REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE MOSTEFA BEN BOULAID BATNA2

Faculté des Mathématiques et Informatique
Departement d'informatique

Thèse de Doctorat

Presentée par:

Wafa LOUAFI

En vue de l'obtention du diplôme de **DOCTORAT Science** en :

Filière: Informatique

Option: Intellegence Artificielle

TITRE

Machine Learning Pour La Détection Des Communautés

Soutenue publiquement le : 06/06/2024

Devant le jury composé de:

Dr. Riadh HOCINE	MCA	Université de Batna2	Président
Pr. Faiza TITOUNA	Professeur	Université de Batna2	Rapporteur
Pr. Mohamed Nadjib KOUAHLA	Professeur	Université de Guelma	Examinateur
Dr. Ouahab KADRI	MCA	Université de Batna2	Examinateur
Dr. Leila BOUSSAAD	MCA	Université de Batna1	Examinateur
Dr. Zakaria LABOUDI	MCA	Université d'Oum El Bouagh	Examinateur

Année Universitaire: 2023/2024

PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
UNIVERSITY BATNA 2 MOSTEFA BEN BOULAID

Faculty of Mathematics and Computer Science
Computer Science Department

Dissertation

Presented by:

Wafa LOUAFI

Submitted in partial fulfillment of the requirements for **Doctorate Science** en :

Field:Computer Science

Option: Artificial Intelligence

TITLE

Machine Learning For Community Detection

Defended : 06/06/2024 In front of a jury composed of:

Dr. Riadh HOCINE	MCA	University of Batna2	President
Pr. Faiza TITOUNA	Professor	University of Batna2	Reporter
Pr. Mohamed Nadjib KOUAHLA	Professor	University of Guelma	Examiner
Dr. Ouahab KADRI	MCA	University of Batna2	Examiner
Dr. Leila BOUSSAAD	MCA	University of Batna1	Examiner
Dr. Zakaria LABOUDI	MCA	University of Oum El Bouagh	Examiner

Year Acadimic: 2023/2024

Dedication

“

I dedicate this work to To my beloved parents, Amar and Houria, whose unwavering support, love, and encouragement have been the foundation of my journey. Your sacrifices and guidance have shaped me into the person I am today. This thesis is a testament to your belief in me.

To my loving husband, Radouen, who has been my rock and source of endless inspiration. Your patience, understanding, and belief in my abilities have been instrumental in my pursuit of this academic milestone.

To my incredible children, Taym, Louai, and Larine, who bring joy and purpose to my life. Your endless curiosity and boundless energy remind me of the importance of lifelong learning. This thesis is dedicated to you, as a symbol of my commitment to creating a better future for you and generations to come.

To my brother, Fares and my sisters Sana and her husband Faycel, Leila and her husband Imed, Nor and her husband Aymen, who have been my pillars of strength throughout this journey.

To my beloved nieces and nephews, Iyad, Jad, Mayar, Miral, Yaniss and Ossayed, who have brought so much happiness into our lives.

To my caring stepfather, Saad, my stepmother, Aicha, and my wonderful step-siblings, Djemel, Didina, Loulou and Donia, and their children.

To my dear friends, both near and far, including Houda, who have cheered me on and provided unwavering support. Your presence in my life has made the journey more meaningful and memorable.

To my beloved extended family, including my parent, aunts, uncles, cousins, and to all those who have crossed my path and left an indelible mark.

Thanks!

”

WAFA

Acknowledgements

I am deeply grateful to Almighty God for granting me the faith, courage, and patience necessary to complete this work.

I sincerely thank my supervisor, Professor Faiza Titouna of Batna2 University. Her unwavering support, advice, constructive comments, and dedication to my progress have been invaluable in bringing this undertaking to a successful conclusion.

I would also like to express my sincere gratitude to the Doctor Hocine Riadh for graciously chairing the defense committee and supervising this important phase of my academic career.

To the members of my committee, Professor Kouahla Mohamed Nadjib, Professor at the University of Guelma; Dr.Kadri Ouahab, Senior Lecturer at the University of Batna2; Dr.Boussaad Leila, Senior Lecturer at the University of Batna1; and Dr. Laboudi Zakaria, Senior Lecturer at the University of Oum El Bouagh, I am grateful for their time and effort in examining my work.

I would like to thank all those who provided valuable information and suggestions that contributed to the improvement of this manuscript.

I greatly appreciate my family for their encouragement, prayers, and confidence in my abilities, which have kept me going.

Finally, I would like to express my gratitude to all those who played a role, large or small, in the development and progress of this work.

May Allah bless.

ملخص

حظي المجال المزدهر لتحليل الشبكات الاجتماعية باهتمام كبير ، مع التركيز بشكل خاص على المجال الحاسم للكشف عن المجتمع. تلعب المجتمعات، التي تفسر على أنها مجموعات من العقد المترابطة بشكل وثيق مع روابط أضعف بالشبكة الأوسع، دورا محوريا في فهم تطور هياكل الشبكة, وهو جانب أساسي من تحليل الشبكة.

تجري هذه الأطروحة استكشافا شاملا لخوارزميات الكشف عن المجتمع وطرق التعلم غير الخاضعة للإشراف، ثم الخوض في تطبيق تقنيات التعلم الآلي في هذا المجال. يتم تقديم طريقتين متميزتين: واحدة تركز على اكتشاف المجتمعات المتداخلة والأخرى على تحديد المجتمعات المنفصلة والجدير بالذكر أن أساليبنا تتضمن اختيار العقد الحيوية من خلال الرسوم البيانية الفرعية ، والتي تعمل بشكل فردي على كل عقدة ، بينما تحدث عملية التجميع نفسها على مستوى الشبكة بأكملها. يستفيد التنفيذ من العديد من تقنيات التعلم الآلي غير الخاضعة للإشراف، بما في ذلك التجميع الهرمي وk-means، مما يعرضها للكفاءة وسهولة التنفيذ.

تكمن الميزة البارزة لأساليبنا في تفوقها الواضح، وتحقيق دقة وأداء معززين مقارنة بالمنهجيات المعاصرة. ولإثبات هذا الادعاء ، قمنا بإجراء تقييمات دقيقة على كل من مجموعات بيانات الشبكة الحقيقية والتركيبية. بالإضافة إلى هذه الإنجازات ، يفتح البحث سبلا للاستكشاف المستقبلي في فهم التطبيقات الأوسع للكشف عن المجتمع ضمن هياكل الشبكة المتطورة.

الكلمات المفتاحية: الكشف عن المجتمعات، التعلم الآلي، الشبكات الاجتماعية، التجميع، المجتمعات المتداخلة.

Résumé

Le domaine en plein essor de l'analyse des réseaux sociaux a suscité une attention considérable, se concentrant particulièrement sur le domaine critique de la détection de communautés. Les communautés, comprises comme des regroupements de nœuds étroitement interconnectés avec des liens plus faibles vers le réseau global, jouent un rôle central dans la compréhension de l'évolution des structures de réseau, un aspect essentiel de l'analyse des réseaux.

Cette thèse mène une exploration approfondie des algorithmes de détection de communautés et des méthodes d'apprentissage non supervisé, plongeant ensuite dans l'application de techniques d'apprentissage automatique dans ce domaine. Deux méthodes distinctes sont présentées : l'une axée sur la détection de communautés chevauchantes et l'autre sur l'identification de communautés disjointes. Notamment, nos méthodes impliquent la sélection de nœuds vitaux à travers des sous-graphes, opérant individuellement sur chaque nœud, tandis que le processus de regroupement lui-même se déroule globalement sur l'ensemble du réseau. L'implémentation exploite diverses techniques d'apprentissage automatique non supervisé, notamment le regroupement hiérarchique et k-means, démontrant ainsi leur efficacité et leur facilité de mise en œuvre.

La caractéristique remarquable de nos méthodes réside dans leur supériorité démontrée, atteignant une précision et une performance améliorées par rapport aux méthodologies contemporaines. Pour étayer cette affirmation, des évaluations méticuleuses ont été menées sur des ensembles de données de réseaux réels et synthétiques. Au-delà de ces réalisations, la recherche ouvre des perspectives pour des explorations futures visant à comprendre les implications et applications plus larges de la détection de communautés au sein des structures de réseau en évolution.

Mots clés : Détection de communauté, Apprentissage automatique, Réseaux sociaux, Regroupement, Communautés chevauchantes.

Abstract

The burgeoning field of social network analysis has garnered considerable attention, with a particular focus on the critical area of community detection. Communities, construed as clusters of closely interconnected nodes with weaker ties to the broader network, play a pivotal role in understanding the evolution of network structures, an essential aspect of network analysis.

This thesis conducts a comprehensive exploration into community detection algorithms and unsupervised learning methods, subsequently delving into the application of machine learning techniques in this domain. Two distinct methods are presented: one focused on detecting overlapping communities and the other on identifying disjoint communities. Notably, our methods involve the selection of vital nodes through subgraphs, operating individually on each node, while the clustering process itself occurs globally across the entire network. The implementation leverages various unsupervised machine learning techniques, including hierarchical clustering and k-means, showcasing efficiency and ease of implementation.

The standout feature of our methods lies in their demonstrated superiority, achieving enhanced accuracy and performance compared to contemporary methodologies. Substantiating this claim, meticulous evaluations were conducted on both real and synthetic network datasets. Beyond these achievements, the research opens avenues for future exploration in understanding the broader implications and applications of community detection within evolving network structures.

Keywords **:** Community detection, Machine Learning, Social Networks, Clustering, Overlapping communities.

Contents

List of Figures

List of Tables

List of Algorithms

General Introduction

The foundations of mathematics have given rise to a complex realm known as Graph Theory (GT), a domain where nodes and edges intertwine to depict intricate relationships among entities. Graphs stand as fundamental constructs, providing a framework to unravel the intricate web of connections present in diverse contexts. Within the domain of social networks, graphs find profound application as potent representations of interconnected individuals and their affiliations. In these networks, nodes symbolize individuals, while edges encapsulate the intricate threads that bind them together. These networks weave a diverse tapestry of relationships encompassing friendships, professional alliances, familial connections, and the interactions transpiring across digital platforms (Tutte, 2001).

Social network encompass a concept in organizational psychology that revolves around the establishment and maintenance of social connections, offering pathways to access information and resources (Utz & Breuer, 2019). In the field of social network analysis, the spotlight is on the concept of community detection.This invaluable tool aids in uncovering insightful patterns. Visualize a network as an intricate web of connections between people or entities.

Community Detection (CD) dissects this web into smaller clusters, unveiling latent patterns and spotlighting groups of interconnected entities. This approach offers insights not only into interconnected groups but also into individual and group behaviors within the network (Fortunato, 2010). Their objective is expansive. It aids in comprehending how information flows within a network and even facilitates predictions about future occurrences. It serves as a navigational tool for efficiently exploring networks, allowing us to better understand the complex social systems that interconnect our world. This concept of community detection transcends a singular domain, it's akin to a compass guiding decision-making across various fields. Whether unraveling societal dynamics, devising effective marketing strategies, or conducting intricate network analysis, community detection remains a pivotal tool. In essence, community detection serves as a specialized instrument, enabling us to navigate and comprehend the intricate realm of social networks. However, finding an optimal solution to the graph partitioning task remains a challenge, often classified as NP-complete, particularly for extensive graphs(Newman & Girvan, 2004). Nonetheless, a diverse array of algorithms has emerged to trove satisfactory solutions in many scenarios.

Traditional approaches to community identification in graphs have played a pivotal role in understanding complex structures and patterns within intricate networks. These methods, predating modern machine learning techniques, offer foundational insights into how connections manifest across diverse fields. However, they are not without their trade-offs. While well-suited for smaller networks with distinct hierarchies and clear communities, these methods may face challenges as networks grow larger and more intricate. Issues related to computational speed, scalability, and managing a diverse range of community structures may arise. As we encounter increasingly complex network challenges, the integration of multifaceted network data becomes imperative. This involves elements like network topology and node semantics, posing challenges for conventional methods to efficiently handle high-dimensional and diverse attribute data. In response, the incorporation of machine learning methods has garnered attention, enabling the processing of high-dimensional network data and acquiring reduced-dimensional network representations (Jin et al., 2021).

The realm of community detection in networks constitutes an ever-evolving domain fraught with intricate challenges, encompassing the development of detection algorithms, validation and interpretation of results, accuracy in detection, and the handling of community overlap across diverse network types. Amid these complexities, Machine Learning (ML) methodologies emerge as a promising solution, offering heightened precision and efficacy in the community detection process. By leveraging labeled data, ML algorithms excel in identifying subtle patterns that may elude human observation, as noted by (Butler, Davies, Cartwright, Isayev, & Walsh, 2018). This not only enhances the accuracy of community allocation outcomes but also facilitates more adept management of complex network configurations.

The incorporation of machine learning techniques further strengthens the capacity to unveil concealed relationships within networks, leading to more refined and accurate community identification, as highlighted by Ghouchan Nezhad Noor Nia, Jalali, Mail, Ivanisenko, and Kübel (2022). Importantly, these ML methods effectively address challenges posed by intricate network structures where traditional techniques may falter. As machine learning models continuously evolve through learning and adaptation, their proficiency in discerning latent community structures within diverse networks becomes increasingly robust, presenting a valuable proposition for addressing the complexities inherent in community detection in networks.

In this context, our thesis presents two distinct approaches to community detection. The first, Density Based Overlapping Community Detection (DBOCD) relies on local density and uses clustering techniques to identify overlapping communities (Louafi & Titouna, 2023a). The second approach, named PCMeans, introduces an innovative methodology based on influential nodes, hierarchical clustering, and k-means algorithms (Louafi & Titouna, 2023b). PCMeans adeptly navigates the complexities of community detection by efficiently revealing disjoint communities. The empirical validation of our algorithms is substantiated through thorough comparisons on both real-world and synthetic datasets. This validation underscores their potency,

precision, and efficiency, highlighting their potential across diverse network scenarios. These approaches stand alongside contemporary algorithms, providing novel insights into community detection while addressing limitations posed by gradual convergence.

This thesis is structured into two main parts: the first part meticulously explores the state of the art regarding the interplay between graphs, social networks, community detection, and machine learning. Comprising three elaborately crafted chapters, each contributes a crucial element to our comprehensive understanding of this subject. The second part introduces our innovative proposals in two dedicated chapters.

Chapter 1: Graph Theory and Social Networks

Chapter 1 serves as a gateway to explore the fascinating convergence of graph theory and the dynamic realm of social networks. It begins by establishing the foundational cornerstones of graph theory, providing precise definitions alongside practical illustrations. Seamlessly transitioning, the chapter navigates the intricate landscapes of social networks, unveiling a spectrum of relationships ranging from collaboration to kinship. Guided by graph metrics such as similarity and centrality, this journey provides profound insights into network dynamics and community interconnections. The chapter also addresses challenges and outlines future possibilities in comprehending intricate social interactions through graph-based analysis.

Chapter 2: Machine Learning

In Chapter 2, we delve into the domain of machine learning, unveiling various methodologies, including supervised, unsupervised, and reinforcement learning. A detailed comparison between these methods is conducted, with a specific focus on elucidating the intricacies of the K-means algorithm.

Chapter 3: Community Detection

Chapter 3 delves deeply into the current landscape of community structures, navigating through the diverse spectrum of methods used for community detection. The journey begins with a clarification of the myriad definitions surrounding the concept of community, subsequently proceeding to its formal delineation. This chapter extensively explores a plethora of approaches and techniques developed for community detection, highlighting their multifaceted nature. We study different applications of machine learning in the field of community detection.

Chapter 4: DBOCD - Density-Based Overlapping Community Detection

Chapter 4 introduces ”DBOCD,” a novel method designed for detecting overlapping communities in social networks, integrating a measure of node importance grounded in local density. Our approach begins by computing the local density of each node within the network. Subsequently, it employs a clustering algorithm to discern both overlapping and disjoint communities based on the local density of their constituent nodes. We assess the effectiveness of our approach across a range of social network datasets. Our results demonstrate the high acceptability

of our approach, showcasing its capability to identify both disjoint and overlapping communities (Louafi & Titouna, 2019, 2023a).

Chapter 5: PCMeans - A Novel Community Detection Algorithm

Chapter 5 introduces ”PCMeans,” a novel approach that integrates influential nodes, hierarchical clustering, and k-means algorithms. This innovative methodology navigates the intricate landscape of community detection. The approach consists of three phases: the identification of influential nodes through localized PageRank, the segmentation of data points via hierarchical clustering, and the refinement of clusters using k-means. The resulting framework achieves a delicate balance between complexity and efficiency, purposefully designed to meet the requirements of community detection.

In conclusion, a final section encapsulates the essence of this thesis, revisiting pivotal aspects discussed throughout and emphasizing the central proposition. Additionally, this section outlines a range of short-term and long-term prospects for future exploration and advancement in this captivating field.

Part I

State of the Art

Chapter 1

Graph Theory and Social Network

1.1 Introduction

Graph theory, a foundational branch of mathematics, traces its roots to the groundbreaking work of Leonhard Euler. Euler, a Swiss mathematician and physicist born on April 15, 1707, in Basel, Switzerland, and passing away on September 18, 1783, in St. Petersburg, Russia, made significant contributions to various fields of mathematics and science during the 18th century. Euler's 1741 publication, "The Bridges of Königsberg," marked a pivotal moment in the history of graph theory, introducing concepts like Eulerian circuits and paths, which profoundly shaped the mathematical landscape (Euler, 1741, 1956). With this article, Euler catalyzed the inception of graph theory, revolutionizing mathematical thought and providing a powerful tool for comprehending relationships within interconnected structures (Alexanderson, 2006).

Euler's innovative work not only resolved a specific puzzle but also established the foundation for an entire discipline, driving systematic analysis and abstraction. The concept of graphs, as introduced by Euler, proved instrumental in modeling real-world phenomena. Over time, graph theory has evolved, yielding a diverse range of concepts, including Eulerian graphs, tree structures, Hamiltonian graphs, and more (Majeed & Rauf, 2020). These concepts unveil the mathematical framework that underpins the intricate web of connections in our world.

In today's context, graphs are omnipresent, serving as fundamental elements across various domains. They find application in social network analysis, data analytics, community detection, network dynamics, and pattern recognition, reflecting their indispensability in contemporary research and problem-solving (Sarma, 2012).

In this chapter, we provide a comprehensive foundation for understanding the essentials of graph theory and explore their application in the context of social networks. We will delve into graph theory fundamentals and metrics, including similarity measures, graph distances, and centrality measures, essential for social network analysis. Furthermore, we will explore the world of social networks, understanding their emergence and key characteristics.

1.2 Foundations of graph theory

Graph theory is like a tool in math that helps us understand graphs. These graphs are made up of points (called nodes or vertices) and lines (called edges) that connect them (Browet, 2014). Think of it like a puzzle where the points are pieces, and the lines connect the pieces together. In simple terms, a graph shows us how different things are connected. We can figure this out by looking at degrees and adjacency, which tell us how points relate to each other. Paths and cycles show the routes that points take, helping us see how the graph is structured.

Concepts like connectivity and components highlight groups of points that stick together, giving us insights into different communities within a network. Graphs aren't just about math; they're a cool way to understand real-world connections. They pop up in many areas, showing us networks and relationships we might not notice otherwise (West et al., 2001). The essence of

graph theory lies in its capacity to transform these seemingly abstract connections into tangible mathematical entities, facilitating rigorous analysis and interpretation (Newman, Strogatz, & Watts, 2001).

