COMMUNITY DETECTION USING GENETIC ALGORITHMS

A Thesis submitted in partial fulfillment of the requirements
for the degree of Master of Technology
in
Computer Technology

Submitted by:

NILANJAN MANDAL
Exam Roll No. : M6TCT24018
Class Roll No. : 002110504023
Registration No. : 160094 of 2021-2022

Under the Guidance of

Prof. Nirmalya Chowdhury
Department of Computer Science and Engineering
Jadavpur University

Department of Computer Science and Engineering
Faculty of Engineering and Technology
Jadavpur University, Kolkata, India

May, 2024

Department of Computer Science and Engineering
Faculty of Engineering and Technology
Jadavpur University, Kolkata 700032

CERTIFICATE OF APPROVAL

(Only in case the thesis is approved)

This is to certify that the thesis entitled **" Community Detection Using Genetic Algorithms"** is a bonafide record of work carried out by Nilanjan Mandal in partial fulfillment of the requirements for the award of the degree of Master of Technology in Computer Technology in the Department of Computer Science and Engineering, Faculty of Engineering and Technology, Jadavpur University during the period July 2023 to May 2024. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed, or conclusion herein but approve the thesis only for the purpose for which it has been submitted.

_____

(Sign of Examiner)
Date:

_____

(Sign of Examiner)
Date:

TO WHOM IT MAY CONCERN

This is to certify that the work embodied in this thesis entitled **"Community Detection Using Genetic Algorithms"** has been satisfactorily completed by Nilanjan Mandal (Univ Reg No: 160094 of 2021-2022; Class Roll No: 002110504023 ; Exam Roll No: M6TCT24018) for partial fulfillment of the requirements for the award of the Master of Technology in Computer Technology degree of the Department of Computer Science and Engineering, Faculty of Engineering and Technology, Jadavpur University, during the academic year 2021-24.

Prof. Nirmalya Chowdhury
Supervisor
Dept. of Computer Science and Engineering
Jadavpur University

Prof. Nandini Mukhopadhyay
Head of the Department
Dept. of Computer Science and Engineering
Jadavpur University

Prof. Dipak Laha
DEAN, Faculty of Engineering and Technology
Jadavpur University

ACKNOWLEDGEMENT

Firstly, I would express my deep and sincere gratitude to my supervisor Prof. Nirmalya Chowdhury, Department of Computer Science and Engineering, Jadavpur University, without whose continuous support and encouragement this work would not have been possible. His assistance, valuable suggestions, and personal guidance throughout the duration of the project have played a pivotal role, without which I never would have been able to reach this far.

I would also like to thank Prof. Nandini Mukhopadhyay, Head of the Department of Computer Science and Engineering, and Prof. Dipak Laha, DEAN of Faculty of Engineering and Technology, Jadavpur University, for providing me with all the facilities and for their support to the activities of this research.

I am thankful to my parents who have always been my constant source of support and inspiration. Last, but not the least, I would like to thank all my friends, classmates, and respected teachers for their valuable suggestions and helpful discussions.

Regards,
Nilanjan Mandal
Class Roll No.: 002110504023
Exam Roll No.: M6TCT24018
University Reg. No.: 160049 of 2021-22
Department of Computer Science and Engineering
Jadavpur University

_____-

Signature

DECLARATION OF ORIGINALITY AND COMPLIANCE OF ACADEMIC ETHICS

I hereby declare that this thesis contains a literature survey and original research work by the undersigned candidate, as part of Master of Technology in Computer Technology Degree of the Department of Computer Science and Engineering.
All information in this document has been obtained and presented in accordance with academic rules and ethical conduct.
I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name : Nilanjan Mandal

Class Roll Number : 002110504023

Exam Roll Number : M6TCT24018

Thesis Title : Community Detection Using Genetic Algorithms

Signature With Date :

I dedicate this thesis to my parents for their continuous support and encouragement throughout my journey

Contents

1 Overview of Community Detection **1**

1.1 Introduction of Community Detection 1

1.2 Differences Between Community Detection and Clustering . . 5

1.3 Methods in the Field of Community Detection 8

1.4 Evaluation and Validation of Detected Communities 12

1.5 Applications of Community Detection 25

2 Utilization of Graphs and Networks in Community Detection **29**

2.1 Graph 30

2.2 Networks 34

2.3 Usage of Graphs in Community Detection 44

2.4 Usage of Networks in Community Detection 47

3 Aim and Objective of the Thesis **50**

3.1 Motivation 50

3.2 Aim and Objective 51

3.3 Contribution 51

3.4 Thesis Organization 52

4 Introduction to Binary Attributed Network **53**

4.1 Introduction 53

4.2 Applications of Binary Attributed Network 54

5 Procedure of Community Detection in Binary Attributed Network **62**

5.1 Introduction 62

5.2 Methodologies and Techniques 63

6 Various Methods for Optimization 66

6.1 Introduction 66
6.2 Different Techniques of Optimization 68
6.3 Single-Objective Optimization Method 74
6.4 Multi-objective Optimization Method 76
6.5 Genetic Algorithm (GAs) 81
6.6 Non-dominated Sorting Method 90
6.7 (Non-dominated Sorting Genetic Algorithm II(NSGA-II) 97
6.8 (Non-dominated Sorting Genetic Algorithm III (NSGA-III) . . 99

7 Survey of Literatures 103

8 The Proposed Method 112

8.1 Genetic Algorithm on Community Detection Problem 112
8.2 Single-objective and Multi-objectives 113
8.3 Identification of Overlapping Nodes 115
8.4 Chromosome 117
8.5 Objective Functions 119
8.6 Population Initialization 122
8.7 Genetic Operators 124
8.8 Algorithm 127

9 Data Collection and Preparation 131

9.1 Introduction 131
9.2 Description about the Data 148

10 Experimental Results 149

10.1 Evaluation Metrics 149
10.2 Comparison Between Different Methods 150

11 Conclusion and Scope for Future Works 152

Bibliography **155**

List of Figures

1.1 Community Detection 3
1.2 Different method of Community Detection 5

2.1 Directed and Undirected Graph 31
2.2 Weighted Graph 32
2.3 Bipartite Graph 33
2.4 Complete Graph 34
2.5 Social Network 36
2.6 Biological Network 38

4.1 Example of an Attribute Network 54

6.1 Pareto-optimal front 79
6.2 Non-domnation example 80
6.3 Population, Chromosome and gene 82
6.4 Crossover 86
6.5 Mutation 87
6.6 Non Dominated Sorting 94
6.7 NSGA-II 98
6.8 Crowding Distance 98
6.9 NSGA 3 Pareto set 100
6.10 Solutions selected from the splitting front after non-domination sortiing 101
6.11 NSGA3 (solutions) 102

8.1 Procedure to determine whether node 4 is an overlapping candidate node 117
8.2 Tanimoto Similarity 121

List of Tables

9.1 Benchmark Datasets 148

10.1 Experimental Results in Terms of Density 150
10.2 Experimental Results in Terms of Ext. Modularity 150

Chapter 1

Overview of Community Detection

1.1 Introduction of Community Detection

Community detection is a crucial field of study within network science. It focuses on identifying clusters of nodes that exhibit higher levels of connectivity with each other compared to the broader network. This process is vital for comprehending the structural and operational characteristics of intricate systems like social networks, biological networks, and technological networks. This essay will delve into the fundamental principles and methodologies of community detection, along with their practical uses and associated difficulties.

Communities are a prevalent feature in numerous networks, and a single network may encompass multiple communities. For instance, when we examine our social media accounts and identify the individuals with whom we interact frequently, they form a tightly-knit community.

Machine learning algorithms can leverage community detection to identify groups of entities exhibiting similar traits and extract these groupings for diverse applications.

The initial step in community detection involves establishing an appropriate

metric for gauging community structure. A community can be defined as a subgroup of nodes that exhibit stronger internal connections compared to their connections with the rest of the network. This can be assessed using various metrics such as modularity [1], conductance, or density. Modularity stands out as one of the most commonly utilized measures for evaluating community structure. It quantifies the disparity between the observed number of edges within communities and the anticipated number of edges in a random network with a similar degree distribution. A high modularity score signifies well-defined communities, while a low score indicates a network that is either random or poorly structured.

The concept of modularity was initially introduced by Newman and Girvan in 2004 [2], and it has subsequently evolved into one of the most extensively employed metrics in community detection. Modularity is quantified on a scale from -1 to 1, with higher values signifying a more effective partitioning of the network into communities. Nevertheless, modularity does come with certain limitations, including a resolution limit, which implies that it might not effectively detect smaller communities within larger ones.

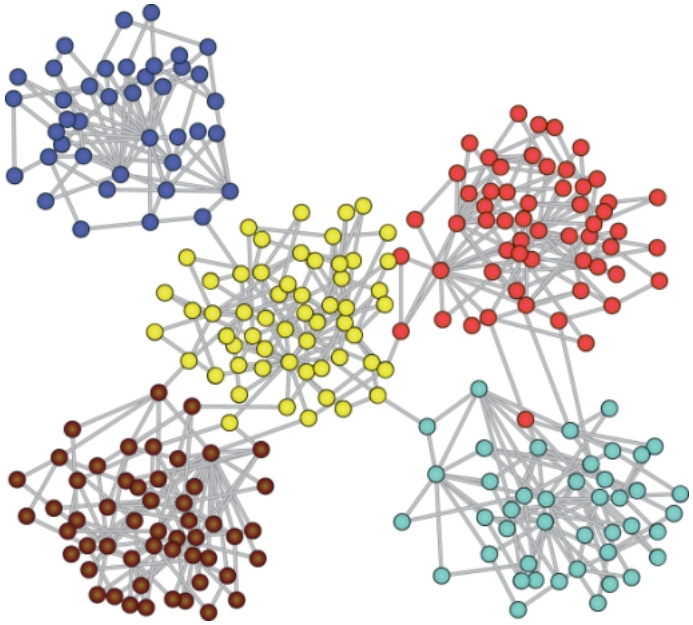


Figure 1.1: Community Detection
[3]

Once a community structure metric is established, the subsequent task involves pinpointing the actual communities within the network. There exists a multitude of algorithms and methodologies for community detection, each possessing its own set of advantages and drawbacks. Some of the most popular algorithms are grounded in graph partitioning, spectral clustering, hierarchical clustering, and probabilistic modeling. Graph partitioning algorithms[4] segment the network into a fixed number of communities by optimizing an objective function linked to the community structure metric. Spectral clustering algorithms[5] utilize the eigenvectors of the adjacency matrix to identify densely connected groups of nodes. Hierarchical clustering algorithms [6] construct a hierarchy of nested communities by iteratively combining or dividing nodes based on their similarity. Probabilistic modeling algorithms [7] adopt a generative model for the network and deduce the most probable communities through Bayesian inference or maximum likelihood estimation.

The selection of an algorithm and methodology for community detection hinges on the unique attributes of the network, encompassing aspects like its scale, density, sparseness, and diversity, along with the objectives of the analysis. These objectives could include recognizing functional modules, forecasting the establishment or removal of connections, or contrasting diverse networks. Additionally, the efficacy of community detection algorithms is contingent on several factors, including data quality, data preprocessing, algorithm initialization, and parameter tuning.

Although there have been significant strides in the realm of community detection, numerous challenges and unresolved inquiries persist. Among these challenges, the resolution limit problem stands out, whereby many algorithms struggle to identify communities that are below a specific size or possess a limited signal-to-noise ratio. Another formidable challenge is the issue of overlapping community detection, focusing on the recognition of nodes that simultaneously pertain to multiple communities. Furthermore, assessing the performance of community detection algorithms is frequently intricate due to the absence of definitive ground-truth data or the subjective nature of evaluation criteria.

To summarize, community detection constitutes a pivotal domain within network science, striving to pinpoint clusters of nodes exhibiting stronger internal connections than with the broader network. This endeavor is fundamental in comprehending the structural and operational attributes of intricate systems, including social networks, biological networks, and technological networks. While a plethora of algorithms and techniques exist for community detection, each harboring its own merits and drawbacks, there persist numerous challenges and unresolved issues in this field. These challenges beckon

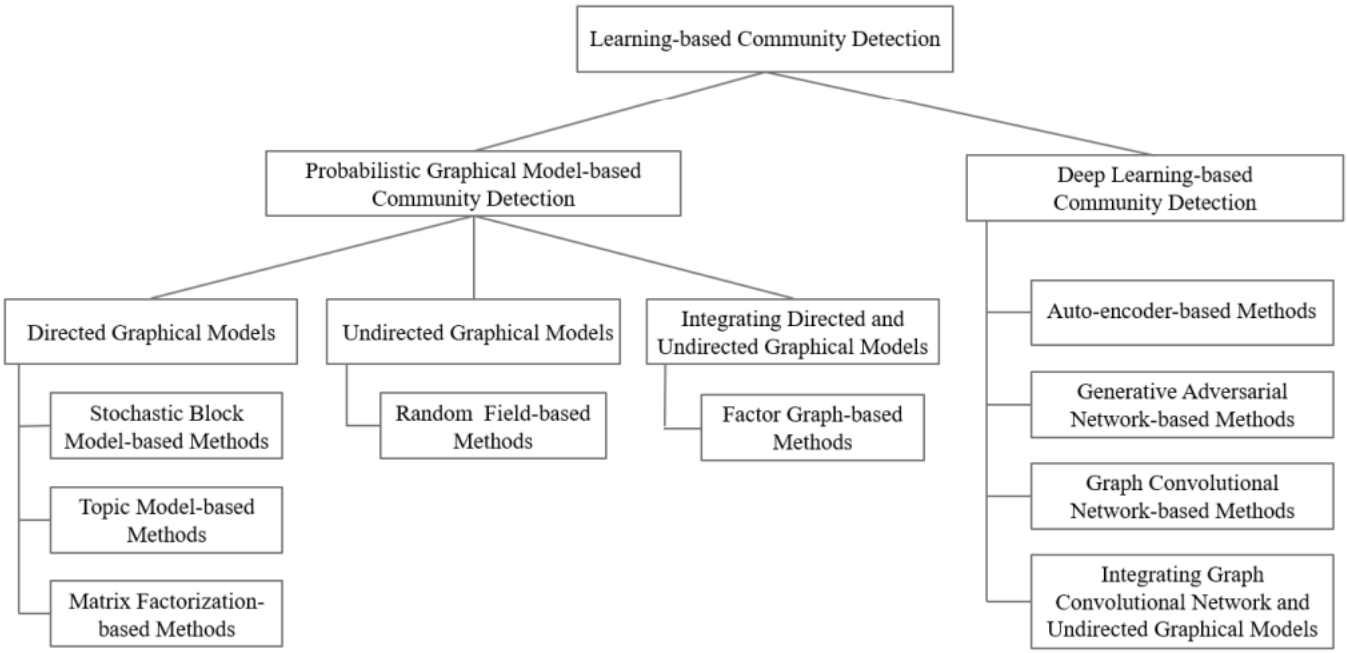


Figure 1.2: Different method of Community Detection [8]

further research and development efforts.

1.2 Differences Between Community Detection and Clustering

Community Detection and Clustering are two related but distinct concepts used in the analysis of networks and data. Let's explore the differences between these two terms:

1. **Network vs. Data:**

   Community Detection: Primarily applied in the context of network science, community detection focuses on identifying groups of nodes within a network that are more densely connected to each other than to nodes outside the group. It is concerned with uncovering the inherent structure of a network, such as social networks or biological networks.

Clustering: Clustering, on the other hand, is a broader data analysis technique used in various fields, not limited to networks. It involves grouping similar data points or objects based on certain criteria, often in high-dimensional spaces. Clustering can be applied to datasets, documents, images, and more.

2. **Purpose:**

   Community Detection: The primary goal of community detection is to reveal the underlying organization or modular structure of a network. It helps in understanding the relationships between nodes and identifying functional groups or communities within the network.

   Clustering: Clustering aims to group data points that exhibit similarities, making it useful for tasks like data segmentation, pattern recognition, and simplifying complex datasets.

3. **Algorithms and Techniques:**

   Community Detection: Community detection algorithms are tailored for network analysis and often consider connectivity and topological properties. Common methods include modularity-based, spectral, and hierarchical clustering algorithms.

   Clustering: Clustering algorithms are diverse and designed for different types of data. Examples include K-means, hierarchical clustering, DBSCAN, and agglomerative clustering, each suited to specific data characteristics and applications.

4. **Application Domain:**

   Community Detection: Mainly used in the study of social networks, biological networks, recommendation systems, and other network-related fields.

   Clustering: Widely employed in machine learning, data mining, natural language processing, image analysis, and other data-driven domains.

Community detection and clustering are methodologies employed in data analysis to aggregate akin objects or entities. Nonetheless, there exist notable distinctions between these two approaches.

Clustering involves partitioning a collection of items into clusters or groups determined by their resemblance. The objective of clustering is to identify clusters of items that maximize similarity within the group and minimize similarity between different groups. Clustering is commonly applied in unsupervised learning scenarios, where there are no predetermined labels for the items under consideration.

Conversely, community detection entails the identification of clusters of nodes within a network that exhibit stronger interconnections among themselves than with the wider network. This process finds frequent application in network analysis, where nodes signify entities, and edges symbolize relationships among these entities. The aim of community detection is to pinpoint sets of nodes that possess functional relationships or common attributes. [9]

Community detection places its emphasis on both the network structure and the resemblance among objects within a community while clustering centers on the similarity among the objects being grouped. In essence, community detection considers the connections between nodes within a network, a di-

mension that clustering does not take into account.

Another disparity between these two approaches lies in their typical applications. Community detection is frequently employed within the realm of extensive, intricate networks that comprise numerous nodes and edges. In contrast, clustering finds utility in a broader array of scenarios, extending to smaller datasets or non-network data.

1.3 Methods in the Field of Community Detection

Community detection stands as a foundational challenge in network analysis and boasts applications across diverse domains such as social network analysis, biological networks, recommendation systems, and beyond. Its primary objective is to unearth densely interconnected sets of nodes, referred to as communities or clusters, within a network. These communities typically signify nodes with shared traits, akin interests, or cohesive interactions. This essay delves into several pivotal techniques employed in the realm of community detection.

1. **Modularity Optimization:** Modularity optimization ranks among the most commonly employed methods for community detection. This technique hinges on the concept of enhancing a quality metric known as modularity, which quantifies the level of community structure within a network. Modularity is computed as the disparity between the proportion of edges residing within communities and the anticipated proportion of edges in a random network. The process of maximizing modularity revolves around dividing the network into communities that exhibit a greater edge density within them compared to what would be

expected. Renowned algorithms such as the Louvain method and the Girvan-Newman algorithm rely on modularity optimization principles.

2. **Hierarchical Clustering:** Hierarchical clustering is a methodology designed to construct a hierarchical framework of communities within a network. It initiates by treating each node as an independent community and subsequently merges communities in successive iterations, contingent upon a similarity measure between them. This amalgamation process persists until a predefined termination condition is satisfied. Hierarchical clustering yields a dendrogram, offering a representation of the community hierarchy across multiple levels of detail. This technique facilitates the exploration of communities at different scales, spanning from smaller subgroups to more extensive overarching communities.

3. **Spectral Clustering:** Spectral clustering is an approach that exploits the eigenvalues and eigenvectors of a graph's Laplacian matrix to discern communities. This method entails converting the network into a reduced-dimensional representation through spectral decomposition and subsequently employing a clustering algorithm in this transformed space. Spectral clustering offers the advantage of accommodating networks with intricate connectivity patterns and the ability to identify communities not solely linked by robust edges. It particularly excels when communities exhibit distinctive spectral characteristics.

4. **Modularity-based Greedy Algorithms:** Modularity-based greedy algorithms, exemplified by the Louvain method and the Newman-Girvan algorithm, offer swift and scalable solutions for community detection. These algorithms follow an iterative process in which they optimize the

modularity function by relocating nodes between communities to maximize the increase in modularity. The Louvain method, in particular, has gained prominence owing to its efficiency and capacity to manage extensive networks. Its operation encompasses two phases: a local search phase for node movement within communities and a global optimization phase for community merging, guided by modularity gain.

5. **Overlapping Community Detection:** In certain networks, nodes can concurrently pertain to multiple communities, leading to the existence of overlapping communities. Overlapping community detection approaches endeavor to recognize these communities and encapsulate the inherent network structure overlap. These methods frequently employ probabilistic models in which nodes possess membership probabilities for each community. Algorithms like COPRA (Community Overlap PRopagation Algorithm) and OSLOM (Order Statistics Local Optimization Method) are frequently employed for the detection of overlapping communities.

6. **Density-based Approaches:** Density-based strategies center on the detection of high-density regions within a network, which are considered as communities. The objective of these methods is to uncover dense areas characterized by strong internal connections while exhibiting sparser connections with the broader network. Techniques such as DBSCAN (Density-Based Spatial Clustering of Applications with Noise) and OPTICS (Ordering Points To Identify the Clustering Structure) exemplify density-based clustering algorithms that can be tailored for community detection purposes.

7. **Evolutionary Algorithms:** Evolutionary algorithms, drawing inspiration from the principles of natural evolution, can be harnessed for community detection. These algorithms leverage methodologies like genetic algorithms, particle swarm optimization, or simulated annealing to enhance a fitness function that gauges community quality. Through iterative evolution of a population comprising potential solutions, evolutionary algorithms traverse the search space and ultimately approach near-optimal community structures. These approaches prove especially valuable in the context of extensive and intricate networks, where alternative techniques may encounter challenges in attaining optimal solutions.

To sum up, community detection plays a pivotal role in network analysis, finding applications across various domains. The techniques discussed here provide a comprehensive overview of the diverse methods employed to unearth communities within networks. Whether it's modularity optimization, hierarchical clustering, spectral clustering, density-based methods, or evolutionary algorithms, each approach carries its own set of advantages and limitations. The selection of the appropriate technique hinges on the network's characteristics, the targeted community properties, and the specific objectives of the analysis. By applying these techniques, researchers can glean valuable insights into the intricate structure and organization of complex networks, unveil concealed relationships, and fathom the dynamics of communities in diverse real-world systems.

1.4 Evaluation and Validation of Detected Communities

Community detection is a demanding task in network analysis, aiming to unveil meaningful and cohesive sets of nodes within a network. However, to ensure the efficacy of community detection algorithms, it is imperative to evaluate and validate the quality and reliability of the detected communities. Evaluation and validation techniques play a pivotal role in appraising the performance of these algorithms, comparing different approaches, and validating the identified communities against established ground truth or external criteria. This process entails the assessment of community quality, reliability, and accuracy. Proficient evaluation and validation empower researchers to instill confidence in the results and gain insights into the performance of community detection algorithms. This section delves into various evaluation measures and techniques commonly employed in the field.

1. **Internal Evaluation Measures:** Internal evaluation measures appraise the quality of detected communities solely based on the network's inherent structure, requiring no external information or ground truth. These measures concentrate on assessing the compactness and separability of communities by evaluating their internal structure and connectivity. Modularity is a frequently used metric, where higher modularity values suggest superior community structure. Additional measures like conductance, cut ratio, and normalized cut gauge the connectivity and boundary quality of communities. Internal evaluation measures provide insights into the quality of identified communities solely through the network's topology. Some prevalent internal evaluation measures encompass:

- **Modularity:** Modularity is a fundamental and commonly employed concept in community detection. It plays a pivotal role as an evaluation metric, quantifying the quality of a network's division into communities. Modularity gauges the disparity between the observed count of edges within communities and the anticipated count of edges in a random network. A higher modularity value signifies a more robust and effective community structure.

  The notion of modularity was initially introduced by Newman and Girvan in 2004 as an evaluation metric to assess the excellence of network partitions. Over time, it has evolved into one of the most favored and broadly embraced metrics in the field of community detection research.

  Modularity operates under the premise that in a sound community structure, there should be a surplus of edges within communities compared to what would be anticipated by random chance. It accomplishes this by contrasting the actual count of edges within communities with the anticipated count of edges under the assumption of random edge distribution. The expected number of edges is computed using a null model, which might be the configuration model or the Erdős-Rényi random graph model. This null model provides a basis for estimating the random edge distribution within the network.

  The modularity score, typically represented as Q, is formulated as the disparity between the actual count of edges within communities and

the anticipated count of edges based on the null model. A positive Q value signifies a greater edge density within communities compared to what would be expected by chance, hinting at the existence of meaningful communities.

Community detection algorithms strive to maximize the modularity score by optimizing the assignment of nodes to communities. Multiple optimization techniques, including the Louvain algorithm, spectral clustering, and genetic algorithms, have been devised to identify the partition that maximizes modularity. These algorithms seek to attain the most cohesive and distinct community structure within a network.

- **Conductance:** Conductance serves as a frequently employed metric in community detection, focusing on the evaluation of identified communities or partitions within a network. It quantifies the degree to which a community exhibits internal cohesion and separation from the broader network. A reduced conductance value signifies a more robust and tightly-knit community structure.

  The concept of conductance originates from graph theory and finds particular relevance in the realm of community detection. In this context, the objective is to uncover sets of nodes that display strong internal connections while maintaining sparse connections with nodes outside the community. Conductance is a pertinent metric for assessing the degree of internal cohesion and isolation within such communities.

  Conductance is defined as the ratio of the number of edges that connect nodes within the community to the total number of edges

incident on nodes in the community. Mathematically, it can be expressed as:

Conductance = (Number of edges between community and non-community nodes) / (2 * Min(Vol(C), Vol(V C)))

where: - Number of edges between community and non-community nodes is the count of edges that connect nodes within the community to nodes outside the community. - Vol(C) is the sum of degrees of nodes within the community. - Vol(V C) is the sum of degrees of nodes outside the community.

The conductance value ranges between 0 and 1. A conductance value close to 0 indicates a well-connected community with few edges crossing its boundary, while a value close to 1 suggests a weak community structure, with many edges connecting nodes inside and outside the community.

In the field of community detection, the primary aim is to discover a network partition that minimizes the conductance value for each community. A lower conductance value implies a more tightly knit and cohesive community structure, indicating that nodes within the community are strongly interconnected while having limited connections outside the community.

- **Normalized Cut:** Normalized Cut stands as a prevalent metric in community detection for assessing the quality of identified communities within a network. It gauges the equilibrium between connectivity within communities and the demarcation between distinct communities. The objective is to locate a partition that minimizes the Normalized Cut value, signifying the presence of well-defined

and clearly separated communities.

