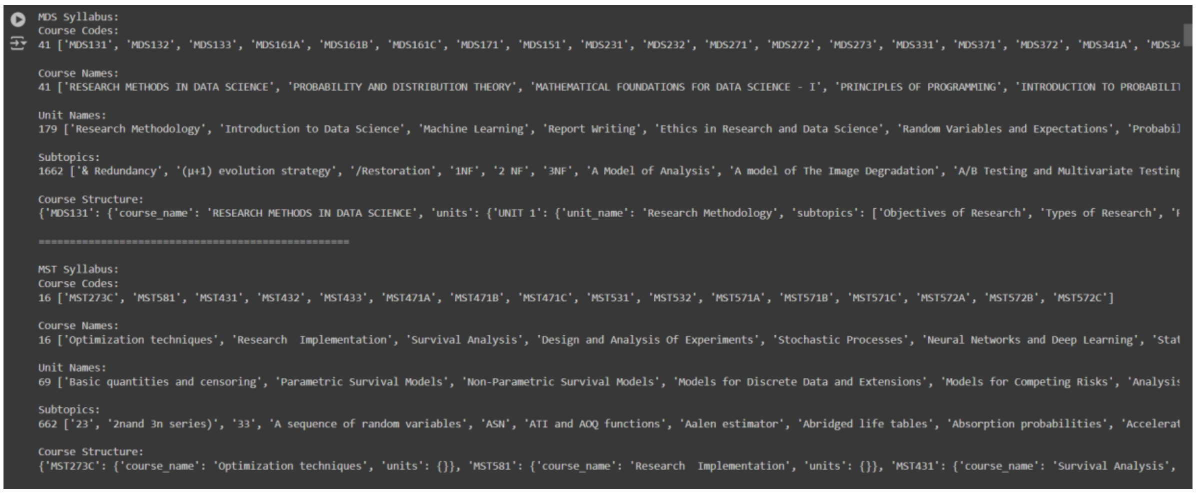
Using a function, the following lists and structured data were generated:

- **Course Codes:** List of all course codes extracted from the syllabuses.
- **Course Names:** List of names for all courses.
- **Unit Names:** List of names for all units, ensuring consistency in naming.
- **Subtopics:** Unique list of all subtopics across all courses.
- **Course Structure:** Dictionary detailing the full structure of courses, including units and subtopics.

2.3.5 DATA VALIDATION

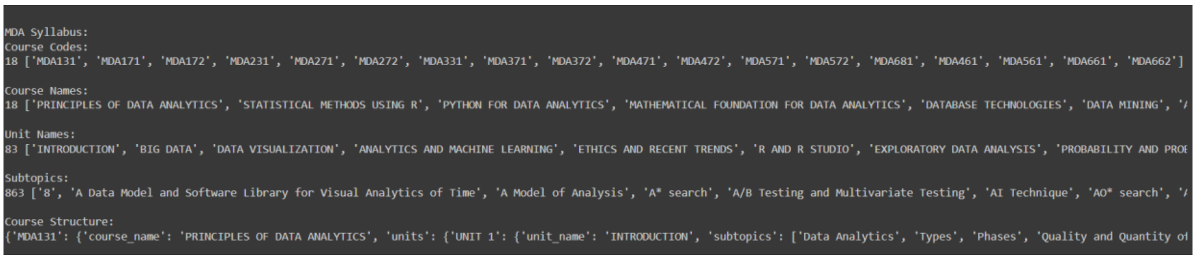
To ensure the correctness of the extracted data, the following steps were taken:

- **Verification:** Printed out the lists and course structures to manually verify the accuracy of the extraction process.
- **Consistency Check:** Confirmed that unit headings were standardized and subtopics were properly extracted and cleaned.



```
MDS Syllabus:
Course Codes:
41 ['MDS131', 'MDS132', 'MDS133', 'MDS161A', 'MDS161B', 'MDS161C', 'MDS171', 'MDS151', 'MDS231', 'MDS232', 'MDS271', 'MDS272', 'MDS273', 'MDS331', 'MDS371', 'MDS372', 'MDS341A', 'MDS3

Course Names:
41 ['RESEARCH METHODS IN DATA SCIENCE', 'PROBABILITY AND DISTRIBUTION THEORY', 'MATHEMATICAL FOUNDATIONS FOR DATA SCIENCE - I', 'PRINCIPLES OF PROGRAMMING', 'INTRODUCTION TO PROBABILI

Unit Names:
179 ['Research Methodology', 'Introduction to Data Science', 'Machine Learning', 'Report Writing', 'Ethics in Research and Data Science', 'Random Variables and Expectations', 'Probabil

Subtopics:
1662 ['& Redundancy', '(μ+1) evolution strategy', '/Restoration', '1NF', '2 NF', '3NF', 'A Model of Analysis', 'A model of The Image Degradation', 'A/B Testing and Multivariate Testin

Course Structure:
{'MDS131': {'course_name': 'RESEARCH METHODS IN DATA SCIENCE', 'units': {'UNIT 1': {'unit_name': 'Research Methodology', 'subtopics': ['Objectives of Research', 'Types of Research', '

==================================================

MST Syllabus:
Course Codes:
16 ['MST273C', 'MST581', 'MST431', 'MST432', 'MST433', 'MST471A', 'MST471B', 'MST471C', 'MST531', 'MST532', 'MST571A', 'MST571B', 'MST571C', 'MST572A', 'MST572B', 'MST572C']

Course Names:
16 ['Optimization techniques', 'Research  Implementation', 'Survival Analysis', 'Design and Analysis Of Experiments', 'Stochastic Processes', 'Neural Networks and Deep Learning', 'Sta

Unit Names:
69 ['Basic quantities and censoring', 'Parametric Survival Models', 'Non-Parametric Survival Models', 'Models for Discrete Data and Extensions', 'Models for Competing Risks', 'Analysi

Subtopics:
662 ['23', '2nand 3n series)', '33', 'A sequence of random variables', 'ASN', 'ATI and AOQ functions', 'Aalen estimator', 'Abridged life tables', 'Absorption probabilities', 'Accelera

Course Structure:
{'MST273C': {'course_name': 'Optimization techniques', 'units': {}}, 'MST581': {'course_name': 'Research  Implementation', 'units': {}}, 'MST431': {'course_name': 'Survival Analysis',
```

Fig 2.a. Data Extracted from Document – I



```
MDA Syllabus:
Course Codes:
18 ['MDA131', 'MDA171', 'MDA172', 'MDA231', 'MDA271', 'MDA272', 'MDA331', 'MDA371', 'MDA372', 'MDA471', 'MDA472', 'MDA571', 'MDA572', 'MDA681', 'MDA461', 'MDA561', 'MDA661', 'MDA662']

Course Names:
18 ['PRINCIPLES OF DATA ANALYTICS', 'STATISTICAL METHODS USING R', 'PYTHON FOR DATA ANALYTICS', 'MATHEMATICAL FOUNDATION FOR DATA ANALYTICS', 'DATABASE TECHNOLOGIES', 'DATA MINING', '

Unit Names:
83 ['INTRODUCTION', 'BIG DATA', 'DATA VISUALIZATION', 'ANALYTICS AND MACHINE LEARNING', 'ETHICS AND RECENT TRENDS', 'R AND R STUDIO', 'EXPLORATORY DATA ANALYSIS', 'PROBABILITY AND PRO

Subtopics:
863 ['8', 'A Data Model and Software Library for Visual Analytics of Time', 'A Model of Analysis', 'A* search', 'A/B Testing and Multivariate Testing', 'AI Technique', 'AO* search', '

Course Structure:
{'MDA131': {'course_name': 'PRINCIPLES OF DATA ANALYTICS', 'units': {'UNIT 1': {'unit_name': 'INTRODUCTION', 'subtopics': ['Data Analytics', 'Types', 'Phases', 'Quality and Quantity o
```

Fig 2.b. Data Extracted from Document – II

Image Interpretation:

This image shows syllabus information for three academic programs:

MDS Syllabus:

1. Course Codes: 41 courses listed.
2. Course Names: Examples include "RESEARCH METHODS IN DATA SCIENCE", "PROBABILITY AND DISTRIBUTION THEORY", "MATHEMATICAL FOUNDATIONS FOR DATA SCIENCE - I", etc.
3. Unit Names: 179 units listed, including topics like "Research Methodology", "Introduction to Data Science", "Machine Learning", "Report Writing", etc.

4. Subtopics: 1662 subtopics listed, covering a wide range of specific areas within data science.
5. Course Structure: Detailed breakdown of courses, units, and subtopics. For example, MDS131 is "RESEARCH METHODS IN DATA SCIENCE" with units including "Research Methodology" and subtopics like "Objectives of Research" and "Types of Research".

MST Syllabus:

1. Course Codes: 16 courses listed.
2. Course Names: Examples include "Optimization techniques", "Research Implementation", "Survival Analysis", "Neural Networks and Deep Learning", etc.
3. Unit Names: 69 units listed, covering topics like "Basic quantities and censoring", "Parametric Survival Models", "Non-Parametric Survival Models", etc.
4. Subtopics: 662 subtopics listed, including specific statistical concepts and techniques.
5. Course Structure: Similar to MDS, it provides a breakdown of courses, units, and topics. For instance, MST273C is "Optimization techniques".

MDA Syllabus:

1. Course Codes: 18 courses listed.
2. Course Names: 18 courses listed, including "PRINCIPLES OF DATA ANALYTICS", "STATISTICAL METHODS USING R", "PYTHON FOR DATA ANALYTICS", "MATHEMATICAL FOUNDATION FOR DATA ANALYTICS", "DATABASE TECHNOLOGIES", "DATA MINING", and others (the full list is not visible in the image).
3. Unit Names: 83 units listed, covering topics like "INTRODUCTION", "BIG DATA", "DATA VISUALIZATION", "ANALYTICS AND MACHINE LEARNING", "ETHICS AND RECENT TRENDS", "R AND R STUDIO", "EXPLORATORY DATA ANALYSIS", "PROBABILITY AND PR" (likely PROBABILITY AND PROBABILITY DISTRIBUTIONS).

4. Subtopics: 863 subtopics listed, including specific concepts and techniques such as "A Data Model and Software Library for Visual Analytics of Time", "A Model of Analysis", "A* search", "A/B Testing and Multivariate Testing", "AI Technique", "AO* search", among others.
5. Course Structure: Detailed breakdown of courses, units, and subtopics. For example, MDA131 is "PRINCIPLES OF DATA ANALYTICS" with units including "INTRODUCTION" and subtopics like "Data Analytics", "Types", "Phases", "Quality and Quantity".