1.2.1 Concept of a graph

The concept of a graph unfolds as a foundational construct of great significance in computer science and mathematics. A graph is a dynamic arrangement, meticulously composed of vertices (or nodes) intricately linked together by edges. These two fundamental components, vertices and edges, orchestrate a rich and intricate interplay, forming the very essence of graph theory (Wicker & Kim, 2002).

Nodes, symbolizing unique entities or elements, form the backbone of the graph, defining its organizational structure. These nodes could represent diverse entities, such as individuals in a social network or cities in a transportation system, embodying the subjects under examination (Albert, Jeong, & Barabási, 1999).

Edges, dynamic conduits linking nodes, play a crucial role in bringing entities together in a network of relationships. These edges symbolize connections, interactions, or associations between entities, forming a network of complex dependencies.

Formally, a graph G is defined as a pair (V, E), where V denotes a set of vertices, and E signifies a set of edges. Each edge is formed by selecting a pair of distinct vertices from the set V. The visual representation of a graph, as shown in Figure 1.1b with six nodes and eight edges, provides an intuitive understanding of how nodes are connected by edges.

1.2.2 Adjacency matrix

The adjacency matrix serves as a crucial bridge between the abstract world of graphs and the structured domain of matrices. In its binary form, it expresses the connections, interactions, and dependencies that govern the dynamics of graphs, whether simple unweighted or nuanced, weighted graphs.

In graph theory, particularly in undirected, unweighted graphs G with n nodes, the matrix representation known as the adjacency matrix A $(n \times n)$ emerges. This matrix accurately reflects the graph G, with vertices corresponding to nodes represented in both rows and columns (Wallis, 2007).

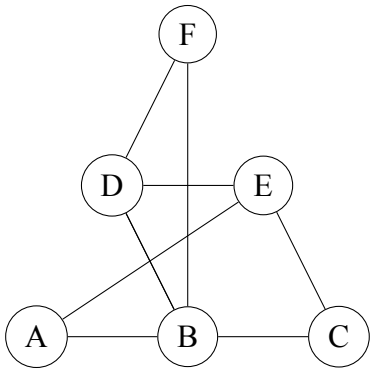
For unweighted graphs, the entries in the adjacency matrix convey a binary message, where $M_{ij} = 0$ denotes no connection, while $M_{ij} = 1$ signifies the presence of an edge, reflecting the interaction of nodes within the graph.

$$M_{ij} = \begin{cases} 1, & \text{if a connection exists between vertices } i \text{ and } j. \\ 0, & \text{otherwise.} \end{cases} \tag{1.1}$$

Figure 1.1 presents an example of an unweighted graph and its corresponding adjacency matrix. In this example, we use an adjacency matrix denoted as M to represent a graph comprising six vertices identified as A, B, C, D, E, and F. If there is a connection between vertex A and vertex B, the entry at the intersection of row 1 and column 2 in matrix M is assigned the value 1. Similarly, if there is a connection between vertex B and vertex E, the entry at the intersection of row 2 and column 5 is assigned the value 1. It is important to highlight that the diagonal entries in the matrix are consistently set to 0, as a vertex is not considered adjacent to itself.

$$M = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

(a) Adjacency matrix.

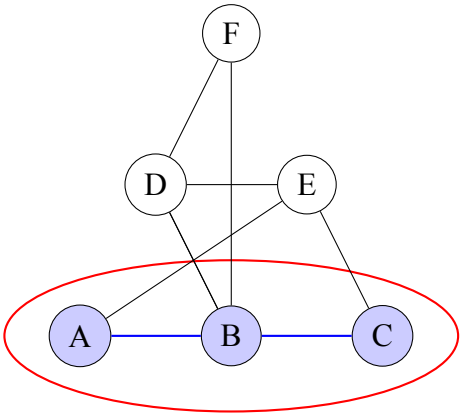


(b) Unweighted graph corresponding to the matrix.

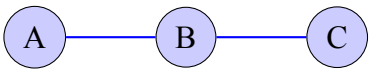
Figure 1.1: Graphic representation and corresponding matrix.

1.2.3 Concept of subgraph

A subgraph constitutes a subset of both the vertices and edges originating from the original graph, functioning as a scaled-down representation encompassed within the broader graph (Tutte, 2001). While a subgraph possesses its distinct collection of vertices and edges, it generally isn't regarded as a characteristic of the original graph in the same vein as traits like the graph's degree or density. Figure 1.2b represents a subgraph extracted from the graph shown in Figure 1.1b.



(a) Graph.



(b) Subgraph.

Figure 1.2: Extracting a subgraph from a graph.

Subgraphs are invaluable in graph theory for their ability to simplify complex graph structures. These substructures serve as pivotal tools, aiding in the identification of recurring patterns or structures within a graph. Furthermore, they offer an avenue to scrutinize the properties of specific vertex or edge subsets.

Beyond simplifying graph structures, subgraphs play a crucial role as representations of subnetworks or communities within expansive networks. They provide valuable insight into the intricate organization and functioning of complex networks.

1.2.4 Graph density

Graph density, within the framework of a graph $G = (V, E)$, serves as a pivotal metric, quantifying the level of interconnectivity within the network. It provides a quantitative depiction of the abundance of links that knit together the fabric of nodes and edges, offering insights into the depth of the graph's connectivity (Zhao, 2023).

The computation of a graph's density involves a fundamental comparison, dividing the observed count of edges by the maximum conceivable count of edges. This division yields a density value between 0 and 1. An empty graph, where vertices stand isolated without connecting edges, manifests a density of 0, signifying a minimal value that represents a dearth of connections. In contrast, a complete graph, where every pair of vertices is bound by a link, garners a density of 1, indicating comprehensive connectivity.

Graph density, measured by the formula 1.2, acts as a barometer of a graph's cohesion. This metric captures how intricately nodes are linked, ranging from scenarios where isolation prevails to ones where every possible connection has been realized. The formula for density D is calculated using the number of edges ($|E|$) and the count of nodes ($|V|$).

$$D = \frac{|E|}{|V|(|V|-1)/2} \tag{1.2}$$

1.2.5 Search algorithm

The search algorithm is a category of algorithms used to explore and analyze graphs. Two of the most commonly used search algorithms are Breadth-First Search (BFS) and Depth-First Search (DFS). These two algorithms can be used to systematically traverse a graph and identify important information about its structure and content (Everitt & Hutter, 2015).

1.2.5.1 Breadth-first search (BFS)

BFS stands for Breadth-First Search, and it is a graph traversal algorithm used in computer science. It explores all the vertices of a graph in breadthward motion, meaning that it visits all the neighbors of a vertex before moving on to their neighbors (S. J. Russell & Norvig, 2010).

The algorithm starts at the root (selecting some arbitrary node as the root in the case of a

graph) and explores the neighbor nodes at the present depth before moving on to nodes at the next depth level. It uses a data structure called a queue to keep track of the nodes to be explored. The BFS algorithm can be presented in four steps:

1. Enqueue the starting node and mark it as visited.
2. Dequeue a node from the queue and visit it.
3. Enqueue all the adjacent nodes of the dequeued node that haven't been visited and mark them as visited.
4. Repeat steps 2 and 3 until the queue is empty.

BFS is often used to find the shortest path between two nodes in an unweighted graph, and it is also employed in various other applications such as network broadcasting, web crawling, and puzzle solving. The algorithm is presented in Algorithm 1.1.

Algorithm 1.1. BFS Algorithm.

```
1:  procedure BFS(G, s)                          ▷ G is the graph, s is the source node
2:      Initialize an empty queue Q
3:      Mark all vertices as not visited
4:      Enqueue the source node s and mark it as visited
5:      while Q is not empty do
6:          Dequeue a vertex v from Q
7:          for each adjacent vertex u of v do
8:              if u is not visited then
9:                  Enqueue u and mark it as visited
10:             end if
11:         end for
12:     end while
13: end procedure
```

1.2.5.2 Depth-First Search (DFS)

DFS stands for Depth-First Search, another graph traversal algorithm used in computer science. Unlike BFS, DFS explores as far as possible along each branch before backtracking. It can be implemented using either recursion or an explicit stack data structure (S. J. Russell & Norvig, 2010). The DFS algorithm can be presented in five steps:

1. Start at a source node and mark it as visited.
2. Explore one of the unvisited neighbors.
3. If the chosen neighbor has unvisited neighbors, repeat step 2 for this neighbor.
4. If the chosen neighbor has no more unvisited neighbors, backtrack to the previous node and explore its other unvisited neighbors.
5. Repeat steps 2-4 until all nodes are visited.

DFS can be used to solve various graph-related problems, such as finding connected compo-

nents, detecting cycles, and traversing a graph in topological order. However, it doesn't guarantee finding the shortest path in a graph because it may delve into a branch before exploring other paths. It is generally less memory-intensive than BFS because it does not require storing all nodes at the current level. The algorithm is presented in Algorithm 1.2.

Algorithm 1.2. DFS Algorithm.

```
1: procedure DFS(G, start)          ▷ Depth-First Search on graph G starting from start
2:     Create a set visited to keep track of visited nodes.
3:     DFS-Visit(G, start, visited)
4: end procedure
5: procedure DFS-Visit(G, current, visited)
6:     Mark current as visited.
7:     Process node current                    ▷ e.g., visit or perform an operation.
8:     for all neighbor of current do
9:         if neighbor is not in visited then
10:            DFS-Visit(G, neighbor, visited)
11:        end if
12:    end for
13: end procedure
```

The choice between BFS and DFS depends on the specific problem to solve and the characteristics of the graph being processed.

1.2.6 Type of graphs

Graph theory encompasses different graph types, each possessing unique characteristics and applications. Here are a few prevalent examples:

1. **Undirected Graph:** In an undirected graph, edges lack directionality; they simply denote a connection between two vertices. If there exists an edge between vertex A and vertex B, it signifies that vertex A is linked to vertex B, and vice versa. Figure 1.1b presents an undirected graph.
2. **Connected Graphs (Conex):** A connected graph is one in which, for any pair of nodes, a sequence of edges can be traced to traverse from one vertex to another. Figure 1.1b presents a conex graph.
3. **Directed Graphs (Oriented):** A directed graph, also referred to as an oriented graph, is characterized by edges that possess directionality. In simpler terms, each edge is represented as an ordered pair of vertices (u, v), where u serves as the starting point, and v as the destination. Consequently, movement along the edge is restricted to a single direction, specifically from u to v. Figure 1.3 presents an oriented graph.
4. **Weighted Graphs** A weighted graph is a type of graph in which each edge is assigned a weight or cost. These weights can represent various metrics, such as distances, expenses, or capacities. Weighted graphs frequently find application in optimization sce-

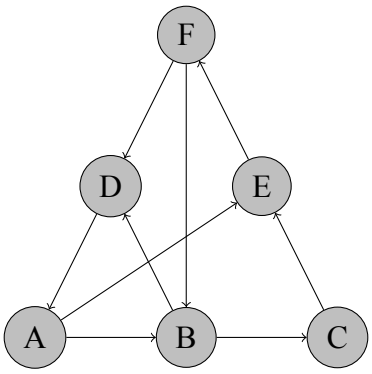
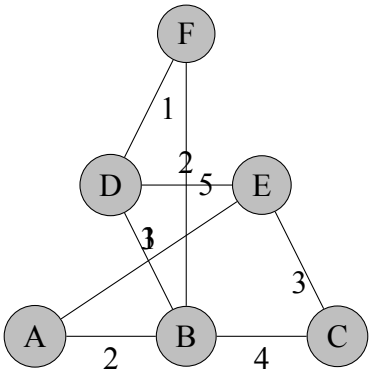


Figure 1.3: Oriented Graph.

narios, where the objective is to identify the shortest or longest path according to these assigned weights, or to address related optimization challenges.

The utility of the matrix isn't confined to unweighted graphs alone. It extends into weighted graphs, where entries represent not only connectivity but also the weight or cost associated with each edge (L. Zhou, Wang, Qu, Huang, & Liu, 2020).



(a) Weighted Graph.

$$\begin{bmatrix} 0 & 2 & 0 & 0 & 1 & 0 \\ 2 & 0 & 4 & 3 & 0 & 2 \\ 0 & 4 & 0 & 0 & 3 & 0 \\ 0 & 3 & 0 & 0 & 5 & 0 \\ 1 & 0 & 3 & 5 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(b) Adjacency Matrix.

Figure 1.4: Weighted Graph and his Adjacency Matrix.

In the example shown in Figure 1.4a, their adjacency matrix presented in Matrix 1.4b becomes a canvas portraying the intricacies of weights and relationships. The weights are depicted as numerical values associated with the edges. The edge connecting vertex A and vertex B carries a weight of 2, while the edge linking vertex D and vertex F possesses a weight of 1. These weights enable computations for determining the optimal path, whether it be the shortest or longest, between two vertices or for solving other optimization problems.

1.3 Graph measures & metrics

A graph metric functions as a numerical measure or trait intricately associated with a graph, offering a quantifiable representation that sheds light on specific facets of its structure or behavior. These metrics assume a pivotal role in the comprehensive analysis and comparison of

diverse graphs, providing valuable insights into the intricacies of the network's properties and dynamics (Wills & Meyer, 2020).

The realm of graph theory presents a rich tapestry of graph metrics, with each metric meticulously designed to address particular analytical needs. This diversity in metrics allows researchers and analysts to choose specific measures that align with the nuances of their graph-based investigations, ensuring a nuanced and tailored approach to graph analysis and interpretation. Figure 1.5 illustrates this variety, showcasing the richness and versatility of metrics available for in-depth graph analysis.

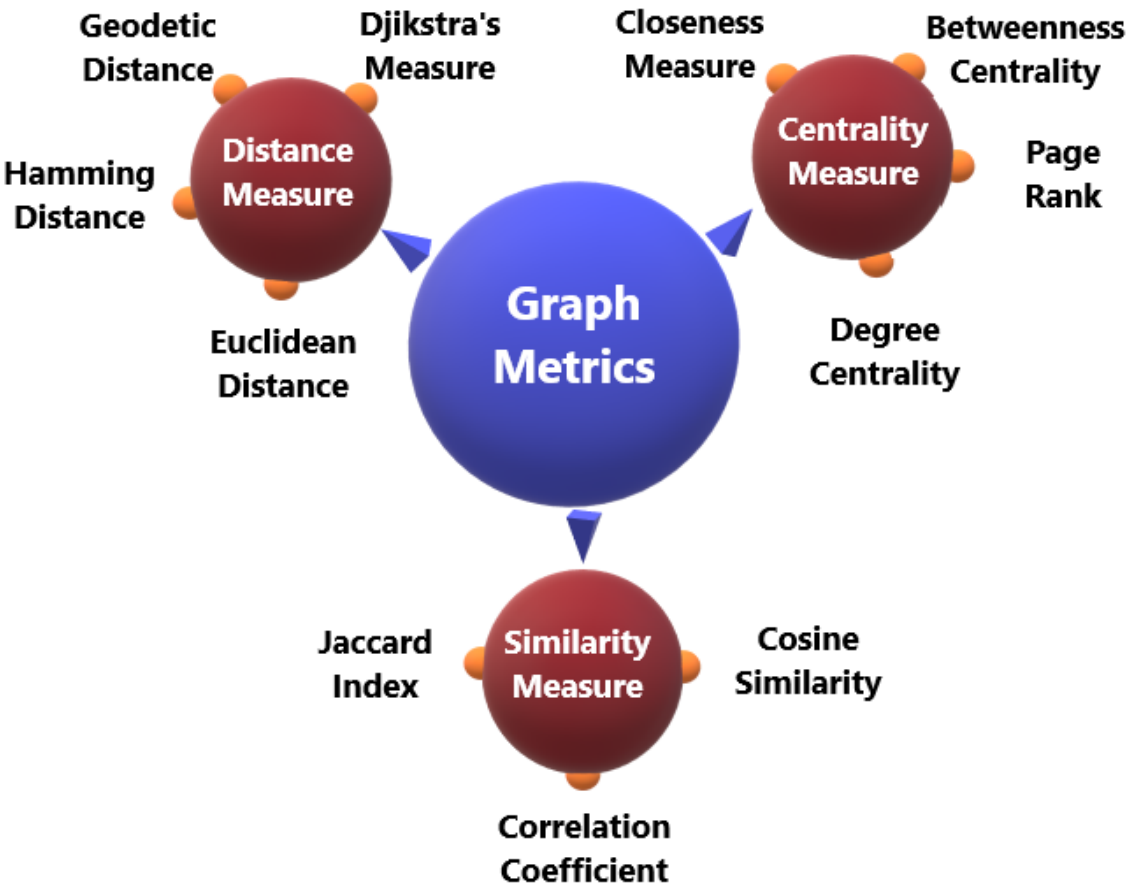


Figure 1.5: Graph measures and metrics.

1.3.1 Similarity measures

Similarity measures are used to quantify the degree of similarity between two objects or sets of objects (Bunke & Messmer, 1993). In the context of graph theory, similarity measures are often employed to compare the structural or functional similarities between different nodes, subgraphs, or entire graphs. We delineate three frequently employed methods for gauging node similarity: Jaccard Index, Cosine Similarity, and Pearson Correlation Coefficien.

1.3.1.1 Jaccard index

The Jaccard Index, also known as the Jaccard similarity coefficient, was introduced by Paul Jaccard (1901), a Swiss botanist born in Lausanne. This index quantifies the similarity between two sets of data by calculating the ratio of the size of their intersection to the size of their union, their values yields within the range of 0 to 1, with 0 indicating that the two sets share no common elements, and 1 signifying that the two sets are identical. Mathematically, the Jaccard index between two sets A and B is expressed as:

$$\text{Jacc}(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{1.3}$$

Widely used in machine learning and data science, the Jaccard index serves as a valuable tool in applications such as clustering, classification, and recommendation systems. It gauges the similarity between two sets of items, such as movies, songs, or products, based on their shared elements (Prasetya, Wibawa, & Hirashima, 2018). Furthermore, it plays a significant role in text mining and natural language processing tasks, such as measuring document similarity, where it aids in comparing the resemblance of two documents based on their contained words.

1.3.1.2 Cosine similarity

Cosine similarity functions as a metric to measure the similarity between two non-zero vectors in an inner product space by evaluating the cosine of the angle formed between them. This measure is extensively applied in the fields of information retrieval and natural language processing (Tackx, 2018). The formula for computing cosine similarity between two vectors, denoted as u and v, is expressed as follows:

$$\text{Cosine Similarity}(u, v) = \cos(\theta) = \frac{\mathbf{u} \cdot \mathbf{v}}{|\mathbf{u}||\mathbf{v}|} = \frac{\sum_{i=1}^{n} u_i v_i}{\sqrt{\sum_{i=1}^{n} u_i^2}\sqrt{\sum_{i=1}^{n} v_i^2}} \tag{1.4}$$

The output of cosine similarity ranges from -1 to 1. A value of 1 suggests that the two vectors are identical, 0 indicates orthogonality (no correlation), and -1 signifies diametric opposition, indicating a negative correlation between the vectors. This metric finds particular utility in text mining and natural language processing tasks, often employed to assess the similarity between two documents or two sets of word vectors. It aids in tasks such as document comparison and word vector analysis.