The concept of Normalized Cut finds its roots in spectral graph theory, a field that harnesses the eigenvalues and eigenvectors of the graph Laplacian matrix to scrutinize the connectivity patterns in a network. The graph Laplacian matrix serves as a representation of the network's structural characteristics and frequently forms the foundation for various community detection algorithms. It enables the analysis of network connectivity and the exploration of meaningful partitions.

The computation of the Normalized Cut value involves aggregating the cut values across all communities and subsequently normalizing it by the sum of the sizes of these communities. The cut value signifies the count of edges that traverse the boundaries separating communities. In this context, the size of a community is defined as the sum of the degrees of the nodes residing within that community. This formula provides a quantitative measure of the balance between internal cohesion and external separation in the community structure.

Mathematically, the Normalized Cut can be expressed as:

Normalized Cut = (Sum of cut values for all communities) / (Sum of sizes of all communities)

A lower Normalized Cut value indicates a better community structure, where the communities are tightly connected internally and well-separated from each other. The Normalized Cut metric encourages the formation of communities with strong internal connections and limited inter-community connections.

2. **External Evaluation Measures:** External evaluation measures come into play when there is an opportunity to compare the identified communities with a known ground truth or external criteria. These measures prove particularly valuable in cases where a ground truth partition is accessible, as is often the case with synthetic networks or benchmark datasets. Metrics like Normalized Mutual Information (NMI), Adjusted Rand Index (ARI), and F1 score serve to gauge the alignment between the detected communities and the ground truth partition. NMI quantifies the mutual information between the two partitions, while ARI adjusts for chance agreement. These measures deliver a quantitative evaluation of the similarity between the detected communities and the ground truth, providing valuable insights into the algorithm's performance.

    - **Normalized Mutual Information (NMI):** Normalized Mutual Information (NMI) is a prevalent and widely adopted metric for appraising the effectiveness of community detection algorithms. It quantifies the similarity between the identified communities and the ground truth communities, delivering a quantitative assessment of the accuracy of the community detection outcomes. NMI serves as a valuable tool for evaluating the quality of detected communities in comparison to the established ground truth.

      NMI draws its foundation from the concept of mutual information, a metric designed to assess the statistical dependence between two variables. In the context of community detection, these variables correspond to the detected communities and the ground truth communities. NMI is instrumental in quantifying the extent of shared information between these two sets of communities, providing a

quantitative measure of their similarity or dissimilarity. This allows for a rigorous evaluation of the quality of the detected communities in relation to the ground truth.

The NMI value ranges between 0 and 1, where a higher value indicates a greater similarity between the detected communities and the ground truth communities. A value of 1 indicates a perfect match between the two sets of communities, while a value of 0 indicates no similarity.

To calculate NMI, the contingency table is constructed, which captures the number of common nodes between each pair of detected and ground truth communities. The contingency table is then used to compute the mutual information and the entropy values, which are used to calculate the NMI score.

The formula for calculating NMI is as follows:

$NMI = [2 * (MI(D,G)/(H(D) + H(G))]$

where MI(D, G) is the mutual information between the detected communities (D) and the ground truth communities (G), and H(D) and H(G) are the entropies of the detected and ground truth communities, respectively.

- **Adjusted Rand Index (ARI):** The Adjusted Rand Index (ARI) is a statistical measure that finds common use in the evaluation of community detection algorithms. It examines the similarity between the detected communities and the ground truth communities, considering chance agreement and offering a normalized score. ARI is valuable in providing a comprehensive and adjusted assessment of the quality of community detection results, taking into account

random agreement that may occur by chance.

ARI stems from the Rand Index concept, which quantifies the similarity between two data clusterings. Nevertheless, the Rand Index alone doesn't factor in chance agreement stemming from random node-to-cluster assignments. ARI rectifies this limitation by adjusting for chance agreement, thus offering a more robust and comprehensive evaluation metric for assessing the quality of community detection results. It provides a normalized score that accounts for random agreements, making it particularly valuable in community detection evaluation.

The ARI value ranges between -1 and 1, where a value close to 1 indicates a high agreement between the detected and ground truth communities, a value close to 0 indicates agreement no better than random chance, and a value close to -1 indicates a complete disagreement between the detected and ground truth communities.

To calculate the ARI, a contingency table is constructed that captures the number of pairs of nodes with the same community assignment in both the detected and ground truth communities, as well as the pairs of nodes with different community assignments in both sets. The contingency table is then used to calculate the ARI score using the following formula:

$$ARI = (Index - Expected_Index)/(Index_{max} - Index_{Expected})$$

where Index is the Rand Index, $Index_{Expected}$ is the expected value of the Rand Index under the assumption of random cluster assignments, and $Index_{max}$ is the maximum possible value of the Rand Index.

The ARI score ranges from -1 to 1, where a score of 0 indicates a random clustering, a score of 1 indicates a perfect match between the detected and ground truth communities, and negative scores indicate a disagreement beyond what would be expected by chance.

- **F1 Score:** The F1 score is a commonly employed metric for assessing the performance of binary classification tasks, including community detection. It amalgamates precision and recall, furnishing a balanced gauge of the algorithm's proficiency in accurately identifying positive instances while mitigating the occurrence of false positives and false negatives. The F1 score proves to be a valuable metric for evaluating the effectiveness of community detection algorithms.

  Within the realm of community detection, the F1 score finds frequent application in appraising the quality of the detected communities by juxtaposing them with the ground truth communities. It evaluates how effectively the algorithm discerns the nodes that pertain to specific communities, offering valuable insights into the accuracy of community detection results. The F1 score is instrumental in quantifying the algorithm's performance in relation to the established ground truth, specifically regarding its ability to identify nodes belonging to particular communities.

  The F1 score is calculated based on precision and recall. Precision is the ratio of true positives (the number of nodes correctly identified as belonging to a community) to the sum of true positives and false positives (the number of nodes incorrectly identified as belonging to a community). Recall, also known as sensitivity or true positive rate, is the ratio of true positives to the sum of true positives and

false negatives (the number of nodes belonging to a community that was not identified as such).

The F1 score is the harmonic mean of precision and recall, providing a single value that balances the trade-off between them. It is calculated using the following formula:

$F1Score = 2 * (Precision * Recall)/(Precision + Recall)$

The F1 score ranges from 0 to 1, where a score of 1 indicates perfect precision and recall, meaning all the nodes in the community were correctly identified, and no nodes outside the community were incorrectly labeled. A score of 0 indicates poor performance, with either low precision, low recall, or both.

3. **Stability Analysis:** Stability analysis focuses on assessing the stability and resilience of the detected communities by scrutinizing their consistency across multiple executions or alterations of the network. This typically entails running the community detection algorithm multiple times, either on the same network with varying random initializations or on perturbed versions of the network. Metrics like the Variation of Information (VI) or the Jaccard coefficient are often employed to evaluate the stability of the detected communities. Greater stability reflects more dependable and uniform community detection outcomes, indicating the robustness of the algorithm.

4. **Robustness Analysis:** Robustness analysis serves as a means to gauge the stability and dependability of community detection algorithms by scrutinizing their performance in the face of diverse network perturbations. These perturbations might include the addition or removal of

edges, nodes, or communities within the network. Robustness analysis plays a crucial role in identifying algorithms capable of effectively managing noise and fluctuations in network structures, offering insights into the resilience of these algorithms.

5. **Overlapping Community Evaluation:** Evaluating the quality of overlapping communities presents a more intricate challenge compared to non-overlapping ones. Conventional evaluation metrics are ill-suited for overlapping communities as they typically assume non-overlapping partitions. To address this, specialized measures have been devised to assess the quality of overlapping communities. One such metric is Coverage, which quantifies the proportion of nodes covered by the detected communities in relation to the ground truth. Another metric is the F1 measure, which takes both precision and recall into account to capture true positives and false positives within the detected communities. Evaluating overlapping communities necessitates a focus on the extent of overlap and the precision of community assignments, accounting for the intricacies of these structures.

6. **Visual Evaluation:** Visual evaluation techniques offer an intuitive means of appraising the quality of detected communities. Network visualization tools like Gephi, Cytoscape, or NetworkX can be leveraged to visually explore the community structure and pinpoint any irregularities or incongruities. Visualization empowers researchers to attain a more profound comprehension of the network's organization and to uncover possible errors or misclassifications. Visual evaluation proves especially beneficial in recognizing outliers, areas of overlap, or isolated communit-

ies that may necessitate further scrutiny and analysis.

7. **Real-World Validation:** Real-world validation encompasses the evaluation of detected communities within the framework of a particular application or domain. This form of validation typically engages domain experts who possess the knowledge and expertise relevant to the specific domain. These experts evaluate the communities by drawing upon their domain-specific insights and expertise. They assess the coherence, relevance, and interpretability of the communities, ultimately determining whether the detected communities align with their expectations and domain-specific knowledge. Real-world validation plays a pivotal role in guaranteeing that the detected communities carry practical significance and can deliver valuable insights for the intended application or use case.

In summary, the evaluation and validation of detected communities play a pivotal role in shedding light on the performance and dependability of community detection algorithms. These processes aid researchers in the selection of suitable algorithms for networks with distinct characteristics and in comparing different approaches. Evaluation and validation represent critical phases in community detection that serve to gauge the quality, reliability, and importance of the detected communities.

A diverse range of techniques, including internal and external evaluation measures, stability analysis, overlapping community evaluation, visual evaluation, and real-world validation, constitutes a comprehensive toolkit for assessing and validating the detected communities. Researchers should thoughtfully combine these techniques, taking into account the specific attributes of the network, the nature of the communities, and the analysis

objectives. Through the meticulous application of these rigorous evaluation and validation approaches, researchers can ensure the resilience and effectiveness of their community detection algorithms, leading to the extraction of valuable insights from the identified communities.

By embracing both internal and external evaluation measures, researchers can appraise the quality and precision of the detected communities and derive meaningful inferences concerning the underlying network structure.

1.5 Applications of Community Detection

Community detection, sometimes referred to as graph clustering or graph partitioning, stands as a fundamental challenge within network analysis. Its implications extend across an array of domains, encompassing social network analysis, biological networks, recommendation systems, and information retrieval. Within the scope of this essay, we will delve into the myriad applications of community detection and underscore the profound importance of this field in unraveling the intricacies of complex systems.

1. **Social Network Analysis:** In the contemporary landscape, social networks have evolved into an integral aspect of our daily lives, with platforms such as Facebook, Twitter, and LinkedIn serving as global connectors of individuals. In this context, community detection emerges as a pivotal tool for comprehending the structure and evolution of social networks. By pinpointing communities within these networks, we gain insights into social groups, online communities, and influential figures. This wealth of information holds considerable value, offering opportunities for targeted marketing, the identification of key opinion leaders, the discovery of online communities with shared interests, and the study of information dissemination and influence propagation within these networks.

2. **Biological Networks:** The realm of community detection boasts a wide array of applications in the realm of biological networks, encompassing domains like protein-protein interaction networks, gene co-expression networks, and metabolic networks. Through the identification of functional modules or protein complexes within these networks, researchers

are empowered to glean profound insights into the intricacies of biological processes. Community detection within biological networks plays a pivotal role in elucidating the mechanisms underlying diseases, the identification of potential drug targets, and the prediction of protein functions. This application holds significant promise for advancing our understanding of the biological world.

3. **Recommendation Systems:** Community detection techniques are extensively harnessed in recommendation systems to elevate the quality of personalized recommendations. Through the clustering of users into communities rooted in their preferences and behaviors, recommendation algorithms can discern users with akin preferences and furnish recommendations grounded in the choices of other members within the same community. This methodology elevates recommendation accuracy and facilitates the exploration of specialized or niche communities, amplifying the efficacy of recommendation systems.

4. **Information Retrieval:** Community detection assumes a pivotal role in the realm of organizing and retrieving information from extensive and intricate datasets. The segmentation of documents, web pages, or citations into communities empowers researchers to establish hierarchical structures, construct topic models, and form citation networks, all of which contribute to streamlined information retrieval. Techniques centered around community-based information retrieval serve to enhance search relevance, simplify the exploration of interrelated topics, and bolster the foundation for knowledge discovery in large and complex datasets.

5. **Network Security:** The application of community detection techniques

in network security serves as a robust means to unearth anomalous or malicious activities within networks. Through the identification of communities exhibiting analogous activity patterns, network administrators can zero in on potential threats, discern coordinated attacks, and enhance the security posture of the network. The utilization of community-based anomaly detection bolsters the network's capacity to identify intrusions and fortify its defenses against cyber threats, ultimately fortifying network security.

6. **Urban Planning and Transportation:** Community detection finds practical applications in the domains of urban planning and transportation systems. Through the examination of mobility patterns, traffic dynamics, and transportation networks, communities encompassing regions, neighborhoods, or transportation hubs can be discerned. This data proves invaluable for urban planners, enabling them to fine-tune transportation infrastructure, pinpoint areas of traffic congestion, and enhance the provision of public transportation services. The insights derived from community detection contribute to more efficient and effective urban planning and transportation management.

7. **Online Social Media Analysis:** Amidst the proliferation of social media platforms, community detection emerges as a pivotal tool for comprehending online interactions and user behavior. The identification of communities within social media networks empowers researchers to dissect sentiment trends, pinpoint influential users or opinion leaders, scrutinize the patterns of information diffusion, and discover online communities revolving around specific topics or events. This wealth

of information proves instrumental for endeavors such as social media marketing, crisis management, and the comprehension of online discourse, offering valuable insights into the dynamics of the digital realm.

8. **Collaborative Filtering:** Community detection techniques are harnessed in collaborative filtering algorithms that underpin recommender systems. Through the identification of communities comprising users with akin preferences, collaborative filtering algorithms can furnish recommendations rooted in the choices of other members within the same community. This approach substantially enhances recommendation accuracy and fosters higher levels of user engagement, amplifying the effectiveness of recommender systems.

In summary, community detection serves as a versatile tool with applications spanning a multitude of domains. It offers invaluable insights into the intricacies of complex systems, shedding light on their structure, dynamics, and organization. The identification of communities within networks empowers us to unearth concealed patterns, gain a deeper understanding of social interactions, optimize systems, and make well-informed decisions. The ongoing progress in community detection algorithms and techniques holds the promise of further advancements in these application areas, ushering in new prospects for research and innovation.

Chapter 2

Utilization of Graphs and Networks in Community Detection

Graphs and networks serve as powerful mathematical models that facilitate the comprehension of intricate systems and relationships. Comprising nodes and edges connecting these nodes, give rise to a network structure. These mathematical constructs find extensive applications across various disciplines, spanning computer science, engineering, social sciences, and biology. Their versatility enables the analysis and visualization of complex data in a meaningful manner, unveiling patterns and relationships that might elude direct observation in raw data. In essence, graphs and networks stand as indispensable tools, aiding our understanding of the intricacies of our surroundings.

2.1 Graph

Definition

A graph, in mathematical terms, constitutes a model comprising nodes (or vertices) and edges (links or arcs) establishing connections between these nodes. The ubiquity of graphs as tools for representation lies in their capacity to depict relationships and dependencies among objects or entities. Across fields like computer science, engineering, and social sciences, graphs play an indispensable role. They facilitate the visualization and analysis of intricate data, enabling the identification of patterns, trends, and predictions regarding the behavior of the modeled system. The practical applications of graphs span diverse domains, ranging from optimizing transportation networks and communication systems to modeling biological and social systems.[10]

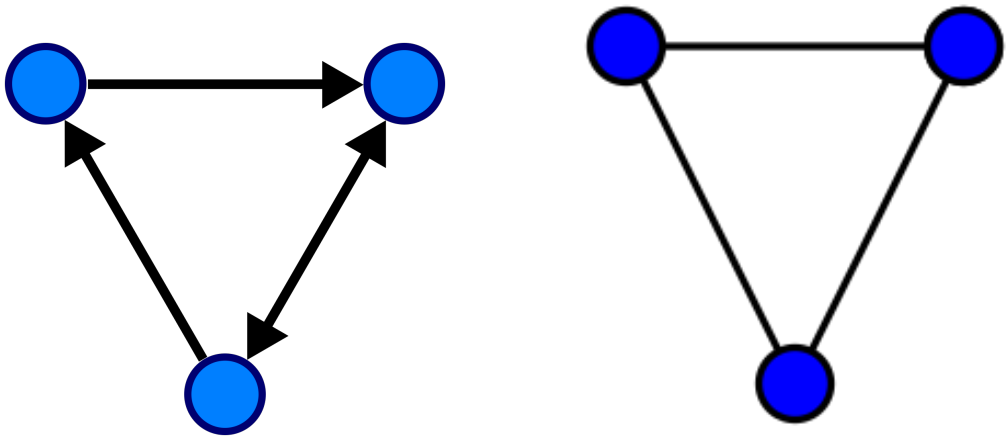
Types of Graphs

There are many types of graphs, each suited to specific applications and scenarios. Some common types of graphs include:

2.1.0.1 Directed Graph (Digraph):

A directed graph, commonly referred to as a digraph, is a specific type of graph characterized by edges that possess an associated direction. In simpler terms, it is a graph wherein the relationship between two nodes is unidirectional. This implies that an edge connecting node A to node B does not necessarily indicate the existence of an edge connecting node B to node A. Directed graphs

find frequent application in scenarios where the direction of the relationship holds significance, such as in electrical circuits, transportation networks, or causal relationships between variables. Within a directed graph, the in-degree of a node signifies the number of edges pointing toward that node, while the out-degree of a node indicates the number of edges pointing away from that node.



(a) Directed Graph [11]

(b) Undirected Graph [12]

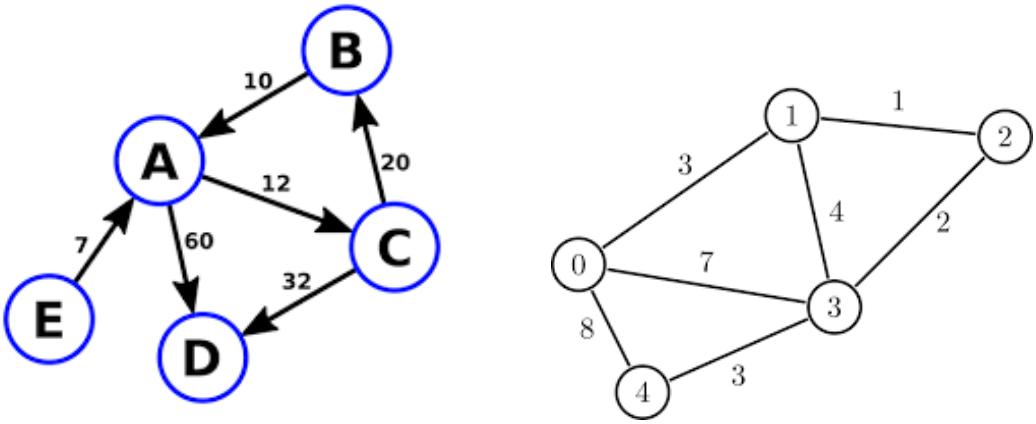
Figure 2.1: Directed and Undirected Graph

2.1.0.2 Undirected Graph:

An undirected graph is a type of graph where edges do not have a direction associated with them. In other words, it is a graph where each edge connects two nodes, and the relationship between the two nodes is bidirectional. This means that if there is an edge connecting node A to node B, there is also an edge connecting node B to node A. Undirected graphs are often used to represent relationships or connections where the direction of the relationship does not matter, such as social networks or comm. networks.

2.1.0.3 Weighted Graph:

A weighted graph is a specific type of graph in which each edge is assigned a weight or cost. Essentially, it is a graph wherein each edge is endowed with a numerical value representing a certain characteristic of the relationship between the connected nodes. These weights can denote distances, travel time, cost, the strength of a connection, or any other pertinent metric. Weighted graphs find frequent application in modeling real-world scenarios where the edges between nodes carry varying degrees of importance or significance.



(a) Weighted - Directed Graph [13]

(b) Weighted - Undirected Graph [14]

Figure 2.2: Weighted Graph

2.1.0.4 Bipartite Graph:

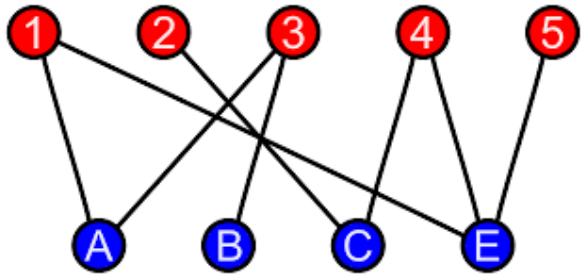


Figure 2.3: Bipartite Graph

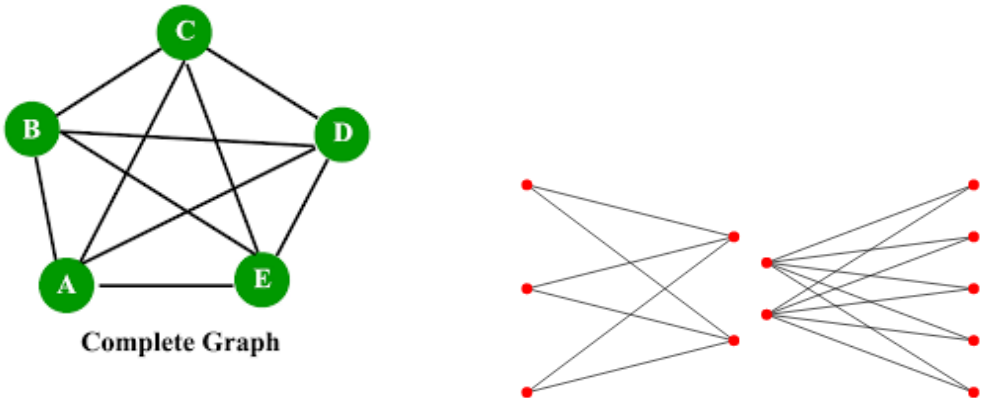
[15]

A bipartite graph is a specific type of graph in which the nodes can be segregated into two distinct sets, with each edge, connecting a node from one set to a node from the other set. Essentially, it is a graph that allows for the partitioning of nodes into two independent groups or subsets, and every edge establishes a connection between a node in one subset to a node in the other subset. The representation of a bipartite graph typically involves denoting it as a pair of sets (X, Y), where X and Y represent the two disjoint sets of nodes.

2.1.0.5 Complete Graph:

A complete graph is a specific type of simple undirected graph characterized by the presence of an edge connecting every pair of distinct nodes. In simpler terms, it is a graph in which each node is directly linked to every other node within the graph. A complete graph with n nodes is denoted by (K_n), where n is the number of nodes in the graph. The number of edges in a complete graph can be determined by the formula $n(n-1)/2$, since each node is connected to $(n-1)$ other nodes, and we divide by 2 to account for the fact that each edge is counted twice. Complete graphs possess numerous properties and

applications in both mathematics and computer science. They frequently serve as a foundational reference point for analyzing various types of graphs. As an illustration, a complete graph is often employed as a benchmark for gauging the connectivity or complexity of other graphs, providing a standard for comparison in graph theory and related fields.



(a) Complete Graph [16]

(b) Complete-Bipartite Graph [17]

Figure 2.4: Complete Graph

2.2 Networks

Definition

In the realm of data, a network denotes a specific type of data structure designed to depict relationships or connections between entities. Graphs are commonly employed to represent networks, where nodes symbolize the entities, and edges signify the connections or relationships between them. Networks find applications in modeling various systems, including but not limited to social networks, transportation networks, biological networks, and

communication networks. The complexity of networks varies, contingent upon the quantity of entities and connections within the system. The analysis of networks facilitates a deeper understanding of the structure and function of intricate systems, enabling the identification of pivotal nodes or pathways that hold significant roles in the system. [18]

Graph Vs. Network :

While the terms "graph" and "network" are frequently used interchangeably, their meanings can vary slightly in different contexts. In the realm of mathematics, a graph constitutes a collection of vertices (or nodes) linked by edges. In computer science, a graph takes on the role of a data structure encompassing nodes and their connections. On the other hand, "network" generally denotes a broader concept, encapsulating the structure and interactions among entities within a system. In the domain of data analysis, a network represents a specific type of data structure illustrating relationships or connections between entities, often visualized using a graph. In summary, the distinction between "graph" and "network" relies on the specific context and application of these terms.

Types of Network

There are various types of networks that can be classified based on their characteristics, structure, and applications [19]. Here are some common types of networks:

2.2.0.1 Social Network:

A social network is a specific type of network that serves as a representation of the relationships or connections between individuals or groups of individuals. The structure of social networks can be effectively captured using graphs, where the nodes symbolize the individuals or groups, and the edges signify the connections or relationships between them [20]. Within a social network, these connections may encompass diverse types of relationships, ranging from friendships and family ties to professional contacts and other forms of social interaction. Social networks find applications in various fields, including marketing, politics, and social science research, where gaining insights into the relationships and interactions between individuals holds significant value. The advent of digital platforms like Facebook, Twitter, and LinkedIn exemplifies the integration of social networks into the modern landscape, emphasizing their relevance and impact. [21]

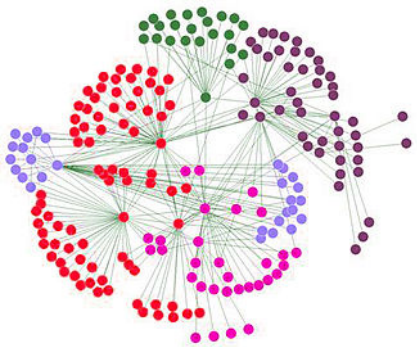


Figure 2.5: Social Network
[22]

2.2.0.2 Communication Network:

A communication network is a specialized type of network designed to enable the exchange of information or messages among entities, which could include individuals, organizations, or devices. Various manifestations of communication networks exist, encompassing platforms like telephone networks, email systems, messaging applications, or social media platforms. The structure and dynamics of communication networks can be effectively modeled and analyzed using graphs, where nodes symbolize the entities, and edges signify the communication channels or connections between them. Within a communication network, the connections between entities can be either directed or undirected, representing diverse communication links such as phone lines and email addresses.