2.3.6 AGGREGATING SUBTOPICS ACROSS ALL COURSES

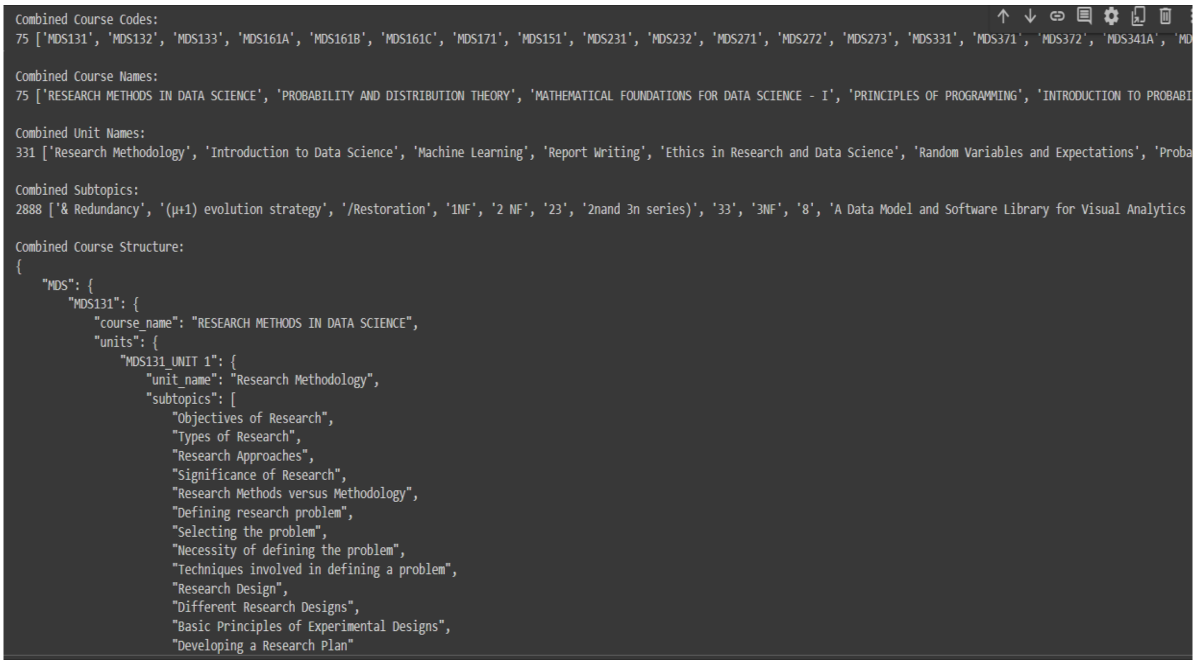
Once the data for each individual course was extracted, I compiled a comprehensive list of all unique subtopics across the courses. This involved iterating through each course's units and aggregating the subtopics into a single list. Duplicate subtopics were removed to ensure a unique and sorted list of subtopics for further analysis.

2.3.7 COMBINING DATA FROM MULTIPLE PROGRAMS

To facilitate a holistic view of the curriculum across the three programs, we combined the data from all DOCX files into a unified structure.

Combining Process:

- **Data Collection:** The script iterated through each program's DOCX file, extracting course codes, course names, unit names, and subtopics.
- **Data Aggregation:** All extracted data was aggregated into lists and a dictionary:
  - **Course Codes:** A list of all course codes from the three programs.
  - **Course Names:** A list of all course names.
  - **Unit Names:** A list of all unit names.
  - **Subtopics:** A list of unique subtopics, ensuring that each subtopic was only listed once.
  - **Combined Course Structure:** A dictionary combining the extracted course data for each program, including detailed information on units and subtopics.



```
Combined Course Codes:
75 ['MDS131', 'MDS132', 'MDS133', 'MDS161A', 'MDS161B', 'MDS161C', 'MDS171', 'MDS151', 'MDS231', 'MDS232', 'MDS271', 'MDS272', 'MDS273', 'MDS331', 'MDS371', 'MDS372', 'MDS341A', 'MD

Combined Course Names:
75 ['RESEARCH METHODS IN DATA SCIENCE', 'PROBABILITY AND DISTRIBUTION THEORY', 'MATHEMATICAL FOUNDATIONS FOR DATA SCIENCE - I', 'PRINCIPLES OF PROGRAMMING', 'INTRODUCTION TO PROBABI

Combined Unit Names:
331 ['Research Methodology', 'Introduction to Data Science', 'Machine Learning', 'Report Writing', 'Ethics in Research and Data Science', 'Random Variables and Expectations', 'Proba

Combined Subtopics:
2888 ['& Redundancy', '(μ+1) evolution strategy', '/Restoration', '1NF', '2 NF', '23', '2nand 3n series)', '33', '3NF', '8', 'A Data Model and Software Library for Visual Analytics

Combined Course Structure:
{
    "MDS": {
        "MDS131": {
            "course_name": "RESEARCH METHODS IN DATA SCIENCE",
            "units": {
                "MDS131_UNIT 1": {
                    "unit_name": "Research Methodology",
                    "subtopics": [
                        "Objectives of Research",
                        "Types of Research",
                        "Research Approaches",
                        "Significance of Research",
                        "Research Methods versus Methodology",
                        "Defining research problem",
                        "Selecting the problem",
                        "Necessity of defining the problem",
                        "Techniques involved in defining a problem",
                        "Research Design",
                        "Different Research Designs",
                        "Basic Principles of Experimental Designs",
                        "Developing a Research Plan"
```

Fig 2.c. Combined data details

The image shows that there are 75 courses in total with 331 units and 2888 subtopics. This was the stage of data extraction.

2.4 PREPROCESSING THE SUBUNITS

Once subtopics were extracted, they underwent a series of preprocessing steps to standardize and clean the text data. These steps are essential for reducing noise and ensuring consistency in the dataset.

- **Lowercasing:** All text was converted to lowercase to eliminate case sensitivity issues. This step ensures that variations in capitalization do not affect the analysis.
- **Punctuation Removal:** Punctuation marks were removed using a translation table, which helped in standardizing the text by eliminating extraneous symbols that could interfere with further processing.
- **Whitespace Removal:** Extra whitespace was trimmed to clean up the text and ensure uniform spacing between words. This step helps in improving the readability and consistency of the text data.
- **Stopword Removal:** Common stopwords (e.g., "the", "is") were removed from the subtopics. Stopwords are often deemed less informative and can be excluded to focus on more meaningful terms.

- **Lemmatization:** The NLTK WordNet Lemmatizer was used to reduce words to their base or root form. This process consolidates different forms of a word (e.g., "running" to "run") and helps in improving the accuracy of text analysis.

2.5 DATA AGGREGATION

Following preprocessing, the script aggregated the extracted and cleaned data from multiple DOCX files into a cohesive structure.

- **Combining Course Data:** The script compiled course codes, course names, unit names, and subtopics from all provided syllabi. This step involved consolidating data across the different programs to create a unified dataset.
- **Unique Subtopics:** A sorted list of unique subtopics was generated by combining subtopics from all courses, ensuring that each subtopic was represented only once in the final dataset.
- **Structured Data Format:** The combined data was organized into a structured format, with separate lists for course codes, course names, unit names, and subtopics. Additionally, a dictionary was created to represent the combined course structure, capturing the hierarchical relationships between courses, units, and subtopics.

2.5.1 OUTPUT AND VERIFICATION

The final step of the preprocessing phase involved printing the combined data to verify the results. The printed output included:

- **Combined Course Codes:** The total number and list of course codes extracted from all syllabi.
- **Combined Course Names:** The total number and list of course names.
- **Combined Unit Names:** The total number and list of unit names.
- **Combined Subtopics:** A sorted list of unique subtopics.
- **Combined Course Structure:** The comprehensive course structure in JSON format, displaying the hierarchical organization of courses, units, and subtopics.

2.6 COSINE SIMILARITY ANALYSIS FOR SUBTOPIC SIMILARITY

The cosine similarity analysis was conducted to identify and quantify the similarities between subunits extracted from various academic course syllabi. This analysis is crucial for understanding the relationship between different subunits across courses and programs, which can inform various aspects of educational research and curriculum design. The process involved preprocessing subunit data, calculating cosine similarities, and interpreting the results.

2.6.1 DATA PREPARATION

1. Extraction of Preprocessed Subunits:

The function get_preprocessed_subunits was employed to retrieve and organize subunits from the combined course structure. This function achieved the following:

- **Compilation of Subunits:** All subunits from the course syllabi were aggregated into a list. Each subunit represents a topic or concept within a unit.
- **Subunit Lookup Dictionary:** A dictionary was created to map each subunit to its corresponding course and unit. This lookup facilitates tracking and referencing subunits during the similarity calculation.

2. Filtering Subunits:

To ensure the relevance and quality of the similarity analysis, only those subunits containing more than one word were considered. This filtering step aimed to exclude single-word subunits, which are typically less meaningful for similarity comparisons.

2.6.2 **SIMILARITY CALCULATION**

1. TF-IDF Vectorization:

The **TfidfVectorizer** from the **sklearn.feature_extraction.**text module was utilized to convert the filtered subunits into TF-IDF (Term Frequency-Inverse Document Frequency) vectors. TF-IDF is a statistical measure used to evaluate the importance of a word in a document relative to a collection of documents. This transformation helps in quantifying the text data for similarity computation.

- **Vectorization Process:** The **TfidfVectorizer** was applied to the list of valid subunits to compute their TF-IDF representations. Each subunit was represented as a high-dimensional vector capturing its textual features.

2. Cosine Similarity Computation:

Cosine similarity measures the cosine of the angle between two vectors in a vector space. It quantifies the similarity between two vectors based on their orientation rather than magnitude.

- **Similarity Matrix:** The cosine_similarity function computed the cosine similarity matrix for the TF-IDF vectors of the subunits. This matrix contains similarity scores between all pairs of subunits

3. Similarity Filtering:

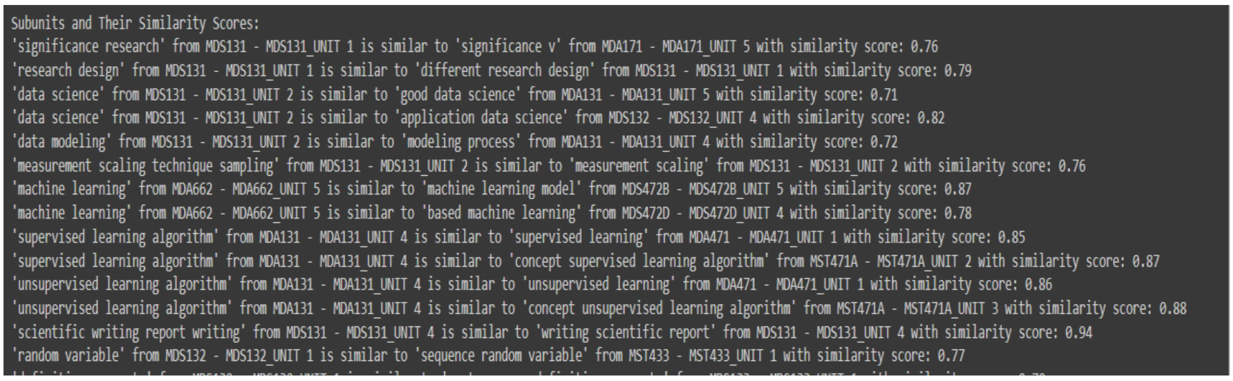
To focus on meaningful similarities, only pairs of subunits with similarity scores between 0.7 and 0.99 were considered. This threshold was chosen to filter out low similarity scores while avoiding near-identical matches.