1.3.1.3 Pearson correlation coefficient

The pearson correlation coefficient, commonly referred to as Pearson's r, is a statistical metric that measures the strength of linear correlation between two sets of node attributes (Hahs-Vaughn, 2023). This coefficient is computed by dividing the covariance of the two vectors by

the product of their standard deviations. The coefficient of the Pearson correlation of X and Y is computed by the formula:

$$r = \frac{\text{cov}(X, Y)}{\sigma_X \cdot \sigma_Y} \tag{1.5}$$

where:

- $\text{cov}(X, Y)$ is the covariance between X and Y;
- σ_X is the standard deviation of X;
- σ_Y is the standard deviation of Y.

The resulting value of this coefficient ranges from -1, indicating a perfect negative correlation, to 1, signifying a perfect positive correlation.

1.3.2 Distance measures

Distance measurements are frequently employed in network analysis to assess the connectivity of a graph. The shortest path distance between two vertices within a graph signifies the smallest number of edges required to transition from one vertex to the other. It is alternatively known as graph distance or path length (Vandaele, Saeys, & De Bie, 2021).

Various methods exist to calculate the distance between vertices in a graph. Four frequently used methods include Dijkstra's Algorithm, Geodetic Distance, Hamming Distance, and Euclidean Distance.

1.3.2.1 Dijkstra's algorithm

Dijkstra's algorithm is a popular method for finding the shortest path between two vertices in a graph with non-negative edge weights. The algorithm, named after its inventor Edsger Dijkstra, is commonly used in network routing and pathfinding applications (Shu-Xi, 2012; Fatima & Maurya, 2023).

Here's an overview of how Dijkstra's Algorithm works:

1. **Initialization:** Assign a provisional distance value to each vertex. Set the distance of the initial vertex to 0 and the distances of all other vertices to infinity.
2. **Priority Queue:** Create a priority queue to keep track of vertices with their tentative distances as priorities.
3. **Exploration:** For exploration, we need four steps:
   - Extract the vertex with the smallest tentative distance from the priority queue.
   - For each neighboring vertex not yet visited, calculate its tentative distance by summing the distance of the current vertex and the weight of the edge connecting them. If this calculated distance is smaller than the previously assigned distance, update the tentative distance.
   - Add the updated vertex and its new tentative distance to the priority queue.

- Repeat the following steps until the priority queue is empty.

4. **Termination:** The algorithm terminates when the destination vertex is extracted from the priority queue, or when the priority queue becomes empty.
5. **Path Reconstruction (Optional):** If needed, reconstruct the shortest path from the source to the destination using the information stored during the algorithm's execution.

When dealing with non-negative and connected graphs, Dijkstra's Algorithm ensures the discovery of the shortest path. However, it is crucial to acknowledge its limitations, as the algorithm may produce inaccurate results when applied to graphs with negative edge weights. In such instances, alternative algorithms might prove more effective.

1.3.2.2 Geodetic distance

Geodesic distance is the shortest distance between two points on a curved surface, such as the surface of the Earth. It is also known as the shortest path distance or the great circle distance. The geodesic distance is calculated along the surface of the Earth, taking into account the curvature of the Earth (Crane, Livesu, Puppo, & Qin, 2020).

In graph theory, the geodesic distance is the shortest path between two vertices in a graph, also known as the graph geodesic or the distance between two nodes.it's an essential concept in network analysis, as it helps determine the shortest path between two nodes in a network (Jiawei Han, 2012). The geodesic distance is determined by:

$$d(u,v) = \min_{p \in P_{u,v}} \sum_{i=1}^{k-1} w(i, i+1) \tag{1.6}$$

Where:

- $P_{u,v}$ denotes the set of all paths from node u to node v;
- $w(i, i+1)$ signifies the weight of the edge between nodes i and $i+1$ within path p.

1.3.2.3 Hamming distance

The Hamming distance serves as a metric for quantifying the dissimilarity between two strings of equal length. It assesses the number of positions at which the corresponding symbols differ, essentially measuring the minimum number of substitutions required to transform one string into the other (Hamming, 1950). Applied in various domains, including error correction codes, data compression, DNA sequence comparison, and cryptography, Mathematically, this distance between two equal-length strings, denoted as $d_H(s,t)$, is defined as:

$$d_H(s,t) = \sum_{i=1}^{n} [s_i \neq t_i] \tag{1.7}$$

Where:

- s and t represent the two strings of length n,
- $[s_i \neq t_i]$ is an indicator function that equals 1 when the i^{th} symbol of s differs from the i^{th} symbol of t, and 0 otherwise.

For instance, consider two equal-length strings: ”111000” and ”1**0**1**1**00”. The Hamming distance between these strings is 2, as they diverge at the 2nd and 4th positions. In the previous example, the Hamming distance is 2. It summarizes the positions where the corresponding symbols are dissimilar.

1.3.2.4 Euclidean distance

Euclidean distance is a metric used to quantify the dissimilarity or distance between two points or nodes in a multi-dimensional space. This distance measurement is derived from the Pythagorean theorem and involves calculating the square root of the sum of squared differences between the coordinates of two nodes (Danielsson, 1980). Mathematically, the Euclidean distance between two nodes i and j in a graph with n nodes denoted as $DistE(i, j)$, is computed as follows:

$$DistE(i, j) = \sqrt{\sum_{k=1}^{n} (i_k - j_k)^2} \tag{1.8}$$

In machine learning and data science applications, the Euclidean distance serves as a versatile measure with widespread use. Tasks such as clustering, classification, and dimensionality reduction benefit from its utility. This distance metric enables the computation of distances between diverse data points, whether they involve images, sounds, or text, by considering their feature vectors.

1.3.3 Centrality measures

The concept of centrality in a graph involves evaluating the significance of a vertex or edge in the overall structure. There are various measures of centrality, each offering a distinct perspective on importance. Some measures consider the number of edges linked to a vertex, while others assess a vertex’s role in facilitating connections among others (Kempf-Leonard, 2004).

In the field of social network analysis, graph centrality is of crucial importance as it enables the identification of key nodes or actors within a network. Measures of centrality serve as valuable tools for understanding the positioning of individuals in a graph relative to others. Notable centrality measures include closeness, betweenness, PageRank, and degree.

1.3.3.1 Closeness centrality

Closeness centrality serves as an indicator of a node’s proximity to all other nodes in a graph, signifying its ability to quickly interact with its neighboring nodes. This measure is computed

by taking the inverse of the sum of the lengths of the shortest paths between the node of interest and all other nodes within the graph. In essence, the more central a node is, the closer it is to all other nodes in terms of communication efficiency (Eballe & Cabahug, 2021).The formal expression for closeness centrality C(u) of node u is as follows:

$$C(u) = \frac{1}{\sum_v d(u,v)} \tag{1.9}$$

This centrality measure quantifies how well a node can reach other nodes in the graph, making it a vital metric for understanding the accessibility and influence of nodes within a network. It is often used in network analysis to identify nodes that are most central or important in the network, as they are closer to all the other nodes and can quickly reach them.
Additionally, it helps identify nodes that are most vulnerable or critical in the network, as removing these nodes can cause the network to become less connected or more fragmented.

1.3.3.2 Betweenness centrality

Betweenness centrality stands as a pivotal concept within the realm of graph theory, serving as a crucial measure of a node's significance. It delves into the node's role in facilitating the transmission of information throughout a network or graph. A node assumes a central position if it becomes a necessity for numerous shortest paths between pairs of nodes to pass through it. Specifically, betweenness centrality quantifies the number of times a particular node lies on the shortest path connecting any two other nodes within the graph (Bader, Kintali, Madduri, & Mihail, 2007). The mathematical expression for betweenness centrality, denoted as $C_B(V)$ for a node V, takes the following form:

$$C_B(V) = \sum_{i,j,i\neq j} \frac{\sigma_{ij}(v)}{\sigma_{ij}} \tag{1.10}$$

This centrality measure plays a pivotal role in identifying nodes that serve as essential bridges or intermediaries in networks, making it a vital tool in network analysis.

1.3.3.3 PageRank centrality

PageRank centrality, originally introduced by Brin and Page (1998), reflects how users navigate the web and evaluate web page rankings. In this framework, web pages serve as nodes in a graph, and the hyperlinks connecting them represent the edges. The significance of nodes is evaluated by PageRank based on the assumption that a node's importance correlates with the expected sum of the importance of all its linked nodes, considering edge directions.

The assigned PageRank value indicates the probability of randomly accessing nodes. Within the realm of graph theory, PageRank recursively calculates a normalized and propagated value for each node in a given graph (Henni, Mezghani, & Gouin-Vallerand, 2018).

The formula for PageRank in a directed graph is defined by:

$$PR(u) = (1-d) + d\left(\frac{PR(v_1)}{L(v_1)} + \frac{PR(v_2)}{L(v_2)} + \ldots + \frac{PR(v_n)}{L(v_n)}\right) \tag{1.11}$$

where:

- $PR(u)$ is the PageRank score of node u,
- d is the damping factor (commonly set to 0.85),
- $v_1, v_2, \ldots, v_n$ are nodes with directed edges pointing to node u, and
- $L(v_i)$ is the number of outgoing edges from node v_i.

While PageRank is originally designed for directed graphs (Kadavankandy, Avrachenkov, Prokhorenkova, & Raigorodskii, 2015; Grolmusz, 2015), it can be adapted for undirected graphs by considering each edge bidirectional and adjusting the formula accordingly. In an undirected graph, the formula for PageRank can be modified by replacing $L(v_i)$ with the degree of node v_i (the total number of edges incident to v_i). The adapted formula for PageRank in an undirected graph is:

$$PR(u') = (1-d) + d\left(\frac{PR(v_1)}{\deg(v_1)} + \frac{PR(v_2)}{\deg(v_2)} + \ldots + \frac{PR(v_n)}{\deg(v_n)}\right) \tag{1.12}$$

Diverging from the global PageRank, which gauges the significance of web pages across an entire web graph, the local variant concentrates on calculating PageRank scores within a specific subset of web pages or nodes within a network. This localized methodology enables tailored ranking centered around a particular starting node or a predefined set of nodes. Essentially, Local PageRank, as a derivative of the PageRank algorithm, directs its attention to ranking nodes within a subgraph instead of the all-encompassing global ranking of web pages in the original PageRank algorithm.

1.3.3.4 Degree centrality

Degree centrality is a measure used in network analysis to quantify the importance of a node within a graph based on the number of connections it has. It assesses the number of edges connected to a node, indicating the extent of its connectivity within the network (Zhang & Luo, 2017). The degree centrality (C_D) of a node is the count of edges incident to it. Often, degree centrality is normalized by dividing the degree of a node by the total possible number of connections in the network. This normalization facilitates comparisons of node centrality across networks of different sizes. Mathematically, the degree centrality of a node v is calculated as follows:

$$C_D(v) = \frac{\text{Number of edges connected to node } v}{\text{Total possible edges in the network}} \tag{1.13}$$

This measure centrality is a widely adopted and straightforward metric in network analysis. Nodes with elevated degree centrality are perceived as more central and influential due to their

numerous direct connections. This is particularly evident in social networks, where nodes with high degree centrality often symbolize individuals enjoying popularity or engaging in numerous social interactions.

1.4 Comparative analysis of graph metrics

The selection of a metric for graph analysis is a critical decision influenced by specific research objectives and the inherent attributes of the graph under investigation. Each metric introduces a unique set of advantages and limitations, rendering them suitable for distinct scenarios. Careful consideration and application of these metrics enable a nuanced exploration of the intricate aspects of graph structures, contributing to a profound understanding across various domains, with a particular emphasis on the field of community detection. Table 1.1 outlines the commonalities among these three metrics, shedding light on their interconnected relationships and contributing to a nuanced exploration of graph characteristics. Simultaneously.

Metric Aspect	Description
Theoretical Foundation	All three metrics are rooted in graph theory. This theoretical foundation provides a robust mathematical framework for comprehensively understanding the properties encapsulated by these metrics.
Network Analysis	These metrics play a crucial role in network analysis, unraveling the complexities of interconnected systems. They provide valuable insights into the relationships and structures within networks, aiding in the analysis of various domains, including social networks, transportation planning, bioinformatics, and computer vision.
Visualization Aid	This metrics serve as essential tools for graph visualization. They highlight key features within graphs, making them instrumental in visually interpreting complex structures and patterns.
Broad Applications	The utility of these metrics extends across diverse domains, reflecting their versatility. From social network analysis to transportation planning, bioinformatics, and computer vision, these metrics find applications in a wide array of fields.
Graph Algorithms	Graph algorithms, including Dijkstra's shortest path, betweenness centrality, PageRank, and graph isomorphism, compute these metrics.

Table 1.1: Aspects and descriptions of common graph metrics.

In Table 1.2, a comprehensive examination is undertaken to elucidate the distinctions among Similarity, distance, and centrality with respect to two distinct measures. The table meticulously presents and contrasts the characteristics of each measure, shedding light on the nuanced variations between these fundamental concepts in the context of the broader study. This analytical

exploration serves to enhance our understanding of the intricate interplay and divergences inherent in this measures, contributing valuable insights to the overall comprehension of these key metrics in various analytical domains.

Metric Interaction	Description
Similarity vs. Distance	Graph similarity evaluates the resemblance between two graphs, while graph distance computes the shortest path between vertices. These metrics are complementary, with similarity identifying structurally similar graphs, and distance revealing connectivity and accessibility patterns.
Distance vs. Centrality	Graph distance measures shortest path length, while centrality gauges vertex or edge importance. These metrics are connected, as central vertices tend to have shorter distances to others. However, centrality captures broader significance, including a vertex's role in connecting graph components or facilitating information flow.
Similarity vs. Centrality	Graph similarity identifies structural resemblance, while centrality assesses vertex or edge significance. Combining these measures identifies critical structural features; vertices central to both similar graphs may maintain overall structure.

Table 1.2: Interaction between graphical metrics.

1.5 Network centrality and its relation to communities

Network centrality is a fundamental concept in graph theory, which we explained in Section 1.3.3. It enables us to understand the structural and functional importance of network elements. Centrality measures help us understand the role of nodes and edges in facilitating information flow, influence, and communication (Gupta, Singh, & Cherifi, 2016). The centrality of a network is closely linked to the concept of communities within a graph. Communities are subsets of nodes that are more densely connected to each other than to nodes outside the community.

Measures of centrality can provide valuable information about the structure and organization of communities:

- **Community Detection:** Centrality measures can help identify central nodes within communities. Nodes with high centrality might serve as community hubs or connectors that facilitate interactions between different communities.
- **Boundary Nodes:** Nodes exhibiting high betweenness centrality frequently reside on the boundaries between distinct communities within a network. These nodes serve a crucial function by facilitating and maintaining connections between various communities, thereby acting as key mediators in the overall network structure.
- **Community Core:** Nodes with high eigenvector or PageRank centrality within a com-

munity might indicate its core members or influential individuals who have a significant impact on the community's dynamics.

- **Communication Flow:** Closeness centrality can indicate nodes that are well-positioned to quickly disseminate information within a community. These nodes can expedite communication and collaboration.

Understanding the significance of individual nodes within a graph is enriched by network centrality measures. These measures provide valuable insights into the roles nodes play in facilitating information flow, influence, and communication. Moreover, their connection to communities enhances our comprehension of community structure, communication dynamics, and the overall functioning of complex networks.

1.6 Social networks (SN)

Social Network (SN) play a pivotal role in shaping human interactions, communication patterns, and the sharing of information. These dynamic platforms provide diverse opportunities for individuals to establish and maintain relationships, exchange ideas, collaborate, seek support, and engage in various forms of social and cultural activities.

1.6.1 Definitions

Various scholars have proposed distinct definitions of ”social networks”, each contributing unique perspectives to their nature.

According to Boyd and Ellison (2007), ”social networks” encompass applications or websites specifically designed to facilitate user interactions. These platforms serve as hubs for entities engaged in multi-directional communication through network links, encompassing various forms of relationships such as text messages, videos, images, and more.

Fortunato (2010) defines ”social networks” as systems where individuals or groups are interconnected through social relationships, forming graph structures with communities at their core. Meanwhile,

Alotaibi and Rhouma (2022) characterizes ”social networks” as social structures comprising individuals, organizations, or groups linked through diverse relationship types. These relationships span friendships, familial ties, professional connections, or even virtual interactions online. The scope of social networks ranges from small, close-knit groups to extensive communities or even global-scale platforms.

The implications of ”social networks” extend across various disciplines, influencing fields such as sociology, psychology, communication studies, marketing, public health, and beyond. They have fundamentally transformed the way people connect, communicate, and interact, exerting influence on social behaviors, information dissemination, and even political movements. As complex structures, social networks emerge from intricate interactions and connec-

tions among individuals, organizations, or entities, encompassing various types of relationships and giving rise to distinctive characteristics that shape their dynamics and properties.

1.6.2 Social Network Analysis (SNA)

Social Network Analysis (SNA) serves as a methodological framework designed for the exploration of relationships among individuals or nodes within a network. Recognizing the adaptability of these nodes, SNA proves effective in examining both organizational and inter-organizational phenomena. Diverse methodologies within SNA have been applied across various domains, encompassing the analysis of network structures, tracking changes in user relationships, investigating citations in scientific article writing, community detection, exploring commercial contexts such as connections between customers, products, and vendors, and developing systems for recommending products or films to customers.

Toch, Lerner, Ben-Zion, and Ben-Gal (2019) provides a comprehensive exploration of methods and applications of machine learning, enabling large-scale analysis of human mobility data within the SNA framework. In a separate investigation, Rios, Aguilera, Nuñez-Gonzalez, and Graña (2019) focuses on enhancing the semantics of SNA to identify influencers within online social networks. Additionally, B. Liu, Zhou, Ding, Palomares, and Herrera (2019) proposes a decision support model for large-scale groups, utilizing SNA to detect and resolve conflicts based on trust relationships among social network users. Offering a distinctive perspective on social networks, Froehlich and Gegenfurtner (2019) emphasizes their role as a means of social support, facilitating a seamless transition from education to the workplace.

1.6.3 Emergences of social networks

The emergence of social networks is driven by human interactions and affiliations. When individuals connect with each other, whether through friendships, collaborations, familial ties, or professional relationships, a network starts to form. This network evolves as more connections are established, resulting in a web of interrelated nodes representing individuals or entities and edges symbolizing connections between them (Ubaldi, Burioni, Loreto, & Tria, 2021). Social networks can emerge naturally in various scenarios:

- **Friendship Networks:** Individuals establish connections based on shared interests, common experiences, or personal affinities.
- **Collaboration Networks:** Professionals in academic, research, or creative fields collaborate on projects, leading to the formation of networks that connect their work.
- **Online Social Networks:** Digital platforms provide a virtual space for individuals to connect, share, and interact, leading to the formation of online communities.

1.6.4 Characteristics of social networks

Social networks exhibit distinct characteristics that influence their structure and behavior:

- **Nodes and Edges:** Nodes represent individuals, entities, or elements, while edges signify the relationships or connections between them (Alotaibi & Rhouma, 2022).
- **Clustering:** Social networks often exhibit clustering, where nodes are more likely to be connected to nodes that are already connected to each other. This phenomenon creates groups or clusters within the network (Bramer, 2007).
- **Small World Phenomenon:** The small world phenomenon implies that any two individuals in a social network can be connected by a relatively short chain of acquaintances. This characteristic leads to the concept of ”six degrees of separation” (Amin et al., 2015).
- **Homophily:** Individuals in social networks tend to connect with others who share similar attributes, such as interests, demographics, or affiliations (Boyd & Ellison, 2007).
- **Scale-Free Distribution:** Some nodes in social networks have a disproportionately high number of connections, following a power-law distribution. These nodes are often referred to as ”hubs” and play a crucial role in network dynamics (Van den Heuvel & Sporns, 2013).
- **Community Structure:** In social networks, it's common to observe the presence of a community structure, characterized by nodes within a community having more interconnectedness among themselves compared to nodes outside of that community. These communities can embody diverse social units or functional groups within the network (Fortunato, 2010).