2.2.0.3 Transportation Network:

A transportation network is a specialized type of network designed to model the infrastructure and connections involved in transporting goods or people between distinct locations. Such networks can manifest in various forms, including roadways, railways, air routes, waterways, or pipelines. Graphs can effectively represent transportation networks, with nodes symbolizing locations and edges denoting transportation links or connections. Within a transportation network, the connections between locations can be either directed or undirected, representing diverse transportation modes like roads, bridges, tunnels, or airports. Analyzing transportation networks aids in understanding the efficiency, connectivity, and accessibility of the transportation infrastructure.

2.2.0.4 Biological Network:

A biological network serves as a valuable representation of the intricate interactions and relationships among biological entities like genes, proteins, metabolites, or cells. It offers a powerful tool for studying and analyzing diverse biological processes, including gene regulation, metabolic pathways, and protein-protein interactions. Graphs provide an effective means to represent biological networks, with nodes depicting biological entities and edges symbolizing interactions or relationships. These networks may exhibit characteristics such as directionality, weight, and various interaction types, including activation, inhibition, or physical binding. Analyzing biological networks contributes to a deeper understanding of complex biological systems and their underlying processes.

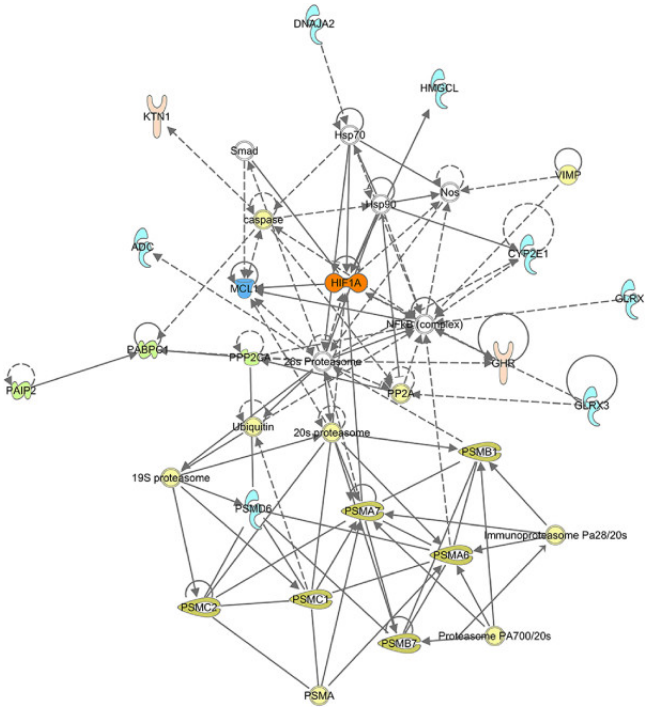


Figure 2.6: Biological Network
[23]

2.2.0.5 Citation Network:

A citation network serves as a structured representation of the connections and relationships among scholarly articles or publications based on citations. In this network, individual articles are depicted as nodes, and citations from one article to another are represented by edges. The analysis of citation networks provides valuable insights into patterns of scientific collaboration, the development of research fields, and the influence of specific articles or researchers within a given domain. Researchers leverage citation networks to identify pivotal articles or contributors, trace the impact of research over time, and unveil emerging trends in scientific inquiry. In the digital era, the importance of citation networks has grown significantly, facilitated by online databases and search engines that streamline access to and analysis of extensive collections of scholarly articles.

2.2.0.6 Co-occurance Network:

A co-occurrence network is a type of network that represents the co-occurrence of words or terms within a set of documents or texts. In this network, individual words or terms are depicted as nodes, and edges symbolize co-occurrences between pairs of words or terms. Co-occurrence networks are subject to analysis for studying patterns of word usage, performing topic modeling, and conducting sentiment analysis. They are instrumental in identifying significant words or terms within a document corpus, pinpointing key topics or themes, and unveiling relationships between words or topics. Co-occurrence networks find applications in diverse fields, including natural language processing, information retrieval, and computational linguistics.

Network types based on characteristics:

2.2.0.7 Small-world Network:

Small-world networks are a type of network that exhibit both high local clustering and short path lengths between nodes. They are characterized by the "small-world" phenomenon, where any two nodes in the network are typically connected by only a few intermediate nodes, even in networks with a large number of nodes.

2.2.0.8 Scale-free Network:

Scale-free networks are a type of network that exhibits a power-law degree distribution, where a few nodes (called "hubs") have a very high degree of connectivity, while most nodes have only a few connections. These networks are characterized by the presence of hubs, which play a critical role in the structure and function of the network.

2.2.0.9 Random Network:

Random networks are a type of network where edges between nodes are randomly distributed. In a random network, the degree of each node is typically distributed normally, with most nodes having a degree close to the average degree. Random networks are often used as a null model for comparing other types of networks, as they provide a baseline for comparison to determine whether a network is significantly different from random.

2.2.0.10 Modular Network:

Modular networks, also known as community networks, are a type of network where nodes are grouped into densely connected subgroups or communities, with weaker connections between communities. In other words, a modular network consists of a collection of sub-networks, or modules, that are themselves highly connected, but with relatively fewer connections between modules.

2.2.0.11 Hierarchical Network:

Hierarchical networks are a type of network where nodes are organized into hierarchical layers or levels, with connections typically occurring between nodes within the same level or between adjacent levels. Hierarchical networks are characterized by a nested structure, where smaller sub-networks are connected to form larger sub-networks, and so on. This type of network is commonly found in biological systems, such as neural networks, gene regulatory networks, and ecological networks.

2.2.0.12 Spatial Network:

Spatial networks are a type of network where the nodes are located in physical space, and the edges represent some kind of spatial relationship or proximity between nodes.

2.2.0.13 Temporal Network:

Temporal networks are a type of network where the edges or connections between nodes change over time. In other words, a temporal network captures the dynamics of a system by representing the patterns of interactions between

nodes at different points in time. Temporal networks are important because they allow us to study how the structure and function of a network change over time, and how these changes affect the behavior of the system.

2.2.0.14 Complex Network:

Complex networks are networks that exhibit non-trivial structural and dynamic properties, often arising from the interactions between nodes and their environment. These networks are characterized by features such as degree distributions that are highly skewed, the presence of community structure, and the emergence of patterns of connectivity at multiple scales.

2.2.0.15 Attribute Network:

Attribute networks are a type of network where nodes are associated with attributes or features that describe their properties or characteristics. In attribute networks, the nodes are connected based on the similarity or dissimilarity of their attributes. These networks can be used to represent a wide range of systems, including social networks, biological networks, and technological networks.

2.2.0.16 Multiview Network:

Multiview networks are a type of network where there are multiple types of edges or multiple ways of representing the relationships between nodes. In other words, a multiview network captures the structure and dynamics of a system using more than one type of connection or representation. These networks can be found in many real-world systems, such as social networks, transportation networks, and ecological networks. Applications of multiview

network analysis include community detection, link prediction, and the identification of key nodes or views that drive the behavior of the system.

2.2.0.17 Signed Network:

Signed networks are a type of network where the edges have both positive and negative signs, representing the presence of positive and negative relationships between nodes. In other words, signed networks capture the positive and negative interactions between nodes, such as trust and distrust, friendship and enmity, or cooperation and conflict.

2.2.0.18 Weighted Network:

Weighted networks are a type of network where the edges are associated with weights or values that quantify the strength or intensity of the relationships between nodes. Weighted networks capture not only the presence or absence of connections but also the degree of similarity or dissimilarity between nodes. Examples of weights that can be used in weighted networks include the frequency of interactions, the strength of ties, or the similarity of attributes.

These are just a few examples of the many types of networks that can be classified based on their characteristics. The classification of networks based on their characteristics can be useful in understanding their properties, behaviors, and applications in different fields.

2.3 Usage of Graphs in Community Detection

Community detection, a fundamental task in network analysis, aims to identify groups of nodes with dense connections within a graph. While network graphs are the most commonly studied type of graph in community detection, there are various other types of graphs that can be used to capture different types of relationships and dependencies among entities. In this section, we explore the usage of different types of graphs in community detection beyond traditional network graphs.

1. **Bipartite Graphs:** Community detection in bipartite graphs involves identifying clusters of nodes within two disjoint sets, revealing meaningful patterns in relationships. This can be achieved through one-mode or two-mode projections, where nodes are connected based on shared neighbors. Algorithms like Bipartite Louvain, Bipartite Spectral Clustering, and Bipartite Stochastic Block Model are adapted for this purpose. Applications include revealing user-item interactions in recommender systems, co-authorship patterns, or customer-product relationships.

2. **Hypergraphs:** Hypergraphs, as extensions of conventional graphs, enable edges to connect more than two nodes. In community detection, hypergraphs prove useful in representing intricate relationships involving multiple entities simultaneously. For instance, in social networks, hyperedges can depict clusters of individuals engaged in joint activities or collaborations. The application of community detection in hypergraphs facilitates the identification of cohesive node groups linked by hyperedges, uncovering communities characterized by shared interests, collaborations, or interactions.

3. **Weighted Graphs:** Weighted graphs involve assigning weights to edges, signifying the strength or intensity of connections between nodes. Within community detection, weighted graphs offer supplementary insights into the importance of relationships. The assigned weights can convey diverse factors like interaction frequency, similarity metrics, or levels of trust. Incorporating edge weights allows community detection algorithms to unveil groups with robust interactions or shared attributes, accommodating the nuanced strengths inherent in relationships.

4. **Directed Graphs:** Directed graphs represent asymmetrical relationships among nodes, with edges possessing a designated direction indicating information or influence flow. When applied to community detection, directed graphs factor in edge directionality to pinpoint node groups with potent influence or dependencies. For example, in citation networks, directed edges showcase citation flows between papers. The exploration of community detection in directed graphs can unveil clusters of influential papers or research topics characterized by robust citation relationships.

5. **Multilayer Graphs:** Multilayer graphs amalgamate diverse relationships or layers into a unified graph, with each layer denoting a distinct interaction or dependency among nodes. When applied to community detection, multilayer graphs account for interactions across different layers to pinpoint communities manifesting consistent patterns across multiple relationships. This methodology facilitates the examination of intricate systems characterized by entities engaging through diverse channels or modalities.

6. **Temporal Graphs:** Temporal graphs capture the dynamic nature of re-

lationships by incorporating time information into the edges. Each edge represents a specific time period during which a connection exists between nodes. Community detection in temporal graphs allows the analysis of evolving communities over time, uncovering patterns of temporal cohesiveness or transitions between different groups. This approach is useful in studying evolving social networks, dynamic collaborations, or topic evolution in text corpora.

7. **Attribute Graphs:** Attribute graphs introduce extra attributes or node features alongside the connectivity information. Nodes are linked to sets of attributes or features, encompassing traits like age, gender, interests, or geographic location. In community detection for attribute graphs, both connectivity patterns and node attributes are utilized to pinpoint communities sharing similar attribute profiles. This methodology facilitates the revelation of communities characterized by shared characteristics or preferences.

In summary, the scope of community detection extends beyond conventional network graphs, encompassing various graph types that can capture a range of relationships and dependencies among entities. The exploration of diverse graph structures offers a deeper understanding of complex systems, facilitating the identification of meaningful communities based on specific interaction types, relationships, or attributes. The selection of a particular graph type depends on the domain, data characteristics, and research goals, enabling a nuanced exploration of community structure and dynamics.

2.4 Usage of Networks in Community Detection

Community detection is a widely studied topic in network analysis, where the goal is to identify groups of nodes with dense connections within a network. While graphs are a fundamental representation of networks, there are various types of networks that can be used to model different real-world scenarios and data sources. In this section, we explore the usage of different types of networks in community detection, highlighting their unique characteristics and applications.

1. **Social Networks:** Social networks delineate relationships among individuals or entities, where nodes depict individuals and edges signify social connections like friendships, collaborations, or interactions. Community detection within social networks proves crucial for comprehending social structures, pinpointing cohesive groups, and examining information diffusion, opinion dynamics, or social influence. Its applications span diverse domains, including online social networks, organizational networks, and friendship networks.

2. **Citation Networks:** Citation networks portray the connections among academic papers, with nodes representing papers and edges symbolizing citations from one paper to another. The implementation of community detection in citation networks is instrumental in recognizing sets of papers sharing similar research topics, influential papers, or clusters of interconnected scientific communities. This approach proves valuable for bibliometric analysis, comprehending knowledge diffusion, and exploring research trends and collaborations within the academic landscape.

3. **Biological Networks:** Biological networks illustrate the connections among biological entities like genes, proteins, or metabolic pathways, encompassing gene regulatory networks, protein-protein interaction networks, and metabolic networks. The application of community detection in biological networks yields valuable insights into functional modules, regulatory mechanisms, and protein complexes. This methodology contributes to comprehending disease mechanisms, facilitating drug discovery, and exploring intricate biological processes.

4. **Internet Networks:** Internet networks depict the relationships among routers, autonomous systems, or web pages. Employing community detection in internet networks facilitates the examination of the internet's structure and dynamics, pinpointing significant nodes or hubs, and investigating information flow, network resilience, or internet routing. This practice is pivotal for efficient network management, the enhancement of internet infrastructure, and the optimization of network performance and security.

5. **Recommender Networks:** Recommender networks encapsulate user-item interactions and preferences in recommendation systems, where nodes signify users and items, and edges represent user-item interactions or ratings. The application of community detection in recommender networks aids in the identification of user groups with akin preferences, clustering of items, and the generation of personalized recommendations. This approach is beneficial in various domains, including e-commerce, movie and music recommendations, and personalized marketing.

6. **Transportation Networks:** Transportation networks depict the intercon-

nections within transportation infrastructure, including road networks, airline networks, or public transportation systems. Detecting communities in transportation networks assists in pinpointing nodes with similar connectivity patterns, transportation hubs, or transportation communities. This analysis is valuable for urban planning, route optimization, and enhancing overall transportation efficiency.

7. **Communication Networks:** Communication networks represent the exchanges between individuals or entities using communication channels like email, phone calls, or online messaging. Detecting communities in communication networks is valuable for analyzing communication patterns, identifying closely connected groups, and exploring information diffusion or rumor spreading. Its applications extend to areas such as social media analysis, cybersecurity, and optimizing communication network structures.

In summary, community detection is versatile and can be applied to a range of network types, each representing distinct relationships and interactions. The selection of a specific network type depends on factors such as the domain, data source, and research goals, providing a nuanced exploration of community structure and dynamics in diverse real-world contexts.

Chapter 3

Aim and Objective of the Thesis

3.1 Motivation

In an attribute network, specifically a node-attributed network where each node is associated with a set of binary attributes, the thesis places emphasis on community detection. The unique aspect of this focus lies in considering not only structural similarity but also attribute similarity. Despite many real-life complex networks having attributes, the prevailing community detection techniques predominantly concentrate on structural similarities within the network, neglecting the additional layer of attribute information.

Attributed networks are pervasive across various domains, including social media platforms like Twitter, Facebook, and LinkedIn, as well as academic citation networks such as Cora, Citeseer, or WebKB. Recognizing the prevalence of attributed networks, it becomes imperative to devise effective solutions for addressing the challenge of community detection within these networks.

3.2 Aim and Objective

This thesis aims to create an efficient technique for identifying the intrinsic community structure within networks. The process involves the processing and development of datasets containing attributed graph networks. To accomplish this objective, several key points must be addressed:

- Modify the dataset into a proper CSV file i.e. there should be 2 files, one must contain the edge list and another must contain the node-attribute list.
- Predefine scalar variables that quantify the influence of structural and attribute similarity. Include parameters for the range of values in comparison to the maximum similarity value, as well as the impact of 1st and 2nd order similarity in the network.
- For datasets containing ground truth, preprocess them to include community ID and a list of nodes associated with each community after the preprocessing step.

3.3 Contribution

In this project, addressing the challenge of learning from binary attributed network data, we propose an effective learning mechanism. The model developed is designed for community detection, employing multi-objective optimization to optimize four distinct objectives. Among these, three pertain to structural similarity, while the fourth focuses on attribute similarity.

3.4 Thesis Organization

The structure of the thesis is as follows:

- **Chapter 1**: Provides an overview of Community Detection, discussing its fundamentals, applications across various fields, and the diverse techniques employed in the process.
- **Chapter 2**: Explores different types of graphs and networks, elucidating their role and application in Community Detection.
- **Chapter 3**: Outlines the aim, objectives, and motivation behind the undertaken work.
- **Chapter 4**: Delves deeper into Binary Attributed networks, providing comprehensive insights.
- **Chapter 5**: Focuses on Community Detection within the realm of Binary Attributed Networks.
- **Chapter 6**: Offers a detailed exploration of multi-objective optimization, including the utilization of Genetic Algorithms.
- **Chapter 7**: Conducts a literature survey, specifically addressing binary attributed networks.
- **Chapter 8**: Explores the proposed method and details the data collection and preparation process.
- **Chapter 9**: Provides an analysis of the experimental results.
- **Chapter 10**: Concludes the thesis and sheds light on potential avenues for future work in **Chapter 11.**

Chapter 4

Introduction to Binary Attributed Network

4.1 Introduction

A binary node-attributed network is a specific type of network structure in which individual nodes are distinguished by a set of binary attributes. These binary attributes typically take on values of 0 or 1, providing a binary representation for various characteristics or features associated with each node. These attributes may encompass a diverse range of node-specific information, including but not limited to gender, age, occupation, or group membership. This network configuration allows for the nuanced characterization of nodes based on binary attribute values, facilitating a comprehensive understanding of the network's structural and attribute-based dynamics.

The task of community detection in binary node-attributed networks poses a considerable challenge, primarily due to the inherent complexity introduced by the binary attributes associated with each node. Traditional clustering algorithms encounter limitations in their direct application to such networks, as the binary nature of the attributes introduces intricacies that demand care-

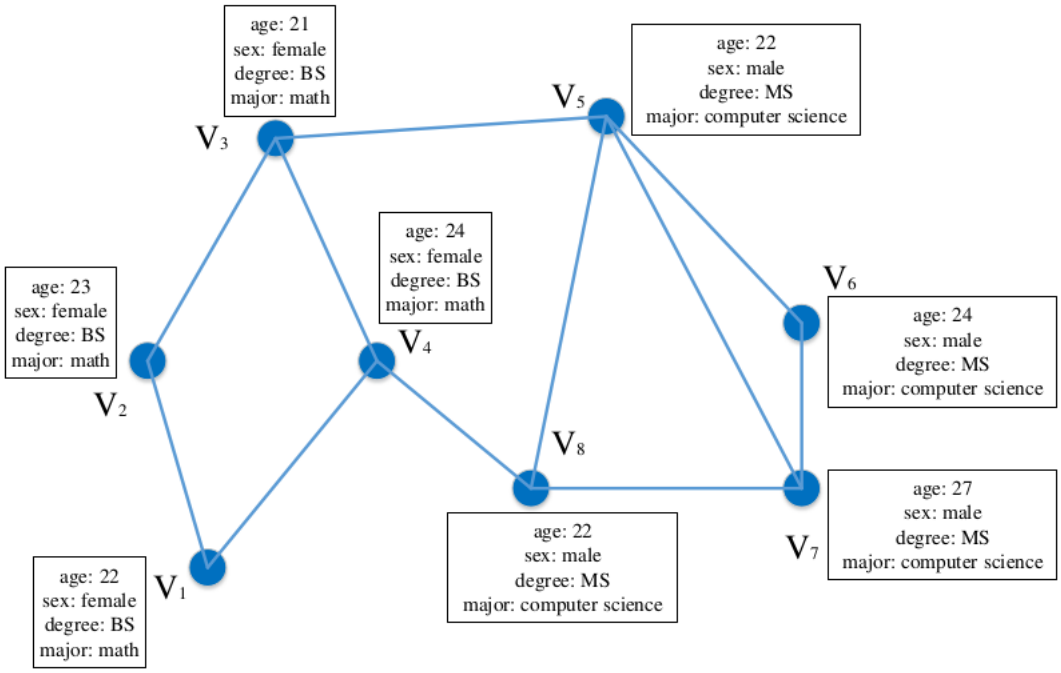


Figure 4.1: Example of an Attribute Network
[24]

ful consideration during the community identification process. The binary attributes contribute an additional layer of information that necessitates specialized approaches for effective and accurate community detection within this network framework.

4.2 Applications of Binary Attributed Network

Binary node-attributed networks find widespread applications across diverse domains, including but not limited to social network analysis, biological network analysis, and recommender systems. These networks serve as valuable models for understanding complex systems, where nodes exhibit distinct properties or behaviors. Exploring the influence of these properties on connection patterns and the development of communities within the network is a key aspect of their utility. Let's delve into a few examples illustrating the applications of binary attribute networks in various domains:

1. **Social Networks:**Within social networks, binary attribute networks serve as effective tools for representing relationships among individuals grounded in shared attributes or characteristics. Consider a co-authorship network as a prime example, where connections between researchers are established based on whether they have jointly authored a publication. This instantiation results in a binary attribute network that adeptly encapsulates collaboration patterns within the academic community.

   In this context, the binary attributes could signify the presence or absence of a collaborative effort between two researchers. Nodes in the network represent individual researchers, and edges denote instances where a shared publication exists. By employing binary attributes, the network not only delineates the structural connections but also encodes specific information about collaborative engagements.

   This approach proves valuable for social network analysis, offering insights into the collaborative dynamics among researchers. Researchers with shared attributes, such as co-authorship, are linked in the network, facilitating the identification of collaborative communities and the exploration of how shared characteristics contribute to the formation of cohesive groups within the scholarly domain.

2. **Biological Networks:**Within the realm of biology, binary attribute networks play a pivotal role in unraveling the intricacies of gene regulatory networks, protein-protein interaction networks, and metabolic networks. These networks leverage binary attributes to signify the existence or non-existence of specific attributes or interactions between biological

entities, offering a nuanced understanding of functional relationships and regulatory mechanisms.

Consider, for instance, a gene regulatory network where genes are represented as nodes, and regulatory interactions are denoted by edges. Binary attributes in this context could represent the activation or inhibition of gene expression, providing a concise yet informative way to capture the regulatory dynamics within the network. Nodes with shared attributes, such as co-regulation, are connected, enabling the identification of functional modules or regulatory clusters.

Similarly, in a protein-protein interaction network, binary attributes may signify the presence or absence of physical binding between proteins. This binary representation facilitates the exploration of protein complexes and interaction patterns. In a metabolic network, binary attributes can capture whether specific reactions occur, shedding light on the functional relationships between different metabolites.

By incorporating binary attributes into these biological networks, researchers gain a powerful tool for studying the underlying mechanisms that govern biological processes. The binary attribute network approach allows for a focused analysis of specific interactions, aiding in the identification of regulatory patterns, functional modules, and key players within these intricate biological systems.

3. **Recommender Systems:** In the realm of recommender systems, binary attribute networks emerge as indispensable tools for understanding and enhancing user-item interactions. Within these networks, the interactions between users and items are encapsulated through binary edges,

where the presence or absence of a connection signifies user engagement with a particular item.

Consider a scenario where users are nodes, items are nodes, and edges represent user-item interactions. In this context, binary attributes play a pivotal role by indicating whether a user has interacted with a specific item. This binary representation encapsulates valuable information about user preferences and item attributes, forming the foundation for personalized recommendations.

The binary attribute network approach allows recommender systems to discern patterns of user engagement, and identify items with similar attributes or those preferred by users with comparable interactions. This, in turn, facilitates the generation of tailored recommendations by leveraging the shared characteristics captured in the binary attribute network. For instance, users connected by shared interactions or preferences in the binary attribute network are likely to receive recommendations based on the items preferred by their counterparts.

In essence, binary attribute networks provide a nuanced framework for recommender systems to navigate and comprehend user-item interactions, offering a robust foundation for the delivery of personalized and context-aware recommendations.

4. **Online Social Media:** Within the realm of online social media platforms, binary attribute networks serve as powerful models for capturing connections and interactions. Consider the context of a follower network on Twitter, where users are nodes, and the presence of a connection signifies a follower relationship between two users. This binary representation

encapsulates the essence of social interactions, allowing for the analysis of information diffusion, influence dynamics, and community detection.

In a follower network, each user-node is characterized by binary attributes, indicating the presence or absence of a follower relationship with other users. This binary attribute network becomes a valuable substrate for studying how information propagates through the network, identifying influential users based on their follower connections, and discerning communities formed around shared interests or attributes.

For instance, users connected by follower relationships in the binary attribute network are likely to share common interests or preferences. Analyzing these connections can unveil cohesive groups or communities within the larger social media platform, providing insights into the structure and dynamics of the online social network.

In essence, binary attribute networks in the context of online social media platforms empower researchers and analysts to delve into the intricacies of social connections, allowing for a nuanced exploration of information diffusion, influence patterns, and community structures based on shared attributes.

5. **Collaboration Networks:** Within collaborative environments, binary attribute networks serve as instrumental tools for capturing the nuances of collaborations between individuals or organizations. Imagine a co-working network where entities are nodes, and connections between them indicate the presence or absence of collaborations on various projects. This binary attribute network becomes a rich representation, enabling the exploration of collaboration patterns and the identification

of potential partnerships.

In a co-working network, each node is associated with binary attributes that signify whether collaboration exists with other nodes. This binary attribute network provides a detailed perspective on the collaborative landscape, allowing researchers and analysts to discern the structure of collaborations and uncover patterns within the network.

For instance, connections in the binary attribute network may denote shared projects or joint initiatives, shedding light on the dynamics of collaborative efforts. Analyzing this network can reveal clusters or communities of entities that frequently collaborate, fostering a deeper understanding of the collaborative landscape within the co-working environment.

In summary, binary attribute networks in collaborative environments offer a valuable lens for studying the presence or absence of collaborations. They provide insights into collaboration patterns, facilitate the identification of potential partnerships, and contribute to a nuanced understanding of the dynamics within co-working networks.

6. **Marketing and Customer Analytics:** In the realm of customer behavior analysis, binary attribute networks serve as powerful tools for gaining insights into the intricacies of customer preferences and interactions. Picture a scenario where each customer is a node in the network, and connections between them are established based on the presence or absence of specific attributes or interactions with products. This binary attribute network becomes a valuable framework for unraveling customer segmentation, pinpointing influential customers, and forecasting

preferences.

In this context, each node in the binary attribute network represents a customer, and the binary attributes associated with them capture key aspects of their behavior or preferences. These attributes could signify interactions with specific products, preferences for certain features, or engagement in particular activities. Analyzing the binary attribute network allows researchers and marketers to discern patterns in customer behavior and make informed decisions based on these patterns.