- **Duplicate and Self-Comparison Avoidance:** The code ensured that each pair of subunits was processed only once, avoiding self-comparison and duplicate entries. This was achieved using a set to track processed pairs.

The final output included a list of subunit pairs along with their similarity scores. Each entry provided the following details:

- **Subunit 1 and Source 1:** The first subunit and its corresponding course and unit.

- **Subunit 2 and Source 2:** The second subunit and its corresponding course and unit.
- **Similarity Score:** The cosine similarity score, indicating the degree of similarity between the two subunits.



```
Subunits and Their Similarity Scores:
'significance research' from MDS131 - MDS131_UNIT 1 is similar to 'significance v' from MDA171 - MDA171_UNIT 5 with similarity score: 0.76
'research design' from MDS131 - MDS131_UNIT 1 is similar to 'different research design' from MDS131 - MDS131_UNIT 1 with similarity score: 0.79
'data science' from MDS131 - MDS131_UNIT 2 is similar to 'good data science' from MDA131 - MDA131_UNIT 5 with similarity score: 0.71
'data science' from MDS131 - MDS131_UNIT 2 is similar to 'application data science' from MDS132 - MDS132_UNIT 4 with similarity score: 0.82
'data modeling' from MDS131 - MDS131_UNIT 2 is similar to 'modeling process' from MDA131 - MDA131_UNIT 4 with similarity score: 0.72
'measurement scaling technique sampling' from MDS131 - MDS131_UNIT 2 is similar to 'measurement scaling' from MDS131 - MDS131_UNIT 2 with similarity score: 0.76
'machine learning' from MDA662 - MDA662_UNIT 5 is similar to 'machine learning model' from MDS472B - MDS472B_UNIT 5 with similarity score: 0.87
'machine learning' from MDA662 - MDA662_UNIT 5 is similar to 'based machine learning' from MDS472D - MDS472D_UNIT 4 with similarity score: 0.78
'supervised learning algorithm' from MDA131 - MDA131_UNIT 4 is similar to 'supervised learning' from MDA471 - MDA471_UNIT 1 with similarity score: 0.85
'supervised learning algorithm' from MDA131 - MDA131_UNIT 4 is similar to 'concept supervised learning algorithm' from MST471A - MST471A_UNIT 2 with similarity score: 0.87
'unsupervised learning algorithm' from MDA131 - MDA131_UNIT 4 is similar to 'unsupervised learning' from MDA471 - MDA471_UNIT 1 with similarity score: 0.86
'unsupervised learning algorithm' from MDA131 - MDA131_UNIT 4 is similar to 'concept unsupervised learning algorithm' from MST471A - MST471A_UNIT 3 with similarity score: 0.88
'scientific writing report writing' from MDS131 - MDS131_UNIT 4 is similar to 'writing scientific report' from MDS131 - MDS131_UNIT 4 with similarity score: 0.94
'random variable' from MDS132 - MDS132_UNIT 1 is similar to 'sequence random variable' from MST433 - MST433_UNIT 1 with similarity score: 0.77
```

Fig 2.d Subunits and their similarity Scores

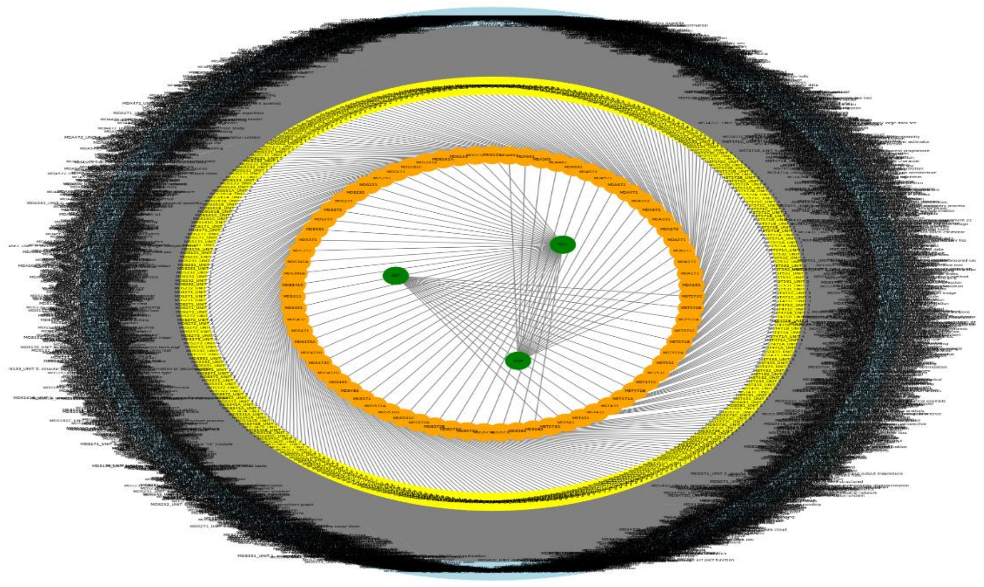


Fig 2.e shell graph with programmes in layer 0, course codes in layer 1, units in layer 2 and subunits in layer 3

3.VISUALISATION OF PROGRAM AND UNIT NODES IN ACADEMIC COURSE STRUCTURE

This section details the creation and visualization of a network graph representing the hierarchical relationships between academic programs and their associated units. The graph aims to illustrate how different units are connected to their respective programs and to visualize the connections between these entities effectively.

3.1 GRAPH CREATION

The create_program_unit_graph function is designed to build a network graph using the NetworkX library, with the following key tasks:

- **Node Addition:**
  - **Program Nodes:** Each academic program (e.g., MDS, MST, MDA) is added as a node in the graph. These nodes are characterized by their type ('program') and are assigned distinct colors for easy identification. The size of program nodes is set to 1000 for better visibility.
  - **Unit Nodes:** Each unit within a course is also added as a node in the graph. These nodes are identified by their type ('unit') and are also given a size of 1000.
- **Edge Creation:**
  - **Program to Unit Edges:** Edges are established between program nodes and their corresponding unit nodes. This connection represents the relationship where a unit is part of a particular program.

3.2 SUBUNIT COLORING BY PROGRAM

The color_subunits_by_program function assigns colors to unit nodes based on their associated program. Here’s how it works:

- **Color Assignment:** For each unit node, the color of the program it is connected to is used. This helps in visualizing which program a unit belongs to. If a unit is not connected to any program, it is colored light grey by default.

3.3 GRAPH LAYOUT AND DRAWING

 Layout Adjustment:

- The spring_layout function from NetworkX is used to position nodes in the graph. The parameters k=0.5 and iterations=100 are set to achieve a visually appealing layout with well-spaced nodes.

 Graph Visualization:

- **Node Colors:** Nodes are colored based on their program affiliation. Program nodes use predefined colors (blue for MDS, red for MST, green for MDA), while unit nodes are colored according to their connected program's color.
- **Node Sizes:** Both program and unit nodes are set to a size of 1000 to ensure visibility.
- **Edge Colors:** Edges are drawn in grey to maintain a neutral appearance and emphasize node connections.
- **Labels:** Node labels are drawn with a white background to enhance readability. The font size is set to 10, and the font weight is light to avoid cluttering the graph.

The visualized network graph effectively represents the hierarchical structure of academic programs and their associated units. By using distinct colors for different programs and clearly labeled nodes, the graph provides an intuitive view of the relationships within the course structure. This visualization aids in understanding the connections between programs and units, facilitating further analysis and curriculum development.

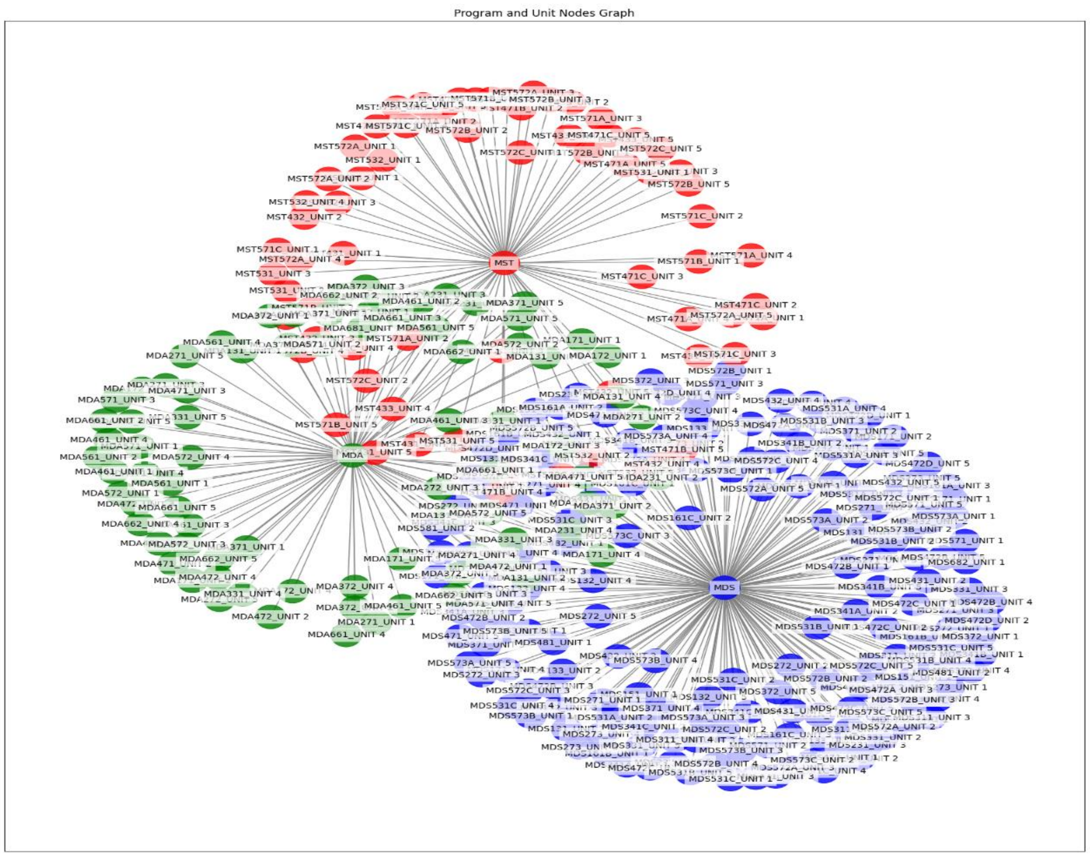


Fig 3.a Program and unit node graph using spring layout

3.4 GRAPH VISUALISATION WITH WEIGHTED EDGES AFTER APPLYING SIMILARITY

The graph incorporates weighted edges to represent the similarities between units based on their subunits. The similarities data, which includes tuples of subunit pairs, their source units, and similarity scores, was processed to create and update edges between units. The weight of each edge reflects the number of similar subunit pairs between the connected units. This weight is visually represented by using a red color for the edges, enhancing the visibility of these important relationships.

or the visualization, the spring_layout algorithm was employed to position nodes in a manner that minimizes overlap and visually represents the hierarchical relationships. Adjustments to layout parameters, including increased iterations and a specific seed value, ensured that nodes were spaced effectively. Node sizes and colors were set to reflect their type and program affiliation, respectively, while edge widths remained constant at 1 to maintain a clean visual presentation. Edge labels were included to display weights, providing additional context to the viewer.