The characteristics of social networks make them a rich area of study in fields such as sociology, anthropology, psychology, and computer science. Analyzing these networks provides insights into social dynamics, information propagation, influence spread, and the interplay between individual behavior and network structure. Arising from the interactions and connections among individuals, social networks showcase a myriad of characteristics that shape their behavior and structure. It is crucial to comprehend these traits to delve into the dynamics and ramifications of social networks across various domains, including but not limited to community detection.

1.6.5 Challenges and future directions

While graph based social network analysis has made significant advancements, several challenges and exciting future directions (Ubaldi et al., 2021), as shown here:

- **Data Privacy and Ethical Concerns**: One of the foremost challenges is ensuring data privacy and addressing ethical concerns. As social network data becomes more accessible for research, preserving the privacy of individuals and safeguarding sensitive information becomes paramount. Striking a balance between data availability and user privacy is a complex task that requires innovative solutions and ethical considerations.
- **Scalability and Big Data**: The scale of social network data continues to expand exponen-

tially, posing scalability challenges. Analyzing massive networks requires efficient algorithms and distributed computing techniques. Researchers must develop scalable methods that can handle the ever-growing volume of data while maintaining the accuracy of analysis.

- **Community Detection in Large Networks:** Real-world networks are often multi-layered and heterogeneous, incorporating various types of relationships and attributes. Developing methods to analyze and integrate information from diverse network layers is an exciting direction. Multi-layer community detection and cross-layer analysis hold promise for uncovering hidden patterns and relationships. Community detection algorithms often face difficulties in handling large networks with millions of nodes and edges. Enhancing the efficiency and accuracy of community detection in such networks is a pressing challenge. Novel algorithms and techniques that can identify communities quickly and accurately in large-scale networks will be pivotal for future research.
- **Dynamic and Temporal Networks:** Real-world social networks are dynamic and evolve over time. Traditional static analysis methods may not capture the temporal aspects and changes in network structures. Developing techniques that account for the temporal dynamics of networks is a crucial area of future research, enabling a more accurate understanding of network evolution.

1.7 Conclusion

In this chapter, we delved into the foundational concepts of graph theory, exploring the intricacies of social networks as dynamic structures. The journey began with the fundamental elements of graphs, from vertices and edges to diverse graph types. We navigated through essential measures and metrics, uncovering the significance of similarity, graph distances, and centrality in understanding network structures.

Our exploration extended to social networks, where graphs serve as a powerful abstraction for unraveling the complexities inherent in social interactions. We dissected various metrics, such as closeness centrality and PageRank centrality, shedding light on the role of nodes within communities. The chapter provided a comprehensive understanding of social networks and their emergent properties, laying the groundwork for the subsequent exploration of their intersection with machine learning.

In the next chapter, the focus will shift to Machine Learning (ML), where the union of data and algorithms unfolds new dimensions for analysis, prediction, and decision-making. The interconnected nature of graph theory and social networks, established here, serves as a springboard for our foray into the world of ML. The knowledge garnered in this chapter will prove instrumental in comprehending the symbiotic relationship between social interactions and the evolving landscape of intelligent data analysis. Let's embark on the exploration of Machine Learning.

Chapter 2

Machine Learning (ML)

2.1 Introduction

At the heart of Artificial Intelligence (AI), Machine Learning (ML) plays a pivotal role in shaping algorithms and models. It empowers computers to extract knowledge from data, predict outcomes, and make decisions autonomously, eliminating the need for explicit programming. This transformative field focuses on enabling machines to analyze intricate data patterns and relationships automatically. The learning process in machine learning unfolds through the iterative extraction of significant features and patterns from extensive datasets, known as training data. These identified patterns form the foundation for constructing mathematical models or algorithms capable of generalizing their understanding. This generalization enables these models to make predictions or decisions when presented with new, unseen data. The refinement of these models occurs through the adjustment of internal parameters guided by patterns and feedback from the training data.

Machine learning's applicability extends across diverse domains, from image and speech recognition to natural language processing, recommendation systems, fraud detection, community detection, autonomous vehicles, and healthcare. Its evolution is driven by the availability of extensive datasets, enhanced computational capabilities, and continuous advancements in algorithms and techniques.

This chapter unfolds as a coherent narrative, commencing with an overview of machine learning and its various categories, with a specific emphasis on unsupervised learning. A comparative analysis of these methods follows, shedding light on their respective strengths and applications. The chapter concludes with considerations of advantages and challenges arising from the combination of unsupervised learning and graph analysis, providing a comprehensive perspective on the field's ongoing development in conjunction with graph theory.

2.2 Machine learning categories

The categorization of machine learning tasks is a pivotal aspect in understanding the diverse applications within the field. These tasks are commonly categorized into three main groups, as illustrated in the figure 2.1: Supervised Learning (SL), Unsupervised Learning (UL), and Reinforcement Learning (RL) (S. Russell, 2016).

The importance of studying these categories lies in delineating the nature of the learning signal or feedback available to a learning system, offering distinct approaches to algorithm training and adaptation. This classification provides a framework for understanding the various methodologies employed in ML, each tailored to specific learning objectives and problem-solving scenarios (Nasteski, 2017).

In the subsequent sections, we delve into each category, exploring their unique characteristics, algorithms, and applications.

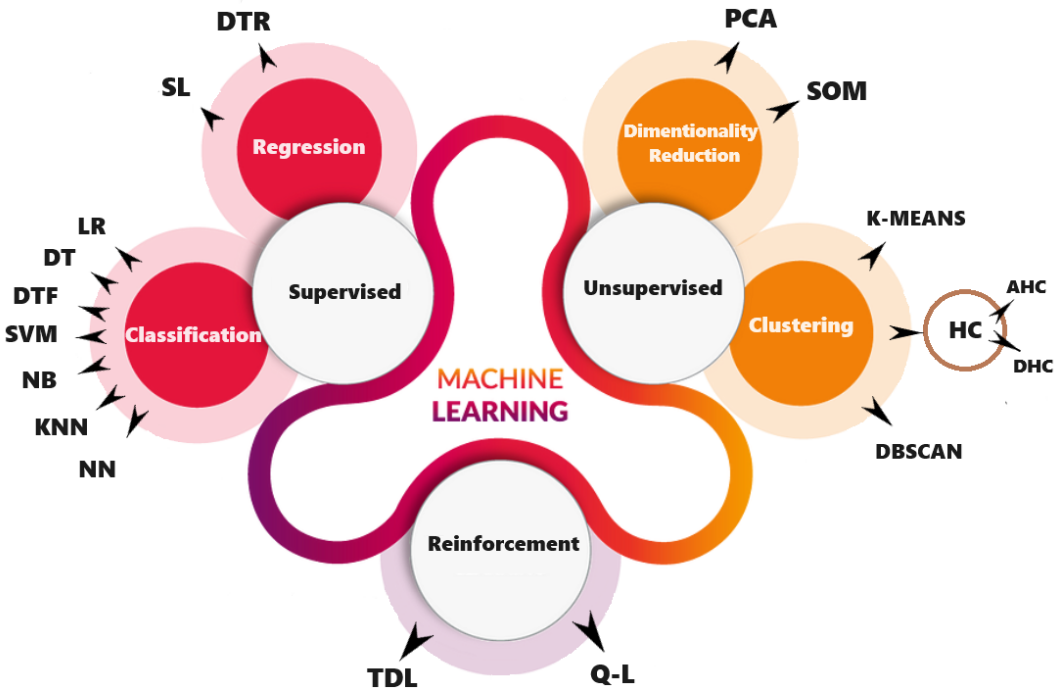


Figure 2.1: Machine learning categories.

2.2.1 Supervised Learning (SL)

Supervised Learning (SL) constitutes a Machine Learning methodology wherein an algorithm is trained to comprehend the connection between input data and corresponding output labels. Within this framework, the algorithm is furnished with a collection of labeled training instances, where each instance comprises input data coupled with its anticipated output. These training instances serve as the algorithm's educational basis, enabling it to extrapolate and provide predictions for fresh, unencountered data points (Cunningham, Cord, & Delany, 2008; Learned-Miller, 2014; Uddin, Khan, Hossain, & Moni, 2019).

The main objective of supervised learning is to enable the algorithm to accurately map input data to the correct output labels based on the patterns and relationships present in the training data. Through an iterative process, the algorithm adjusts its internal parameters to minimize the discrepancy between its predicted outputs and the true labels in the training set. This iterative process is often referred to as model training or model optimization (B. Liu, 2011).

During the training phase, the algorithm analyzes the input features and tries to identify the underlying patterns that lead to the correct output labels. By learning from the labeled training examples, the algorithm can generalize its understanding and make accurate predictions on new, unlabeled data instances.

After the training phase, the supervised learning model can be deployed to predict output labels for new input data. The model applies the learned patterns and relationships to the unseen

data, providing predictions or classifications based on its acquired knowledge.

A wide range of applications in all fields use supervised learning. It can be used for tasks such as image classification, sentiment analysis, spam detection, medical diagnosis, and stock market prediction, among others. The availability of labeled training data is a crucial requirement for supervised learning, as the quality and quantity of the training data greatly influence the performance of the model.

So we can say that the supervised learning is a Machine Learning approach that involves training an algorithm using labeled input-output pairs. It aims to learn the underlying patterns and relationships between input features and output labels, enabling the algorithm to make accurate predictions on new, unseen data. Supervised learning has numerous applications and plays a vital role in solving real-world problems by leveraging the power of labeled training data. Supervised learning is generally performed in the context of **classification** and **regression** (B. Liu, 2011). Indeed, there are highlighted an important distinction between regression and classification algorithms. While both regression and classification are tasks in Machine Learning, they differ in the type of output they aim to predict.

While regression algorithms predict continuous values and classification algorithms predict discrete values, there can be some overlap or variations. For example, logistic regression is a regression algorithm commonly used for binary classification tasks, where it predicts the probability of an instance belonging to a particular class. Additionally, there are regression algorithms that can handle multiple classes, known as multiclass regression or multivariate regression. Understanding the distinction between regression and classification is essential for selecting the appropriate algorithm and techniques based on the nature of the problem and the type of output variable being predicted.

2.2.1.1 Classification

Classification is a fundamental task in machine learning, wherein the objective is to construct a function or model that assigns data instances to predefined classes or categories. This involves considering various parameters or characteristics. Classification algorithms play a crucial role in predicting discrete or categorical values, commonly known as class labels.

The primary aim of classification is to develop a computer system capable of efficiently classifying new, unobserved data instances into their respective classes. This is achieved by leveraging knowledge acquired from a training data set. These algorithms learn from labeled training data to create a decision boundary or decision function that accurately classifies new, unseen instances.

Various real-world applications of classification tasks include email spam detection, sentiment analysis (determining positive or negative sentiment), disease diagnosis (healthy or diseased), and image classification (identifying objects or animals in images).

In the classification process, a Machine Learning algorithm is trained using a labeled training dataset. This dataset comprises input data samples along with their corresponding class labels. The algorithm analyzes and learns the patterns and relationships present in the training data. The goal is to create a decision boundary or decision function capable of effectively separating different classes.

During the training phase, the algorithm adjusts its internal parameters or weights through an optimization process to minimize classification errors. It identifies distinguishing characteristics or features indicative of each class.

Once the classification model has been trained, it can be applied to new, unlabeled data instances for predicting their class memberships. The model utilizes the learned decision boundary or decision function to assign the input features of unseen data to the most appropriate class based on the learned patterns. This process is often referred to as inference or prediction.

Classification algorithms can be categorized into several types based on their underlying principles and methods. The choice of the specific learning algorithm depends on the problem at hand and may include approaches such as Logistic Regression (LR), Decision Trees (DT), Random Forest (RF), Support Vector Machines (SVM), Naive Bayes (NB), K-Nearest Neighbors (KNN), and Neural Networks (NN) (Nasteski, 2017). Each algorithm possesses unique characteristics and is suitable for different types of problems.

2.2.1.1.1 Logistic Regression (LR)

Logistic Regression (LR) is a statistical technique that establishes a model connecting input features with the likelihood of membership in a specific class. It finds frequent application in binary classification tasks and can be expanded to address multiclass classification challenges as needed.

2.2.1.1.2 Decision Trees (DT)

Decision Treess (DTs) are hierarchical structures that use a sequence of binary decisions to classify instances. Each decision node represents a feature and a threshold, and each leaf node corresponds to a class label. Decision trees are interpretable and can handle both categorical and numerical features (Roudiere, 2018; Rokach & Maimon, 2005).

2.2.1.1.3 Random Forest (RF)

Random Forest (RF) is an ensemble learning technique that amalgamates numerous decision trees. It generates a varied ensemble of decision trees by employing random subsets of both training data and features. The ultimate prediction is then determined by aggregating the forecasts of these individual trees, typically leading to enhanced accuracy and resilience.

2.2.1.1.4 Support Vector Machine (SVM)

Support Vector Machine (SVM) stands as a robust algorithm tasked with discovering the optimal hyperplane within a high-dimensional space for the purpose of segregating data into distinct classes. Its objective revolves around maximizing the margin, which is the gap between the decision boundary and the closest data points. SVM exhibits versatility by effectively addressing both linear and non-linear classification challenges, achieved through the application of kernel functions (Jakkula, 2006; Noble, 2006; Mahdavinejad et al., 2018).

2.2.1.1.5 Naive Bayes (NB)

Naïve Bayes (NB) is a probabilistic algorithm that applies Bayes theorem with the assumption of independence among features. It calculates the probability of each class given the input features and selects the class with the highest probability. It is computationally efficient and can handle high-dimensional data (Cornfield, 1967; Rokach & Maimon, 2005).

2.2.1.1.6 K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is a simple yet effective algorithm that classifies instances based on their proximity to the nearest neighbors in the training data. It assigns the majority class label among the k nearest neighbors. KNN is non-parametric and can handle both binary and multiclass classification problems (Samet, 2007; Mahdavinejad et al., 2018).

2.2.1.1.7 Neural Networks (NN)

Neural Networks (NNs) are a fundamental component of Machine Learning and artificial intelligence, inspired by the structure and functioning of the human brain. They are computational models composed of layers of interconnected nodes, often referred to as neurons or artificial neurons. These nodes are organized into layers, including an input layer, one or more hidden layers, and an output layer. Neural Networks components of different Layers type:

- **Input Layer:** This layer receives the initial data or features that the neural network will process.
- **Hidden Layers:** These layers process the input data through a series of weighted connections. Each connection has an associated weight that the network learns to adjust during training.
- **Output Layer:** The final layer produces the network's output, which could be a classification, regression, or some other task depending on the nature of the problem.
- **Activation Function:** Each node in a neural network typically applies an activation function to its input. This introduces non-linearity to the model, enabling it to learn complex patterns.

The training process involves feeding the network with labeled data, allowing it to make predictions, comparing these predictions to the actual labels, and adjusting the weights to minimize

the error. This process is often done using optimization algorithms like stochastic gradient descent. NN have shown remarkable success in various applications, including image and speech recognition, natural language processing, and game playing. Deep learning, a subset of Machine Learning, focuses on neural networks with multiple hidden layers (deep neural networks), allowing them to learn intricate patterns and representations from large datasets

2.2.1.2 Regression

Regression algorithms are specifically designed for predicting continuous numerical values. The primary goal is to model the relationship between input features and a continuous target variable, allowing the algorithm to generate predictions for new, unseen data instances. Regression tasks encompass diverse domains, such as forecasting real estate prices, predicting stock market values, estimating temperature fluctuations, or projecting sales trends (Nasteski, 2017).

These algorithms aim to capture a functional relationship between dependent and independent variables to predict continuous outcomes. The core objective is to determine a mapping function that relates input variables (denoted as x) to a continuous output variable (denoted as y). Unlike classification tasks, where the output is a discrete class label, regression outputs a numerical value, such as a percentage or an absolute quantity. For instance, regression can be employed to predict the likelihood of a machine breaking down (Uddin et al., 2019).

Various regression algorithms exist, each employing distinct methodologies for modeling relationships and generating predictions. Examples encompass Simple Linear Regression (SLR) and Decision Tree Regression (DTR). These algorithms play a pivotal role in tasks where the comprehension and prediction of continuous numerical trends are indispensable.

2.2.1.2.1 Simple Linear Regression (SLR)

Simple Linear Regression (SLR) models the relationship between a dependent variable and a single independent variable. The relationship is typically linear, represented by a straight line. The dependent variable must be a continuous value, while the independent variable can be continuous or categorical (Maulud & Abdulazeez, 2020).

2.2.1.2.2 Decision Tree Regression (DTR)

Decision Tree Regression (DTR) approximates real functions, including class proportions. It uses binary recursive partitioning to divide data into partitions based on splits that minimize the sum of squared deviations from the mean. The process continues recursively until reaching a minimum node size, resulting in terminal nodes that provide predictions (Xu, Watanachaturaporn, Varshney, & Arora, 2005).

2.2.2 Unsupervised Learning (UL)

Unsupervised Learning (UL) is a type of Machine Learning that involves having only input data and no corresponding output variables. In this type of learning, algorithms are left to their

own mechanisms to discover and present the interesting structure within the data; there is no correct answer or teacher (Alloghani, Al-Jumeily, Mustafina, Hussain, & Aljaaf, 2020).

The answers that we seek to predict are not available in the datasets. Here, the algorithm uses unlabeled datasets. The machine is then asked to create its own answers. It proposes responses based on data analysis and grouping. Unsupervised learning consists of two categories of algorithms: Clustering algorithms and Dimensionality Reduction (DR) algorithms (Bonaccorso, 2017).

2.2.2.1 Clustering

Clustering entails the partitioning of a dataset into distinct classes, known as clusters, grounded on their similarities. The objective is to ensure that data objects within the same cluster exhibit maximum similarity, while those in different clusters are as dissimilar as possible. Clustering results in the grouping of similar data and the separation of dissimilar data. Unlike classification tasks, the predefined classes or groups are not known in advance(Madhulatha, 2012; Hardy, 2019; Samba, 2018).

2.2.2.1.1 K-means clustering

K-means clustering, originally proposed by MacQueen et al. (1967), stands out as one of the most renowned and widely used clustering algorithms (a key choice for our work). It represents the fundamental category of centroid-based partitioning algorithms. This method aims to minimize the distance between data points and their centroids within a cluster, serving as one of the simplest machine learning clustering algorithms capable of automatically recognizing groups of similar objects in a database. The algorithm divides the data (objects) into K predefined groups or clusters, specified by the user, ensuring that similar data points are grouped together.

This algorithm is a well-established clustering algorithm that partitions a dataset into K clusters, where each data point belongs to the cluster with the nearest mean. Originally designed for continuous data, K-means can also be adapted for graph-based community detection (Mawloud, 2017). It undeniably stands as the most well-known and widely used partitioning method across various scientific and industrial applications, such as spam detection, filtering, and identifying fake news. This success is attributed to the algorithm's favorable cost-effectiveness ratio. The K-means algorithm is characterized by its speed, ease of implementation, and ability to handle large amounts of data.