For example, connections between nodes might indicate shared preferences for certain products or similar patterns of engagement with the offerings. This network analysis can unveil clusters of customers with comparable preferences, facilitating targeted marketing strategies and personalized recommendations.

In summary, binary attribute networks in customer behavior analysis provide a robust framework for understanding customer segmentation, identifying influential individuals, and predicting preferences. These networks empower businesses to tailor their strategies based on the nuanced insights derived from the binary attributes associated with customer nodes.

In conclusion, the versatility of binary attribute networks extends across various domains, including social networks, biology, recommender systems, online platforms, collaboration networks, and marketing. These networks offer a flexible and insightful framework for modeling and analyzing complex systems, where the presence or absence of specific attributes holds significance in unraveling network dynamics and extracting valuable insights. Whether

deciphering collaboration patterns, predicting customer preferences, or understanding the structure of social connections, binary attribute networks stand as powerful tools that cater to the nuanced intricacies of diverse applications.

Chapter 5

Procedure of Community Detection in Binary Attributed Network

5.1 Introduction

Community detection in binary node-attributed networks is a method employed to identify clusters of nodes in a network that share similar values in their binary attributes. In such networks, each node possesses a set of binary attributes, and the edges connecting nodes lack weights. The application of community detection in these networks spans various domains, encompassing social network analysis, bioinformatics, and information retrieval. This technique proves valuable in discerning patterns of similarity among nodes based on their binary attributes within the network, contributing to a more nuanced understanding of its structural and functional aspects.

The primary goal of community detection is to cluster nodes with akin attribute values, concurrently minimizing the number of edges connecting these clusters. This objective is realized through the maximization of a metric termed modularity. Modularity gauges the disparity between the real count of edges within communities and the anticipated count of edges in a hypothetically randomly connected network. The pursuit of optimizing modularity

facilitates the formation of cohesive groups within the network, enhancing the understanding of its inherent structural patterns.

5.2 Methodologies and Techniques

One strategy for identifying communities in binary node-attributed networks involves the application of **modularity optimization** [25]. Modularity serves as a metric to assess the effectiveness of partitioning a network into distinct communities. This metric quantifies the disparity between the observed number of edges within communities and the anticipated number of edges under random distribution. The process of modularity optimization entails exploring various network partitions and selecting the one that maximizes the modularity score. This approach aims to unveil community structures that exhibit higher internal connectivity compared to what would be expected by chance.

In binary node-attributed networks, modularity optimization involves the utilization of a null model that specifically considers the binary attributes associated with the nodes. This null model operates on the assumption that the likelihood of an edge between two nodes is linked to the similarity of their binary attributes. Subsequently, this null model is employed to generate a random network, enabling a comparison with the actual network for modularity calculation. This tailored approach accounts for the binary nature of node attributes, enhancing the accuracy of modularity optimization in the context of binary node-attributed networks.

Numerous algorithms are available for community detection in binary node-

attributed networks, each exhibiting distinct advantages and limitations. One widely adopted method is the **Louvain algorithm** [26]. This algorithm follows an iterative approach to partition the network into communities, with a primary focus on optimizing the modularity measure. The algorithm commences by assigning each node to an individual community and subsequently proceeds through iterations, merging communities to maximize modularity. This iterative process persists until further modularity improvement becomes unattainable. The Louvain algorithm's effectiveness lies in its ability to systematically enhance the modularity of the network's community structure.

An alternative strategy is the utilization of the **stochastic block model** [27], which posits that the network can be delineated into distinct blocks, each characterized by its probability of edge formation. This model is adaptable to binary node-attributed networks, allowing the estimation of the optimal number of blocks and their respective probabilities. After deriving the optimal model, nodes can be assigned to communities based on their highest probability of affiliation with a specific block.

Detecting communities in binary node-attributed networks involves leveraging the **similarity in attributes** among nodes. In these networks, each node possesses a binary attribute, denoted as 0 or 1. The similarity between two nodes is quantified by the number of identical attributes they share. Nodes exhibiting high similarity are more inclined to belong to the same community.

An alternative strategy for community detection in binary node-attributed networks involves employing techniques explicitly designed to consider the binary attributes of the nodes. An illustration of this approach is the **Binary**

Edge Partitioning (BEP) algorithm, which employs a probabilistic model to partition the edges of the network into communities. The BEP algorithm posits that the probability of an edge belonging to a community is contingent on the similarity of the binary attributes of the nodes it links.

Beyond modularity optimization and the BEP algorithm, several other methodologies prove effective for community detection in binary node-attributed networks. The **Markov Clustering algorithm** [28] employs a random walk process to discern clusters of nodes sharing similar binary attributes. Conversely, the **Hierarchical Agglomerative Clustering algorithm** [29] constructs a hierarchical tree of clusters, considering the similarity of binary attributes among nodes. These diverse approaches offer flexibility in tackling the challenges posed by binary node-attributed networks.

In conclusion, the process of community detection in binary node-attributed networks is a crucial methodology for pinpointing clusters of nodes sharing similar binary attributes. Utilizing the modularity measure as a common metric, various algorithms have been devised to optimize the network's partitioning into communities. The selection of a specific algorithm hinges on the unique characteristics of the network and the desired level of precision in the analysis.

Chapter 6

Various Methods for Optimization

6.1 Introduction

Optimization stands as a foundational and potent concept employed across diverse domains to seek the most favorable solution to a given problem. The process revolves around the maximization or minimization of an objective function while adhering to a set of constraints. The primary aim is to identify the optimal values for decision variables, ensuring the optimization of the objective function within the specified constraints. Optimization holds immense significance in various disciplines, encompassing mathematics, engineering, economics, operations research, machine learning, and beyond. Its extensive applications have transformed and elevated the methodologies for problem-solving and decision-making across a multitude of fields.

Optimization, at its essence, aims to uncover the most efficient, effective, or optimal resolution to a problem within specified constraints. The range of problems addressed by optimization is diverse, encompassing various levels of complexity and nature. These problems may feature linear or nonlinear functions, discrete or continuous decision variables, single or multiple objectives, and deterministic or stochastic conditions. The overarching objective

remains constant – to attain the most favorable outcome, be it in terms of maximizing profits, minimizing costs, enhancing efficiency, or optimizing the allocation of resources.

An integral component of optimization involves creating mathematical models that articulate the specific problem under consideration. These models delineate the objective function, decision variables, and constraints within a mathematical framework. The act of translating real-world problems into mathematical expressions endows optimization with a systematic and rigorous methodology for addressing challenges. This approach facilitates precise analysis, the exploration of solution spaces, and the identification of optimal solutions.

Optimization techniques can be broadly categorized into two main types: deterministic optimization and stochastic optimization. Deterministic optimization pertains to problems where all parameters and conditions are precisely known. Its objective is to identify the optimal solution based on the given inputs and constraints. Conversely, stochastic optimization tackles problems characterized by uncertainty or randomness. It takes into account probabilistic models and aims to find solutions that exhibit robustness or resilience in the face of variations and uncertainties.

Various optimization algorithms and methods have been developed to solve different types of optimization problems. These include linear programming, nonlinear programming, integer programming, dynamic programming, genetic algorithms, simulated annealing, particle swarm optimization, and many more. Each algorithm has its strengths and limitations, and the choice of algorithm depends on the problem characteristics, size, complexity, and computational requirements.

The significance of optimization in real-world applications cannot be overstated. In the field of engineering, optimization plays a pivotal role in the design of structures, circuits, and systems, ensuring optimal performance and efficiency. Within economics, optimization is instrumental in tasks such as resource allocation, production planning, and portfolio management. In the realm of transportation, optimization is indispensable for refining routes, schedules, and logistical operations. Moreover, in machine learning, optimization is critical for tasks such as model training, hyperparameter tuning, and enhancing predictive accuracy. These examples represent just a fraction of the myriad applications where optimization techniques have exerted a substantial influence.

In summary, optimization stands as a potent tool that empowers us to discover optimal solutions for intricate problems. It encompasses a diverse array of techniques and algorithms designed to refine objectives while adhering to constraints. The impact of optimization on problem-solving methodologies across various fields has been transformative, facilitating more efficient and effective decision-making processes. Its applications are expansive and varied, transcending disciplinary boundaries and finding relevance across industries. With ongoing technological advancements and enhanced computational capabilities, optimization remains a crucial force in shaping the evolving landscape of our world.

6.2 Different Techniques of Optimization

Optimization involves identifying the most favorable solution among a set of possible options [30]. This concept holds significant importance in numerous disciplines, including engineering, economics, operations research, and com-

puter science. Optimization problems come in various forms, encompassing linear programming, nonlinear programming, and combinatorial optimization. Over time, a multitude of optimization techniques has been devised to address and resolve these diverse problem types.

1. **Brute Force Search:** The process of brute force search entails a meticulous examination of every conceivable solution within the problem space to pinpoint the optimal one. Although this method assures the identification of the best solution, it frequently involves high computational costs, rendering it impractical, particularly when dealing with large-scale problems.

2. **Greedy Algorithms:** Greedy algorithms operate by making locally optimal decisions at each step, aiming to reach a global optimum. While these algorithms are known for their simplicity and efficiency, it's important to note that they may not consistently yield the absolute best solution. Their effectiveness is particularly prominent in scenarios where opting for the locally optimal choice at each step ultimately leads to an overall optimal solution [31].

3. **Gradient-Based Methods:**In optimization problems where the objective function is differentiable, gradient-based methods, exemplified by techniques like gradient descent [32], come into play. These methods follow an iterative approach, continually refining the solution by traversing the path of the steepest descent of the objective function. Widely applied in fields like machine learning and deep learning, gradient-based methods prove valuable for fine-tuning model parameters.

4. **Evolutionary Algorithms:** Evolutionary algorithms [33] draw inspira-

tion from the mechanisms of natural evolution, incorporating principles like selection, crossover, and mutation. These algorithms operate by maintaining a population of potential solutions and iteratively enhancing them across generations. Genetic algorithms and particle swarm optimization stand out as notable examples of evolutionary algorithms extensively employed in solving optimization problems.

5. **Simulated Annealing:** Simulated annealing serves as a probabilistic optimization method inspired by the annealing process observed in metallurgy. The algorithm initiates with an initial solution [34] and systematically explores the search space. Notably, it considers both superior and inferior solutions with a defined probability. Simulated annealing is recognized for its ability to navigate away from local optima, increasing the likelihood of converging towards a global optimum.

6. **Linear Programming:** Linear programming stands as a mathematical method employed for optimizing a linear objective function within the bounds of linear equality and inequality constraints. The primary objective is to identify values for decision variables that either maximize or minimize the objective function while adhering to the provided constraints. Widely applied in resource allocation, production planning, and logistics [35], linear programming proves to be a valuable tool in various domains.

7. **Nonlinear Programming:** Nonlinear programming, as an optimization technique [36], comes into play when dealing with problems featuring nonlinear objective functions or constraints. In contrast to linear programming, nonlinear programming introduces greater complexity,

necessitating advanced methods like the simplex algorithm, traditionally employed for solving linear programs. This technique finds application in scenarios where linear programming falls short due to the presence of nonlinear elements.

8. **Combinatorial Optimization:** Combinatorial optimization [37] is a field dedicated to identifying the optimal solution within a discrete set of choices. Illustrative examples of combinatorial optimization problems encompass the traveling salesman problem and the knapsack problem. Solving these intricate problems often entails the application of diverse techniques, including dynamic programming, branch and bound, and genetic algorithms, tailored to the specific nature of the optimization challenge at hand.

9. **Integer Programming:** Integer programming [38] is an extension of linear programming designed for scenarios where decision variables must assume integer values. This approach proves invaluable when optimization challenges entail discrete or combinatorial decision-making. Mixed Integer Linear Programming (MILP) and branch-and-bound algorithms stand out as prominent techniques routinely employed to tackle the intricacies of integer programming problems.

10. **Convex Optimization:** Convex optimization is concerned with problems in which both the objective function and constraints exhibit convexity [39]. This characteristic ensures that any local minimum within the solution space is also a global minimum, streamlining the optimization process. The utility of convex optimization spans diverse domains, encompassing machine learning, signal processing, and control systems.

11. **Metaheuristic Algorithms:** Metaheuristic algorithms [40] represent versatile optimization techniques designed to address a wide range of problems without being inherently tied to specific domains. Drawing inspiration from natural phenomena, social behaviors, or random processes, these algorithms include ant colony optimization, particle swarm optimization, and simulated annealing. They find applicability in situations where problem structures are intricate or when other optimization techniques prove unsuitable.

12. **Constraint Programming:** Constraint programming [41] is geared towards resolving optimization challenges entailing constraints with logical connections among variables. Its objective is to identify a feasible solution that adheres to all constraints while optimizing the objective function. Constraint programming finds utility in addressing scheduling, resource allocation, and combinatorial optimization problems.

13. **Dynamic Programming:** Dynamic programming is a technique used for solving optimization problems by breaking them down into smaller subproblems and solving them iteratively. It is particularly effective for problems with overlapping subproblems, allowing for efficient computation of the optimal solution. Dynamic programming has applications in areas such as robotics, resource management, and bioinformatics.

14. **Quadratic Programming:** Quadratic programming deals with optimization problems that involve quadratic objective functions and linear constraints. It is widely used in finance, portfolio optimization, and engineering applications. Quadratic programming algorithms, such as the active set method and interior-point methods, are employed to effi-

ciently solve these problems.

15. **Swarm Intelligence:** Swarm intelligence algorithms are inspired by the collective behavior of social insect colonies, such as ants and bees. These algorithms involve a population of agents that communicate and cooperate to find optimal solutions. Particle swarm optimization (PSO) and ant colony optimization (ACO) are examples of swarm intelligence techniques widely used in optimization problems.

16. **Stochastic Optimization:** Stochastic optimization [42]techniques are used when the problem involves uncertain or random parameters. These methods consider probability distributions and make decisions based on expected values or optimization under uncertainty. Stochastic gradient descent, simulated annealing, and genetic algorithms with probabilistic operators are common stochastic optimization methods.

17. **Heuristic Search:** Heuristic search algorithms [43] are designed to swiftly discover satisfactory solutions by navigating the problem space using heuristics or rule-of-thumb strategies. While they do not guarantee optimal solutions, these algorithms prioritize efficiency in finding acceptable solutions. Prominent examples of heuristic search algorithms encompass A* search, tabu search, and genetic algorithms incorporating heuristic approaches.

In summary, optimization plays a pivotal role across various fields, offering a multitude of techniques for solving optimization problems [44]. The selection of an appropriate optimization method hinges on the nature of the problem and the intricacy of the constraints. Optimal solutions can be efficiently identified by employing the right optimization technique tailored to the specific

requirements of the problem at hand.

6.3 Single-Objective Optimization Method

Single-objective optimization stands as a potent technique employed to ascertain the optimum solution for problems featuring a solitary objective function. This foundational concept in optimization revolves around maximizing or minimizing a distinct objective, all within the constraints defined by the problem. Its applications span diverse fields, including engineering, economics, operations research, and data science. This essay delves into the principles, methodologies, and applications that characterize single-objective optimization.

Central to single-objective optimization is the objective function, serving as the bedrock that encapsulates the quantity slated for maximization or minimization. This function, tailored to the nuances of the problem, manifests as a mathematical formula or computational model. It systematically quantifies the performance or quality measure pertinent to the system or process undergoing optimization. The overarching aim revolves around determining the optimal values for decision variables, thereby attaining the most favorable outcome as dictated by the objective function.

At the heart of single-objective optimization are the decision variables, akin to adjustable levers or inputs that wield influence over the objective function's optimization. These variables, instrumental in shaping the outcome, operate within the confines of predefined constraints. Constraints, acting as the regulatory framework, delineate the boundaries and conditions integral to the optimization problem. Whether in the form of equality constraints,

dictating specific relationships between variables, or inequality constraints, prescribing bounds or limitations, these conditions encapsulate the optimization landscape.

To tackle a single-objective optimization problem, an array of methods and algorithms is at one's disposal. A prevalent approach involves mathematical programming, wherein the conundrum is cast into a mathematical model. Subsequently, optimization techniques are employed to unearth the optimal solution. This realm encompasses linear programming (LP), nonlinear programming (NLP), and mixed-integer programming (MIP), which stand out as conventional methods within the domain of single-objective optimization.

Linear programming specializes in scenarios where both the objective function and constraints exhibit linearity. Its utility spans resource allocation, production planning, and network optimization. Nonlinear programming broadens the spectrum by accommodating nonlinear objective functions and constraints, facilitating the optimization of more intricate systems. Meanwhile, mixed-integer programming integrates discrete decision variables, catering to situations demanding choices from a finite set of options.

The applications of single-objective optimization are extensive and varied. Within engineering, it finds utility in optimizing the design of structures, circuits, and systems, aiming to achieve peak efficiency or performance. In economics, single-objective optimization aids in discerning the optimal allocation of resources, whether to maximize profits or minimize costs. In the realms of data science and machine learning, its application extends to fine-tuning hyperparameters of models, optimizing feature selection, and enhancing predictive accuracy.

6.4 Multi-objective Optimization Method

Multi-objective optimization presents a problem-solving paradigm where the simultaneous optimization of multiple objective functions is pursued [45]. Many real-world problems inherently involve the consideration of several objectives that may, at times, conflict with one another. For instance, in manufacturing, the optimization goals might encompass minimizing production costs while concurrently reducing production time and maximizing product quality.

In contrast to the conventional optimization approach, which focuses on identifying a solitary optimal solution aligned with a single objective function, multi-objective optimization recognizes the complexity of problems involving multiple objectives. Instead of seeking a singular solution, this approach endeavors to discover a set of optimal solutions that strike a balance among the diverse objectives.

Numerous methodologies exist for tackling multi-objective optimization problems, with Pareto optimization standing out as a prominent approach rooted in the Pareto principle[46]. This principle posits that enhancing one objective inevitably involves sacrificing another. In essence, a trade-off exists among different objectives, and the optimal solution is situated on the Pareto frontier. This frontier denotes the set of solutions that cannot be improved in any single objective without deteriorating another.

Additional strategies for addressing multi-objective optimization encompass evolutionary algorithms, notably genetic algorithms [47] and particle swarm optimization[48]. Drawing inspiration from natural selection and swarm be-

havior, these algorithms embark on a quest for the optimal solution. They achieve this by iterating over a population of candidate solutions and incrementally enhancing them through a dynamic process involving selection, mutation, and crossover.

The realm of multi-objective optimization finds applications across diverse fields such as engineering, finance, and management. Its significance lies in empowering decision-makers to weigh multiple objectives when making critical decisions, facilitating the delicate balance of competing interests. Nevertheless, it is essential to acknowledge that the computational expense of multi-objective optimization, particularly in dealing with expansive datasets, underscores the need for a meticulous selection of the optimal optimization technique and parameters tailored to the specifics of each problem.

Mathematical Equations

Multiobjective optimization involves finding the set of solutions that optimizes multiple objective functions. Multiobjective functions can be maximizing or minimizing in nature. There are some constraints for a multiobjective problem identifying feasible solutions out of a large set of candidates, where the problem can be modeled in terms of arbitrary constraints. It is represented mathematically as follows:

$$min/max\ F(x) = (f_1(x),\ f_2(x), ..., f_k(x))^T$$

$$subject\ to \begin{cases} g_1(x) \leq 0 \\ g_2(x) \leq 0 \\ .... \\ g_p(x) \leq 0 \end{cases}$$

where x is the vector of decision variables, $(f_1(x), f_2(x), ..., f_k(x))^T$ are the objective functions to be minimized or maximized, $(g_1(x), g_2(x), ..., g_p(x)$ are the constraints, and X is the feasible region of the decision variables.

Pareto-optimal solution

A Pareto optimal solution refers to a specific outcome in a multi-objective optimization scenario. In such a setting, where there are multiple objectives or criteria to be considered, a Pareto optimal solution represents a unique state where no further improvement can be made in one objective without causing a detriment to at least one of the other objectives. In simpler terms, a Pareto optimal solution is a point of equilibrium where it stands out as the best compromise, excelling in certain objectives, and any attempt to enhance one aspect would result in a setback in another. This concept is derived from the Pareto principle, which underscores the inherent trade-offs between different objectives in complex decision-making situations.

The set of all Pareto optimal solutions is called the Pareto front or Pareto set. These solutions represent the trade-off between conflicting objectives in the optimization problem, and they provide decision-makers with a set of options to choose from, depending on their preferences and priorities.
Pareto optimal solutions find practical applications in various real-world scenarios, including portfolio optimization, engineering design, and resource

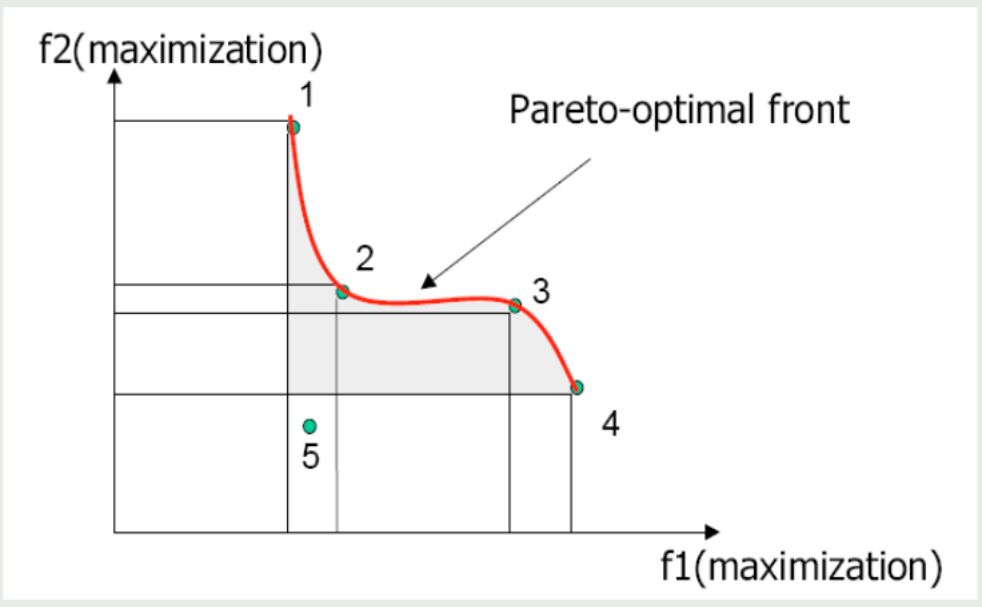


Figure 6.1: Pareto-optimal front
[49]

allocation. In the context of portfolio optimization, where the aim is to maximize return on investment while simultaneously minimizing risk, the principles of Pareto optimality become particularly relevant. Given the inherent conflict between maximizing returns and minimizing risk, a collection of Pareto optimal solutions provides investors with a spectrum of choices. This set enables investors to select a portfolio that aligns most closely with their specific risk-return preferences, striking a balance that suits their financial goals and risk tolerance.

Discovering Pareto optimal solutions in multi-objective optimization problems can be achieved through diverse methods, including mathematical programming techniques, evolutionary algorithms, and heuristic search algorithms. The selection of a specific method hinges on factors such as the intricacy of the problem, the quantity of objectives and constraints involved, and the targeted balance between accuracy and efficiency. Tailoring the approach to the unique characteristics of each problem ensures a more effective

and suitable exploration of the Pareto optimal solution space.

Domination Relationship

A vector $x = (x_1, ...x_k)$ is considered to dominate another vector $(y_1, ..., y_k)$ if and only if there is at least one m such that $x_j \leq y_j, \forall\, j \in \{1, ..., k\}$ and $x_m < y_m$, and the vector y dominated by vector x is denoted by $x \prec y$. X is a Pareto optimal solution if it is not dominated by any other solution.

Among a set of solutions P, the non-dominated set of solutions P' are those that are not dominated by any solution in set P. Solution x is called non-dominating with respect to all the solutions if there exists no solution y that dominates x. The non-dominated set of the entire search space S is the globally Pareto optimal set.

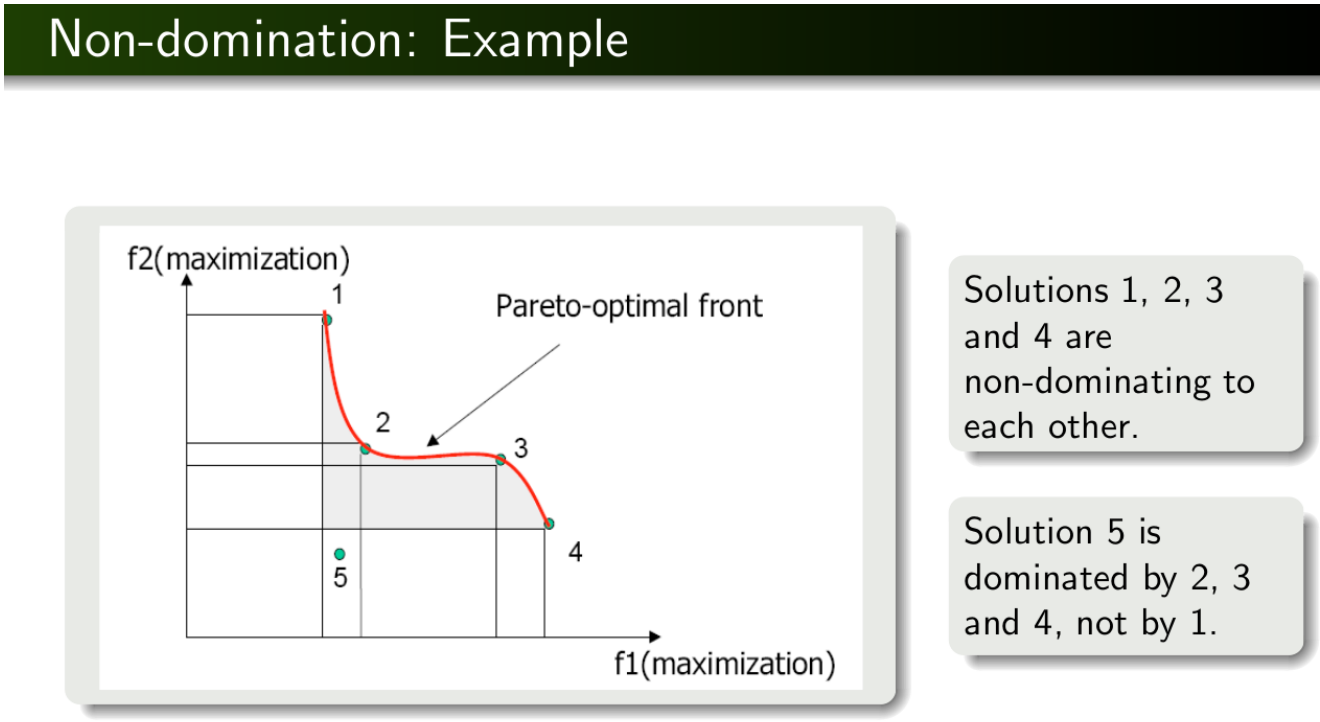


Figure 6.2: Non-domnation example
[50]

6.5 Genetic Algorithm (GAs)

The genetic algorithm, a computational approach inspired by natural selection and genetics, serves as a potent tool for addressing optimization challenges. The algorithm initiates by establishing a population of potential solutions. Subsequently, it applies selection, crossover, and mutation operators to iteratively evolve and enhance these solutions across generations. By evaluating fitness functions, the algorithm identifies and retains the most promising solutions while eliminating weaker ones, mirroring the biological principle of survival of the fittest. Genetic algorithms have demonstrated success across diverse domains, encompassing engineering, economics, biology, and computer science, cementing their role as indispensable tools for tackling intricate problems.