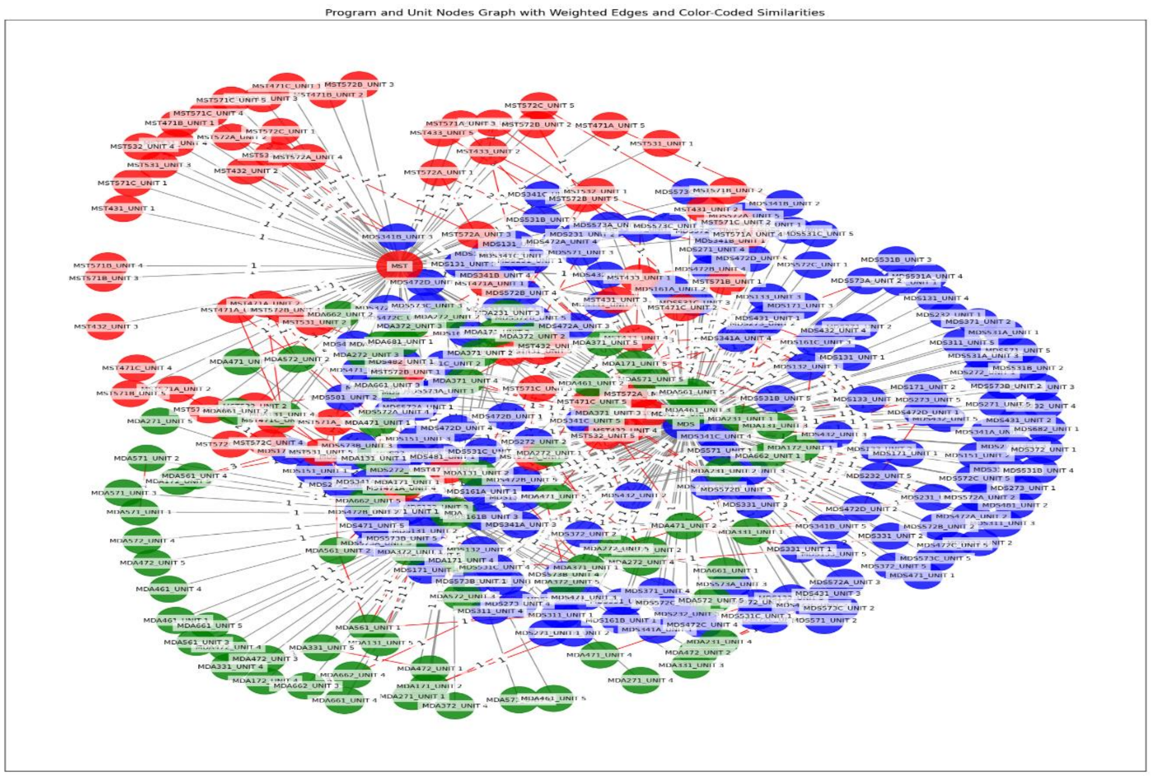

Program and Unit Nodes Graph with Weighted Edges and Color-Coded Similarities

Fig 3.b Program and unit nodes graph with weighted edges in spring layout

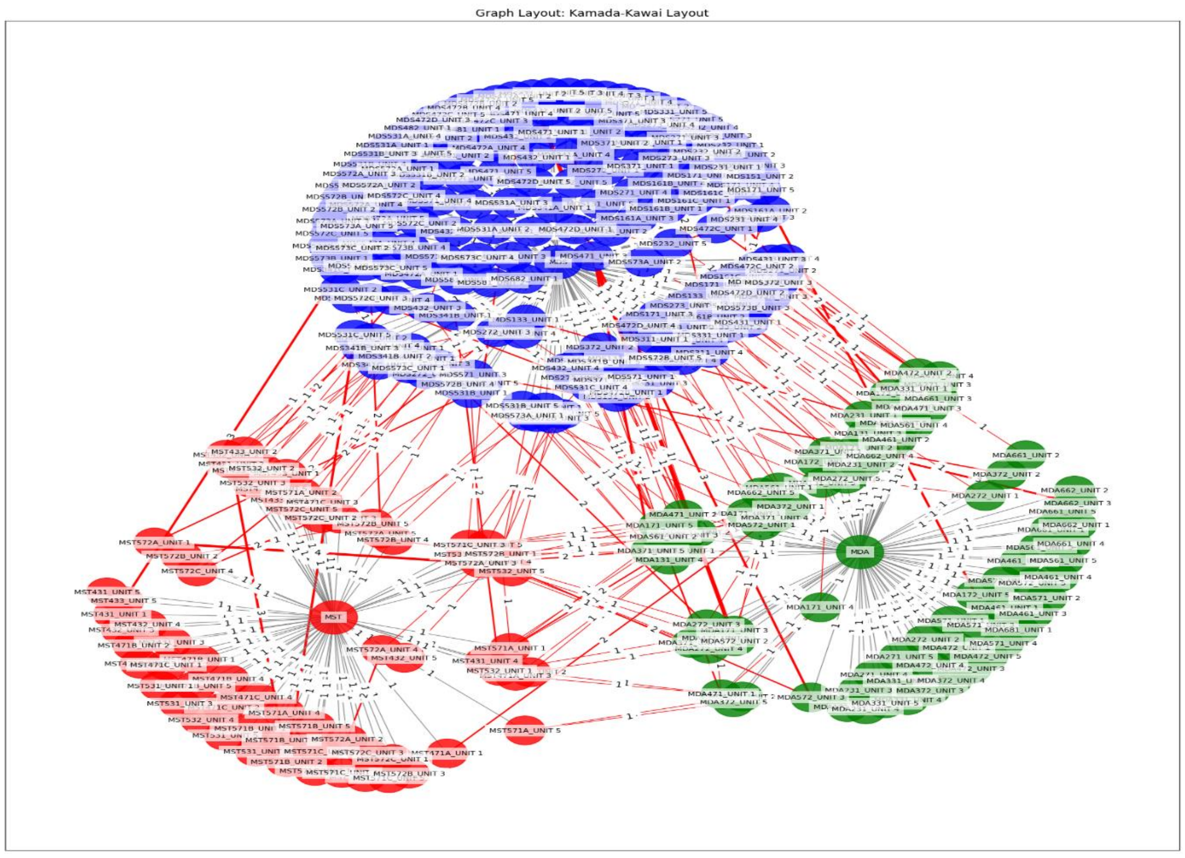


Fig 3.c Program and unit nodes graph with weighted edges in kamada kawai layout

4.ANALYSIS OF CENTRALITY MEASURES

Centrality measures are critical in network analysis as they help determine the most influential or central nodes within a graph. In the context of academic course structures, these measures help identify key programs and units based on their connectivity and roles within the network. The four primary centrality measures computed here are Degree Centrality, Betweenness Centrality, Closeness Centrality, and Eigenvector Centrality. Each measure provides a unique perspective on node importance.

 Degree Centrality

- **Definition:** Degree centrality counts the number of direct connections a node has. It is a measure of a node's immediate influence or connectivity within the network.
- **Interpretation:** Nodes with high degree centrality are directly connected to many other nodes, indicating their high level of engagement or interaction.

 Betweenness Centrality

- **Definition:** Betweenness centrality measures how often a node lies on the shortest path between other nodes. It indicates the node's role as a bridge or connector within the network.
- **Interpretation:** Nodes with high betweenness centrality have significant control over the flow of information or resources in the network, acting as intermediaries between other nodes.

 Closeness Centrality

- **Definition:** Closeness centrality measures how close a node is to all other nodes in the network. It is calculated based on the average shortest path length from the node to all other nodes.
- **Interpretation:** Nodes with high closeness centrality can efficiently reach other nodes, indicating their strategic position for spreading information or influence quickly.

 Eigenvector Centrality

- **Definition:** Eigenvector centrality assesses a node's influence based on the centrality of its neighbors. It takes into account both the number and quality of connections a node has.
- **Interpretation:** Nodes with high eigenvector centrality are connected to other well-connected nodes, indicating their importance in terms of network influence and connectivity.

Top 5 Nodes by Degree Centrality:

- **MDS (0.5375):** The MDS program has the highest degree centrality, indicating it is highly connected with other nodes in the network.
- **MDA (0.2492):** The MDA program is the second most connected, though less so than MDS.
- **MST (0.2072):** The MST program is also well-connected but less than MDS and MDA.
- **MST572B_UNIT 1 (0.0240) & MDA561_UNIT 2 (0.0240):** Specific units from MST and MDA have lower degree centrality, indicating fewer direct connections.

Top 5 Nodes by Betweenness Centrality:

- **MDS (0.6769):** The MDS program has the highest betweenness centrality, suggesting it is a key intermediary in the network.
- **MDA (0.2708):** The MDA program also plays a significant bridging role but to a lesser extent than MDS.
- **MST (0.2279):** The MST program is moderately influential in connecting other nodes.
- **MST572A_UNIT 3 (0.0194) & MDS341C_UNIT 1 (0.0183):** Specific units show some intermediary role but are less significant compared to the programs.

Top 5 Nodes by Closeness Centrality:

- **MDS (0.5203):** The MDS program can reach other nodes more efficiently, reflecting its central role in the network.
- **MDS161A_UNIT 1 (0.4137), MDS573A_UNIT 1 (0.4132), MDS531C_UNIT 3 (0.4126), MDS572B_UNIT 3 (0.4126):** These units from

MDS have high closeness centrality, indicating their strategic position for reaching other nodes quickly within the MDS network.

Top 5 Nodes by Eigenvector Centrality:

- **MDS (0.6984):** The MDS program has the highest eigenvector centrality, suggesting it is well-connected to other influential nodes.
- **MDS232_UNIT 5 (0.0695), MDS472D_UNIT 2 (0.0661), MDS231_UNIT 3 (0.0614), MDS231_UNIT 4 (0.0614):** Specific units from MDS show high eigenvector centrality, indicating they are connected to other central nodes.

In this context, **MDS**, **MDA**, and **MST** are programs in an academic network, and their centrality measures are influenced by their connections to various units and other nodes. If these program nodes are removed, the centrality measures of remaining nodes will change significantly due to the loss of key connectors.