Figure 2.2 illustrates the K-means algorithm's process, the first crucial step involves randomly selecting k objects to serve as initial centroids. The algorithm then proceeds to examine each object in the data set and assigns it to the nearest cluster. Various methods can measure the distance between the object and the centroid, with the Euclidean distance being the most popular. The centroids of the newly formed clusters are continuously recalculated, iterating this process until no further changes occur (Ahmed, Seraj, & Islam, 2020).

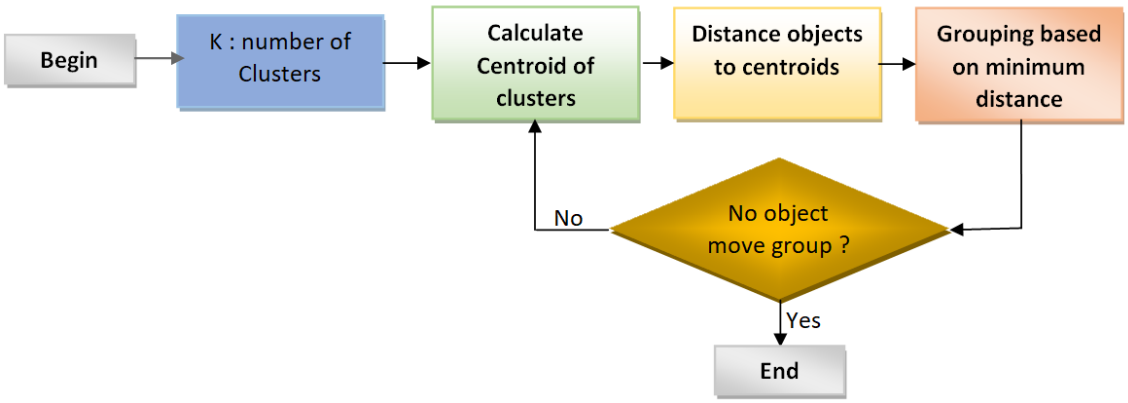


Figure 2.2: Flowchart illustrating the process of the K-means algorithm.

The k-Means algorithm is elucidated in the following steps (Syarif, Prugel-Bennett, & Wills, 2012; Yahi, 2019; Khatir & Nait-bahloul, 2018):

Step 1: Choose the number of clusters (k).

Step 2: Randomly select k points and set them as centroids (Select k random points in the data as centroids: Then, randomly assign each cluster's centroid).

Step 3: Calculate the distance between each object and all centroids (using the Euclidean method, for example).

Step 4: Assign each object to the centroid of the nearest cluster.

Step 5: Recalculate the centroids of the newly formed clusters. The calculation of centroids for the clusters is done by taking the average of all data points that belong to each cluster.

Step 6: Repeat steps 3 to 5 until the algorithm converges or reaches a maximum number of iterations. After a few iterations, the algorithm finds a stable partitioning of the data set, and we say that the algorithm has converged.

To apply K-means for community detection in graphs, we must map graph nodes into feature vectors suitable for K-means clustering. This can be achieved by extracting relevant graph-based features, such as node attributes or structural properties. In the upcoming chapter, we will delve into the intricacies of using the K-means clustering algorithm to identify and analyze communities.

2.2.2.1.2 DBSCAN

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a highly versatile unsupervised clustering algorithm widely recognized for its capability to discover clusters of diverse shapes and effectively handle noisy datasets. Initially proposed by Ester, Kriegel, Sander, Xu, et al. (1996), it operates by identifying clusters based on the density of data points.

The algorithm categorizes data points into core points, border points, or noise points. Core points have a sufficient number of nearby points within a user-defined radius (eps), while border points have fewer neighbors but fall within the neighborhood of a core point. Noise points fail to meet the criteria for core or border points. The DBSCAN sequence unfolds as follows:

1. Select an unvisited data point from the dataset. Collect neighboring points within the specified radius (eps) and assess if they surpass a minimum threshold (minPts). If so, the point is identified as a core point, and a new cluster is established.
2. Expand the cluster iteratively by exploring neighbors of core points. If a neighbor is also a core point, their neighbors are added to the cluster, continuing until no more core points are found.
3. Repeat steps 1-2 for all unvisited data points until each has been visited.
4. Unvisited data points not assigned to any cluster are classified as noise points.

DBSCAN offers several advantages over traditional clustering algorithms. It excels at identifying clusters with diverse shapes and robustly handles data noise and outliers. Additionally, it eliminates the need to predefine the number of clusters, making it particularly suitable for datasets with an unknown cluster count.

The effectiveness of DBSCAN relies on selecting appropriate values for the eps and minPts parameters. These parameters determine the density criteria for core points and the minimum points required to constitute a cluster. As the algorithm is sensitive to parameter choice, different settings may yield different outcomes (Madhulatha, 2012; Scicluna & Bouganis, 2015).

2.2.2.1.3 Hierarchical Clustering (HC)

The Hierarchical Clustering (HC) algorithm is an unsupervised machine learning technique used to group unlabeled data points with similar characteristics into clusters. Unlike some other clustering methods, HC takes a distinctive approach by constructing a hierarchy of clusters through the progressive merging or division of data points.

The versatility of the Hierarchical Clustering algorithm extends across diverse domains, encompassing biology, pattern recognition, genomic data analysis, and other fields where uncovering similar groups within data is essential (Hardy, 2019). A notable advantage of this method is its deterministic nature, as it is not subject to specific parameter initializations. Moreover, the flexibility of not requiring a fixed number of classes a priori enhances its applicability (Boubou, 2007). There are two primary approaches in HC:

A. Agglomerative Hierarchical Clustering (AHC)

In Agglomerative Hierarchical Clustering (AHC), the process begins by treating each data point as an individual cluster. Subsequently, similar clusters are iteratively merged to form larger clusters. This merging continues until all data points are grouped into clusters (Murtagh & Contreras, 2012).

B. Divisive Hierarchical Clustering (DHC)

In Divisive Hierarchical Clustering (DHC), the algorithm starts with all data points grouped into a single cluster. It then systematically divides clusters into smaller subclusters. This division process is repeated iteratively until each data point resides in its own individual cluster.

2.2.2.2 Reducing Dimensionality (RD)

This assignment revolves around the process of condensing input data into a more compact representation space. Beyond the advantage of conserving storage capacity, dimensionality reduction serves the purpose of enhancing data representation. The application of Reducing Dimensionality (RD) proves instrumental in addressing the complexities linked to the curse of dimensionality(Hardy, 2019).

This curse encapsulates the challenges encountered when dealing with high-dimensional data. One illustrative issue is the diminished efficacy of nearest neighbor search algorithms when applied to datasets with a substantial number of dimensions (Jonathan, Goldstein, Ramakrishnan, & Shaft, 1999). Various techniques exist for accomplishing dimensionality reduction, include PCA and SOM.

2.2.2.2.1 Principal Component Analysis (PCA)

Principal Components Analysis (PCA) is a statistical technique used for the analysis and visualization of datasets comprising multiple quantitative variables. Particularly valuable in exploring multivariate data, where each variable represents a distinct dimension, PCA synthesizes these insights into a limited set of new variables known as principal components. These principal components are essentially linear combinations of the original variables, and their number is typically equal to or less than the number of original variables. PCA operates by reducing the dimensions of a multivariate dataset to just two or three principal components. This reduction enables graphical visualization while retaining as much pertinent information as possible from the initial dataset and minimizing the loss of valuable information.

2.2.2.2.2 Self-Organizing Maps (SOM)

Self-Organizing Maps (SOM), commonly referred to as Kohonen maps, facilitate the nonlinear projection of individuals from a training set onto a structured topological space known as a map. This space is typically defined by a two-dimensional regular grid, where each node can be associated with a position in the feature space. The method offers a meaningful representation of data in two dimensions and is frequently employed in exploratory data analysis (Kohonen, 1982).

2.2.3 Reinforcement Learning (RL)

Reinforcement Learning (RL) is commonly used to teach machines to perform a sequence of steps. It differs from supervised and unsupervised learning. Scientists program an algorithm

to perform a task, providing positive or negative feedback as it learns how to accomplish the task. The programmer sets the rules for rewards but allows the algorithm to decide the steps to maximize the reward and accomplish the task (Barto & Sutton, 1998; Kaelbling, Littman, & Moore, 1996). in subsequent sessions, we will mention two reinforcement methods learning:

2.2.3.1 Q-Learning (Q-L)

Quality-Learning (Q-L) constitutes a variant of model-free reinforcement learning and can be regarded as an asynchronous dynamic programming approach. Its primary function is to empower agents to acquire knowledge about optimal actions within Markovian environments through the experiential understanding of action outcomes. Importantly, this learning process transpires without the necessity to construct comprehensive maps of the domains in question (Watkins & Dayan, 1992).

2.2.3.2 Temporal Difference Learning (TDL)

Temporal Difference Learning (TDL), an unsupervised learning method frequently employed in the realm of reinforcement learning, plays a pivotal role in forecasting the cumulative reward expected in future interactions with an environment. Beyond its primary application in reward prediction, TDL exhibits versatility and applicability in predicting various other quantities related to sequential data. At its core, TDL operates as a mechanism for acquiring the capability to forecast a specific quantity based on the forthcoming values of a given signal or state. This process involves iteratively updating predictions as new information becomes available through the ongoing interactions with the environment (Sutton, 1988).

2.3 Comparison between categorization of machine learning

Supervised Learning, Unsupervised Learning, and Reinforcement Learning are three distinct paradigms within the field of Machine Learning, each employing unique approaches and finding applications in diverse domains. When evaluating various Machine Learning (ML) methods, it becomes crucial to consider factors such as their fundamental principles, strengths, weaknesses, and suitability for specific problem types. The subsequent table (Table 2.1) provides a comparative overview of prevalent ML methods, shedding light on their features and applications. The choice of a particular ML method depends on factors such as the nature of the problem at hand, data availability, computational resources, and the intended outcome. Combining different methods or techniques is common to effectively tackle intricate tasks, leveraging the diverse characteristics of each approach.

2.4 Combining unsupervised learning with graph analysis

The integration of unsupervised learning techniques with graph analysis provides a versatile approach for uncovering hidden structures in complex networks. By leveraging the strengths of both paradigms, researchers can enhance their understanding of the organization, behavior,

	SL	UL	RL
Description	Learns from labeled data with input-output pairs.	Learns from unlabeled data to discover patterns or structures.	Learns through trial and error interactions with an environment.
Highlights	Can make accurate predictions when labeled data is available.	Can uncover hidden relationships and structures in data.	Can learn optimal strategies in dynamic and uncertain environments.
Limitations	Relies on the availability of labeled data, may not generalize well.	No explicit output labels, interpretation may be subjective.	Requires careful reward design, can be sample inefficient.
Applicability	Well-suited for classification and regression when labeled data is abundant.	Useful for clustering, anomaly detection, and dimensionality reduction.	Suitable for sequential decision-making tasks like robotics and game playing.
Training Data	Labeled data	Unlabeled data	Feedback through rewards
Objective	Prediction Classification	Pattern discovery	Maximize cumulative reward.
Examples	Image classification	Community detection	Game playing
Advantages	Clear objectives	Discover hidden patterns	Sequential decision-making
Disadvantages	Requires labeled data	Lack of clear objectives	Exploration vs. exploitation

Table 2.1: Comparison of common machine learning methods.

and relationships within various types of networks. Clustering approaches include K-Means clustering, hierarchical clustering, and others functioning without labeled training data, categorizes similar data points and finds applications in tasks such as data segmentation, community detection, and pattern recognition. The choice of a specific clustering algorithm depends on the characteristics of the data and the nature of the problem (Philip, Han, & Faloutsos, 2010).

Hybrid approaches, integrating unsupervised learning with graph analysis, leverage the strengths of both paradigms, exemplified by using community detection algorithms for preprocessing before applying clustering methods, ultimately enhancing accuracy in uncovering complex network structures.

2.4.1 Integration strategies

Integrating unsupervised learning into graph analysis provides a better understanding of the social structure within the network, revealing clusters of nodes that are more densely connected

to each other than the rest of the network. This information can be valuable in a variety of areas, including targeted marketing, recommendation systems and understanding social dynamics. this integration involves various strategies:

- **Feature Extraction:** Unsupervised learning techniques can be employed to extract informative features from graph data. These features can include graph-based metrics, node attributes, or even embeddings learned through techniques.
- **Preprocessing:** Unsupervised learning can aid in preprocessing tasks, such as dimensionality reduction, noise reduction, and outlier detection, before applying graph analysis methods.
- **Hybrid Models:** Combining unsupervised learning models with traditional graph algorithms or Machine Learning methods can leverage the strengths of both approaches. For example, using unsupervised clustering results as inputs for subsequent community detection algorithms.
- **Graph-Based Learning:** Unsupervised learning can be incorporated into graph-based learning frameworks, where graph structure and node attributes are jointly considered to solve tasks like node classification and link prediction.

2.4.2 Applications of combining

The integration of unsupervised learning with graph analysis has found applications in various domains, including:

- **Bioinformatics:** Identifying functional modules in biological networks.
- **Social Networks:** Detecting communities and influential nodes in social networks.
- **Recommendation Systems:** Enhancing personalized recommendations using user-item interaction graphs.
- **Fraud Detection:** Identifying anomalous patterns in financial transaction networks.
- **Community Detection:** In social network analysis, the integration of unsupervised learning and graph analysis is often employed for community detection. Consider a social network where nodes represent individuals, and edges represent connections between them (friendship, interaction, etc.). By applying unsupervised learning techniques, such as clustering algorithms like k-means or DBSCAN clustering, on the graph data, one can identify communities or groups of individuals who share similar characteristics, interests, or patterns of interaction.

2.4.3 Advantages from the combining

Integrating unsupervised learning into graph analysis offers a number of advantages:

- **Dimensionality Reduction:** Unsupervised learning methods like Principal Component Analysis (PCA) can reduce the high dimensionality of graph data, enabling better visual-

ization and interpretation (Teletin, Czibula, Albert, & Bocicor, 2018).

- **Cluster Identification:** Unsupervised clustering algorithms can help identify potential communities within a graph by grouping nodes with similar attributes or connectivity patterns.
- **Anomaly Detection:** Unsupervised techniques can highlight nodes or edges that deviate from the expected patterns, aiding in the identification of outliers or anomalies.
- **Embedding Learning:** UL are capable of learning vector representations of nodes. These representations capture both structural and semantic information within a graph. Subsequently, these embeddings can be utilized for downstream graph analysis tasks.
- **Enhanced Insight:** Unsupervised learning techniques can uncover hidden patterns and relationships that traditional graph analysis might miss.
- **Flexibility:** The combination of methods provides a flexible framework that can adapt to various types of network data.
- **Comprehensive Understanding:** By utilizing diverse techniques, researchers can achieve a comprehensive understanding of complex networks.

2.4.4 Challenges from the combining

Indeed, the promising fusion of unsupervised learning and graphical analysis is not devoid of challenges:

- **Algorithm Selection:** Choosing the right combination of unsupervised learning and graph analysis methods can be complex and context-dependent.
- **Computational Complexity:** Some hybrid approaches might require substantial computational resources due to the combination of techniques.
- **Interpretability:** Combining different methods can sometimes make the interpretation of results more challenging.

2.5 Conclusion

Machine learning is a powerful field, empowering computers to discern patterns and make predictions or decisions without explicit programming. The spectrum of machine learning algorithms encompasses three fundamental categories: supervised learning, unsupervised learning, and reinforcement learning. The selection of a category depends on the nature of the available data and the chosen learning method.

In unsupervised learning, the primary goal is to discover and extract meaningful patterns from unlabeled data. This pursuit involves two main classes of methods: clustering and dimensionality reduction. Clustering algorithms, such as K-means, group similar data points based on their intrinsic characteristics, revealing natural groupings. Simultaneously, dimensionality reduction techniques reduce the number of features while preserving crucial information, en-

hancing data visualization, and simplifying feature engineering.

Within the realm of unsupervised learning, K-means clustering stands out as a prominent and widely used algorithm. Its key function is to segment data points into K-means clusters, unraveling inherent structures within the data. This is particularly relevant in the context of community detection, which we will look at in detail in the next chapter.

Chapter 3

Community Detection

3.1 Introduction

Community detection, also referred to as community identification or clustering, constitutes a pivotal undertaking in network analysis. It is a systematic endeavor aimed at delineating groups of nodes or individuals within a network, characterized by dense interconnections amongst themselves and comparably sparse links to the broader network fabric. This intricate pursuit underpins the foundational framework of network comprehension and extends its ramifications across diverse domains, spanning social network analysis, biology, computer science, and sociology.

At its core, the crux of community detection revolves around unveiling the latent architecture of a network. This is executed through the meticulous partitioning of the network into discrete and cohesive communities or clusters. These communities, in essence, epitomize congregations of nodes that stand united by shared attributes, akin interests, or synchronous functions within the intricate network weave. The endeavor of community detection assumes the mantle of deciphering the intricate threads that weave these clusters together, casting light on the cohesive forces that mold them and the inherent dynamics that propel their growth and evolution.

As we delve into the chapters ahead, the exploration of community detection unfurls as an essential odyssey into the heart of network structures and their manifestations in the real world. The ramifications span from dissecting the patterns of interactions in social ecosystems to comprehending the organizational principles underlying intricate biological networks. This chapter unfurls as an illuminating excursion into the landscape of community detection, unraveling its methodologies, significance, and multifarious applications.

3.2 Concept of communities

In the realm of graph theory, a ”community” denotes a subset or cluster of nodes within a graph that demonstrates a heightened level of interconnectedness or similarity compared to nodes external to the group. These communities are alternatively termed ”clusters,” ”modules,” or ”subgraphs.” Within graph structures, communities play a pivotal role in unveiling concealed patterns and relationships, thereby fostering a deeper comprehension of the intricate systems they represent and facilitating diverse applications across various domains (Fortunato, 2010).

The concept of communities holds particular significance in the exploration of complex networks, encompassing domains such as social networks, biological networks, citation networks, and more. The identification of communities within a graph unveils latent structures, patterns, or groups of interconnected entities, offering valuable insights for diverse analytical purposes, including:

Social Network Analysis: Within social networks, communities often represent clusters of individuals sharing similar interests, social circles, or affiliations. Identifying these communities proves invaluable in unraveling the dynamics of social connections, offering insights that can be

leveraged for optimizing targeted advertising and enhancing content recommendations (Ubaldi et al., 2021).

Biology: Within biological networks, communities may correspond to functional modules within a biological system, such as protein-protein interaction networks. The identification of these modules sheds light on the organization and functioning of biological systems (Sah, Singh, Clauset, & Bansal, 2014).

Recommendation Systems: Community detection proves useful in recommendation systems by pinpointing groups of users with similar preferences or behaviors, enabling personalized recommendations (Gorripati & Vatsavayi, 2017).

Citation Networks: In academic citation networks, communities may align with distinct research areas or subfields. This assists researchers in exploring the structural layout of academic knowledge (Gao, Pan, Wang, Yang, & Zhang, 2021).

Anomaly Detection: Communities offer a means to identify anomalies or outliers within a network. Nodes that do not align with any community or serve as bridges between multiple communities may warrant particular attention (Z. Chen, Hendrix, & Samatova, 2012).