Initialization

Initialization is a crucial phase in the application of a genetic algorithm (GA), as it involves the creation of the initial population of candidate solutions within the problem's search space. The quality of this initial population significantly influences the performance and efficiency of the GA.

In a GA, the population is typically generated randomly, aiming to cover the diverse range of potential solutions in the search space. The primary goal of initialization is to establish a varied set of candidate solutions that effectively represent the problem's landscape.

Several methods have been proposed for the initialization phase in GAs, each with its advantages and suitability for different scenarios. These methods include:

1. **Random Initialization**: This straightforward approach involves randomly creating individuals in the population. While simple, it may lack diversity and struggle to explore the entire solution space effectively.

2. **Latin Hypercube Sampling**: This method ensures a more even distribution of candidate solutions across the search space. It divides the search space into regions and assigns a single solution to each region, reducing redundancy.

3. **Heuristic Initialization**: Some problems benefit from initializing the population using problem-specific heuristics. These heuristics leverage domain knowledge to generate candidate solutions that are likely to be closer to the optimal ones.

The choice of the initialization method depends on the specific characteristics of the problem domain and the complexity of the search space. An appropriate initialization strategy contributes to the GA's ability to explore diverse solutions and converge toward optimal or near-optimal solutions efficiently.

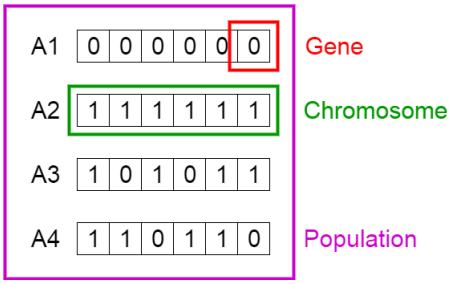


Figure 6.3: Population, Chromosome and gene
[47]

Fitness Function

The fitness function plays a pivotal role in genetic algorithms (GAs) as it serves as a critical metric to assess the quality of individuals within the population. This function is tailored to the specific problem domain and aims to quantify how effectively an individual solution addresses the given problem.

The primary objective of the fitness function is to provide a numerical representation of the solution's quality. It encapsulates the problem's goals and constraints, allowing the GA to distinguish between better and worse solutions. The fitness value assigned to each individual acts as a guide for the algorithm, influencing the selection of individuals for the reproduction process.

During the selection phase, individuals with higher fitness values are granted a greater probability of being chosen for reproduction. This mimics the concept of "survival of the fittest" in nature, where individuals better adapted to their environment have a higher chance of passing on their genes to the next generation.

As the GA progresses through multiple generations, the emphasis on individuals with higher fitness values promotes the evolution of the population towards improved solution quality. Over time, this iterative process aims to refine the solutions, ultimately converging towards optimal or near-optimal outcomes for the given problem.

In summary, the fitness function is a crucial element in the success of genetic algorithms, guiding the algorithm toward solutions that align with the problem's objectives and constraints. Its careful design is essential for the algorithm's ability to effectively explore the solution space and converge to

satisfactory solutions.

Selection

Selection is a pivotal operation within genetic algorithms (GAs), playing a key role in determining which individuals from the current population will contribute to the creation of the next generation. Various selection methods are employed in GAs, each with its unique approach. Common selection methods include roulette wheel selection, tournament selection, rank-based selection, and more.

The fundamental principle governing the selection process is to assign higher probabilities of being chosen to individuals with superior fitness values. This strategic emphasis on fitter individuals aims to guide the evolution of the population towards optimal or near-optimal solutions. The selection operation is vital for mimicking the natural process of favoring individuals better adapted to their environment.

1. **Roulette Wheel Selection**: This method involves associating a portion of the roulette wheel with each individual, where the size of the portion is proportional to the individual's fitness. A spin of the wheel then determines the selected individual.

2. **Tournament Selection**: In tournament selection, a predefined number of individuals are randomly selected from the population. The individual with the highest fitness among the chosen set is then chosen for reproduction.

3. **Rank-Based Selection**: This method involves ranking the individuals

based on their fitness, and selection is performed according to the rank. The probability of selection is often inversely proportional to the rank.

By employing these selection methods, GAs ensure that individuals with better fitness values have a greater chance of being selected to contribute their genetic material to the next generation. This iterative process, coupled with crossover and mutation, aims to refine the population over successive generations and converge toward optimal solutions. The selection operation is a crucial element in the efficiency and success of genetic algorithms.

Crossover

In genetic algorithms (GAs), the crossover operation is a pivotal process that emulates the biological mechanism of exchanging genetic information between two parents to generate offspring. This operation mirrors the natural process of reproduction where genetic material from both parents combines to produce a new generation.

The typical procedure for crossover involves selecting a random point along the chromosomes of the parent individuals. Subsequently, the genetic material on either side of this randomly chosen point is exchanged between the parents, resulting in the creation of offspring with a combination of genetic traits from both parents.

This mimicking of biological reproduction contributes to the diversification of genetic material within the population, introducing new combinations of traits that may lead to improved solutions. The effectiveness of the crossover operation is influenced by factors such as the choice of the crossover method, the representation of the problem, and the characteristics of the genetic encoding. Overall, the crossover operation plays a crucial role in the

evolutionary process of genetic algorithms, fostering diversity and facilitating the exploration of the solution space.

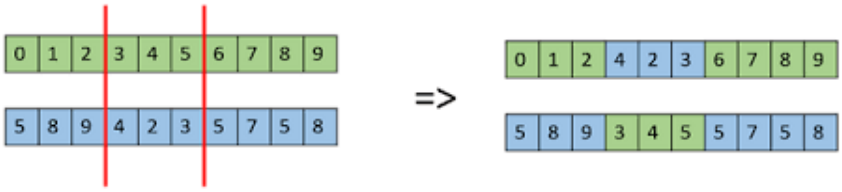


Figure 6.4: Crossover
[51]

This process generates two new individuals that combine the genetic information of their parents. Crossover is an essential operator in genetic algorithms as it allows for the exploration of the search space and the creation of new solutions that may not have been present in the initial population.

Mutation

In genetic algorithms, mutation is a crucial process aimed at introducing random changes to the genetic material of individuals in the population. The primary purpose of mutation is to enhance diversity within the population, preventing the algorithm from converging too quickly and potentially getting trapped in local optima. This stochastic operation plays a vital role in exploring new regions of the solution space.
The mutation process typically involves the following steps:

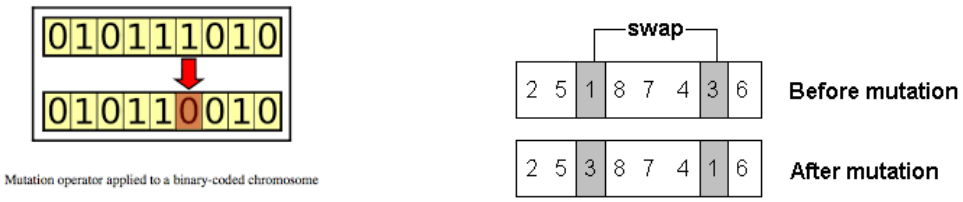
1. **Random Selection of Genes**: A gene or multiple genes within an individual's chromosome are randomly selected for mutation.

2. **Alteration of Gene Values**: The selected gene's value is then altered or mutated to a new random value within the permissible range. This introduces variability in the individual's genetic makeup.

Mutation is particularly important in maintaining genetic diversity, as it provides a mechanism for exploring novel solutions. Without mutation, a genetic algorithm might converge prematurely to a suboptimal solution, especially if the initial population lacks the necessary genetic diversity.

The rate at which mutation occurs, often referred to as the mutation rate, is a parameter that influences the balance between exploration and exploitation in the genetic algorithm. A higher mutation rate promotes greater exploration by introducing more random changes, while a lower mutation rate emphasizes exploitation by relying more on crossover and selection.

Effective mutation strategies depend on the nature of the problem being solved, and various mutation techniques, such as bit-flip mutation and non-uniform mutation, offer different ways of introducing randomness into the genetic material. The choice of mutation strategy and rate is a crucial consideration in designing a robust genetic algorithm tailored to specific optimization tasks.



(a) Bit Mutation **(b)** Swap Mutation

Figure 6.5: Mutation
[52]

The mutation rate is an important parameter to control the degree of exploration and exploitation in the search space. A higher mutation rate increases the exploration power of the algorithm while a lower mutation rate promotes the exploitation of promising solutions. Mutation can help the algorithm to discover new solutions that are not present in the initial population.

Replacement

In the context of Genetic Algorithms (GAs), replacement is a pivotal phase involving the selection of individuals from the current population to be replaced by newly generated offspring. The overarching objective is to strike a balance between preserving population diversity and enhancing overall fitness through the introduction of potentially superior solutions.
Two fundamental replacement strategies are commonly employed in Genetic Algorithms:

1. **Generational Replacement**: In this strategy, the entire current population is entirely replaced by the offspring generated in the current iteration. The new population is composed solely of the offspring, allowing for a complete transition from one generation to the next.

2. **Steady-State Replacement**: In contrast to generational replacement, steady-state replacement involves replacing only a subset of the current population with the newly generated offspring. Typically, a small number of individuals are selected for replacement, and the remaining individuals are carried over to the next generation. This approach allows for a more gradual evolution of the population.

The choice between these replacement strategies has implications for the dynamics of the genetic algorithm. Generational replacement tends to introduce a higher degree of exploration, as the entire population is refreshed in each generation. Steady-state replacement, on the other hand, emphasizes a more continuous and incremental evolution by gradually introducing new individuals.

The effectiveness of a replacement strategy depends on the problem at hand, and factors such as the rate of convergence, the desired level of diversity, and the balance between exploration and exploitation should be considered when selecting an appropriate replacement mechanism. Tailoring the replacement strategy to the characteristics of the optimization problem contributes to the overall efficiency and performance of the genetic algorithm.

Termination

In Genetic Algorithms (GAs), termination denotes the concluding phase of the algorithm, where a specified criterion is met, prompting the algorithm to cease its iterations. Termination conditions are pivotal for determining when the GA has sufficiently explored the solution space or when further iterations are unlikely to yield significant improvements. Several factors can influence the termination of a GA:

1. **Number of Generations**: A common termination criterion is a predefined maximum number of generations. Once the algorithm has undergone a set number of iterations or generations, it terminates, regardless of other factors. This approach is useful when the algorithm is expected to converge within a specific timeframe.

2. **Fitness Level**: Termination based on fitness level involves specifying a target fitness value. If any individual in the population reaches or exceeds this target fitness, the algorithm terminates. This approach assumes that achieving a certain level of fitness is indicative of a satisfactory solution.

3. **Convergence of Population**: Convergence-based termination considers the stability or convergence of the population. The algorithm can terminate when the population's diversity falls below a certain threshold, suggesting that further iterations may not significantly improve the solutions.

4. **Resource Constraints**: Termination can also be guided by external factors such as time constraints, computational resources, or budget limitations. If the algorithm exceeds predefined resource limits, it terminates to avoid excessive computation.

5. **Solution Stability**: Some termination conditions involve monitoring the stability of the best solution over consecutive generations. If the best solution remains unchanged for a certain number of iterations, it may indicate that the algorithm has reached a plateau, prompting termination.

The choice of termination criteria depends on the specific characteristics of the optimization problem, the desired level of solution quality, and the available computational resources. Implementing appropriate termination conditions ensures that the GA strikes a balance between exploring the solution space adequately and avoiding unnecessary computational overhead.

6.6 Non-dominated Sorting Method

Non-dominated Sorting is a pivotal stage in Multi-Objective Optimization (MOO) algorithms, particularly in the context of Evolutionary Multi-Objective Optimization (EMO) algorithms. This process involves categorizing solutions to an optimization problem into distinct groups or fronts, primarily determ-

ined by their Pareto dominance relationship.

Here's an elaboration on the key aspects of Non-dominated Sorting:

1. **Pareto Dominance Relationship:** The fundamental principle behind Non-dominated Sorting is rooted in the Pareto dominance relationship. In a multi-objective optimization scenario, a solution is said to dominate another if it is superior in at least one objective and not worse in any other. Non-dominated solutions, often termed Pareto-optimal solutions, represent a set where no solution is dominated by another.

2. **Categorization into Fronts:** Non-dominated Sorting organizes solutions into fronts or layers, where the first front comprises non-dominated solutions (Pareto-optimal solutions), the second front contains solutions that are only dominated by solutions in the first front, and so on. This process results in a series of fronts, each representing a different level of Pareto optimality.

3. **Objective of Non-dominated Sorting:** The primary objective of Non-dominated Sorting is to establish a clear ranking among solutions based on their Pareto dominance relationships. This facilitates the identification of trade-offs and compromises between conflicting objectives. The resulting fronts provide a comprehensive view of the solution space, aiding decision-makers in choosing solutions that cater to various preferences.

4. **Handling Multiple Objectives:** Non-dominated Sorting becomes particularly crucial in multi-objective scenarios where there is no single solution that optimally satisfies all objectives simultaneously. The sort-

ing process helps in distinguishing solutions that represent different trade-offs and trade spaces, enabling decision-makers to make informed choices based on their preferences.

5. **Evolutionary Algorithms:** In EMO algorithms, such as Genetic Algorithms for Multi-Objective Optimization, Non-dominated Sorting is an integral step during the selection process. It guides the evolution of solutions by preserving non-dominated individuals and ensuring their continued participation in the optimization process.

In summary, Non-dominated Sorting is a fundamental mechanism in the realm of multi-objective optimization, aiding in the systematic organization of solutions based on their Pareto dominance relationships. This categorization is instrumental in exploring the diverse trade-offs and compromises inherent in multi-objective optimization problems.

Pareto Dominance is a fundamental concept in Multi-Objective Optimization, and it is used to define the relationship between two solutions. Solution A is said to dominate solution B if and only if A is not worse than B in any objective and is strictly better in at least one objective. In other words, A is said to be Pareto-dominant to B if A is at least as good as B in all objectives and strictly better than B in at least one objective. For example, if we have two solutions, A and B, and solution A has a better objective value than solution B in one objective, and they are equal in the other objective, then solution A dominates solution B.

The Non-dominated Sorting algorithm operates by partitioning the population of solutions into distinct fronts, each representing a unique level of Pareto optimality. Here's a breakdown of how the algorithm functions:

1. **Initialization:** Begin with the entire population of solutions in the optimization problem.

2. **Pareto Dominance Comparison:** Compare each solution with every other solution in the population based on Pareto dominance.
   Identify solutions that are non-dominated, i.e., solutions that are not inferior to any other solution in the population based on the defined objectives.

3. **Front Assignment:** Assign non-dominated solutions to the first front (Front 1) since they represent the highest level of Pareto optimality.
   Remove these solutions from the population.

4. **Repeat for Subsequent Fronts:** Iterate the process for the remaining solutions in the population. For each iteration, identify solutions that are dominated only by solutions in the previous front but not by any other solution in the current population.
   Assign these solutions to the next front (Front 2) and remove them from the population.

5. **Continue Iteration:** Continue this process iteratively until all solutions are assigned to fronts. Each front contains solutions that are non-dominated by solutions in the same front or any other front.

6. **Front Ranking:** The resulting fronts are now ranked based on their level of Pareto optimality, with Front 1 containing the non-dominated solutions that are not outperformed by any other solution in the population.

7. **Trade-Off Analysis:** The different fronts provide a spectrum of solutions, each representing a distinct trade-off between conflicting objectives.

Decision-makers can analyze these fronts to understand the trade-offs and compromises inherent in the solution space.

8. **Preservation for Evolutionary Algorithms:** In the context of Evolutionary Multi-Objective Optimization (EMO) algorithms, such as Genetic Algorithms, the Non-dominated Sorting algorithm guides the selection of solutions to be preserved for the next generation. Non-dominated solutions are prioritized for their potential to contribute to Pareto front exploration.

In essence, the Non-dominated Sorting algorithm systematically organizes solutions into fronts, facilitating a comprehensive understanding of the Pareto-optimal solution space in multi-objective optimization problems. To perform

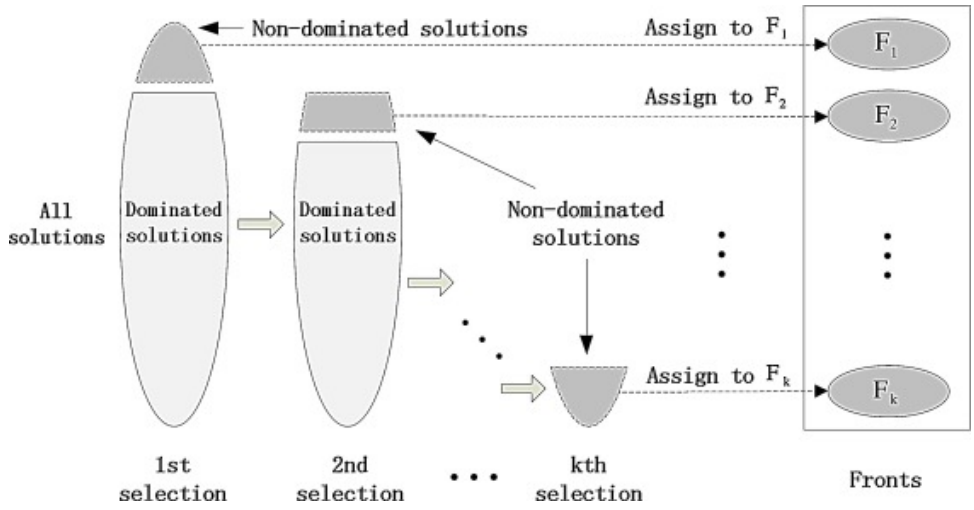


Figure 6.6: Non Dominated Sorting [53]

Non-dominated Sorting, we start by comparing all solutions in the population pairwise. We can use the Pareto dominance relationship to determine which solutions are dominated and which ones are not. We then assign a rank to each solution based on the number of solutions that dominate it. A solution with a rank of 1 is in the first front, and a solution with a rank of 2 is in the

second front, and so on. If two or more solutions have the same rank, we can use a crowding distance measure to decide which one to keep.

The crowding distance measure is used to maintain diversity in the population by keeping solutions that are well-distributed in the objective space. The crowding distance of a solution is the average distance to its nearest neighbors in the objective space. Solutions with a higher crowding distance are kept in the population, while solutions with a lower crowding distance are removed.

The non-dominated Sorting algorithm is used in many Multi-Objective Optimization algorithms, including NSGA-II, MOEA/D, and SPEA2. It is an efficient and effective way to handle multi-objective optimization problems and to obtain a set of non-dominated solutions that represents the trade-offs between different objectives. Non-dominated Sorting is an essential tool in the field of Multi-Objective Optimization, and it has been widely studied and applied in various fields, including engineering, finance, and computer science.

Fast Non-dominated Sorting: Fast Non-dominated Sorting (FNS) is a high-performance algorithm designed for efficiently categorizing solutions into non-dominated fronts in the context of multi-objective optimization problems. In such problems, where multiple conflicting objectives need to be optimized simultaneously, solutions are evaluated based on vectors of objective function values. The primary aim is to identify and rank solutions according to their non-dominance status.

Here's an overview of how the Fast Non-dominated Sorting algorithm operates:

1. **Initialization:** Begin with the entire set of solutions in the optimization

problem.

2. **Sorting:** Divide the solutions into multiple fronts based on non-dominance. A solution is placed in Front 1 if it is not dominated by any other solution. Solutions dominated only by those in Front 1 are assigned to Front 2, and so on.

3. **Fast Sorting:** FNS employs optimization techniques to expedite the sorting process.
   It optimizes the computational complexity, making it particularly suitable for scenarios with large populations and high-dimensional solution spaces.

4. **Ranking:** Solutions within each front are further ranked based on various criteria, such as crowding distance.
   Crowding distance is a measure of the density of solutions in the objective space and helps to maintain diversity among non-dominated solutions.

5. **Preservation for Evolutionary Algorithms:** In the context of Evolutionary Multi-Objective Optimization (EMO) algorithms like Genetic Algorithms, the FNS algorithm assists in selecting solutions to be carried forward to the next generation.
   Non-dominated solutions are crucial for preserving diversity and exploring the Pareto front efficiently.

6. **Efficiency Considerations:** FNS is designed to provide computational efficiency, enabling the handling of large solution sets with reduced computational cost.

The efficiency gains are particularly beneficial for evolutionary algorithms that involve repeated evaluations of large populations.

In summary, Fast Non-dominated Sorting is a powerful algorithm that efficiently handles the classification and ranking of solutions in multi-objective optimization problems. Its optimization techniques make it well-suited for scenarios with computationally demanding requirements, contributing to the effective exploration of the Pareto-optimal solution space.

The FNS algorithm was proposed by Kalyanmoy Deb in 2002 and is an extension of the Non-dominated Sorting Genetic Algorithm (NSGA) proposed by Deb in 2000. The FNS algorithm is designed to address the main limitation of NSGA, which is the high computational cost of sorting the solutions into non-dominated fronts.

The FNS algorithm works by maintaining a set of frontiers that represent the non-dominated solutions. The algorithm iteratively selects the non-dominated solutions in each frontier and adds them to the next frontier until all the solutions have been classified into non-dominated frontiers. The algorithm uses a fast sorting technique to efficiently identify the non-dominated solutions in each frontier.

6.7 (Non-dominated Sorting Genetic Algorithm II(NSGA-II)

NSGA-II (Non-dominated Sorting Genetic Algorithm II) [54] stands as a prominent multi-objective optimization algorithm, initially introduced in 2002 by Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. This algorithm represents an advancement over its predecessor, NSGA (Non-dominated Sorting Genetic Algorithm), and has gained widespread recognition and adoption in the field of multi-objective optimization.

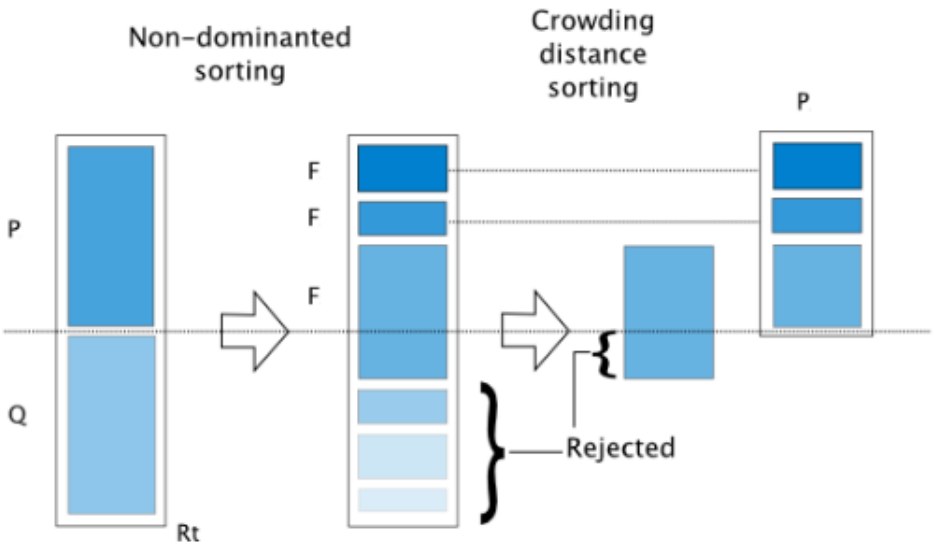


Figure 6.7: NSGA-II
[55]

NSGA-II uses the concept of non-dominated sorting to rank individuals based on their fitness in multiple objective functions. It also introduces a novel crowding distance measure to maintain diversity among the solutions in the population. The algorithm starts with an initial population of individuals, which are evaluated based on the multiple objective functions. The non-dominated sorting procedure is then applied to the population, which sorts the individuals into different fronts based on their non-dominated status.

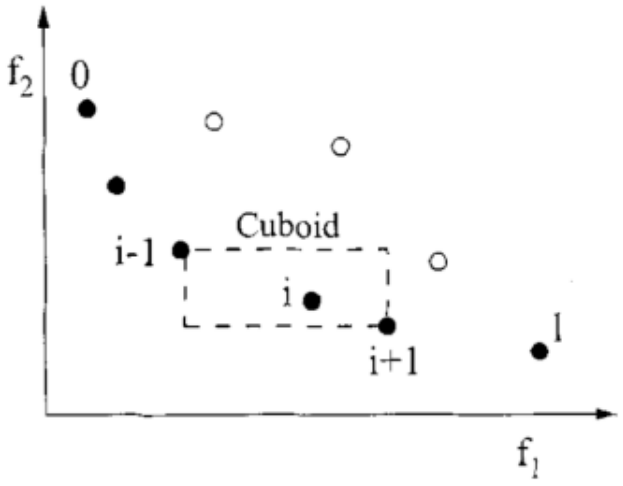


Figure 6.8: Crowding Distance
[56]

After the non-dominated sorting procedure, the crowding distance measure is used to select individuals for the next generation. The crowding distance measures the density of individuals around a particular solution in the population, and it is used to promote diversity among the solutions. The algorithm then proceeds with the application of genetic operators, such as crossover and mutation, to generate new offspring solutions, which are then evaluated based on the multiple objective functions.