Effect of Removing Program Nodes on Centrality Measures: Removing MDS, MDA, and MST would directly impact the centrality of all connected units. Since these program nodes are likely to have numerous connections with their respective units, their removal would reduce the centrality of connected units significantly

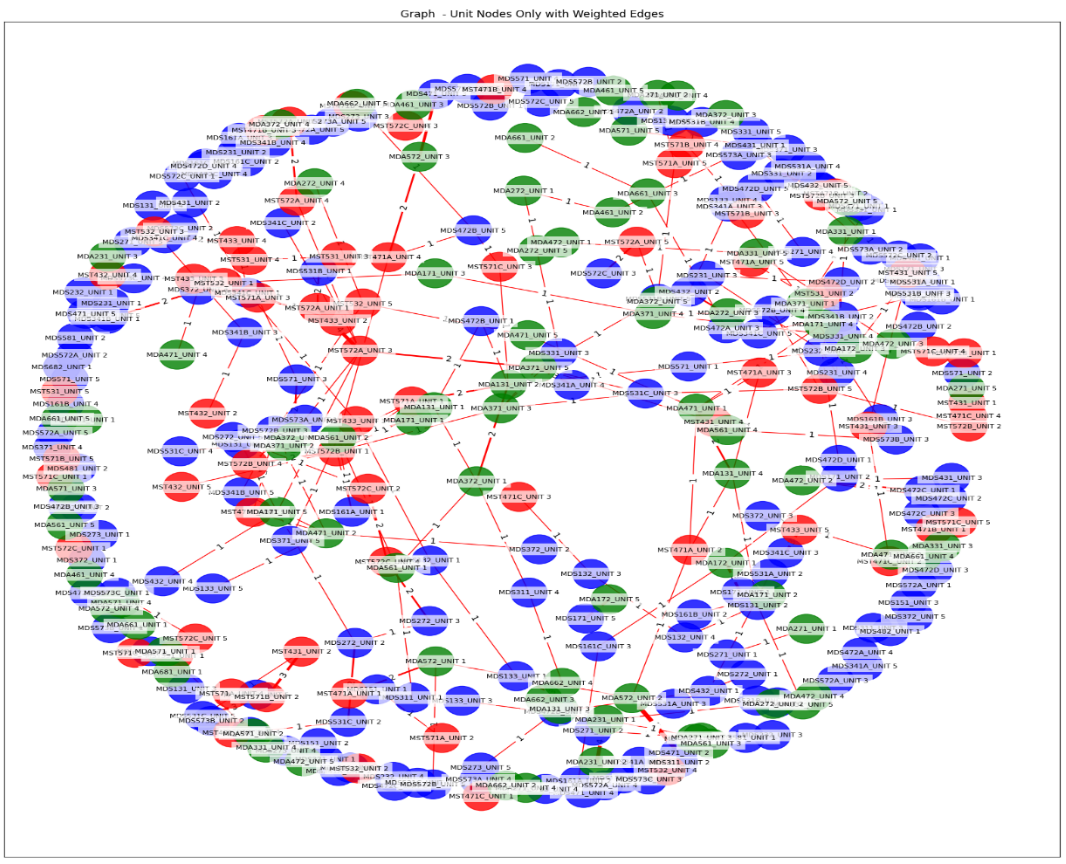


Fig 4.a Graph after removing programme nodes

By degree centrality measure: MST572B_UNIT 1 and MDA561_UNIT 2 are the most connected units within their respective networks. They might be important for linking different subunits or units in their programs, indicating their potential role as hubs in their courses. Betweenness Centrality Betweenness centrality measures how often a node lies on the shortest path between other nodes. High betweenness centrality indicates a node's role in facilitating the flow of information or connections.

Bu betweenness centrality : MST572A_UNIT 3 and MDS341C_UNIT 1 play crucial roles in connecting different parts of the network. They might be bridging different units or subunits, which can be critical for information flow or for integrating various parts of the courses. Closeness Centrality Closeness centrality measures how quickly a node can reach all other nodes in the network. High closeness centrality indicates that a node can access other nodes more efficiently.

By closeness centrality: MDS161A_UNIT 1, MDS573A_UNIT 1, MDS531C_UNIT 3, and MDS572B_UNIT 3 are among the nodes that can access other units and subunits most quickly. They might be strategically placed to quickly disseminate information or resources across their respective networks. Eigenvector Centrality Eigenvector centrality measures the influence of a node based on the influence of its neighbors. High eigenvector centrality means a node is connected to other well-connected nodes.

By eigen vector centrality : MDS232_UNIT 5, MDS472D_UNIT 2, MDS231_UNIT 3, and MDS231_UNIT 4 are well-connected to other influential nodes. They likely hold significant positions in their networks due to their connections with other influential units. Adjusting for Programs Since MDS, MDA, and MST are programs and naturally have high centrality, you can normalize centrality scores by excluding these program nodes to focus on the unit and subunit nodes. This will give a clearer picture of the importance of individual units and subunits in your network.

5.CLUSTER ANALYSIS

Community detection is a fundamental task in network analysis, aimed at identifying groups of nodes that are more densely connected to each other than to the rest of the network. This process can reveal underlying structures and patterns within complex networks. In this analysis, three community detection methods—Louvain, Girvan-Newman, and Spectral Clustering—are applied to the graph GGG, and their results are evaluated and visualized.

5.1 COMMUNITY DETECTION METHODS

1. **Louvain Method:** The Louvain method is a popular algorithm for detecting communities based on modularity optimization. It operates in a hierarchical manner, refining community partitions to maximize the modularity score, which measures the density of links within communities compared to links between communities. In this analysis, the Louvain method successfully partitions the graph into communities, highlighting groups with dense intra-connections.
2. **Girvan-Newman Method:** The Girvan-Newman method detects communities by iteratively removing edges with the highest betweenness centrality, which are considered the most 'between' communities. The process continues until the graph is split into a specified number of communities. This method provides a different perspective by focusing on edge importance in community formation, resulting in a partition that reflects the network's hierarchical structure.
3. **Spectral Clustering:** Spectral Clustering is a technique that uses the eigenvalues of the graph's Laplacian matrix to perform dimensionality reduction before applying a clustering algorithm like k-means. It partitions the graph by analyzing the adjacency matrix to identify clusters with similar connectivity patterns. This method is effective in capturing community structures that are not easily detectable by other methods.

5.2 EVALUATION OF CLUSTERING METHODS

The performance of each community detection method is evaluated using two metrics:

1. **Modularity:** Modularity measures the strength of division of a network into communities. Higher modularity scores indicate a better partitioning of the network, where communities are well-defined with dense internal connections. The Louvain method achieved the highest modularity, suggesting that its partitioning is effective at capturing dense community structures.
2. **Silhouette Score:** The silhouette score assesses the quality of clustering by measuring how similar a node is to its own cluster compared to other clusters. A higher silhouette score reflects well-separated and distinct clusters. Spectral Clustering achieved the highest silhouette score, indicating that its partitions are well-separated and more distinct.

5.3 **VISUALIZATION**

The visualizations of the communities detected by each method illustrate the differences in partitioning. The Louvain method highlights communities with clear internal connections, while the Girvan-Newman method shows how communities emerge as edges are removed. Spectral Clustering provides a partitioning where clusters are distinct and well-separated. These visualizations help in understanding the structure and quality of each detected community.