Various algorithms and methodologies exist for community detection in graphs, with the selection of a specific method contingent upon the unique characteristics of the graph and the analytical objectives. Common approaches encompass modularity-based methods, spectral clustering, hierarchical clustering, and more.

3.3 Objectives and importance of community detection

Community detection refers to the process of identifying groups of nodes within a network that are more densely connected to each other than to the rest of the network. The objectives and importance of community detection can be summarized as follows (Shi, Yu, Cai, Yan, & Wu, 2011; Shi, Yan, Cai, & Wu, 2012; Moayedikia, 2018):

- **Structure Identification:** Community detection aims to identify groups of nodes within a network that are more densely connected to each other than to nodes outside the group. These groups, known as communities or clusters, represent cohesive subsets of nodes that share common attributes, interactions, or roles.
- **Pattern Recognition:** By identifying communities, researchers can discern recurring patterns in network data. This can reveal how entities interact, form relationships, and influence each other, offering a deeper understanding of the network's underlying dynamics.
- **Information Flow Analysis:** Community detection helps analyze how information or influence spreads within a network. Nodes within the same community tend to share information more readily, and studying these patterns of information flow can aid in understanding trends, propagation mechanisms, and potential bottlenecks.

- **Visualization:** Communities provide a concise representation of complex network structures. Visualizing communities can simplify the comprehension of intricate relationships and allow researchers to focus on specific subsets of nodes, aiding in exploratory analysis.
- **Prediction and Recommendation:** Communities can be used to predict missing links or future interactions. If nodes within a community exhibit similar behavior or interests, the likelihood of connections forming between them in the future increases. This predictive power can extend to recommendation systems in various domains.
- **Network Understanding:** Community detection provides a lens through which researchers can dissect the intricate organization of networks. Understanding communities sheds light on how nodes collaborate, share information, form alliances, and influence one another.
- **Identifying Key Players:** Communities often have influential nodes that act as connectors between different clusters. Detecting these key players, also known as ”hubs,” helps in pinpointing nodes with substantial impact on the network’s structure and functioning.
- **Domain-Specific Insights:** In fields like sociology, biology, marketing, and economics, community detection reveals social, biological, or economic subdivisions within networks. These insights can drive targeted interventions, marketing strategies, or policy decisions.
- **Anomaly Detection:** Communities can help identify anomalies or outliers within networks. Nodes that do not align with the characteristics of their community may indicate unusual behavior, fraud, or emerging trends that merit investigation.
- **Network Evolution:** Studying how communities change over time enables researchers to track network evolution. This longitudinal analysis aids in understanding shifts in relationships, identifying emergent trends, and adapting strategies accordingly.
- **Uncovering Insights from Network Structures** Community detection plays a crucial role in unraveling the hidden structures and relationships within complex networks. By identifying communities, we can gain insights into the organization, interactions, and behaviors of nodes in a network.
- **Real-world applications and implications** The significance of community detection extends across various domains. It provides valuable information in social network analysis, aiding in understanding information flow, influence propagation, and the formation of social groups. Additionally, it finds applications in biological networks, identifying functional modules within protein interaction networks. Community detection also holds relevance in recommendation systems, fraud detection, and other data-driven applications.

3.4 Traditional approaches for community detection

Conventional methods for community detection in network analysis have undergone examination by physicists, sociologists, and computer scientists. These methods are designed to

partition a network into distinct sub-structures, with the goal of identifying clusters of nodes that exhibit a higher degree of similarity to one another than to nodes outside these clusters (Jin et al., 2021).

Numerous conventional techniques are available for community detection, each rooted in distinct paradigms. These encompass partitioning approaches, hierarchical methods, overlapping methods, spectral methods, modularity optimization methods, and others (Bothorel, Brisson, & Lyubareva, 2021), the table 3.1 present this techniques with their algorithms.

Approach	**Algorithm**
Partitioning	* **Girvan-Newman Algorithm**: Edge removal based on betweenness centrality. * **Louvain Method**: Maximizes modularity through node reassignment. * **Kernighan-Lin Algorithm**: Swaps nodes between communities to reduce edge cut. * **FastGreedy Algorithm**: Greedily adds nodes to communities for modularity optimization.
Hierarchical	* **Agglomerative Clustering**: Merges nodes or communities based on similarity. * **Divisive Algorithms**: Divide communities iteratively based on certain criteria. * **Walktrap Algorithm**: Hierarchical clustering based on random walks.
Overlapping	* **Clique Percolation Method**: Detects communities based on shared k-cliques. * **Node Overlap Algorithms**: Assign nodes to multiple communities based on affinity scores. * **Link Clustering**: Identifies communities in networks based on link similarity. * **Walktrap Algorithm**: Also overlapping algorithm.
Spectral	* **Spectral Clustering**: Uses eigenvalues/vectors of Laplacian matrices to cluster nodes. * **Normalized Cut Algorithm**: Divides nodes into communities by minimizing normalized cut.
Modularity Optimization	* **Newman-Girvan Algorithm**: Edge removal to maximize modularity. * **Clauset-Newman-Moore Algorithm**: Greedy modularity optimization for large networks. * **Infomap Algorithm**: Models random walks to find communities that minimize information flow. * **Louvain Method**: Also a modularity optimization technique.

Table 3.1: Community detection algorithms and approaches.

The underpinning principles of translational machine learning methods have provided a groundwork for the in-depth analysis of connectivity patterns in diverse domains. Acknowledging the flexibility of algorithms to align with multiple approaches simultaneously, we proceed to introduce and elucidate traditional community detection methods. In doing so, we offer a comprehensive overview of the constituent algorithms that characterize these methods. Preceding modern machine learning methods, these techniques have laid the foundation for dissecting connectivity patterns across diverse domains. Considering that an algorithm may align with multiple approaches simultaneously. Subsequently, we introduce these traditional community detection methods and provide insights into the constituent algorithms that define them.

3.4.1 Partitioning

Partitioning is a traditional method for detecting non-overlapping communities where each node belongs to a single community. These methods seek to optimise certain criteria, such as modularity, which quantify the quality of the community structure. Partitioning methods involve iteratively moving nodes between communities in order to improve the chosen optimisation criterion. Commonly used algorithms in this category include the Girvan-Newman algorithm, the Kernighan-Lin algorithm and various clustering approaches.

3.4.2 Hierarchical Clustering

Hierarchical clustering is a technique employed to categorize nodes into communities based on their similarity or proximity, and it finds widespread applications across various domains. The process initiates with each node forming its own community, and through iterative steps, communities are merged based on a similarity measure. Unlike treating communities as isolated entities, hierarchical community detection explores the nested structure of networks. It organizes communities hierarchically, resembling a tree structure, where each level of the hierarchy unveils different levels of granularity in the organization of the network. The complexity of hierarchical community detection lies in defining the hierarchical levels and selecting appropriate algorithms. Striking a balance between granularity and interpretability remains a persistent challenge, with method evaluation representing an active area of research.

Playing a crucial role in unveiling insights into network organization across various scales, hierarchical communities offer valuable information. Within organizational networks, they expose the structures of departments, teams, and individual connections within a company. In citation networks, these communities provide visibility into themes, sub-themes, and individual contributions to research. Commonly applied algorithms in this category encompass agglomerative clustering, the Walktrap algorithm, and various others.

3.4.3 Overlapping communities

In numerous real-life scenarios, network nodes may concurrently belong to multiple communities. These overlapping communities mirror the intricate nature of social relationships, where individuals frequently engage in various social circles. The detection of overlapping communities seeks to pinpoint and delineate these multifaceted affiliations, providing a more accurate portrayal of network dynamics. However, this introduces distinct challenges compared to the detection of non-overlapping communities. Traditional algorithms often struggle to capture multiple node memberships, leading to fragmented results. Additionally, ongoing research focuses on defining the degree of overlap and developing suitable evaluation measures.

Overlapping communities find applications across diverse fields. In social media analysis, for instance, identifying users' diverse interests and connections allows for personalized content recommendations. In biological networks, overlapping communities may represent genes participating in multiple pathways.

3.4.4 Sepctral

Traditional in nature, spectral methods stand out as a common approach in community detection. Leveraging the eigenvalues and eigenvectors of a graph's adjacency matrix or Laplacian matrix, these methods unveil hidden community structures within the network. Notably, spectral methods prove particularly valuable in scenarios where there is a clear separation between groups of nodes.

Within the realm of spectral methods, spectral clustering emerges as a focused technique, aiming to partition nodes into communities based on the eigenvectors of the graph Laplacian. Through a thorough examination of eigenvalues and eigenvectors, spectral methods can pinpoint densely connected groups of nodes that signify distinct communities.

While spectral methods offer valuable insights into the structural properties of a graph and excel in detecting communities with well-defined boundaries, their effectiveness may wane in the face of networks featuring overlapping or hierarchical community structures.

3.4.5 Modularity optimization

Modularity optimization is a method that seeks to maximize the modularity of a network, which measures the quality of a partition by comparing the density of connections within communities to those between communities. The optimization process aims to identify a partition that achieves the highest modularity score, indicating a strong division into cohesive communities.

3.5 Overview of prominent community detection algorithms

A multitude of community detection algorithms have been developed to address diverse network characteristics and objectives. These algorithms employ different approaches, such as modularity optimization, spectral clustering, and probabilistic models, to partition networks into communities.

3.5.1 Louvain Algorithm

This widely used approach for community detection relies on optimizing modularity. It was introduced by Blondel et al. (2008). The method's core concept involves initially assigning each node to its individual community. Subsequently, it assesses every node and calculates the increase in modularity achievable by relocating the node to each of its neighboring communities. The node is then placed within the neighboring community that offers the most substantial modularity gain.

This sequence is reiterated for each node in the network, with the algorithm concluding when no further enhancements in modularity can be attained. At this juncture, the obtained communities are amalgamated to establish a quotient graph, upon which the same community detection procedure is executed. This iterative course is reiterated until the recursive process no longer enhances modularity.

Louvain algorithm is an iterative technique that employs both bottom-up and top-down strategies to identify optimal communities within a network, all while maximizing modularity.

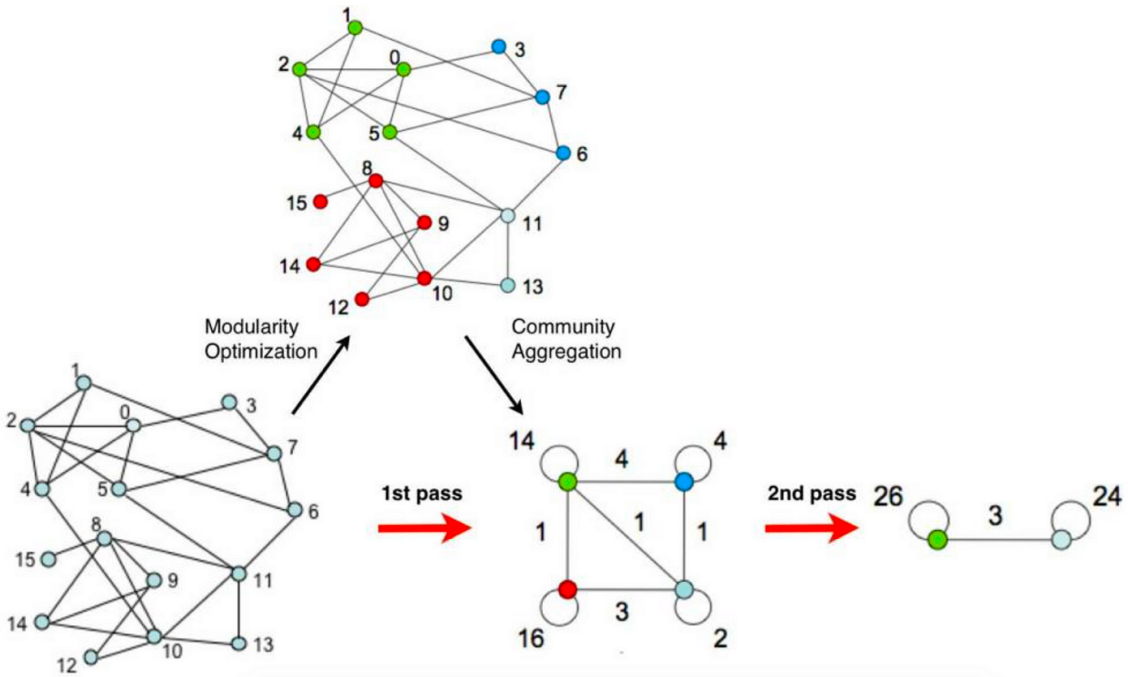


Figure 3.1: Louvain algorithm steps (Blondel et al., 2008)

3.5.2 Label Propagation Algorithm (LPA)

Label Propagation Algorithm (LPA) is an unsupervised classification method used for grouping objects based on their interactions or similarities within a network. This algorithm's core concept involves initially assigning a unique label to each node. Subsequently, in successive iterations, each node updates its label to match the label most commonly observed among its neighboring nodes (or selects a random label in the case of a tie). Over multiple iterations, nodes within the same community frequently converge to possess the same label, effectively signifying their membership in that community (Raghavan et al., 2007). It is renowned for its simplicity and efficiency, rendering it a widely embraced approach for identifying communities within networks. Nonetheless, it exhibits sensitivity to the sequence in which labels are propagated, potentially yielding divergent outcomes depending on the chosen order. To bolster result reliability and consistency, it is often prudent to execute the algorithm multiple times with varied label propagation sequences.

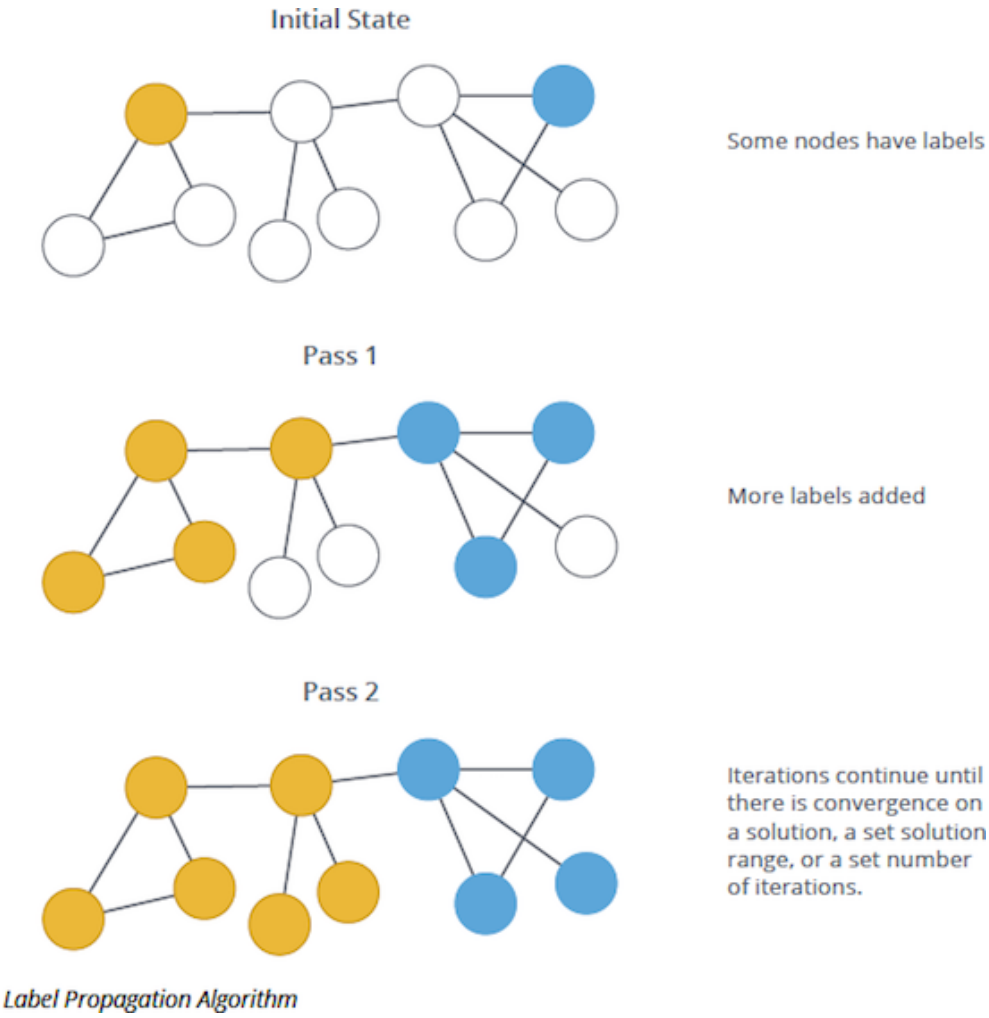


Figure 3.2: Steps in the Label Propagation Algorithm (Raghavan et al., 2007).

3.5.3 Girvan-Newman algorithm

The Girvan-Newman algorithm, also known as the Graph Clustering method, is a community detection algorithm based on the modularity measure. This algorithm aims to identify the most important edges in the network that, when removed, result in a significant division of the existing communities (Newman & Girvan, 2004).

Here are the steps of the Girvan-Newman algorithm:

1. Calculation of Initial Modularity: Modularity is a measure of the quality of node division into communities. In this step, the network's initial modularity is computed.
2. Calculation of Edge Centralities: Edge centralities, such as betweenness centrality, are calculated for all edges in the network. Betweenness centrality quantifies how often an edge is used in the shortest path between two other nodes in the network.
3. Removal of Central Edges: Edges with the highest centralities are removed from the network. This edge removal aims to create divisions between communities by severing the most critical connections.
4. Calculation of Updated Modularity: Following the removal of edges, the modularity of the new network is recalculated.
5. Repetition of Steps 2 to 4: Steps 2 to 4 are repeated until a termination criterion is met. This criterion can involve achieving a desired number of community divisions or reducing modularity below a predefined threshold.
6. Selection of Optimal Partition: At the conclusion of the algorithm, various network partitions are obtained from the edge removal steps. The optimal partition is chosen based on the highest achieved modularity.

The Girvan-Newman algorithm is iterative and can be computationally intensive, especially for large networks. Nevertheless, it is widely adopted because it offers a bottom-up approach to community detection by identifying the most significant edges for the community structure.

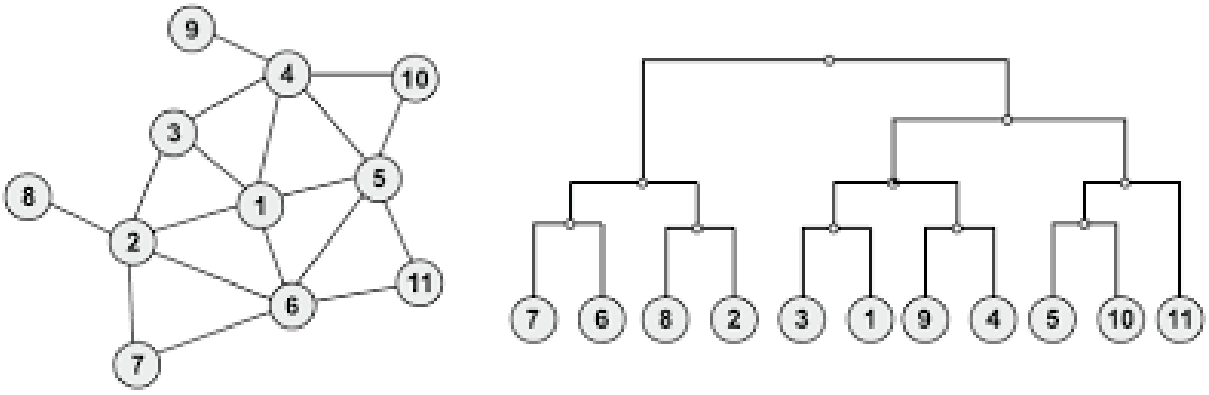


Figure 3.3: Girven-Newman Algorithm (Newman & Girvan, 2004).