NSGA-II continues this process of non-dominated sorting, selection, and genetic operators until a stopping criterion is met. The algorithm typically terminates when a predefined number of generations or a desired level of convergence is achieved.

6.8 (Non-dominated Sorting Genetic Algorithm III (NSGA-III)

NSGA-III [57] is an improved version of the NSGA-II algorithm for multi-objective optimization. It was introduced by Kalyanmoy Deb et al. in 2014.

NSGA-III is designed to address some of the limitations of NSGA-II, particularly in handling problems with a large number of objectives. It uses a reference point-based approach to generate a set of uniformly distributed solutions in the objective space.

NSGA-III performs non-dominated sorting on the population, similar to NSGA-II. However, instead of using crowding distance to maintain diversity, NSGA-III uses a reference point-based crowding distance, which takes into account the distance of each solution to a set of reference points. NSGA-III also uses an adaptive selection scheme, which dynamically adjusts the

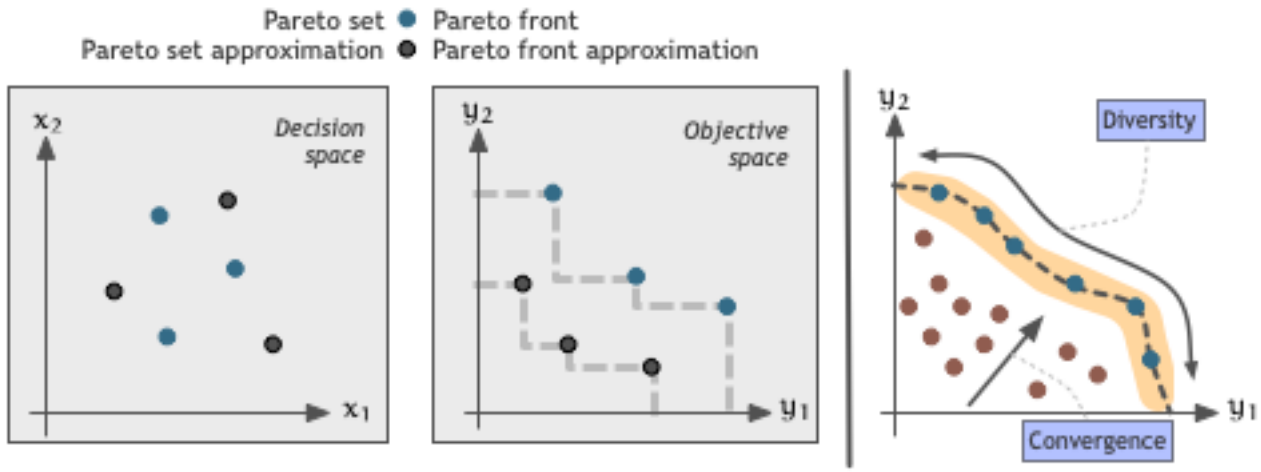


Figure 6.9: NSGA 3 Pareto set
[58]

number of solutions selected from each non-dominated front based on the shape of the front.

In NSGA-III, at first, the non-dominated sorting is done as NSGA-II. Second, a few alternatives must be chosen from the splitting front. The underrepresented reference direction is first filled by NSGA-III. The solution with the lowest perpendicular distance in the normalized goal space is the one that survives if the reference direction has n solutions assigned. In the event that this reference line receives a second solution, it is given at random. As a result, each reference line searches for a good representative non-dominated solution as the algorithm converges.

Determination of reference points in a Hyperplane : To guarantee the diversity of the solutions found, a set of reference points must be established. On a normalized hyperplane, various points are positioned with the same orientation along all axes. The formula for the number of reference points (H) is:

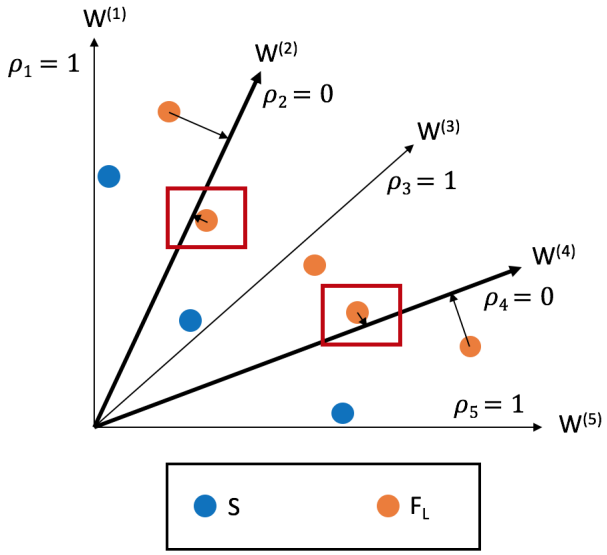


Figure 6.10: Solutions selected from the splitting front after non-domination sortiing [59]

$$H = \begin{pmatrix} C+g-1 \\ g \end{pmatrix}$$

where g is the number of divisions to take into account for each objective axis, and C is the number of objective functions (for 3 objectives and 4 divisions, there will be 15 reference points). The produced reference points will be connected to the solutions when the reference points are placed on the hyperplane and the solutions are defined using a Pareto front.

Normalization of Popular Member: It is a technique used to normalize the crowdedness measure of the non-dominated solutions. This technique involves dividing the crowdedness measure of each non-dominated solution by the sum of the crowdedness measures of all non-dominated solutions in the population. This ensures that the crowdedness measure of each non-dominated solution is normalized to a value between 0 and 1, where 0 indicates that the solution is not crowded and 1 indicates that the solution is highly crowded.

Association among reference points and solutions: Each population member must be connected to a reference once each objective function has been normalized. For each point, a reference line connecting it to the origin point is defined. The perpendicular distance between each reference line and each population member is then calculated. Finally, a population member is paired with the reference point that is closest to them on the reference line.

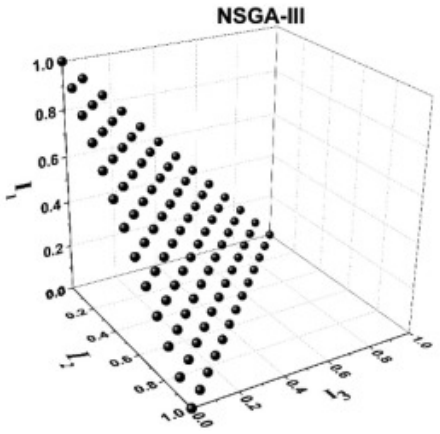


Figure 6.11: NSGA3 (solutions)
[60]

Niche Preservation Operation: One or more solution members may be connected with a reference point, but the solution that is closest to the reference point must be maintained (perpendicular to the reference line).

Genetic Operation: The NSGA-III algorithm's genetic operators will be utilized to create the next generation of children.

Chapter 7

Survey of Literatures

Overlapping Community Detection Using Multiobjective Genetic Algorithm

A popular approach to overlapping community detection is the use of multiobjective genetic algorithms (MOGAs), which are optimization techniques that can simultaneously optimize multiple objectives. In this context, the objectives are typically the identification of communities and the minimization of some measure of overlap between them.

One example of MOGA-based overlapping community detection is the work by [61]. They proposed a MOGA-based algorithm that uses a fitness function that combines modularity, a measure of community structure, and the number of overlaps between communities.

This paper uses the mixed representation [62] to find the overlapping nodes and derive a fuzzy membership value. NSGA2 is used as the implementation model for multiobjective optimization.

Community detection based on spectral clustering with node attributes

Tang and Ding proposed a community detection method [5] that integrates both structural and attribute similarities into a unified framework. Specifically, they first construct a weighted network where nodes represent objects and edges represent their structural similarities. Then, they introduce an attribute similarity matrix to model the similarity between nodes' attributes. The two similarity matrices are then combined to form a unified similarity matrix using a weight parameter.

The authors then propose an objective function that maximizes the modularity of the community structure while considering both types of similarities. They formulate the optimization problem as a binary quadratic programming problem, which can be efficiently solved using existing optimization algorithms.

Community Detection in Complex Networks by Detecting and Expanding Core Nodes Through Extended Local Similarity of Nodes

The paper [63] proposes a novel approach for community detection in complex networks. The method is based on detecting and expanding core nodes through extended local similarity of nodes. The proposed algorithm first identifies the core nodes of the network, which are highly connected and likely to be part of multiple communities. Then, the algorithm expands these core

nodes by adding their neighbors with a high similarity score, resulting in the identification of overlapping communities.

The extended local similarity measure used in the algorithm is defined as a combination of the common neighbors and the Jaccard similarity index between two nodes. The authors also introduce a threshold parameter to control the size of the identified communities.

Community Detection in Attributed Social Networks: a Unified Weight-Based Model and Its Regimes

This paper [64] proposes a unified weight-based model for community detection in attributed social networks, which takes into account both structural and attribute similarities. The proposed model is based on a probabilistic framework that uses a weighted adjacency matrix to represent the network, where the weights are based on both the structural and attribute similarities between nodes.

To optimize the proposed model, the paper presents two mathematical programming approaches: a mixed integer linear programming (MILP) formulation and a continuous quadratic programming (QP) formulation. The MILP formulation aims to find an optimal partition of the network into communities, while the QP formulation aims to find a relaxed solution that approximates the optimal solution.

The paper also proposes three different regimes for the proposed model, depending on the relative importance of structural and attribute similarities. The first regime is the structural regime, where only structural similarities are

considered. The second regime is the attribute regime, where only attribute similarities are considered. The third regime is the mixed regime, where both structural and attribute similarities are considered.

A new Attributed Graph Clustering by using Label Propagation in Complex Networks

The paper [65] proposes a novel method for community detection in complex networks with node attributes. The method is based on label propagation, a popular technique for clustering in unattributed networks, and extends it to incorporate node attributes.

The proposed method starts with randomly assigning each node to a cluster and then propagating the labels based on the similarity of node attributes. The authors introduce a new similarity measure that takes into account both the binary attribute values and their frequencies in the network. This measure is used to update the labels of nodes iteratively until convergence.

Continuous Encoding for Overlapping Community Detection in Attributed Network

The method [66] is based on a continuous encoding framework that combines structural information with attribute information to achieve better clustering results. The continuous encoding framework is based on a graph convolutional network (GCN) that is used to encode both structural and attribute information into a continuous feature space. The GCN takes as input the

adjacency matrix of the network, as well as a feature matrix that encodes the attributes of each node. The authors then use a continuous clustering algorithm to cluster the nodes based on their positions in the feature space.

One of the key advantages of this approach is that it can detect overlapping communities, which is often desirable in real-world networks where nodes can belong to multiple communities.

Overlapping Community Detection in Directed and Undirected Attributed Networks Using a Multiobjective Evolutionary Algorithm

The multiobjective evolutionary algorithm (MOEA) for overlapping community detection proposed by Zhang et al [67]. In this algorithm, a mixed representation-based approach is used to represent nodes as both individuals and groups and a multiobjective fitness function is used to optimize two objectives: community quality and community overlap.

The community quality objective is based on the modularity measure and an attribute similarity method, which measures the strength of community structure in a network. The community overlap objective measures the extent to which nodes belong to multiple communities. These objectives are combined into a single fitness function that is optimized using an MOEA.

The MOEA works using NSGA by generating a population of candidate solutions (i.e., communities) and iteratively improving the solutions through a process of selection, crossover, and mutation. During selection, individuals with higher fitness values are more likely to be selected for reproduction. Dur-

ing crossover, two parent communities are combined to create a new child community. During mutation, a randomly selected node is moved from one community to another to create a new solution.

Extension of Modularity Density for Overlapping Community Structure

A new measure called Modularity Density (MD) has been proposed in this paper [68], which allows nodes to belong to multiple communities simultaneously. The basic idea of MD is to assign a weight to each node that indicates the degree of its membership in different communities. The weight of a node is determined by the density of its connections within each community.

However, the original MD measure still has some limitations. For example, it does not take into account the degree distribution of the network, which can affect the community structure. To address these limitations, an extension of MD has been proposed, called Extended Modularity Density (EMD).

The EMD measure incorporates the degree distribution of the network into the calculation of community structure. It assigns a weight to each node based on its degree and the density of its connections within each community. The weight of a node is determined by the ratio of the number of edges within its community to the expected number of edges based on the degree distribution of the network.

Graph Neural Network Encoding for Community Detection in Attribute Networks

In this paper [69], the authors propose a novel community detection algorithm that uses a graph neural network (GNN) encoding to integrate both the structural and attribute information of nodes in the network. The proposed algorithm is based on the GA method, where the optimization is between a structural similarity metric derived from modularity and an attribute similarity method.

The proposed algorithm is evaluated on both synthetic and real-world datasets, and the results show that it outperforms several state-of-the-art community detection algorithms in terms of modularity score and computational efficiency. The authors also demonstrate the effectiveness of the proposed algorithm in detecting overlapping communities, which is a more challenging task than non-overlapping community detection.

Metrics for Community Analysis: A Survey

The paper [70] presents a comprehensive survey of various metrics used in the analysis of community structures in complex networks. The authors provide an in-depth review of various evaluation measures used to assess the quality of community detection algorithms. They also discuss various application domains where community analysis plays an important role.

The paper begins with an introduction to the concept of community structures in complex networks. Then the authors give a description of different types of metrics for non-overlapping and overlapping networks. The authors

also differentiate between many variations of Modularity. In overlapping CD, they mentioned different types of overlapping modularity and another metric named Flex.

The authors then present a detailed review of various evaluation measures used to assess the quality of community detection algorithms. These measures include modularity, conductance, coverage, homogeneity, and NMI. They also discuss the limitations of these measures and the need for more robust evaluation measures that can handle overlapping communities and community hierarchies.

Community detection in attributed networks considering both structural and attribute similarities: two mathematical programming approaches

The paper [71] proposes two mathematical programming approaches for community detection in attributed networks. The first approach involves formulating the community detection problem as a mixed integer linear programming (MILP) problem. The objective is to maximize the modularity of the resulting communities while ensuring that each node belongs to a single community. The constraints are formulated to ensure that the attribute similarities between nodes are considered in the clustering process.

The second approach is a bi-level programming formulation that considers both structural and attribute similarities between nodes. In this approach, the objective is to maximize the modularity of the communities while minimizing the difference between the attribute similarities of the nodes within each

community. The authors propose an efficient heuristic algorithm to solve this problem.

A Mixed Representation-Based Multiobjective Evolutionary Algorithm for Overlapping Community Detection

In this paper [62], Zhang et al. propose a mixed representation-based multiobjective evolutionary algorithm (Particle Swarm) for overlapping community detection in complex networks. The goal is to detect communities that have overlapping nodes, which can help reveal hidden connections between different groups in a network.

The proposed algorithm uses a mixed representation approach that combines both binary and real-valued encoding to represent the community structures that detect overlapping nodes in a network. This allows for a more flexible and accurate representation of the communities, which can improve the quality of the detected communities.

The algorithm is based on a multiobjective optimization framework that uses the particle swarm optimization technique and the Genetic algorithm's technique to create a new method, namely MR-MOEA. Modularity measures the density of links within communities compared to links between communities, while coverage measures the proportion of nodes that are assigned to communities.

Chapter 8

The Proposed Method

In this section, we begin by providing a brief overview of how a solution based on Genetic Algorithms (GA) can be developed for community detection problems. Following this, we offer a detailed explanation of our proposed method.

8.1 Genetic Algorithm on Community Detection Problem

Using Genetic Algorithms (GAs) for community detection is a fascinating approach. GAs are optimization algorithms inspired by the principles of natural selection and genetics. They work by iteratively improving a population of candidate solutions based on their fitness to the problem at hand.

In the context of community detection, the problem is typically formulated as finding a partition of nodes in a network into cohesive communities, where nodes within a community are densely connected while connections between communities are sparse. This is akin to optimizing a fitness function that measures both intra-community cohesion and inter-community separation.

GAs can be applied to this problem by representing potential community

partitions as individuals in a population. These individuals are encoded in a way that reflects the grouping of nodes into communities. The genetic operators such as selection, crossover, and mutation are then used to evolve this population over generations, to improve the fitness of the community partitions represented by the individuals.

One advantage of using GAs for community detection is their ability to explore a diverse range of potential solutions in parallel. This can be particularly useful in cases where the search space is complex and traditional optimization techniques may get stuck in local optima.

However, like any optimization algorithm, the effectiveness of GAs for community detection depends on various factors such as the choice of representation, the design of the fitness function, and the specific parameters of the algorithm. Additionally, GAs may suffer from issues such as premature convergence or scalability limitations when applied to large-scale networks.

Overall, the use of GAs for community detection represents an intriguing intersection of evolutionary computation and network analysis, offering a promising approach to addressing the challenges inherent in identifying cohesive communities within complex networks.

8.2 Single-objective and Multi-objectives

The majority of community detection algorithms rely on a single objective, which may not yield optimal solutions for intricate networks. [20] suggest that clusters of densely interconnected nodes can be identified as communities if

they fulfill the following four criteria.

- **The Mutuality of Ties:** For a subgroup to be considered a community, its members must exhibit mutual relationships among each other.
- **The Closeness or Reachability:** Within a subgroup, it is essential that at least one member is accessible from other members, whether they are adjacent or nonadjacent.
- **The Frequency of Ties:** The frequency of ties among the members of a subgroup should be high.
- **The relative frequency of ties:** The frequency of connections among members within a subgroup should surpass the frequency of connections between members and individuals outside the subgroup.

Single-objective methods indeed have limitations in capturing the complexity of community structures because they focus solely on optimizing one criterion, often resulting in a bias towards that specific criterion. For instance, methods optimizing solely for modularity might overlook other important properties like density or conductance.

Multi-objective optimization, on the other hand, allows for the simultaneous consideration of multiple criteria, offering a more nuanced understanding of community structures. By generating a diverse set of solutions, multi-objective methods provide users with a range of options, each representing a different trade-off between the various properties of interest.

For example, a multi-objective approach might produce solutions that trade-off between maximizing modularity and minimizing conductance. This allows

users to explore a spectrum of community structures and choose the one that best aligns with their priorities or domain-specific knowledge.

However, it's essential to note that the effectiveness of multi-objective methods relies heavily on how well the different objectives are defined and weighted. Additionally, navigating the resulting set of solutions to identify the most suitable one can be non-trivial and may require additional post-processing or decision-making techniques.

In summary, while both single-objective and multi-objective methods have their merits, multi-objective optimization offers a more comprehensive and flexible approach to community detection, particularly when dealing with complex and multi-faceted community structures.

8.3 Identification of Overlapping Nodes

Mathematically, a node v_k (where $v_k \in V$) is classified as either overlapping or non-overlapping based on whether it satisfies any of the following conditions, respectively:

$$v_k \in C_i \bigcap C_j, \forall\, i \neq i, j \in 1,2,..,n$$
$$v_k \notin C_i \bigcap C_j, \forall\, i \neq i, j \in 1,2,..,n$$

The approach outlined in [62] for detecting overlapping nodes involves a specific procedure. For each node v_i, a set of key neighboring nodes V_i^{KN} and a key neighboring subgraph G_i^{KN} are generated. The set V_i^{KN} comprises certain neighbors of v_i, chosen to maximize the number of common neighboring nodes with v_i. On the other hand, the key neighboring subgraph G_i^{KN} encom-

passes the key-neighboring node of v_i as well as the common neighboring nodes of v_i and V_i^{KN}.

A node v_i is detected as an overlapping node if it satisfies the following two conditions.

- The number of different key neighboring subgraphs of v_i should be greater than 1.
- There should be very few links between any two key neighboring subgraphs of v_i.

Consider two key neighboring subgraphs G_i^{KN} and G_j^{KN} within a graph G with an adjacency matrix A. The count of links or edges between G_i^{KN} and G_j^{KN} is denoted as $edges(G_i^{KN}, G_j^{KN}) = \sum_{m \in G_i^{KN}, n \in G_j^{KN}} A_{mn}$. The edge density, also referred to as Link Closeness, between these two subgraphs is defined by the following equation:

$$LC = max\left\{\frac{edge(G_i^{KN}, G_j^{KN})}{edge(G_i^{KN}, G_i^{KN})}, \frac{edge(G_i^{KN}, G_j^{KN})}{edge(G_j^{KN}, G_j^{KN})}\right\}$$

Consider the network illustrated in Figure 8.1 as an example. For node n_1, there is only one key neighboring subgraph consisting of nodes n_2, n_3, n_4. Since Condition 1 is not satisfied, n_1 is not considered a candidate overlapping node. This evaluation holds true for n_2 and n_3 as well. However, for node n_4, the key neighboring subgraph $G_1^{KN(4)}$, containing nodes n_1, n_2, n_3, is initially identified. Subsequently, upon removing nodes n_1, n_2, and n_3, along with their associated links from the network G, another key neighboring subgraph of n_4 ($G_2^{KN(4)}$) emerges, comprising nodes n_6, n_7, n_8.

Figure 8.1 outlines the primary procedure for determining whether n_4 qualifies as a candidate overlapping node. Since there are no edges between $G_1^{KN(4)}$ and $G_2^{KN(4)}$, the calculation for $edges(G_i^{KN}, G_j^{KN})$ yields $max\{\frac{0}{3}, \frac{0}{2}\} = 0$. This indicates that the link between $G_1^{KN(4)}$ and $G_2^{KN(4)}$ is sparse. Consequently, both conditions are satisfied, leading to the conclusion that n_4 indeed qualifies as an overlapping node.

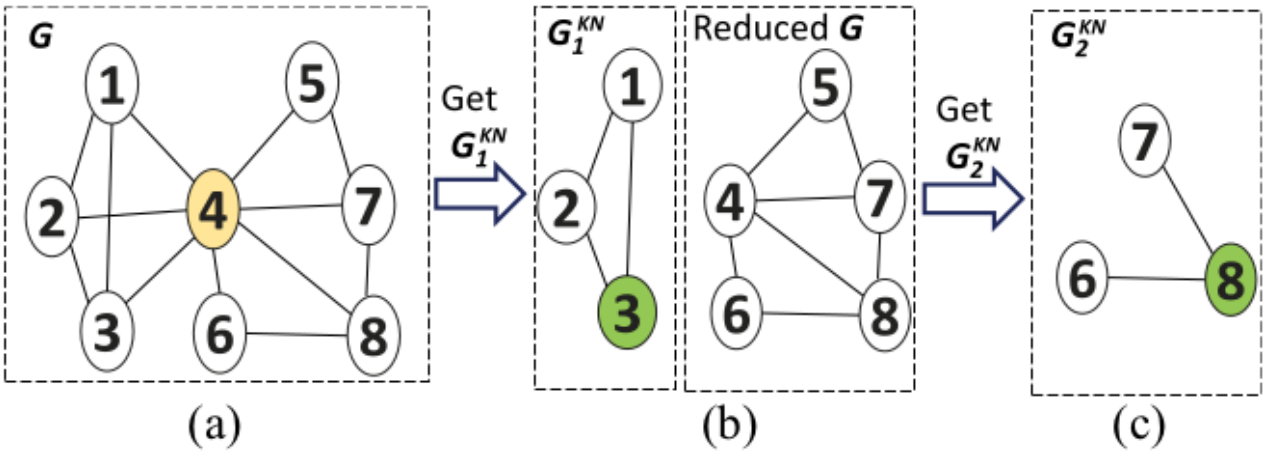


Figure 8.1: Procedure to determine whether node 4 is an overlapping candidate node [72]

8.4 Chromosome

The approach you've outlined involves a multi-stage process for community detection, with a particular focus on handling overlapping nodes. Let's break down each step:

- **Mixed Representation Technique:** This technique likely involves a method for representing nodes and their affiliations with communities in a mixed manner, where nodes can belong to multiple communities simultaneously. Using a dictionary-like data structure, where each key represents

a community ID and the corresponding value is a list of nodes affiliated with that community, allows for the representation of overlapping communities.

- **Identifying Overlapping Nodes:** Through the mixed representation technique, overlapping nodes, i.e., nodes belonging to multiple communities, are identified. These nodes play a crucial role in capturing the interconnectedness between different communities and can provide insights into the underlying structure of the network.

- **Delineating Discrete and Non-overlapping Communities:** Another algorithm is employed to delineate discrete and non-overlapping communities. This likely involves applying a community detection algorithm that aims to partition the network into separate groups of nodes, each representing a distinct community. This step helps in identifying communities that do not share nodes with other communities.

- **Integrating Overlapping Nodes into Communities:** Once discrete communities have been identified, the next step is to integrate the overlapping nodes into their respective communities. This amalgamation process involves assigning overlapping nodes to the communities they are most strongly affiliated with, based on some criteria such as connectivity or similarity.

The use of a chromosome-like data structure, where community IDs are keys and lists of affiliated nodes are valued, facilitates the representation and manipulation of communities throughout this process. This approach allows for a flexible and intuitive way of managing community assignments and handling overlapping nodes within the network.

Overall, this multi-stage process provides a systematic way of detecting and integrating overlapping communities into the broader framework of community detection, enhancing the richness and accuracy of the resulting community structure.

8.5 Objective Functions

The proposed community detection method is founded on a Multi-Objective Genetic Algorithm (MOGA), utilizing the NSGA-III framework. In this approach, four objective functions are employed to maximize intracommunity link density and minimize intercommunity link density. These objectives are detailed below:

- **Maximizing Intracommunity Link Density:**This objective aims to enhance the density of links within communities, fostering stronger internal cohesion.
- **Minimizing Intercommunity Link Density:**This objective strives to reduce the density of links between different communities, promoting segregation and distinctiveness among communities.

The description of the first objective function which is taken from the paper [61] has been given in the following equation:

$$f_1 = \sum_{j=1}^{|C|} \frac{\sum_{i=1}^{|C_j|} (fuzz_{ij} * intDeg_{ij})}{|C_j|}$$

where |C| is the number of communities in a network G = (V, E), consisting of a set of vertices V and a set of edges E. $fuzz_{ij}$ is the fuzzy membership value of the ith node of the jth community. ($intDeg_{ij}$) is the internal degree of ith node of jth community. $\left|C_j\right|$ is the size of jth community.