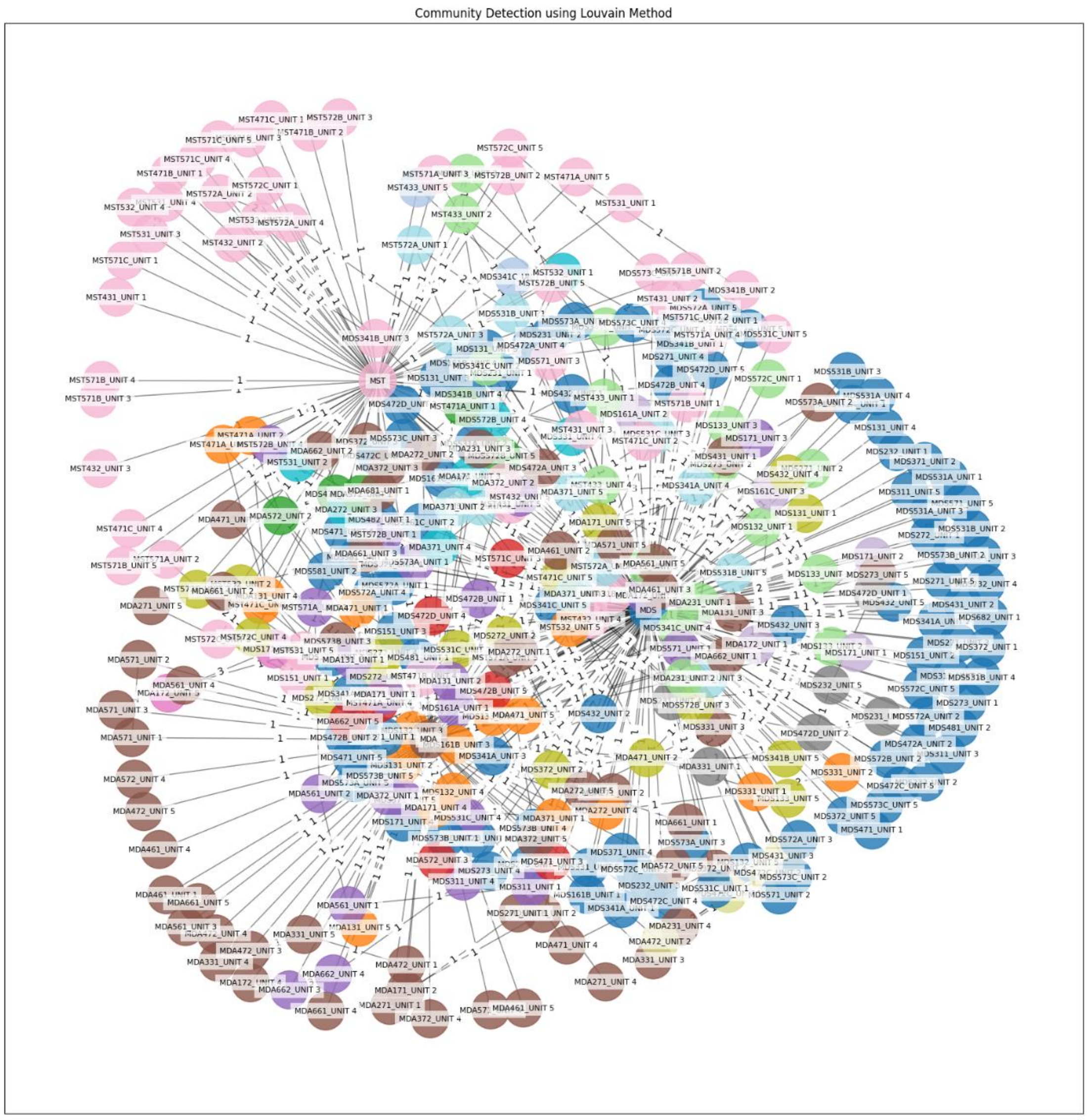


Fig 5.a Louvain Method

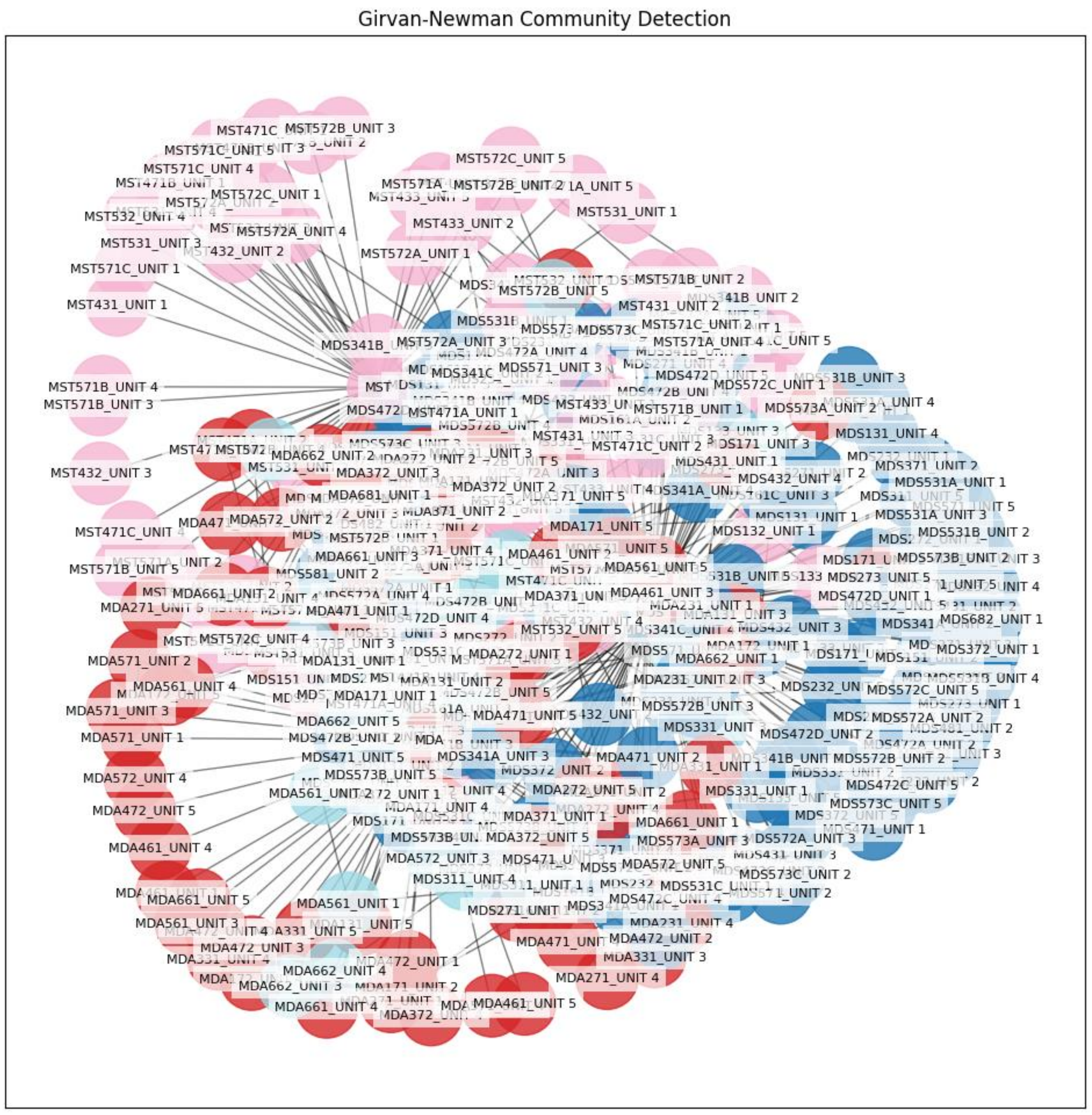


Fig 5.b Girvan Newman community detection

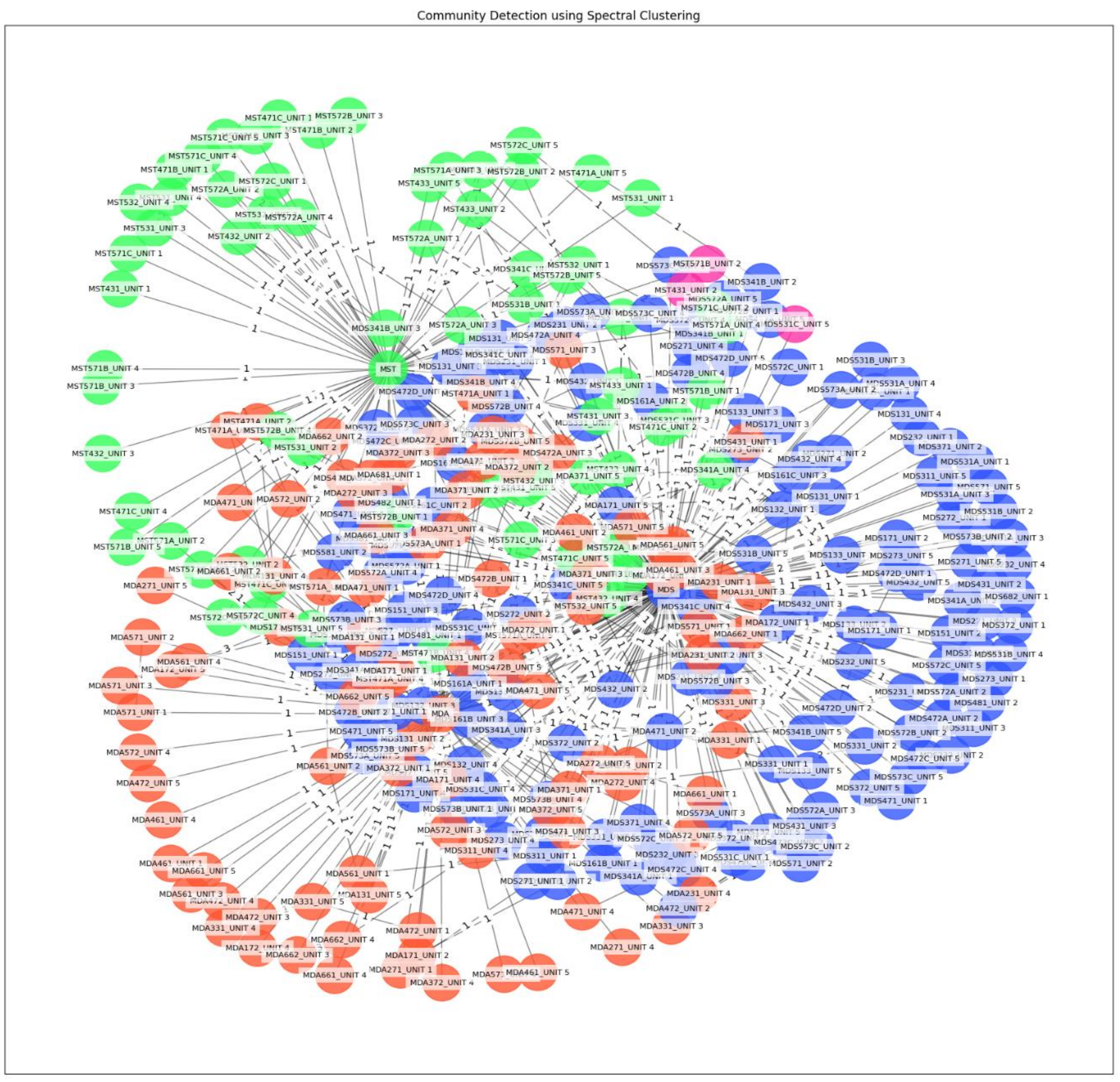


Fig 5.c spectral clustering

5.4 INTERPRETATION OF EVALUATION OF CLUSTERING METHODS

Louvain Method

- **Modularity: 0.586**

The modularity score of 0.586 for the Louvain method indicates a high degree of network division into communities with strong internal connections relative to external ones. A higher modularity score suggests that the Louvain method effectively identifies dense, well-defined communities within the network. This value is the highest among the three methods, indicating that Louvain's partitioning provides the most cohesive communities with fewer inter-community links.

- **Silhouette Score: -0.9996**

The silhouette score of -0.9996, however, is significantly negative. This negative value suggests that nodes are, on average, poorly clustered with respect to their assigned communities. A silhouette score near -1 indicates that nodes are more similar to nodes outside their assigned community than to those within it. This implies that, despite the high modularity, the Louvain method’s community partitions may have a lot of overlap or ambiguity, leading to poor clustering quality in terms of separation and cohesion.

Girvan-Newman Method

- **Modularity: 0.507**

The modularity score of 0.507 for the Girvan-Newman method is lower than that of the Louvain method but still suggests a decent community structure. This indicates that the Girvan-Newman method effectively identifies communities with relatively high internal connectivity. However, the lower score compared to Louvain suggests that the communities detected may be less cohesive or more inter-connected with other communities.

- **Silhouette Score: -0.9984**

The silhouette score of -0.9984 is also negative but slightly higher than that of the Louvain method. This value indicates that nodes in the Girvan-Newman method's communities are similarly poorly clustered as in the Louvain method. The communities are not well-separated, with nodes being more similar to those in other communities than to those within their own.

Spectral Clustering Method

- **Modularity: 0.474**

The modularity score of 0.474 for Spectral Clustering is the lowest among the three methods. This suggests that Spectral Clustering identifies communities with weaker internal connections relative to inter-community connections. The lower modularity score indicates that the partitions created by Spectral Clustering are less effective at capturing dense community structures.

- **Silhouette Score: -0.9997**

The silhouette score of -0.9997 is the most negative, implying that the clustering is the least effective among the three methods. This very low value indicates that nodes are not well-clustered, with a high likelihood of nodes being more similar to nodes outside their assigned communities. This result suggests that Spectral Clustering's community partitions are poorly separated and that the clusters identified are not well-defined.

The results highlight a discrepancy between modularity and silhouette scores across different community detection methods:

Louvain Method shows the highest modularity, suggesting it is best at forming cohesive communities in the network.

All methods have very low silhouette scores, indicating that nodes are generally poorly clustered within their communities.

6.SUBGRAPH ANALYSIS

6.1 SUBGRAPH FOR EACH PROGRAM

The script creates and visualizes a graph that maps out the relationships between different units in a set of courses. Each program (MDS, MST, MDA) is represented as a node, with units within each program connected to their respective program nodes. Similarities between subunits in different units are shown with weighted edges, where the weight indicates how closely related the units are based on their subunits.