3.5.4 Edge Betweenness algorithm

The Edge Betweenness algorithm is a community detection technique focused on identifying critical edges responsible for connecting distinct groups of nodes within a network. This method revolves around computing the betweenness centrality of each edge, which quantifies how many shortest paths traverse a particular edge.

The fundamental concept of the algorithm involves the iterative removal of edges with the highest betweenness centrality until the network separates into distinct communities. The removed edges serve as ”bridges” that connect different communities, while the remaining nodes naturally form separate communities.

It’s worth noting that while the Edge Betweenness algorithm is frequently employed in conjunction with other community detection methods to enhance their efficiency and accuracy, it can be computationally intensive and may exhibit slower performance when applied to large networks.

In a study conducted by Yuan, Chen, Yu, and Jin (2020), they introduced an influence maximization algorithm based on community detection, specifically utilizing the Edge Betweenness algorithm. This approach leverages the K-means algorithm to partition the community and selects the optimal community segmentation based on modularity. After community partitioning, they compute the edge betweenness centrality for each community and identify crucial nodes. These significant nodes from each community constitute the initial set of nodes utilized in their influence maximization algorithm. Ultimately, the Independent Cascade (IC) model is employed for simulations to maximize influence propagation. Experimental results illustrate that this algorithm outperforms certain classical algorithms in terms of influence, all while maintaining low computational complexity, rendering it suitable for large-scale complex networks.

3.5.5 Walktrap algorithm

Walktrap stands as a community detection algorithm crafted to reveal overlapping communities within networks. First introduced by Pons and Latapy (2005), this algorithm utilizes a probabilistic generative model that integrates Bayesian and expectation maximization techniques for identifying these overlapping communities. At its core, Walktrap conceptualizes the network as a stochastic random walk process, where each node represents a state in this process, and the edges denote the probabilities of transitioning from one state to another.

Unlike Edge Betweenness, Walktrap focuses on finding communities based on how frequently nodes are traversed together through random walks. Here’s an overview of how Walktrap operates:

Random walks are simulated from each node in the graph with a fixed walk length. Random walks that visit similar nodes are grouped into communities. Communities are merged into larger communities until a termination criterion is met. The termination criterion can be determined

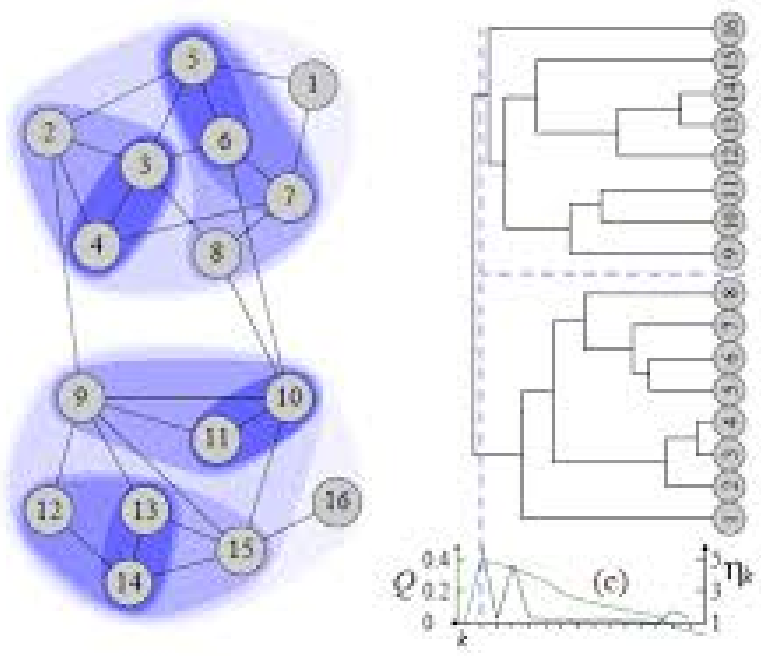


Figure 3.4: Illustrative example with WalKtrap (Pons & Latapy, 2005).

by factors like the number of communities or the difference in modularity between successive iterations. Modularity serves as a quality measure indicating how the connectedness of nodes within a community compares to their connections outside the community.

Walktrap is effective at identifying communities of various sizes and can be applied to both unweighted and weighted graphs. However, it's important to note that it may be slower compared to some other community detection algorithms and may face challenges in identifying communities with weak links or complex hierarchical structures.

3.5.6 Fast Greedy algorithm

The Fast Greedy algorithm is a community detection method designed to efficiently identify communities within networks. It is a widely used technique for detecting communities based on the principle of greedily optimizing modularity (Newman, 2004; Clauset, Newman, & Moore, 2004). The algorithm starts with each node as a separate community. It iteratively merges communities that lead to the greatest increase in modularity until no further improvements can be made. This greedy approach allows the algorithm to quickly form communities by repeatedly selecting the most beneficial merges (Orman & Labatut, 2009).

The Fast Greedy Algorithm is known for its speed and simplicity, making it suitable for analyzing large networks. However, it may not always find the globally optimal solution due to its greedy nature. Nonetheless, it often produces good results and serves as a foundational technique in the field of community detection (Parés et al., 2018).

3.5.7 Clique Percolation Method (CPM)

The Clique Percolation Method (CPM) proposed by Palla et al. (2005), is a community overlapping detection algorithm that exploits cliques, fully connected subgraphs where every node is directly linked to every other node. Communities in networks are identified through the concept of k-cliques, which are cliques with k nodes. These k-cliques serve as fundamental components for building communities in the CPM. If two k-cliques share k-1 nodes, they are deemed adjacent and are considered part of the same community. This process iteratively expands communities by incorporating adjacent k-cliques, ultimately revealing larger community structures.

What distinguishes the CPM is its flexibility in detecting overlapping communities. Nodes can participate in multiple communities if they belong to multiple overlapping k-cliques. This adaptability makes the CPM particularly effective in uncovering communities with intricate and overlapping relationships across various network types.

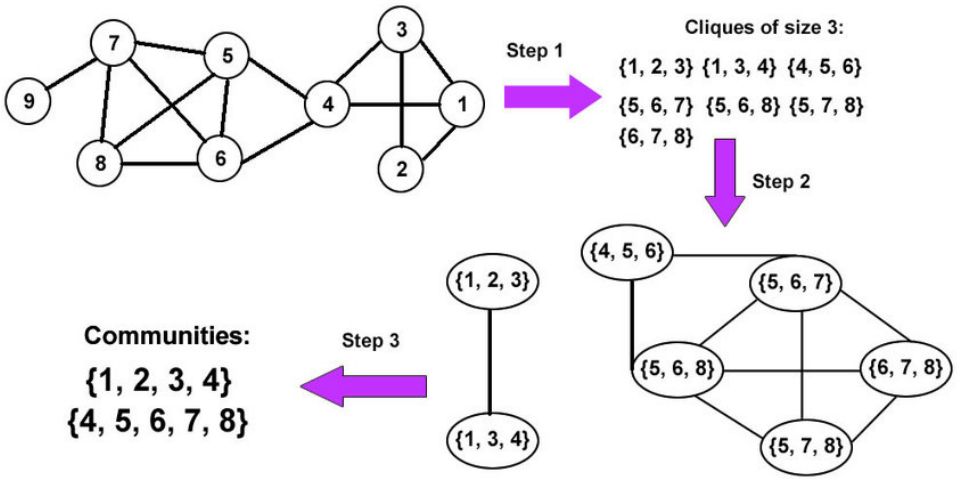


Figure 3.5: An example of CPM with k=3 (Palla et al., 2005).

3.5.8 InfoMap algorithm

The Infomap algorithm proposed by Rosvall and Bergstrom (2008), has gained popularity as a method for community detection within networks. This algorithm employs a random walk methodology, grounded in the principles of information theory, to unearth communities. Its fundamental concept revolves around treating the network as an information flow, with the objective of identifying communities that encapsulate groups of nodes exhibiting substantial information exchange within the cluster while minimizing information exchange between clusters.

3.5.9 Additional algorithms for community detection

3.5.9.1 Disjoint community detection algorithms

This section presents a selection of other algorithms designed for the detection of disjoint communities

H. Li, Zhang, Zhao, and Liu (2021) introduced an improved algorithm named LPA-MNI (Label Propagation Algorithm based on Modularity and Node Importance). This algorithm is specifically crafted for uncovering community structures in complex networks. LPA-MNI aims to address the inherent randomness observed in the original LPA algorithm by incorporating the principles of modularity and node importance. In its initial steps, LPA-MNI utilizes modularity optimization procedures to establish preliminary communities, assigning identical labels to all nodes within a community. In subsequent iterative label propagation phases, LPA-MNI updates labels in a descending order of node importance. In situations where multiple labels are assigned to an equal number of nodes, LPA-MNI computes the importance of each node and selects the label associated with the most influential node for updating. The LPA-MNI algorithm enhances the original LPA algorithm by leveraging modularity and node importance, leading to more accurate detection of community structures within complex networks.

Moosavi, Jalali, Misaghian, Shamshirband, and Anisi (2017) introduced an innovative approach to community detection, focusing on the identification of communalities through the utilization of frequent pattern mining techniques. Their method hinges on the extraction of frequent subgraphs, a process that involves systematically identifying and analyzing recurring patterns within the network structure. By leveraging frequent pattern mining, Moosavi et al. aimed to uncover cohesive substructures or clusters that manifest as frequent motifs within the larger network, providing valuable insights into the underlying community organization. This methodological choice underscores the significance of pattern-based analysis in unraveling the intricate web of relationships within complex networks, laying the foundation for subsequent advancements in community detection algorithms.

The Community Multi-objective Optimization Algorithm (CMCM), introduced by Geng, Bhattacharya, and Pati (2019), stands out as an influential contribution to the field of community detection. This algorithm, grounded in a multi-objective optimization framework, addresses the intricate task of identifying communities within networks by concurrently optimizing two pivotal metrics: modularity and conductance. Modularity, a widely acknowledged measure in community detection, quantifies the quality of the division of a network into communities. Simultaneously, conductance, another crucial metric, assesses the interconnectedness between communities, emphasizing the need for well-connected yet distinct partitions. The brilliance of CMCM lies in its ability to balance and optimize both modularity and conductance simultaneously, providing a nuanced and comprehensive perspective on community structure. Geng et al.'s multi-objective optimization strategy demonstrates a holistic approach to community detection, where the algorithm strives to find solutions that achieve a harmonious trade-off between maximizing the internal cohesion of communities (modularity) and minimizing their external connections (conductance). This nuanced optimization not only enhances the algorithm's robustness but also aligns with the real-world complexity of network structures, where communities exhibit a delicate balance between internal cohesion and external separation. The CMCM

algorithm thus represents a notable advancement in the pursuit of more accurate and insightful community detection methodologies

Sheng, Liu, Chen, Wang, and Zhang (2020) introduced the Inter-node Attraction Community Detection (IACD) method in 2020. Grounded in the novel concept of inter-node attraction, this algorithm employs a series of intricate steps. It commences with the evaluation of node importance, assigning significance to nodes based on their structural and functional roles within the network. Subsequently, the algorithm selects attractive node pairs, identifying pairs of nodes that exhibit strong mutual attraction. The final step involves community division, where the network is partitioned into distinct communities based on the observed inter-node attractions. The IACD algorithm, through its unique approach, provides a nuanced perspective on community structures, emphasizing the dynamic interactions and attractions between nodes.

The Influence Maximization Algorithm, developed by Yuan et al. (2020), focuses on maximizing node influence within a network. This is achieved through the utilization of the edge betweenness algorithm, which is intricately tied to community detection. By strategically identifying edges with high betweenness centrality, the algorithm aims to pinpoint influential nodes that play a crucial role in shaping community structures. This approach provides valuable insights into the key actors driving information flow and connectivity within the network.

Chaudhary and Singh (2021) contribute to the field by applying unsupervised machine learning techniques for community detection specifically in COVID-19 datasets. Leveraging advanced unsupervised learning methodologies, their approach seeks to uncover hidden patterns and structures within the complex network of interactions related to the spread of COVID-19. This application underscores the versatility of community detection algorithms in addressing real-world challenges.

Wu et al. (2021) present the Frequent Pattern-based Approach (FPPM), a methodology that harnesses frequent pattern mining to extract communities in social networks. By identifying recurring patterns within the network, FPPM provides a robust framework for uncovering cohesive substructures and revealing inherent community structures. The emphasis on frequent patterns adds a layer of granularity to the analysis, capturing nuanced relationships within the network.

Akbar, Liu, and Latif (2021) contribute to the field with their algorithm focusing on Modularity Maximization. This approach aims to maximize the modularity metric, a fundamental measure in community detection that assesses the quality of network partitioning. By optimizing modularity, the algorithm seeks to identify communities that exhibit strong internal cohesion while minimizing connections between distinct communities. The pursuit of modularity maximization aligns with the goal of obtaining insightful and meaningful community structures in social networks.

X. Liu, Fu, Wang, and Zhou (2022) propose a community detection algorithm based on en-

tropy concepts, known as the Entropy Gap method. This innovative approach involves analyzing the gap between the entropy of the original network and that of a null model. By quantifying the deviation from random network structures, the algorithm identifies communities that exhibit significant information content and organization. The Entropy Gap method adds a unique perspective to community detection by integrating information-theoretic concepts into the analysis.

X. Zhou, Su, Li, Zhao, and Li (2023) present the Clustering-based Approach (CDBNE), which integrates both network structure and attribute information for comprehensive community detection. This algorithm goes beyond traditional structural considerations and incorporates attribute data associated with nodes. By leveraging a holistic approach that considers both structural connectivity and attribute similarity, CDBNE aims to provide a more comprehensive and nuanced understanding of community structures within complex networks.

3.5.9.2 Overlapping community detection algorithms

This section delves into various algorithms meticulously chosen for their aptitude in identifying overlapping communities:

Gujral, Papalexakis, Theocharous, and Rao (2019) introduced the Hierarchical Agglomerative Community Detection (HACD) algorithm, a highly efficient method designed for the detection of overlapping communities. This algorithm stands out by seamlessly combining local graph information with membership propagation strategies. By integrating these components, HACD excels in uncovering complex community structures where nodes may belong to multiple communities simultaneously. The algorithm's effectiveness lies in its ability to capture nuanced relationships within the network, making it a valuable tool for detecting overlapping communities in diverse settings.

Musdar and Azhari (2015) proposed the Randomized Clustering-based Approach, abbreviated as RCE, as a unique algorithm for detecting overlapping communities. Leveraging a random walk-based strategy, RCE introduces a level of unpredictability into the clustering process. This randomness enables the algorithm to effectively identify overlapping communities by exploring diverse pathways within the network. The approach is particularly valuable in scenarios where traditional, deterministic methods may struggle, providing a versatile solution for uncovering complex community structures.

Ma et al. (2016) developed the Label Propagation Algorithm (LPA) within the context of the LED algorithm for overlapping community detection. LED employs LPA to assign vertices to one or more communities based on network weights and neighbor community assignments. The algorithm's strength lies in its adaptability to diverse network structures, as it leverages local information to iteratively propagate labels, effectively revealing overlapping community memberships. LED serves as a robust tool for identifying communities that exhibit intricate overlaps and intersections.

Van Laarhoven and Marchiori (2016) proposed two algorithms, EMc and PGDc, both tai-

lored for overlapping community detection. EMc relies on the expectation-maximization (EM) algorithm to iteratively refine community assignments, considering the probabilistic nature of node memberships. On the other hand, PGDc employs projected gradient descent to optimize community assignments based on a specified objective function. These algorithms collectively demonstrate the effectiveness of combining optimization techniques with probabilistic modeling for robust detection of overlapping communities.

Kumar, Barman, Sarkar, and Chowdhury (2020) introduced NSGA-II, a Multi-objective Optimization algorithm designed for overlapping community detection. This method leverages multi-objective optimization principles to balance intra-community cohesion and inter-community separation. By considering multiple objectives, NSGA-II aims to discover a diverse set of solutions that represent the trade-off between these conflicting goals. The algorithm's capacity to explore a spectrum of solutions makes it suitable for scenarios where the overlapping community structure exhibits varying degrees of cohesion and separation.

Luo, Lu, Ni, Zhu, and Ding (2020) proposed LCDNN, a Deep Neural Network-based algorithm specifically designed for overlapping community detection. By incorporating both local and global information, LCDNN harnesses the power of deep learning to capture intricate patterns within the network. The algorithm's neural network architecture enables it to learn complex representations, making it adept at handling the intricate relationships and overlaps present in real-world community structures. LCDNN represents a cutting-edge approach to overlapping community detection, capitalizing on the strengths of deep learning methodologies.

Ma et al. (2020) introduced the Likelihood-based Overlapping Community Detection algorithm (LGIEM), a method that utilizes a generative model to describe the underlying structure of overlapping communities. By framing the problem within a likelihood-based framework, LGIEM estimates the probability distribution governing the generation of edges and community memberships. This probabilistic approach allows LGIEM to discern subtle patterns within the network, providing a principled and statistically sound method for identifying overlapping communities. The generative model embedded in LGIEM contributes to its robustness in capturing the intricate nature of overlapping community structures.

3.6 Comparison between traditional methods

3.6.1 Overlapping and hierarchical communities

Overlapping and hierarchical communities represent two distinct yet interconnected concepts that contribute to an enhanced understanding of network dynamics (Louafi & Titouna, 2019). The advancement of research on overlapping and hierarchical communities necessitates the development of innovative algorithms, robust evaluation frameworks, and applications across various disciplines. Researchers aim to address challenges related to scalability, interpretability, and the seamless integration of these concepts Both overlapping and hierarchical

communities serve as valuable lenses through which researchers can explore the intricate landscapes of social networks. By acknowledging the complexities of multiple affiliations and hierarchical structures, researchers can attain deeper insights into the multifaceted nature of social interactions and organizations.

The combination of overlapping and hierarchical community detection enriches our comprehension of network structures. Networks can be investigated at multiple resolutions, simultaneously capturing intricate relationships and broad organizational patterns. This integration presents technical and algorithmic challenges, demanding innovative solutions. The incorporation of overlapping and hierarchical communities extends to various domains. In social networks, it can unveil nuanced interactions while exposing overarching societal structures. In recommendation systems, this approach personalizes suggestions while considering broader content themes.

The amalgamation of hierarchical and overlapping community detection approaches opens avenues for a more comprehensive understanding of network structures. By revealing nested and overlapping communities, this hybrid approach provides richer insights into the intricate relationships present in networks.

3.6.2 Overlapping and disjoint communities

Communities frequently stand for interconnected clusters of individuals within the actual world. The automated identification of network communities proves highly valuable as it can, for instance, illuminate the organization of networks that are exceedingly vast for humans to comprehend manually, even when aided by visualization methods (Gregory, 2009). In contrast, disjoint communities allocate nodes to unique, non-overlapping groups, ensuring that each node belongs to only one community. This simplifies result interpretation, making disjoint community detection particularly useful when the aim is to identify clear and separate groups within a network.