The description of the second objective function that is taken from the paper [61] has been given in the following equation:

$$f_2 = \sum_{j=1}^{|C|} \left[\frac{\sum_{i=1}^{|C_j|} (Sdeg_{ij} - SintDeg_{ij})}{|C_j|} \right]$$

$$Sdeg_{ij} = Deg(v_i) + clustCoeff(v_i) \qquad if\ node\ v_i \in community\ C_j$$

$$SintDeg_{ij} = intDeg_{ij} + intclustCoeff_{ij} \qquad if\ node\ v_i \in community\ C_j$$

where |C| is the number of communities in the network G. $Sdeg_{ij}$ is the sum of degree (Deg) and clustering coefficient of ith node of jth community. $SintDeg_{ij}$ is the sum of the internal degree and internal clustering coefficient of ith node of the jth community. The internal clustering coefficient of a node is defined as the clustering coefficient of a node within a community. $|C_j|$ denotes the number of members (nodes) in the jth community.

The first objective function f_1 is responsible for maximizing intracommunity link density which strongly favors the dense connections within the communities. The fuzzy membership value has been computed according to the following formula using internal degree and internal CC. The term $fuzz_{ij}$ is the fuzzy membership value of the ith node of the jth community which has been used as the weighting factor of this node in this objective function. Higher membership value indicates a higher degree of belongingness toward those communities.

$$fuzz_{ij} = \sum_{j=1}^{|C|} \sum_{i=1}^{|V|} \frac{\left(\frac{intDeg_{ij}}{Deg(v_i)} + \frac{intclustCoeff_{ij}}{clustCoeff_{ij}}\right)}{\sum_{k=1}^{|C|}\left(\frac{intDeg_{ik}}{Deg(v_i)} + \frac{intclustCoeff_{ik}}{clustCoeff(v_i)}\right)}$$

The second objective function f_2 has been used to minimize intercommunity link density. The numerator term is the difference between the sum of internal degree and internal CC from the sum of degree and CC. The degree difference gives a number of external links and the CC difference has been added as a penalty for breaking the neighbor's link.

The third objective function is taken to measure the attribute similarity between nodes. The formula for 3rd objective is given below:

$$S_A = \frac{\sum_{k=1}^{|C|} \sum_{i,j \in k, ij} 2S(i,j)}{\sum_{k=1}^{|C|} r_k(r_k - 1)}$$

$$S(i,j) = Tanimoto(i,j)$$

Tanimoto

Formula:

$$Sim_{Tanimoto}(A,B) = \frac{bothAB}{|A|+|B|-bothAB} = \frac{bothAB}{onlyA+onlyB+bothAB}$$

Range:

$[0.0, 1.0]$

Example:

A	1	0	1	1	1	0	0	0
B	1	1	0	1	1	0	1	0

$$\frac{bothAB}{onlyA+onlyB+bothAB} = \frac{3}{1+2+3} = \frac{3}{6} = 0.5$$

Figure 8.2: Tanimoto Similarity
[73]

8.6 Population Initialization

In a Genetic Algorithm (GA), the quality of the final results can depend on the initial population in several ways:

- Convergence rate: A good initial population can speed up the convergence rate of the algorithm toward the optimal solution. In contrast, a poor initial population can lead to slow convergence, requiring more generations to reach the optimal solution.

- Diversity: A diverse initial population can increase the exploration of the search space, potentially leading to better solutions. Conversely, a population with low diversity may converge prematurely to a suboptimal solution.

- Local optima: Depending on the problem, the search space may contain multiple local optima. A good initial population can help avoid getting stuck in a local optimum and increase the chances of finding the global optimum.

  In summary, the quality of the initial population can have a significant impact on the performance and quality of the final results obtained by the evolutionary process.

  Instead of randomly assigning nodes to communities, nodes are being assigned through a process:

- Choose the 1st node (initial) by checking which node has the highest dominance.

- CoreDominance = $\sum_{i=1}^{|V|}$extJaccard(node, i)

- extJaccard(node1, node2):

  a = adjacency_matrix(node1, node2)

  x = Graph.neighbors(node1) $\bigcap$ Graph.neighbors(node2)

  y = Graph.neighbors(node1) $\bigcup$ Graph.neighbors(node2)

  if a =1:

  return a/b

- Tanimoto(list1, list2):

  A = sum(list1[i]=1 and list2[i]=0)

  B = sum(list1[i]=0 and list2[i]=1)

  C=sum(list1[i]=1 and list2[i]=1)

  if $A+B+C=0$:

  return 0

  else:

  tanimoto_sim = $\frac{C}{A+B+C}$

- similarity(node1, node2):

  sim = c*extJaccard(node1, node2) + (1-c)*Tanimoto(list[node1],list[node2])

  where c is a hyperparameter

- Check the CoreDominance(node) for all nodes and chose the node (say n_i) with highest dominance.

- Check the similarity(node1, node2) between the chosen node and all other nodes and store values (if greater than a predefined threshold value) in a dictionary. Create a community with the nodes with the

highest similarity with the initially chosen one and after that temporarily remove those nodes (for which comparison is already done and in a community currently) from the list of nodes and follow the same process for the rest of the nodes until no node is left to put into a community.

- Then we've to do the process of MixedRepresentation mentioned earlier. With this process, we can find the overlapping nodes by calculating clustering_coefficient with those nodes and every community and lastly, we can put the overlapping nodes to their respective communities aside from those communities to which they already belong.

  We have experimentally shown that the application of similarity-based initialization enhances the quality of the chromosomes.

8.7 Genetic Operators

There are three genetic operators. These are selection, crossover, and mutation. The selection operator is based on the concept of survival of the fittest. It selects chromosomes from a population to generate a mating pool on which crossover and mutation operators are applied. The crossover operator provides low convergence and the mutation operator provides random changes in genes to explore the solution space.

Selection: In this study, a binary tournament selection method has been employed, where pairs of chromosomes are randomly selected to compete, and the winner is chosen to enter the mating pool. This approach not only introduces a level of randomness but also ensures that even chromosomes with lower fitness values have an opportunity to be included in the mating pool. To determine the winners in these tournaments, a ranking-based selection

mechanism has been utilized.

To handle situations where multiple chromosomes have identical rankings, a crowding distance metric has been applied. This metric helps maintain diversity within the population by resolving conflicts among chromosomes with similar performance.

Once the offspring chromosomes have been generated through mating, an elitist strategy inspired by NSGA-II (Non-dominated Sorting Genetic Algorithm II) [54] has been adopted. This strategy involves selecting individuals from both the parent and offspring chromosomes of the current generation. The selected individuals, totaling twice the population size (2N), are then used to construct a new population of chromosomes with a specified size (N) for the subsequent generation. This elitist approach aims to preserve the best-performing individuals across generations while allowing for diversity and exploration within the population.

Crossover: If a randomly generated number falls within the defined crossover probability range (another hyperparameter), the crossover process proceeds. A random crossover point is then selected within the chromosome, excluding the first and last positioned nodes. Subsequently, a one-point crossover operation is performed at the chosen crossover point.

To accommodate the resulting offspring chromosomes, an empty set is initialized to store pairs of output chromosomes. The updated child chromosome is added to this set, denoted as new_child.

This process iterates until the size of the set containing valid child chromosomes (new$_c hild) equals the size of the set containing parent chromosomes. During

Mutation: The mutation operator in this approach comprises two distinct procedures: random mutation and neighbor-based mutation. If a randomly generated number within the range of 0 to 1 meets the mutation probability criterion (another hyperparameter), the mutation process proceeds.

Firstly, a mutation position is randomly selected between the first and last node of a chromosome. If the node at the chosen position is identified as a potential overlapping node, it exclusively undergoes random mutation.

However, if the selected node is determined to be a non-overlapping node, it undergoes neighbor-based mutation. This entails examining whether the selected node has non-overlapping neighbors. If the number of non-overlapping neighbors is greater than 0, the non-overlapping node's label is replaced with the most frequent label among its non-overlapping neighbors. Otherwise, if the node has no non-overlapping neighbors, its label is substituted with a value within the range of 1 to n, where n represents the current number of communities. This approach ensures both random and structured mutations occur, catering to both overlapping and non-overlapping nodes within the chromosome.

8.8 Algorithm

- **Input:**
  1. A_i, i = 1 to N: adjacency matrix of graph G
  2. simC: a static variable to control the participation of structural similarity and attribute similarity
  3. ec = a static variable to control the quantity of overlapping nodes' number
- **Output:** a set of solutions in order
- **Steps:**
  1. Start the initialization process on the graph by calculating and comparing the similarity value among the nodes.
  2. Construct a dictionary to track identified communities.
  3. Execute the Mixed-Representation technique for detecting overlapping nodes.
  4. Assign overlapping nodes to appropriate communities based on Clustering Coefficient comparisons.
  5. Complete the initial population setup.
  6. Calculate Fuzzy Membership values for each node corresponding to their communities.
  7. Implement the three specified objectives.
  8. Employ NSGA-II to optimize the objectives on the initial population.
  9. Extract solutions and evaluate them using appropriate metrics (differentiating between data with and without ground truth).

The above-mentioned algorithm is used in our proposed method for community detection. In this process, an adjacency matrix of the graph is taken as the input. We have also taken two static variables, to control the participation of structural similarity and attribute similarity and to control the number of overlapping nodes. After this process, we get a set of solutions in order.

1. **Input and Setup**:
    - We begin by taking the adjacency matrix of the graph as input.
    - Two static variables are utilized to control the participation of structural and attribute similarity and to manage the number of overlapping nodes, respectively.
2. **Similarity Comparison**:
    - The algorithm commences by comparing the similarity values among the attributes of nodes using the Tanimoto method.
3. **Community Detection**:
    - A dictionary is created to store detected communities, where each community ID serves as a key and a list containing the nodes belonging to that community is the corresponding value.
4. **Overlapping Node Detection**:
    - Utilizing the Mixed-Representation method, overlapping nodes are identified based on network structure, discerning nodes belonging to multiple communities.
5. **Integration of Overlapping Nodes**:

- Detected overlapping nodes are inserted into appropriate communities by comparing their Clustering Coefficients.

6. **Initial Population Setup**:

   - The initial population is prepared for further processing.

7. **Fuzzy Membership Computation**:

   - Fuzzy membership values for each node are computed by evaluating their internal degree and internal clustering coefficient.

8. **Objective Function Implementation and Optimization**:

   - The three objective functions—maximizing intra-community density, inter-community density, and assessing node-attribute similarity—are utilized. These functions leverage the Tanimoto similarity method for attribute comparison.
   - Next, the NSGA-II framework is employed to optimize the objective functions. Solutions are ranked using non-dominated sorting and crowding distance. Subsequently, genetic operations like selection, crossover, and mutation are applied to generate subsequent populations with no duplicate community sets. The newly generated child population is merged with the parent population. The combined population, double the size of the parent population, is ranked using non-dominated sorting and crowding distance. The termination criterion is based on a pre-specified number of iterations.

9. **Solution Evaluation and Selection**:

   - The best solution from the first Pareto front, characterized by the largest density and Extended Modularity value, is obtained.

This structured approach ensures a comprehensive and systematic exploration of community structures within the given graph, ultimately yielding high-quality solutions. By integrating various techniques such as similarity comparison, community detection, and optimization with NSGA-II, our method effectively captures the intricate relationships and attributes present in the network. The careful consideration of overlapping nodes, along with the incorporation of fuzzy membership computation, further enhances the accuracy and robustness of the detected communities. Moreover, the utilization of objective functions addressing intra-community density, inter-community density, and node-attribute similarity ensures a well-rounded evaluation of community quality. Overall, this method provides a reliable framework for community detection tasks, offering insights into the underlying structure and dynamics of complex networks.

Chapter 9

Data Collection and Preparation

9.1 Introduction

The section on Data Collection and Preparation in a project report outlines the process of acquiring, organizing, and preparing the data for analysis. It offers insights into how the data was sourced, collected, and ensured to be suitable for analysis. This section typically includes:

1. **Data Sources**: Determining the origin of the data used to initialize a graph depends on the context and purpose of the graph. Here are some common scenarios:

   (a) **Primary Sources:**If the graph is based on data collected through surveys, experiments, observations, or other direct methods, then the data originates from primary sources. For example, if you're creating a social network graph, the data might come from surveys where individuals list their connections with others.

   (b) **Secondary Sources:**If the graph is based on publicly available datasets, research papers, existing databases, or other pre-existing sources, then the data originates from secondary sources. For instance, if you're creating a graph of academic collaborations, you might use

data from publicly available publications or databases like PubMed or Google Scholar.

(c) **Mixed Sources:**Often, graphs can be initialized using a combination of primary and secondary sources. For instance, if you're building a graph to analyze customer interactions on a website, you might collect primary data through user interactions while also augmenting it with secondary data from sources like web analytics tools or publicly available demographic data.

(d) **Simulated Data:**In some cases, especially during the development and testing phase of a project, simulated data might be used to initialize the graph. This data is generated artificially to mimic real-world scenarios and can be useful for prototyping and testing algorithms.

(e) **Sensor Data:**In scenarios involving IoT (Internet of Things) or sensor networks, data collected from sensors can be used to initialize the graph. This data typically originates from physical sensors measuring various parameters like temperature, humidity, pressure, etc.

Regardless of the data source, it's crucial to ensure that the data used to initialize the graph is reliable, relevant, and representative of the underlying phenomenon or system being modeled. Additionally, ethical considerations such as data privacy and consent should always be taken into account when collecting or using data from any source.

2. **Data Collection Methods:** Describe the methods and techniques used to collect the data. This could involve surveys, interviews, web scraping, data acquisition from APIs, laboratory experiments, or other relevant

approaches.

(a) **Surveys**: Surveys involve collecting data by asking questions directly to individuals or groups. Surveys can be conducted in various formats, including online surveys, paper-based surveys, or interviews. Responses can provide valuable information for populating attributes or relationships in a graph.

(b) **Interviews**: Interviews involve one-on-one or group discussions with participants to gather qualitative or quantitative data. Interviews can be structured, semi-structured, or unstructured, depending on the level of flexibility needed in collecting data.

(c) **Web Scraping**: Web scraping involves extracting data from websites. This technique is useful for collecting large amounts of data from online sources such as social media platforms, forums, news articles, or any website with relevant information. Web scraping tools and libraries can automate the process of collecting and parsing data from web pages.

(d) **Data Acquisition from APIs**: Many online services provide APIs (Application Programming Interfaces) that allow developers to access their data programmatically. By interfacing with these APIs, you can retrieve structured data from platforms such as social media networks, financial databases, weather services, and more.

(e) **Laboratory Experiments**: In scientific research or experimental settings, data can be collected through controlled laboratory experiments. This method allows researchers to manipulate variables and observe their effects on outcomes, providing valuable insights for

graph initialization in fields such as biology, psychology, or physics.

(f) **Observational Studies**: Observational studies involve observing and recording behaviors, events, or phenomena as they naturally occur. This method is often used in fields such as sociology, anthropology, or ecology to collect data on interactions, patterns, or trends in real-world settings.

(g) **Data Fusion**: Data fusion involves integrating information from multiple sources to create a comprehensive dataset. This can include combining data from surveys, sensor networks, public databases, or other sources to enrich the graph with diverse and complementary information.

(h) **Simulations**: In certain scenarios, especially in complex systems or theoretical research, data may be generated through simulations or computational models. Simulations allow researchers to explore hypothetical scenarios, generate synthetic data, or test algorithms under controlled conditions.

The choice of data collection method depends on factors such as the research objectives, available resources, ethical considerations, and the nature of the data being collected. It's important to select methods that are appropriate for the context and ensure that data collection processes adhere to ethical standards and legal requirements.

3. **Data Variables:** Provide an overview of the variables or features present in the dataset. Explain their meanings, types (categorical, numerical), and any transformations or encoding performed on them.

(a) **Node ID**: This variable represents a unique identifier for each node in

the graph. It is typically a numerical or alphanumeric value assigned to uniquely identify each node. This variable is categorical.

(b) **Node Attributes**:

- **Text Features**: These variables represent textual attributes associated with each node, such as node description, content, or labels. Text features are categorical.
- **Numerical Features**: These variables represent numerical attributes associated with each node, such as node size, importance, or weight. Numerical features are numerical.

(c) **Edge Weight**:

- This variable represents the strength or weight of the relationship between pairs of nodes connected by an edge. It can be a numerical value indicating the intensity or significance of the relationship. Edge weights are numerical.

(d) **Edge Type**:

- This variable represents the type or nature of the relationship between pairs of nodes connected by an edge. It can be categorical, indicating different types of relationships such as "friendship," "collaboration," "interaction," etc.

(e) **Timestamp** (if applicable):

- This variable represents the timestamp or time of occurrence associated with edges or nodes in temporal graphs. It can be numerical or datetime type, depending on the format of the timestamp data.

Transformations or encoding performed on these variables may include:

- **Normalization/Standardization**: Numerical features may be normalized or standardized to ensure they have a similar scale, which can help improve the performance of certain machine learning algorithms.
- **One-Hot Encoding**: Categorical variables may be encoded using one-hot encoding to represent them as binary vectors, which can be easier for machine learning algorithms to process.
- **Text Processing**: Text features may undergo preprocessing steps such as tokenization, stemming, stop-word removal, or vectorization (e.g., using techniques like TF-IDF or word embeddings) to convert them into numerical representations suitable for analysis.
- **Feature Engineering**: New features may be derived from existing ones through techniques such as feature extraction or dimensionality reduction to capture additional information or reduce the complexity of the dataset.

The choice of transformations or encoding depends on the specific requirements of the analysis and the algorithms being used. It's essential to carefully preprocess and encode the variables to ensure that they are suitable for the intended analysis tasks.

4. **Data Sampling:** If the data was collected through sampling, the following sampling methodology was employed:

   (a) **Sample Size**: The sample size refers to the number of observations or data points included in the sample. It was determined based on considerations such as the desired level of statistical power, confidence interval width, and feasibility constraints.

(b) **Sampling Technique**:

- **Random Sampling**: Every member of the population has an equal chance of being selected for the sample. This helps ensure that the sample is representative of the population and reduces the risk of selection bias.
- **Stratified Sampling**: The population was divided into subgroups or strata based on certain characteristics (e.g., age, gender, region), and samples were randomly selected from each stratum. This ensures adequate representation of each subgroup.
- **Cluster Sampling**: The population was divided into clusters or groups, and entire clusters were randomly selected to be included in the sample. This approach is efficient for large-scale surveys or geographically dispersed populations.
- **Systematic Sampling**: Individuals were selected from a population list at regular intervals. This method is simple and efficient but may introduce bias if there is a systematic pattern in the ordering of the population list.

(c) **Considerations for Representativeness**: Efforts were made to ensure that the sample is representative of the target population in terms of relevant variables (e.g., demographics, geographic distribution, behavior patterns). Sources of bias such as non-response bias, coverage bias, or sampling frame errors were minimized.

By employing an appropriate sampling methodology, the collected data aims to provide reliable and meaningful insights into the characteristics and relationships within the population of interest.

5. **Data Preprocessing:** Outline the steps taken to preprocess the data before analysis. This may include cleaning the data to remove errors, missing values, or outliers, as well as standardizing or normalizing variables, handling categorical variables, or addressing any other data quality issues.

   Before analysis, the following steps were taken to preprocess the data:

   (a) **Data Cleaning**:

   - **Handling Missing Values**: Missing values were addressed using techniques such as imputation, deletion, or advanced imputation methods like K-nearest neighbors (KNN) imputation.
   - **Error Detection and Correction**: Errors in the data, such as inconsistencies, outliers, or inaccuracies, were identified and corrected through manual inspection or automated algorithms.

   (b) **Handling Outliers**: Outliers were identified using statistical methods like Z-score, IQR, or visualization techniques such as box plots, and treated by either removing them, transforming them, or treating them separately depending on the context of the analysis and domain knowledge.

   (c) **Standardization or Normalization**: Numerical variables were standardized or normalized to ensure they have a similar scale, which can help improve the performance of certain machine learning algorithms and facilitate model convergence.

   (d) **Handling Categorical Variables**: Categorical variables were encoded into numerical values using techniques such as one-hot encoding, label encoding, or ordinal encoding, depending on the

nature of the categorical variables and the requirements of the analysis.

(e) **Feature Engineering**: New features were derived from existing ones through techniques such as feature extraction, dimensionality reduction, or transformation to capture additional information or reduce the complexity of the dataset.

(f) **Data Integration and Aggregation**: Data from multiple sources were integrated and aggregated to create a comprehensive dataset suitable for analysis. This involves merging datasets, aggregating data at different levels of granularity, or combining related variables.

(g) **Data Splitting**: The dataset was split into training, validation, and testing sets to evaluate the performance of machine learning models effectively. This ensures that the model's performance generalizes well to unseen data and helps prevent overfitting.

By preprocessing the data using these steps, the dataset was prepared for analysis, ensuring that it is clean, accurate, and suitable for use in statistical modeling, machine learning, or other analytical tasks.

6. **Data Integration:** If multiple datasets were used, the following process was employed to combine or integrate them:

(a) **Identification of Common Variables or Keys**: Common variables or keys across the multiple datasets were identified. These variables serve as the basis for matching or merging the datasets.

(b) **Data Cleaning and Preprocessing**: Each dataset underwent data cleaning and preprocessing steps to ensure consistency, accuracy,

and compatibility. This involved handling missing values, addressing outliers, standardizing or normalizing variables, and encoding categorical variables.

(c) **Matching or Merging Datasets**: The datasets were matched or merged based on common variables or keys identified in the first step. Different merging techniques were employed, including inner join, outer join, left join/right join, or concatenation.

(d) **Handling Data Conflicts or Inconsistencies**: In cases where there were conflicts or inconsistencies between datasets, appropriate resolutions were applied. This may involve data transformation, standardization, or harmonization to ensure consistency across the integrated dataset.

(e) **Data Validation and Quality Assurance**: After integration, the integrated dataset underwent validation and quality assurance checks to ensure that the merging process was performed accurately and that the integrated dataset meets the requirements of the analysis. This involves verifying the correctness of merged data, checking for data completeness, and assessing the overall data quality.

By following this process of data integration, the multiple datasets were effectively combined or integrated to create a comprehensive dataset suitable for analysis, providing a more complete and informative picture of the underlying phenomenon or system.

7. **Data Validation:** The following measures were taken to validate the data and ensure its quality:

(a) **Data Verification**: Each dataset underwent thorough verification to

ensure its accuracy and reliability. Cross-checking was performed against multiple independent sources, and data entry correctness was verified. Anomalies or outliers indicating data quality issues were identified and addressed.

(b) **Cross-Checking**: Cross-checking involves comparing data points within the same dataset and across different datasets to identify inconsistencies or discrepancies. This process helped detect errors or inconsistencies overlooked during data collection or entry, and validated relationships between variables.

(c) **Assessment of Data Sources**: The reliability and credibility of data sources were assessed to ensure the collected data was trustworthy and accurate. This involved evaluating the reputation, credibility, and expertise of data providers or sources, and scrutinizing data sources for biases, limitations, or errors.

(d) **Data Completeness and Consistency**: Data completeness was assessed to ensure all required variables and observations were present. Missing data were addressed using appropriate techniques such as imputation or deletion. Consistency checks were performed to ensure data values were consistent across different variables and observations.

(e) **Documentation and Metadata**: Comprehensive documentation and metadata were maintained throughout the data validation process, providing transparency and traceability. This included documenting data sources, collection methods, preprocessing steps, and any assumptions or decisions made during the validation process. Metadata such as variable definitions and data quality indicators

were included to facilitate understanding and interpretation of the dataset.

(f) **Iterative Validation Process**: Data validation was an iterative process, with multiple rounds of checks and revisions to ensure the quality and reliability of the dataset. Feedback from domain experts or stakeholders was sought to validate assumptions, identify potential issues, and refine the validation process accordingly.

By implementing these measures, the data validation process aimed to ensure the accuracy, reliability, and integrity of the dataset, thereby enhancing the trustworthiness and usability of the data for analysis and decision-making purposes.

8. **Ethical Considerations:** The following ethical considerations were addressed during the data collection process:

(a) **Informed Consent**: Obtaining informed consent from participants was prioritized to ensure that individuals were aware of the purpose of data collection, how their data would be used, and any potential risks involved. Participants were provided with clear and understandable information about the study, including its objectives, procedures, risks, benefits, and their rights regarding data use and confidentiality. They were given the opportunity to provide consent voluntarily, without coercion or undue influence.

(b) **Data Privacy and Confidentiality**: Protecting data privacy and confidentiality was of utmost importance to maintain trust and respect for individuals' rights. Measures were implemented to safeguard

sensitive information and prevent unauthorized access, use, or disclosure of personal data. Data anonymization or de-identification techniques were employed to remove or encrypt personally identifiable information from the dataset, while access controls, encryption, and secure storage methods were implemented to protect data throughout its lifecycle.

(c) **Compliance with Regulations and Guidelines**: Efforts were made to ensure compliance with relevant regulations, laws, and guidelines governing data collection, such as the General Data Protection Regulation (GDPR), Health Insurance Portability and Accountability Act (HIPAA), or other industry-specific regulations. Researchers familiarized themselves with applicable ethical standards and institutional policies governing research involving human subjects, ensuring that data collection practices adhered to ethical principles of fairness, transparency, respect for autonomy, and beneficence.

(d) **Minimizing Harm and Bias**: Steps were taken to minimize potential harm or negative consequences to participants, including risks of harm to privacy, confidentiality, reputation, or psychological well-being. Data collection procedures were designed to mitigate risks and prioritize participant welfare. Researchers were mindful of potential biases introduced during data collection, analysis, or interpretation, and took steps to minimize bias, ensure objectivity, and promote inclusivity and diversity in research practices.

(e) **Transparency and Accountability**: Transparency in data collection practices involved openly communicating with participants and stakeholders about the purposes, methods, and outcomes of the

research. Researchers provided clear explanations of data collection processes, data handling practices, and any potential uses or disclosures of data. Accountability involved taking responsibility for ethical conduct throughout the research process, including data collection, analysis, reporting, and dissemination. Researchers upheld ethical standards, addressed concerns or complaints, and were prepared to justify their decisions and actions.

By addressing these ethical considerations, researchers upheld principles of respect, integrity, and responsibility in data collection practices, ensuring that research was conducted ethically and responsibly while respecting the rights and dignity of participants.