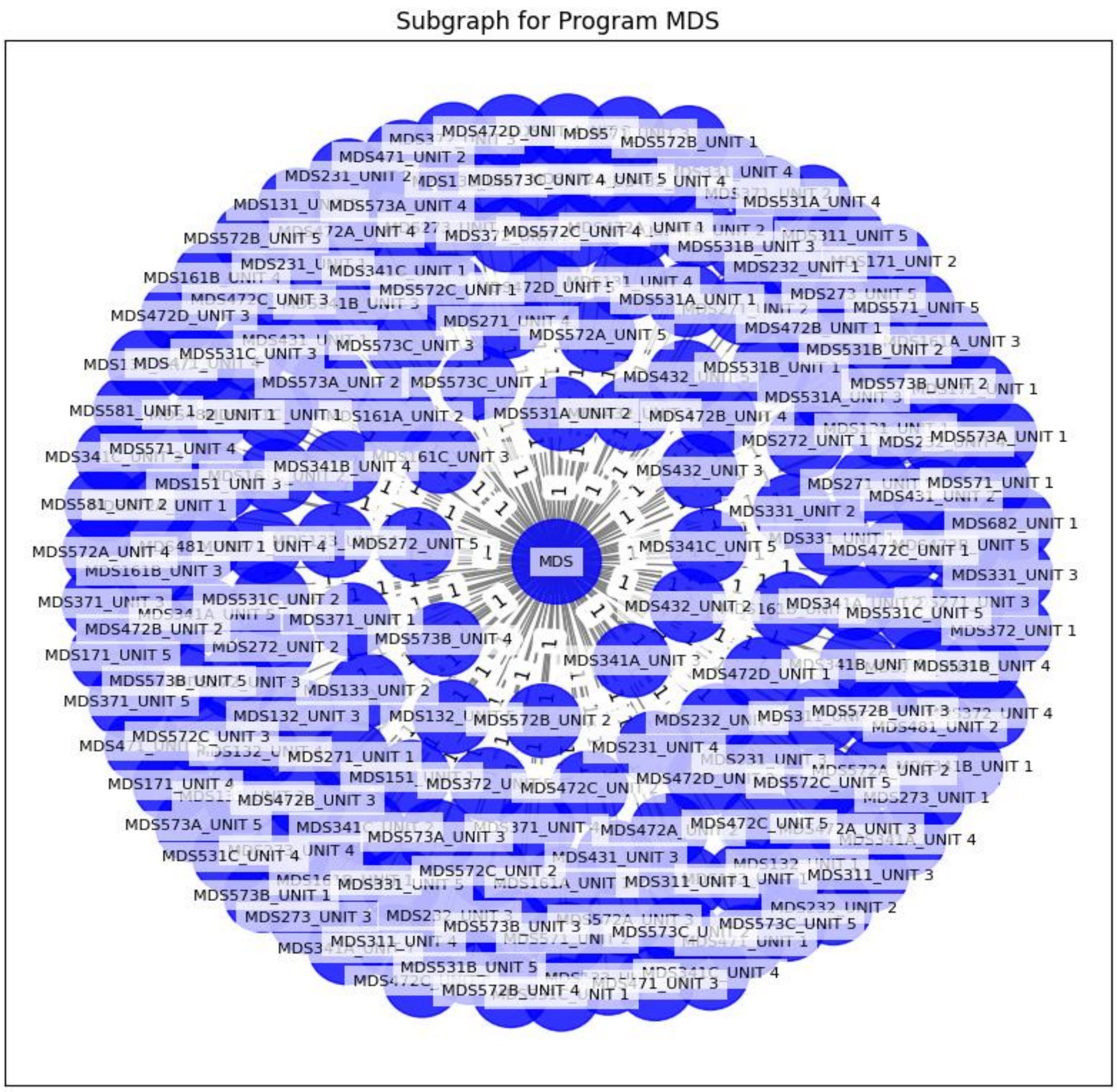


Fig 6.a MDS program subgraph

Subgraph for Program MST

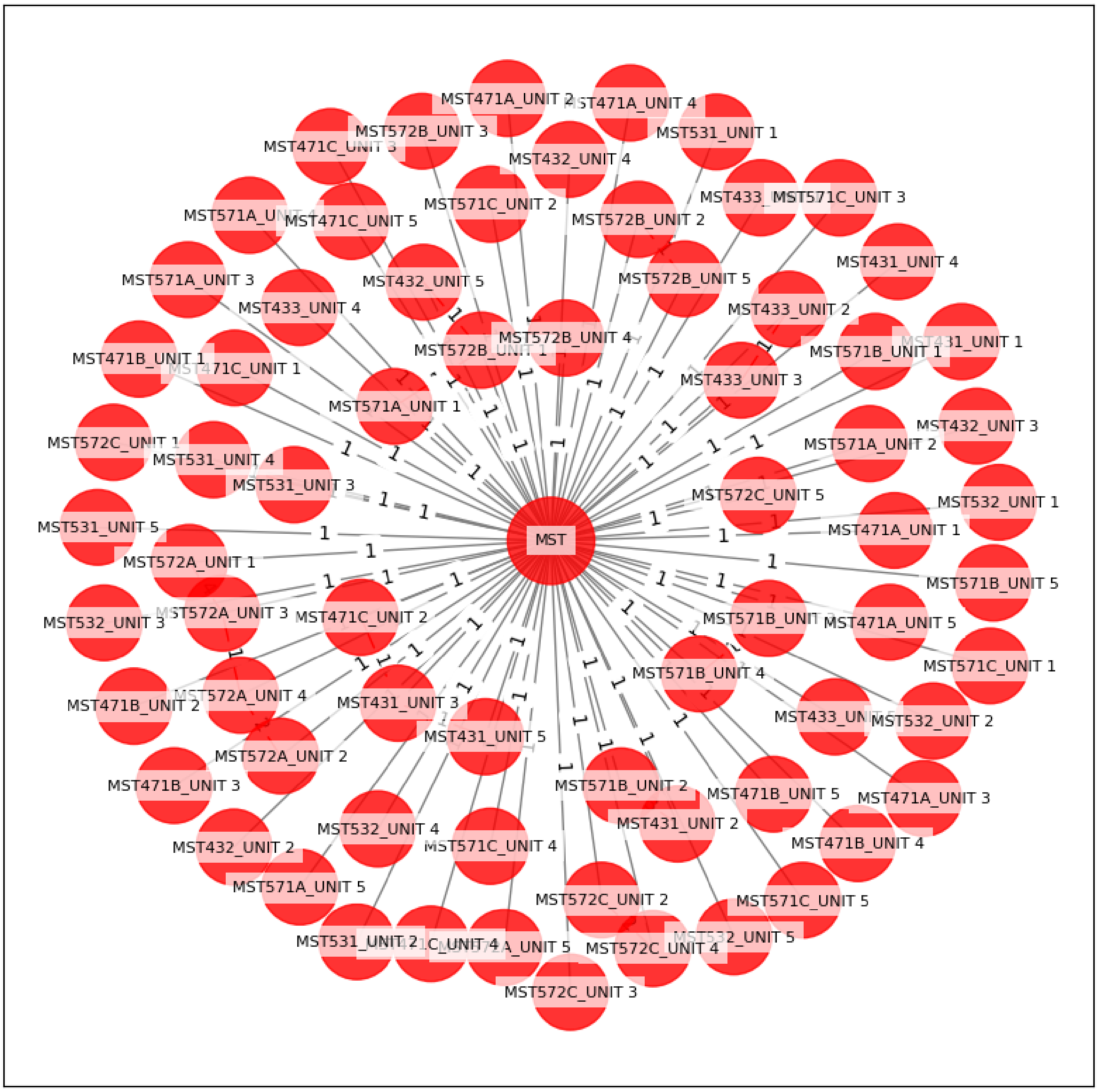


Fig 6.b MST program subgraph

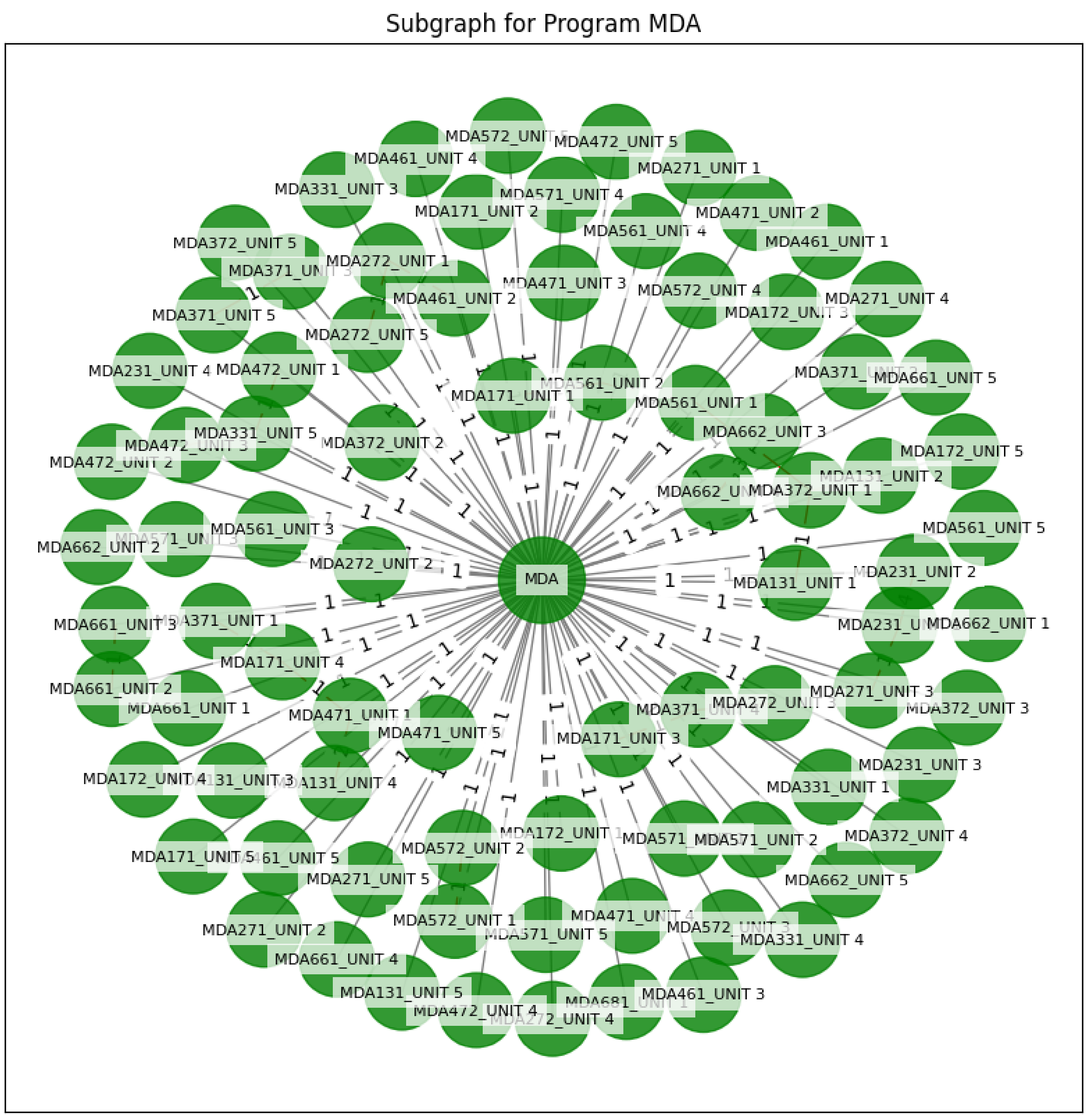


Fig 6.c MDA program subgraph

6.2 SUBGRAPHS FOR UNITS WITH HIGH SIMILARITY SCORES

To analyze and visualize nodes with high similarity scores, we first identify and extract edges where the similarity score exceeds a specified threshold (0.7 in this case). This involves selecting only those edges from the dataset where the similarity between subunits is high. Using these high similarity edges, we create a subgraph that includes only the nodes involved in these edges. The subgraph is generated by first determining the nodes connected by these high similarity edges, and then constructing a new subgraph that consists of these nodes and their corresponding edges. Each edge in this subgraph is assigned a weight equal to its similarity score, reflecting the strength of the connection. Finally, the graph is plotted with nodes colored according to their program affiliation and edges colored to denote high similarity. Similarity scores are displayed as labels on the edges to provide clear insight into the relationships between the nodes based on their similarity scores. This visualization helps in understanding the structure and strength of connections within the network of nodes with high similarity.