9. **Data Storage and Management:** The following methods are generally employed to store and manage data throughout the project:

   (a) **Data Storage Infrastructure**: Data was stored on a dedicated server hosted in a secure data center or cloud environment. This infrastructure provided robust storage capabilities, including high availability, scalability, and data redundancy to ensure data integrity and accessibility.

   (b) **File Formats**: Data files were stored in standardized file formats suitable for the type of data being collected and analyzed. Common formats such as CSV, JSON, or Parquet were used for structured data, while binary formats like HDF5 or Apache Avro were used for more complex or hierarchical data structures.

   (c) **Database Systems**: For structured data requiring relational database management, a relational database management system (RDBMS)

such as PostgreSQL, MySQL, or Microsoft SQL Server was utilized. NoSQL databases such as MongoDB, Cassandra, or Apache HBase were used for large-scale or distributed data processing.

(d) **Version Control Techniques**: Version control systems such as Git or Subversion (SVN) were employed to manage changes to code, scripts, documentation, and other project artifacts. Data versioning techniques may have been implemented to track changes to datasets over time, ensuring reproducibility and traceability.

(e) **Access Control and Security**: Access control mechanisms were implemented to restrict access to data based on roles, permissions, and authentication credentials. Data encryption techniques, secure transmission protocols, and data masking or obfuscation methods were used to enhance data security and protect against unauthorized access.

(f) **Backup and Disaster Recovery**: Regular data backups and disaster recovery procedures were implemented to safeguard against data loss or corruption. Automated backup schedules, off-site backups, and redundant storage configurations were used to ensure data resilience and continuity of operations.

By implementing these data storage and management practices, the project ensured that data was stored securely, managed efficiently, and accessible for analysis, collaboration, and decision-making purposes throughout its lifecycle.

10. **Data Documentation:** Provide information on data documentation, including metadata, data dictionaries, or codebooks that describe the

variables, their definitions, and any relevant information for interpreting the data. The following data documentation practices are:

(a) **Metadata**: Metadata provided essential information about the dataset, including its origin, purpose, creator, contributors, creation date, and update frequency. It also described data attributes such as variable names, data types, units of measurement, and any preprocessing steps applied to the data. Metadata facilitated data discovery, interpretation, and reuse by providing users with the necessary context and understanding of the dataset's characteristics and limitations.

(b) **Data Dictionaries**: A data dictionary was created to define the variables or fields present in the dataset and provide detailed descriptions of their meanings, data types, and permissible values. Each variable was documented in the data dictionary, including its name, label, description, data type, measurement scale, and any relevant codes or categories. Data dictionaries helped users understand the structure and semantics of the dataset, enabling accurate interpretation and analysis.

(c) **Codebooks**: A comprehensive codebook was developed to provide detailed information about the variables, data collection procedures, coding schemes, and any relevant documentation associated with the dataset. The codebook included descriptions of the study design, sampling methods, data collection instruments, variable definitions, data cleaning procedures, and any derived variables or transformations applied to the data. Codebooks served as a reference manual

for understanding the dataset and conducting analyses, documenting the entire data lifecycle and providing insights into the dataset's structure, quality, and reliability.

(d) **Data Documentation Standards**: Standardized formats and conventions were adopted for data documentation to ensure consistency, interoperability, and usability across different datasets and domains. Common standards such as Data Documentation Initiative (DDI), Dublin Core Metadata Initiative (DCMI), and Metadata Object Description Schema (MODS) were followed to describe and organize metadata, data dictionaries, and codebooks in a standardized and interoperable manner.

By providing comprehensive data documentation, including metadata, data dictionaries, and codebooks, the dataset became more accessible, interpretable, and usable for researchers, analysts, and other stakeholders, facilitating informed decision-making and promoting data transparency and reproducibility.

We have used several famous existing datasets and data scraping methods in our research.

9.2 Description about the Data

Table 9.1: Benchmark Datasets

Datasets used in the Experimentation						
Network	Type	# Nodes	# Edges	# Attributes	Ground Truth	# Classes
WebKB Cornell	Citation	195	304	1703	Available	5
WebKB Texas	Citation	187	325	1703	Available	5
Ego 0	Social Network	347	5038	224	Not Available	—
Ego 3437	Social Network	85	452	52	Not Available	—
Ego 686	Social Network	55	145	42	Not Available	—

A subset of WebKB dataset [74] consisting of four subnetworks from two U.S. universities: Cornell and Texas are used. The attribute dimension of all two networks: Texas and Cornell is 1703, which represent the web pages and hyperlinks between them.

Ego Facebook networks are a series of friendship networks. They are chosen from 3 ego networks, consisting of 487 users. The attribute dimension of all networks ranges from 42 and 224.

Chapter 10

Experimental Results

10.1 Evaluation Metrics

To compare the performances of the compared algorithms, the following metrics, including density, Extended modularity (Q_{ov}), normalized mutual information (NMI), and F1 measure are used. The density metric is applied to measure the community detection performance on networks without ground truth, while gNMI and F1 measure are for networks with true labels, and modularity is applied for both types of networks.

10.1.0.0.1 Density The density D of a network is defined as:

$$D = \sum_{k=1}^{c} \frac{l_k}{L}$$

where c is the number of communities, l_k is the number of edges within community k and L represents the total number of edges in the network. The larger D, the more distinct the community structure in the network is.

10.1.0.0.2 Extended modularity

It is one of the frequently used metrics for measuring the quality of the communities. The definition of extended modularity [75] has been given in the

following equation:

$$Q_{ov} = \frac{1}{2m}\sum_{i=1}^{|C|}\sum_{v\in C_i, w\in C_i}\frac{1}{O_v O_w}\left[A_{vw} - \frac{d_v d_w}{2m}\right]$$

where A is the adjacent matrix, C_i denotes a community, m is the number of edges, O_v is the number of communities that node v belongs to, and d_v is the degree of the node v. The larger the value of Q_{ov}, the better the result of overlapping community structure detected.

10.2 Comparison Between Different Methods

The compared algorithms are CEMOV and $MOEA-SA_{ov}$ and our method in terms of Density.

Table 10.1: Experimental Results in Terms of Density

Network	$MOEA-SA_{ov}$	CEMOV	Our Method
Ego 0	0.919	1.133	0.932
Ego 3437	0.678	1.211	0.911
Ego 686	0.752	1.510	0.802
WebKB Cornell	0.953	1.231	0.894
WebKB Texas	0.884	1.146	0.866

MR-MOEA and EMOFM-DK and our method are compared in terms of Extended Modularity [Q_{ov}].

Table 10.2: Experimental Results in Terms of Ext. Modularity

Network	MR-MOEA	EMOFM-DK	Our Method
Ego 0	0.4110	0.4158	0.4584
Ego 686	0.421	0.4228	0.4550
Ego 3437	0.4224	0.4245	0.4910

In this work, our proposed method has generated better results than other methods on Facebook Ego Networks (Ego 0, Ego 686, Ego 3437) in terms of Extended Modularity shown in the Table 10.2. The largest value of Q_{ov} has been

obtained by our method among all three methods for all three datasets. The result presented in Table 10.2 proves that our proposed method outperforms the others in terms of Q_{ov} value.

Table 10.1 shows the experimental result in terms of density. It may be noted that the largest value of density has been obtained by our proposed method is 0.932. However, the result using our proposed method has been ranked second after CEMOV, while our proposed method shows better results than $MOEA-SA_{ov}$ for all five datasets.

In our research, we've devised a community detection approach centered on a multi-objective evolutionary algorithm, with NSGA-II as the underlying framework. Density and Extended Modularity serve as metrics for validating the identified communities. We've formulated three distinct objective functions for this purpose. Our methodology incorporates two primary strategies for community detection: leveraging network structure and node attributes. By integrating both approaches, we aim to enhance the accuracy of community detection. Notably, our method draws upon the insights of overlapping node detection, as outlined in the reference [76], which heavily relies on network structure. Additionally, the computation of fuzzy membership values is grounded in node attribute similarity. To initialize the population, we adopt a similarity-based strategy, while a neighbor-based approach is employed for crossover, mutation, and update operations. Collectively, these techniques enable us to effectively capture the underlying network structure and generate solutions that closely approximate the actual ground truth, particularly in scenarios where such ground truth information is available.

Chapter 11

Conclusion and Scope for Future Works

Community detection in node-attributed networks has garnered considerable attention, driven by the growing influence of social networks and various attributed networks. The prevalence of data from sources such as social media, citations, research, and other relational data underscores the importance of extracting meaningful patterns from network data. In our project, we have undertaken the challenging task of community detection while optimizing objectives within the NSGA-II framework.

Our project focuses on leveraging this optimization framework for community detection on benchmark datasets. The optimization process involves enhancing the accuracy of node labeling and community detection. While we acknowledge that claiming superiority over the best models in the field today may be premature, we are proud to assert that our model demonstrates significant efficacy compared to many existing models.

The NSGA-II framework serves as a robust foundation for our project, enabling the exploration and identification of optimal solutions within the complex landscape of node-attributed networks. By applying our model to benchmark

datasets, we have observed noteworthy success in accurately delineating communities and labeling nodes. The performance achieved underscores the potential and effectiveness of our approach in addressing the challenges posed by real-world attributed networks.

In conclusion, our project represents a meaningful contribution to the evolving field of community detection in node-attributed networks. While we remain humble about surpassing state-of-the-art models, the demonstrated accuracy and effectiveness of our model position it as a viable and competitive solution in the landscape of community detection methodologies.

The future trajectory of our project unfolds with ambitious goals aimed at further enhancing its capabilities. One key avenue of exploration involves delving into link prediction and subgraph prediction, extending the scope beyond community detection. This expansion will not only contribute to the richness of the model but also cater to a broader range of applications within network analysis.

A paramount objective on our roadmap is the pursuit of increased accuracy. We aim to fine-tune and optimize the model to deliver even more precise results. This entails refining the underlying algorithms, exploring advanced optimization techniques, and incorporating insights gained from ongoing research in the field.

Transitioning from the NSGA-II framework to the NSGA-III framework represents a pivotal step in our evolution. This shift is envisioned to bring about

improvements in the model's performance, efficiency, and the quality of solutions generated. By leveraging the advancements introduced in NSGA-III, we anticipate a more robust and adaptive framework for our optimization tasks.

Furthermore, our future endeavors include expanding the training phase to incorporate diverse and extensive node-attribute datasets. Platforms such as Twitter, YouTube, Citeseer, and others offer rich and varied data landscapes. By integrating these datasets into our training regimen, we aim to enhance the model's versatility and generalization, ensuring its efficacy across a spectrum of real-world scenarios.

In essence, the future trajectory of our project is marked by a commitment to innovation and refinement. Through the exploration of new objectives, frameworks, and datasets, we envision a project that continues to evolve, staying at the forefront of advancements in network analysis and optimization.

Bibliography

[1] Jinrong Zhu, Bilian Chen and Yifeng Zeng. “Community detection based on modularity and k-plexes”. In: *Information Sciences* 513 (2020), pp. 127–142. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2019.10.076. URL: https://www.sciencedirect.com/science/article/pii/S0020025519310345.

[2] M. E. J. Newman and M. Girvan. “Finding and evaluating community structure in networks”. In: *Phys. Rev. E* 69 (2 Feb. 2004), p. 026113. DOI: 10.1103/PhysRevE.69.026113. URL: https://link.aps.org/doi/10.1103/PhysRevE.69.026113.

[3] Mark Müller-Linow, Claus Hilgetag and Marc-Thorsten Hütt. “Organization of Excitable Dynamics in Hierarchical Biological Networks”. In: *PLoS computational biology* 4 (Oct. 2008), e1000190. DOI: 10.1371/journal.pcbi.1000190.

[4] Shanon Hong. “An Introduction to Graph Partitioning Algorithms and Community Detection”. In: *Towards Data Science* (Oct. 2022). URL: https://towardsdatascience.com/an-introduction-to-graph-partitioning-algorithms-and-community-detection-29e7c962d10e.

[5] TANG Fengqin DING Wenwen. “Community detection based on spectral clustering with node attributes”. In: *Journal of University of Science and Technology of China* 48.2, 162 (2018), pp. 162–173.

[6] Chaitanya Reddy Patlolla. “Understanding the concept of Hierarchical clustering Technique”. In: *Towards Data Science* (Dec. 2018). URL: https://towardsdatascience.com/understanding-the-concept-of-hierarchical-clustering-technique-c6e8243758ec.

[7] Hans H. Bock. “Probabilistic models in cluster analysis”. In: *Computational Statistics Data Analysis* 23.1 (1996). Classification, pp. 5–28. ISSN: 0167-9473. DOI: https://doi.org/10.1016/0167-9473(96)88919-5. URL: https://www.sciencedirect.com/science/article/pii/0167947396889195.

[8] D. Jin et al. “A Survey of Community Detection Approaches: From Statistical Modeling to Deep Learning”. In: vol. 35. 02. Los Alamitos, CA, USA: IEEE Computer Society, Feb. 2023, pp. 1149–1170. DOI: 10.1109/TKDE.2021.3104155.

[9] Thamindu Dilshan Jayawickrama. “Community Detection Algorithms”. In: *Towards Data Science* (Jan. 2021).

[10] Narsingh Deo. *Graph Theory With Applications To Engineering And Computer Science*. Prentice-Hall, 1979.

[11] Smasongarrison. “Directed Graph Example”. In: *Wikipedia* (Oct. 2022).

[12] Mrmw. “Unirected Graph Example”. In: *Wikipedia* (Apr. 2021).

[13] Mrmw. “Weighted Graph Example”. In: *Wikipedia* (Apr. 2021).

[14] Tom Donovan. “Dijkstra’s Algorithm in an Undirected Graph”. In: *The Startup* (Feb. 2020).

[15] T. Piesk. “Simple bipartite graph”. In: *Wikipedia* (Aug. 2022).

[16] hridyesh bisht. “Basics of Graphs”. In: *Dev.to* (Jan. 2021).

[17] Eric W Weisstein. “Complete Bipartite Graph”. In: *MathWorld–A Wolfram Web Resource* ().

[18] Valery A. Kalyagin Ilya Bychkov. *Network Algorithms, Data Mining, and Applications*. Springer Cham, 2020. ISBN: 978-3-030-37156-2.

[19] Valery A. Kalyagin Ilya Bychkov. *Models, Algorithms, and Technologies for Network Analysis*. Springer Cham, 2017. ISBN: 978-3-319-56828-7.

[20] Stanley Wasserman and Katherine Faust. *Social Network Analysis: Methods and Applications*. Structural Analysis in the Social Sciences. Cambridge University Press, 1994. DOI: 10.1017/CBO9780511815478.

[21] Lu Zheng Song Yang Franziska B. Keller. *Social Network Analysis Methods and Examples*. SAGE Publications, Inc, 2018.

[22] Bob Yirka. “"Social networks may be exaggerating risk of news events"”. In: *Phys.org* (Apr. 2015).

[23] Pål Olsvik et al. “Transcriptional responses to temperature and low oxygen stress in Atlantic salmon studied with next-generation sequencing technology”. In: vol. 14. Nov. 2013, p. 817. DOI: 10.1186/1471-2164-14-817.

[24] Jianyong Sun et al. “Graph Neural Network Encoding for Community Detection in Attribute Networks”. In: (2021).

[25] Xiang Zhang et al. “Modularity optimization in community detection of complex networks”. In: *EPL (Europhysics Letters)* (Sept. 2009).

[26] Luís Rita. “Louvain Algorithm: An algorithm for community finding”. In: *Towards Data Science* (Aug. 2020). URL: https://towardsdatascience.com/louvain-algorithm-93fde589f58c.

[27] Emmanuel Abbe. “Community Detection and Stochastic Block Models”. In: (2022). arXiv: 1703.10146 [math.PR].

[28] Arun Jagota. “Markov Clustering Algorithm:Intuitive description with examples and discussion”. In: *Towards Data Science* (Dec. 2020). URL: https://towardsdatascience.com/markov-clustering-algorithm-577168dad475.

[29] Cory Maklin. “Hierarchical Agglomerative Clustering Algorithm Example In Python”. In: *Towards Data Science* (Jan. 2019). URL: https://towardsdatascience.com/machine-learning-algorithms-part-12-hierarchical-agglomerative-clustering-example-in-python-1e18e0075019.

[30] Sukanta Nayak. “Fundamentals of Optimization Techniques with Algorithms”. In: (2021).

[31] Carmine Cerrone, Raffaele Cerulli and Bruce Golden. “Carousel Greedy: A Generalized Greedy Algorithm with Applications in Optimization”. In: *Computers Operations Research* 85 (Apr. 2017). DOI: 10.1016/j.cor.2017.03.016.

[32] Robert Kwiatkowski. “Gradient Descent Algorithm — a deep dive”. In: *Towards Data Science* (May 2021).

[33] D.B. Fogel. “An introduction to simulated evolutionary optimization”. In: *IEEE Transactions on Neural Networks* 5.1 (1994), pp. 3–14. DOI: 10.1109/72.265956.

[34] Frank Liang. “Optimization Techniques — Simulated Annealing”. In: *Towards Data Science* (Apr. 2020).

[35] Dr. Robert Kubler. “Introduction to Linear Programming”. In: *Towards Data Science* (June 2021).

[36] Bruno Scalia C. F. Leite. “Nonlinear programming: Theory and applications”. In: *Towards Data Science* (Mar. 2022).

[37] Egor Howell. “Breaking Down Combinatorial Optimization: Why Some Problems Are Inherently Difficult”. In: *Towards Data Science* (Apr. 2023).

[38] Chunxue Zhao. “Application of Integer Programming in Project Investment”. In: *WSEAS TRANSACTIONS ON COMPUTER RESEARCH* 8 (July 2020), pp. 111–114. DOI: 10.37394/232018.2020.8.14.

[39] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. DOI: 10.1017/CBO9780511804441.

[40] Sadrach Pierre. “A Guide to Metaheuristic Optimization for Machine Learning Models in Python”. In: *builtin* (Feb. 2022).

[41] Hennie de Harder. “Constraint Programming Explained”. In: *Towards Data Science* (Jan. 2023).

[42] Jason Brownlee. “A Gentle Introduction to Stochastic Optimization Algorithms”. In: *Machine Learning Mastery* (Feb. 2021).

[43] Hamish Thorburn. “Heuristics in Optimisation”. In: *Lancaster University* (Feb. 2020).

[44] Meet Patel. “Optimization: A notorious road to Structured Inefficiency and Transition to Combinatorial Optimization Machine Learning”. In: *Towards Data Science* (Jan. 2022).

[45] Chong Jing Ting. “Multi-Objective Optimization (MOO)”. In: *Towards Data Science* (Mar. 2022).

[46] Elvis David. “The Pareto principle — Game-changer to ML Engineers”. In: *Towards Data Science* (Feb. 2022).

[47] Aseel Salah. “Genetic Algorithms”. In: Nov. 2020. DOI: 10.13140/RG.2.2.25427.81449.

[48] Dr Anupam. “Particle Swarm Optimization: An Interactive Introduction”. In: *Towards Data Science* (Sept. 2020).

[49] Ujjwal Maulik. “Multiobjective Clustering with SVM Based Ensembling for Analysis of Gene Expression Data”. In.

[50] Ujjwal Maulik. “Multiobjective Clustering with SVM Based Ensembling for Analysis of Gene Expression Data”. In.

[51] Tutorials Point. “Genetic Algorithms - Crossover”. In: URL: https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_crossover.htm#.

[52] Srivignesh R. “Genetic Algorithms and its use-cases in Machine Learning”. In: URL: https://www.analyticsvidhya.com/blog/2021/06/genetic-algorithms-and-its-use-cases-in-machine-learning.

[53] Paul Calle. “Non-dominated sorting genetic algorithm II (NSGA-II)”. In: (Oct. 2017). URL: https://oklahomaanalytics.com/data-science-techniques/nsga-ii-explained/.

[54] Kalyanmoy Deb et al. “A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimisation: NSGA-II”. In: vol. 1917. Sept. 2000.

[55] Jonatas Chagas et al. “A non-dominated sorting based customized random-key genetic algorithm for the bi-objective traveling thief problem”. In: *Journal of Heuristics* 27 (June 2021). DOI: 10.1007/s10732-020-09457-7.

[56] Vikas Baghel. “Multiobjective Optimization — New Formulation and Application to Radar Signal Processing”. In: (Apr. 2023).

[57] Kalyanmoy Deb. “An evolutionary many-objective optimization algorithm using reference-point based non-dominated sorting approach, part i: Solving problems with box constraints”. In: *IEEE transactions on evolutionary computation* 18.4 (2014), pp. 577–601.

[58] adanjoga. “An implementation of NSGA-III algorithm in C++ according to the article: Kalyanmoy Deb, Himanshu Jain, An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach”. In: URL: https://github.com/adanjoga/nsga3.

[59] Julian Blank, Kalyan Deb and Proteek Chandan Roy. “Investigating the Normalization Procedure of NSGA-III: 10th International Conference, EMO 2019, East Lansing, MI, USA, March 10-13, 2019, Proceedings”. In: Jan. 2019, pp. 229–240. ISBN: 978-3-030-12597-4. DOI: 10.1007/978-3-030-12598-1_19.

[60] Pranava Chaudhari et al. “Comparison of NSGA-III with NSGA-II for multi-objective optimization of adiabatic styrene reactor”. In: *Materials Today: Proceedings* 57 (2022). International Chemical Engineering Conference 2021 (100 Glorious Years of Chemical Engineering Technology), pp. 1509–1514. ISSN: 2214-7853. DOI: https://doi.org/10.1016/j.matpr.2021.12.047. URL: https://www.sciencedirect.com/science/article/pii/S221478532107718X.

[61] Nirmalya Chowdhury, Debaditya Barman and Ritam Sarkar. “Overlapping Community Detection Using Multiobjective Genetic Algorithm”. In: *IEEE Transactions on Computational Social Systems* 7.3 (2020), pp. 802–817. DOI: 10.1109/TCSS.2020.2989295.

[62] Lei Zhang et al. “A Mixed Representation-Based Multiobjective Evolutionary Algorithm for Overlapping Community Detection”. In: *IEEE Transactions on Cybernetics* 47.9 (2017), pp. 2703–2716. DOI: 10.1109/TCYB.2017.2711038.

[63] Kamal Berahmand, Asgarali Bouyer and Mahdi Vasighi. “Community Detection in Complex Networks by Detecting and Expanding Core Nodes Through Extended Local Similarity of Nodes”. In: *IEEE Transactions on Computational Social Systems* 5.4 (2018), pp. 1021–1033. DOI: 10.1109/TCSS.2018.2879494.

[64] Petr Chunaev, Ivan Nuzhdenko and Klavdiya Bochenina. “Community Detection in Attributed Social Networks: A Unified Weight-Based Model and Its Regimes”. In: *2019 International Conference on Data Mining Workshops (ICDMW)*. 2019, pp. 455–464. DOI: 10.1109/ICDMW.2019.00072.

[65] Kamal Berahmand et al. “A new attributed graph clustering by using label propagation in complex networks”. In: *Journal of King Saud University - Computer and Information Sciences* 34.5 (May 2022), pp. 1869–1883. DOI: 10.1016/j.jksuci.2020.08.013.

[66] Wei Zheng et al. “Continuous Encoding for Overlapping Community Detection in Attributed Network”. In: *IEEE Transactions on Cybernetics* (2022). DOI: 10.1109/TCYB.2022.3155646.

[67] Xiangyi Teng, Jing Liu and Mingming Li. “Overlapping Community Detection in Directed and Undirected Attributed Networks Using a Multiobjective Evolutionary Algorithm”. In: *IEEE Transactions on Cybernetics* 51.1 (2021), pp. 138–150. DOI: 10.1109/TCYB.2019.2931983.

[68] Mingming Chen, Konstantin Kuzmin and Boleslaw K. Szymanski. “Extension of Modularity Density for Overlapping Community Structure”. In: *Proceedings of the 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. ASONAM ’14. Beijing, China: IEEE Press, 2014, pp. 856–863. ISBN: 9781479958764.

[69] Jianyong Sun et al. “Graph Neural Network Encoding for Community Detection in Attribute Networks”. In: *IEEE transactions on cybernetics* PP (Feb. 2021). DOI: 10.1109/TCYB.2021.3051021.

[70] Tanmoy Chakraborty et al. “Metrics for Community Analysis: A Survey”. In: *ACM Comput. Surv.* 50.4 (Aug. 2017). ISSN: 0360-0300. DOI: 10.1145/3091106. URL: https://doi.org/10.1145/3091106.

[71] Petr Chunaev, Timofey Gradov and Klavdiya Bochenina. “Community detection in node-attributed social networks: How structure-attributes correlation affects clustering quality”. In: *Procedia Computer Science* 178 (2020). 9th International Young Scientists Conference in Computational Science, YSC2020, 05-12 September 2020, pp. 355–364. ISSN: 1877-0509. DOI:

https://doi.org/10.1016/j.procs.2020.11.037. URL: https://www.sciencedirect.com/science/article/pii/S1877050920324169.

[72] Lei et al. “A Mixed Representation-Based Multiobjective Evolutionary Algorithm for Overlapping Community Detection”. In: *IEEE Transactions on Cybernetics* 47.9 (2017), pp. 2703–2716. DOI: 10.1109/TCYB.2017.2711038.

[73] OpenEye Scientific. “Similarity Measures”. In: ().

[74] Galileo Mark Namata, Ben London and Lise Getoor. “Collective Graph Identification”. In: *ACM Trans. Knowl. Discov. Data* 10.3 (Jan. 2016). ISSN: 1556-4681. DOI: 10.1145/2818378. URL: https://doi.org/10.1145/2818378.

[75] V Nicosia et al. “Extending the definition of modularity to directed graphs with overlapping communities”. In: *Journal of Statistical Mechanics: Theory and Experiment* 2009.03 (Mar. 2009), P03024. DOI: 10.1088/1742-5468/2009/03/p03024. URL: https://doi.org/10.1088%2F1742-5468%2F2009%2F03%2Fp03024.

[76] Lei Zhang et al. “A Mixed Representation-Based Multiobjective Evolutionary Algorithm for Overlapping Community Detection”. In: *IEEE Transactions on Cybernetics* 47.9 (2017), pp. 2703–2716. DOI: 10.1109/TCYB.2017.2711038.