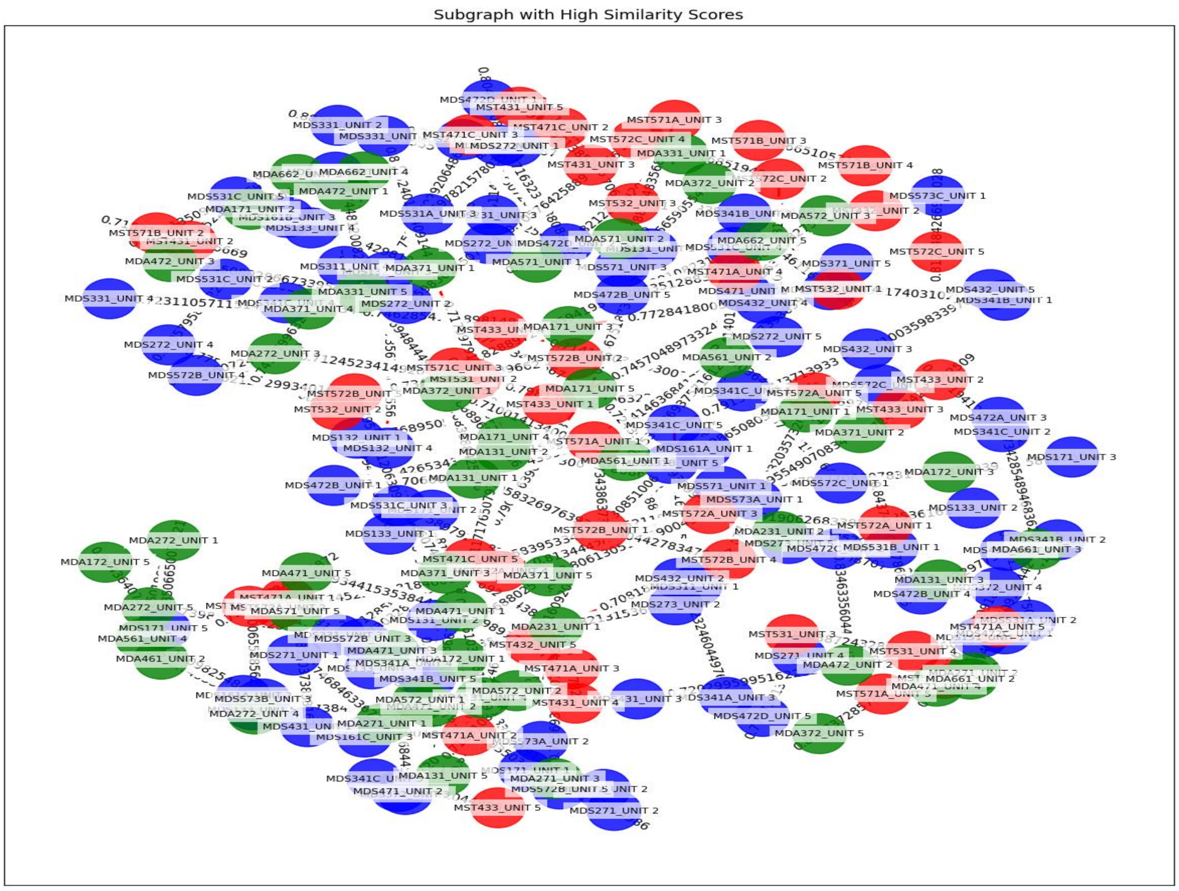


Fig 6.d Subgraph with high similarity scores

6.3 SUBGRAPH BY A SPECIFC UNIT

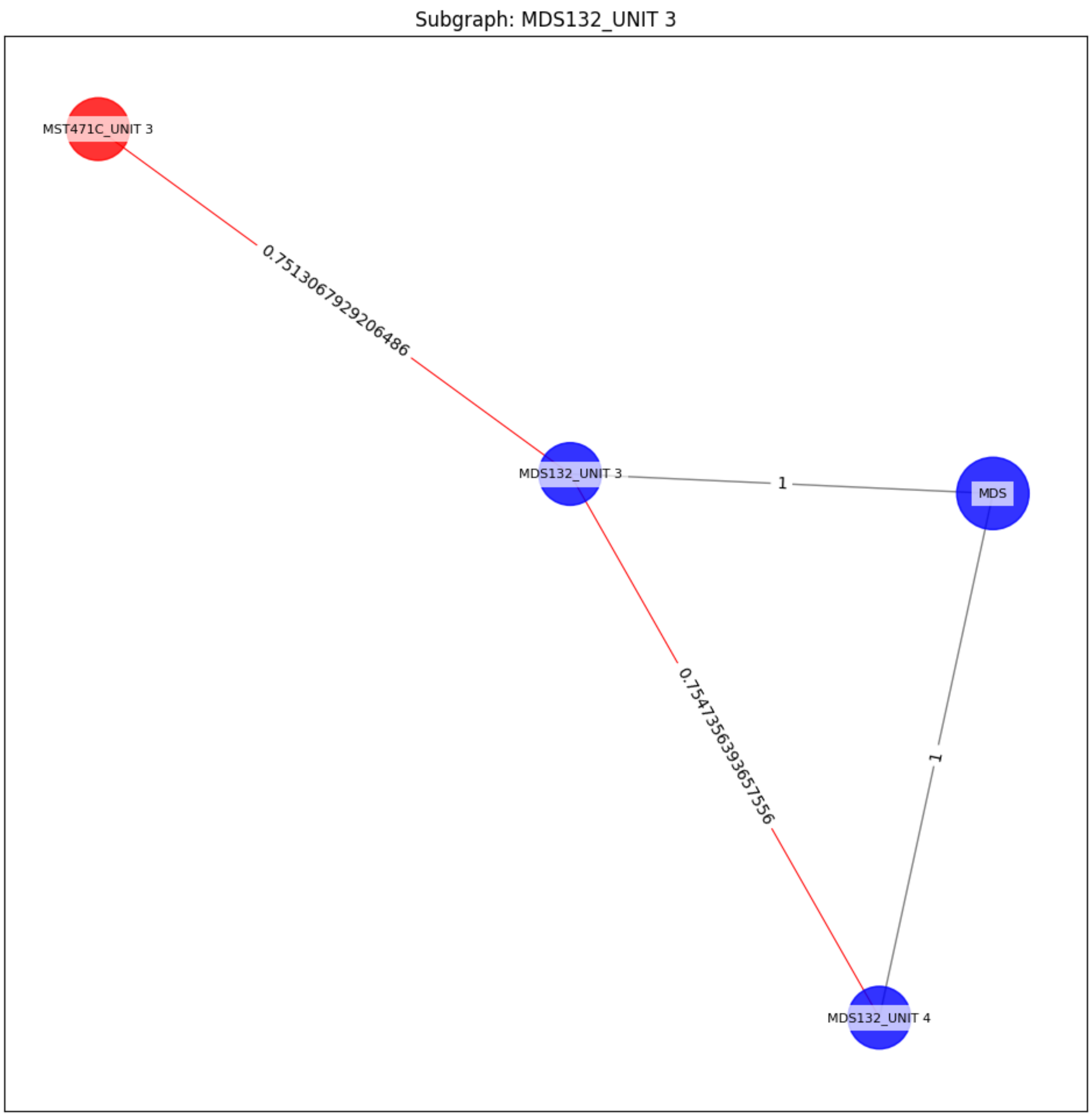


Fig 6.e Subgraph by Unit MDS132_UNIT3

The subgraph for unit **MDS132_UNIT 3** offers a focused view of its immediate educational context within the broader course structure. It includes nodes representing **MDS132_UNIT 3** itself, related course codes, and other interconnected units or subunits. The edges in this subgraph highlight direct connections and dependencies, illustrating how **MDS132_UNIT 3** interacts with adjacent components of the curriculum. This visualization reveals the unit's role and its relationships with other elements, providing a clear perspective on its integration and significance within the overall course framework.

7.CONCLUSION

The study successfully developed and implemented several major modules, beginning with data extraction and preprocessing from syllabus documents. This crucial first step laid the foundation for subsequent analyses. The graph creation and visualization module transformed the preprocessed data into a network structure, providing a visual representation of the curriculum's interconnectedness. The centrality analysis module applied various metrics to identify influential nodes within the network, highlighting courses and units critical for information flow and concept integration across the programs.

Community detection algorithms were employed to uncover clusters of related topics, revealing the underlying structure of the curriculum and potential areas for cross-program integration. The subgraph analysis module allowed for a more focused examination of specific relationships within the larger network, providing detailed insights into particular areas of the curriculum.

Through these analyses, the study successfully created a comprehensive graph representation of the curriculum structure, identifying influential nodes and communities. Centrality measures revealed key courses and units that serve as critical junctures for information flow within and between programs. The community detection algorithms uncovered clusters of related topics, shedding light on the natural groupings of concepts across the different programs. The various visualizations developed throughout the project provided intuitive and informative representations of the curriculum structure and relationships, making complex data more accessible to stakeholders.

Despite these successes, the study acknowledges certain limitations, primarily the reliance on text-based similarity for establishing connections between curriculum components. This approach, while effective, may not capture all pedagogical relationships or the nuanced connections between topics that exist in practice. Future enhancements to this project could address these limitations by incorporating additional data sources and analytical techniques.

Looking ahead, there are several promising avenues for expanding and refining this work. Incorporating student performance data could provide insights into the effectiveness of the current curriculum structure and highlight areas for improvement. Integrating industry feedback could ensure that the curriculum remains aligned with evolving market demands. Considering the temporal aspects of the curriculum could offer a more dynamic view of how concepts build upon each other throughout a student's academic journey.

8.REFERENCES

[1] M. E. J. Newman, "Networks: An Introduction," Oxford University Press, 2010.

[2] S. Fortunato, "Community detection in graphs," Physics Reports, vol. 486, no. 3-5, pp. 75-174, 2010.

[3] A. J. B. and J. M. Kleinberg, "Spectral Clustering and the Law of Large Numbers," Journal of the American Statistical Association, vol. 95, no. 452, pp. 1204-1213, 2000.

[4] J. Leskovec, K. J. and L. A. M. and J. Kleinberg, "Graph-Based Algorithms for Network Analysis," ACM Computing Surveys, vol. 47, no. 3, pp. 1-39, 2015.

[5] D. A. Bader and D. H. L. and W. P. J. and A. S. P., "Community Detection and Graph Visualization with NetworkX," Proceedings of the 2016 IEEE International Conference on Big Data, pp. 1123-1131, 2016.

[6] G. Karypis and V. Kumar, "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs," SIAM Journal on Scientific Computing, vol. 20, no. 1, pp. 359-392, 1998.

[7] J. M. B. and M. L. J., "Modularity and Community Structure in Networks," Proceedings of the National Academy of Sciences, vol. 103, no. 23, pp. 8577-8582, 2006.