The challenge in disjoint community detection lies in accurately dividing nodes into distinct groups while striking a balance between group sizes and optimizing the quality of these communities. Disjoint communities find applications in various fields. For instance, in co-authorship networks, they can represent clusters of researchers with similar research interests, while in collaboration networks, they may indicate departments collaborating on specific projects.

Detecting communities within complex networks is a critical undertaking, and it's essential to distinguish between overlapping and disjoint (non-overlapping) communities, as they each shed light on distinct aspects of network connectivity. The selection between overlapping and disjoint community detection hinges on specific research objectives and the nature of the network under examination. In many cases, networks exhibit a blend of both community types, necessitating a thoughtful choice of algorithms and methodologies.

Ongoing advancements in community detection consistently explore the interplay between overlapping and disjoint communities. Researchers actively devise algorithms that adeptly capture the intricacies of real-world networks while delivering valuable insights into their underlying structures. The decision to opt for overlapping or disjoint community detection ultimately depends on the research query and the characteristics of the network in question. Both community types offer valuable perspectives for comprehending network organization, providing nuanced approaches for unraveling intricate systems.

While overlapping community detection methods possess certain advantages, they may not always be the most appropriate choice for uncovering complex community structures or managing noisy data.

3.7 Strengths and limitations of traditional approaches

Traditional approaches to community detection in graphs have their strengths and limitations, shaping their applicability and effectiveness in different scenarios. Here, we outline the key strengths and limitations of these methods:

3.7.1 Strengths

- **Interpretability:** Traditional methods often provide intuitive results that are easy to interpret. The hierarchical structure obtained from hierarchical clustering, the modularity optimization score, and the spectral embeddings from spectral clustering can offer clear insights into the community organization.
- **Comprehensiveness:** Hierarchical clustering captures community hierarchies at various levels, allowing for a detailed analysis of community relationships. This is particularly valuable when studying networks with nested or overlapping communities.
- **Theoretical Foundation:** These methods are rooted in well-established mathematical and graph theory principles, making them reliable and providing a solid foundation for their application.
- **Benchmarking:** Traditional approaches serve as benchmarks for evaluating the performance of newer, more complex community detection algorithms. Their straightforward nature makes them a useful point of reference for comparison.

3.7.2 Limitations

1. **Scalability:** Traditional methods might struggle with scalability when dealing with large networks. The computational complexity of hierarchical clustering and the eigenvalue decomposition in spectral clustering can be prohibitive for networks with a large number of nodes.
2. **Resolution Limit:** Modularity optimization methods can face the resolution limit issue,

where they tend to detect larger communities and overlook smaller ones. This can lead to oversimplification of the community structure.

3. **Parameter Sensitivity:** Traditional methods often rely on parameters that need to be set beforehand. The choice of parameters can significantly impact the results, and finding the optimal values might not be straightforward.
4. **Limited to Certain Structures:** These methods might not perform well on networks with intricate structures, such as networks with heavily overlapping communities or communities that exhibit non-traditional shapes.
5. **Community Shape Bias:** Spectral clustering assumes that communities are connected and well-separated, which might not hold true for all networks. It can struggle with networks where community boundaries are fuzzy.
6. **Subjectivity in Hierarchical Clustering:** The choice of a threshold for cutting the dendrogram in hierarchical clustering can be subjective and impact the interpretation of results.

Traditional community detection approaches offer a solid foundation for understanding network structures, but they come with trade-offs. They are particularly suitable for smaller networks with clear hierarchies and well-defined communities. However, as network sizes and complexities grow, these methods might face limitations related to computational efficiency, scalability, and their ability to handle diverse community structures.

3.8 Traditional approaches to Machine Learning

The field of community detection continues to evolve, with ongoing research focused on enhancing algorithmic accuracy, scalability, and adaptability to different network structures. Future advances may incorporate Machine Learning techniques, consider dynamic networks, and tackle the challenges of overlapping communities.

Machine learning techniques are being harnessed to enhance the accuracy and efficiency of community detection. By training models on labeled data, Machine Learning algorithms can learn patterns that humans might overlook. This can result in better community assignment and an increased ability to handle complex and diverse network structures.

3.8.1 Deep learning techniques in community analysis

Deep learning, a subset of Machine Learning, has demonstrated its potential in community detection. Graph neural networks (GNNs), in particular, can capture intricate relationships between nodes and learn representations of nodes that are conducive to community detection. GNNs can improve the accuracy of community assignment, especially in networks with rich node attributes and complex interactions.

Deep learning methods are gaining traction in community detection due to their ability to

handle complex graph structures. Some notable deep learning approaches include:

Graph Convolutional Networks (GCNs): Extend convolutional neural networks to graph data, enabling them to capture neighborhood information and detect communities effectively.

Graph Autoencoders: These models leverage autoencoders to learn compact graph representations. They can be used to encode nodes into latent spaces that highlight community structures.

Graph Generative Models: Deep generative models like Graph Variational Autoencoders (VAEs) can learn to generate graphs with specific community structures, aiding in understanding and modeling communities.

3.8.2 Integrating traditional and machine learning approaches

A hybrid approach that combines traditional graph theory methods with Machine Learning techniques is gaining traction. This integration aims to capitalize on the strengths of both paradigms. Traditional methods offer interpretability and domain knowledge, while Machine Learning algorithms can capture complex patterns and relationships. Combining these approaches has the potential to produce more comprehensive and accurate community detection results.

The integration of traditional and Machine Learning approaches often yields improved community detection results. Some strategies include:

Hybrid Models: Combining traditional graph algorithms with Machine Learning methods can leverage the strengths of both paradigms. For example, combining spectral clustering with GNNs can yield more accurate community assignments.

Feature Engineering: Extracting relevant features from the graph and nodes can enhance the performance of Machine Learning models. These features can include node attributes, structural properties, and graph-based metrics.

Ensemble Methods: Combining the predictions of multiple community detection algorithms, including Machine Learning models, can lead to more robust and accurate results.

The integration of Machine Learning techniques into community detection offers promising avenues for advancing the field and extracting more meaningful insights from complex network data.

3.8.3 Leveraging machine learning for improved detection

Machine Learning techniques have been increasingly integrated into community detection to enhance its accuracy and efficiency. Some common approaches include:

Spectral Clustering: This technique utilizes graph Laplacian eigenvalues and eigenvectors to partition the graph into communities. Spectral clustering benefits from Machine Learning

algorithms to optimize the partitioning process.

Graph Neural Networks (GNNs): GNNs leverage deep learning techniques to capture intricate relationships between nodes in a graph. They have shown promise in community detection tasks by learning node embeddings that represent the community structure.

SVM and Random Forests: Traditional Machine Learning models like Support Vector Machines (SVM) and Random Forests can be applied to classify nodes into communities based on features extracted from the graph structure.

3.9 K-means as community detection tool

When applied to graphs, K-means can identify clusters of nodes that are densely connected within themselves and sparsely connected to nodes in other clusters. These clusters often correspond to communities within the graph. By iteratively optimizing cluster assignments and means, K-means seeks to find partitions that maximize intra-cluster similarity and minimize inter-cluster similarity.

K-means offers several benefits for community detection:
Simplicity: K-means is easy to understand and implement, making it a practical choice for initial exploratory analysis.
Scalability: K-means can handle relatively large graphs efficiently, making it suitable for moderate-sized networks.
Quantitative Results: K-means provides quantifiable results, assigning nodes to clusters based on their feature similarities.

However, K-means also has limitations:
Assumption of Equal Size: K-means assumes that each cluster has approximately the same number of nodes, which may not align with real-world community sizes.
Global Optimum: K-means can converge to local optima, leading to suboptimal community partitions.
Hard Assignment: K-means assigns nodes to a single cluster, which may not accurately capture overlapping community structures.

To address some limitations, researchers have developed enhancements and variants of K-means for graph-based community detection:
Weighted K-means: Accounts for node degrees, giving higher weight to nodes with more connections.
Spectral K-means: Utilizes spectral clustering techniques to improve community detection in graphs.
Semi-supervised K-means: Combines labeled and unlabeled data to guide the clustering process.
Fuzzy K-means: Allows nodes to belong to multiple clusters with varying degrees of member-

ship.

K-means can be integrated with network analysis pipelines to complement other graph-based techniques. By combining K-means results with network centrality measures, edge density analysis, and other metrics, a more comprehensive understanding of community structures and their interactions can be achieved. K-means clustering provides a straightforward yet effective approach to community detection in graphs. While it has its limitations, its simplicity and scalability make it a valuable tool for initial exploration and analysis of community structures in networks.

3.9.1 Works employing K-means in community detection

The research on community detection using the K-Means algorithm has been extensively studied in the literature. However, it is important to note that K-Means is primarily used as an unsupervised clustering algorithm and is not specifically designed for community detection in networks.

In some cases, K-Means can be applied to network data using appropriate similarity measures, such as cosine similarity or Euclidean distance between nodes. However, this does not guarantee that the obtained clusters actually correspond to communities in the network.

Community detection in networks is a complex problem, and many other algorithms have been developed specifically for this task. Approaches such as probabilistic modeling, label propagation-based methods, modularity-based community detection algorithms, and spectral clustering techniques are often preferred for community detection in networks.

It is important to choose the appropriate algorithm based on the specific characteristics of the network and the objectives of community detection. While K-Means can be used in some situations, it is often recommended to explore methods that are better suited for community detection in networks to obtain more accurate and meaningful results:

Žalik (2008), a variation of the K-means algorithm designed for non-overlapping community detection, introduces the flexibility to adjust the number of clusters dynamically.

Vilcek (2014) introduced the deep K-means algorithm, a novel graph clustering approach for network community detection. Unlike traditional spectral clustering relying on eigenvector decomposition, Vilcek's method employs a multilayer autoencoder pipeline implemented through recursive K-means clustering. This approach yielded improved accuracy compared to spectral clustering, as measured by normalized mutual information. However, it exhibits increased execution time with larger datasets and requires prior knowledge of the number of communities—a potential future direction could involve automating this determination via modularity function optimization.

L. Li, Fan, Zhangy, and Xia (2016) proposed an enhanced community detection algorithm incorporating Principal Component Analysis (PCA) and the K-means algorithm. Simulation

results indicate superior community detection accuracy, especially in complex networks.

Cai, Zeng, Wang, Li, and Hu (2019) proposed the DDJKM (Density, Degree, Jaccard, and K-means) algorithm for overlapping community detection. This method utilizes node uncertainty based on density, degree, and similarity to describe membership in multiple communities. K-means clustering is then applied to the uncertainty matrix, demonstrating promising accuracy and efficiency in real-world networks.

Hajij, Said, and Todd (2020) presented an innovative method for initializing K-means clustering centers by leveraging the PageRank vector as a centrality measure. This approach demonstrated efficiency and several advantages. Notably, it can be applied to both direct and indirect graphs, harnessing the computational speed originally designed for large graphs. Moreover, it lends itself to generalization across different domains, enhancing its adaptability.

Hajij et al. (2020) harnessed the PageRank vector as a centrality measure to initialize centroids in the K-means clustering algorithm within graphs. This approach offers efficiency and various advantages, including applicability to both directed and undirected graphs, computational speed suitable for massive graphs, and adaptability to metric spaces, making it versatile and straightforward.

The Hybrid DBSCAN Algorithm, Proposed by Aftab, Shuja, Alasmary, and Alanazi (2021), integrates minibatch K-means and DBSCAN methodologies for adaptive community detection in social networks. This hybrid approach harnesses the efficiency of minibatch K-means for initial clustering and subsequently refines the results using the density-based DBSCAN algorithm. The combination of these techniques enhances the algorithm's adaptability, enabling it to effectively identify communities of varying shapes and densities within social networks.

3.10 Assessment measures for community detection

Assessing the effectiveness of community detection algorithms is crucial for quantitatively evaluating their performance and quality. Comparative analysis facilitates the identification of algorithmic strengths, weaknesses, and their applicability to various network types. These evaluations offer a comprehensive overview of the alignment between detected communities and a ground truth partition, providing valuable insights into the accuracy and efficiency of algorithmic results. Commonly used evaluation metrics encompass parameters like modularity, normalized mutual information, and adjusted rand index.

These evaluation metrics collectively offer valuable insights into the performance and quality of community detection algorithms. To obtain a comprehensive assessment, it is advisable to employ multiple evaluation measures that offer different perspectives on the algorithm's effectiveness.

The main difference between overlapping community modularity and disjoint community modularity lies in how nodes in the network are allowed to belong to multiple communities

(overlapping) or just one (disjoint). The choice between these two approaches depends on the nature of the studied network and the objectives of the analysis. Real-world networks often exhibit overlapping community structures, and in such cases, overlapping community modularity may be more appropriate to reflect the complexity of relationships between nodes.

3.10.1 Variation of Information (VI)

Variation of Information (VI) quantifies the dissimilarity between two partitions, such as the detected communities and a ground truth partition. It measures the information required to transform one partition into another. Lower VI values indicate a higher level of agreement with the ground truth, signifying better performance.

$$VI(P,Q) = H(P) + H(Q) - 2 \cdot I(P,Q) \tag{3.1}$$

where:
- P and Q are the partitions being compared.
- $H(\cdot)$ denotes the entropy.
- $I(\cdot,\cdot)$ represents the mutual information.

3.10.2 Normalized Mutual Information (NMI)

Normalized Mutual Information (NMI) is a similarity measure that estimates the similarity between two partitions A and B, where A represents the true network partition and B represents the partition detected by experimental community detection algorithms. It is based on information theory (Danon, Díaz-Guilera, Duch, & Arenas, 2005).

For two partitions A and B of a network, the value of NMI is calculated using equation (3.2) (H. Li et al., 2021):

$$NMI(A,B) = \frac{-2\sum_{i=1}^{C_A}\sum_{j=1}^{C_B} N_{ij}\log\left(\frac{N_{ij}N}{N_i N_j}\right)}{\sum_{i=1}^{C_A} N_i \log\left(\frac{N_i}{N}\right) + \sum_{j=1}^{C_B} N_j \log\left(\frac{N_j}{N}\right)} \tag{3.2}$$

where:

A : represents the true network partition.

B : the partition discovered by community detection algorithms.

C_A : represents the number of communities in partition A.

C_B : denotes the number of communities in partition B.

N : represents the total number of nodes in the network.

N_{ij} : represents the number of common nodes between community i in partition A and community j in partition B.

N_i : is the number of nodes in the actual community i (sum of row i in the matrix N_{ij}).

N_j : is the number of nodes in computed community j (sum of column j).

The value of NMI can range between 0 and 1. The closer the NMI value is to 1, the more similar the two partitions are. In other words, when two partitions A and B are completely different, then NMI(A, B) = 0. If the NMI takes its maximum value of 1, then partition A exactly matches partition B.

3.10.3 F1 Score

The F1 score combines precision and recall to gauge the accuracy of community detection. Precision assesses the fraction of accurately assigned nodes within a community, while recall evaluates the proportion of nodes from a community that are correctly identified. The F1 score, the harmonic mean of precision and recall, ranges from 0 to 1. Higher values indicate more accurate and balanced performance.

$$F1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.3}$$

3.10.4 Conductance

Conductance is a measure of community quality that examines the ratio of edges within a community to the total edges connecting the community to the broader network. Smaller conductance values suggest stronger communities with fewer connections to external nodes, indicative of well-defined structures.

$$\text{Conductance}(C) = \frac{\text{Cut}(C, \overline{C})}{\min(|C|, |\overline{C}|)} \tag{3.4}$$

where:
- C is the community being evaluated.
- $\overline{C}$ is the complement of C.
- Cut$(C, \overline{C})$ is the number of edges between C and $\overline{C}$.

3.10.5 Normalized Cut

Normalized Cut assesses community quality by considering the normalized ratio of cut edges between the community and the remaining network to the community's size. Lower normalized cut values correspond to better community structures, as they reflect fewer edges connecting the community to the outside.

$$\text{Normalized Cut}(C) = \frac{\text{Cut}(C, \overline{C})}{\text{Vol}(C)} + \frac{\text{Cut}(C, \overline{C})}{\text{Vol}(\overline{C})} \tag{3.5}$$

where:
- C and $\overline{C}$ are defined as before.
- Vol(C) is the sum of degrees of nodes in C.
- Vol($\overline{C}$) is the sum of degrees of nodes in $\overline{C}$.

3.10.6 Rand Index (RI)

The Rand Index (RI) quantifies the similarity between the detected communities and a ground truth partition by accounting for true positive and true negative assignments. It measures the proportion of correctly assigned node pairs relative to all possible pairs. Rand Index values vary between 0 and 1, with higher values denoting greater agreement with the ground truth.

$$\text{Rand Index} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{3.6}$$

where:

- TP is the number of true positive assignments.
- TN is the number of true negative assignments.
- FP is the number of false positive assignments.
- FN is the number of false negative assignments.

3.10.7 Adjusted Rand Index (ARI)

The Adjusted Rand Index (ARI) is a measure used to evaluate the similarity between two data clusterings. It assesses the agreement between the true labels of data points and the labels assigned by a clustering algorithm, while considering chance.

The Rand Index (RI) is a basic measure that calculates the proportion of pairs of data points that are either concordant (in the same cluster in both the predicted and true clusterings) or discordant (in different clusters in either the predicted or true clustering). The Adjusted Rand Index adjusts the Rand Index for chance, providing a corrected measure that takes into account the expected similarity between clusterings due to random chance.

The formula for the Adjusted Rand Index involves the Rand Index, the total number of pairs of data points, and correction terms. It produces a score between -1 and 1, where 1 indicates perfect agreement between the two clusterings, 0 indicates the level of agreement expected by chance, and negative values suggest less agreement than expected by chance.

The ARI is commonly used in the evaluation of clustering algorithms, particularly when the ground truth (true labels) is available for comparison. It provides a more robust assessment than the Rand Index alone, especially when dealing with datasets of different sizes and varying numbers of clusters.

3.10.8 Modularity

Q is a widely used metric for evaluating community detection. It measures the difference between the number of edges within communities and the expected number of edges in a random graph with the same degree distribution. Positive values of Q indicate better-defined communities than would be expected by chance. there is two types of Modularity: overlapping community modularity and disjoint community modularity:

3.10.8.1 Overlapping Modularity

Overlapping Modularity pertains to the possibility for a node to be part of multiple communities simultaneously (M. Chen, Kuzmin, & Szymanski, 2014). In this context, a node can have edges (connections) with multiple different groups of nodes, and overlapping community modularity measures to what extent this overlap is taken into account in community detection. This allows for representing network structures where nodes have multiple roles or are involved in different subgroups at the same time.

$$Q_M = \sum \left(\frac{e_{ij} - \frac{a_i \cdot b_j}{2m}}{2m} \right) \delta(C_i, C_j) \tag{3.7}$$

Q_M: Overlapping modularity.
$\sum$: Summation symbol.
e_{ij}: Fraction of edges between nodes in communities i and j.
a_i: Fraction of edges within community i.
b_j: Fraction of edges within community j.
$2m$: Total number of edges in the network.
$\delta(C_i, C_j)$: Kronecker delta function (1 if $i = j$, 0 otherwise)

3.10.8.2 Disjoint Modularity (Q)

Disjoint community Modularity (Q) assumes that each node in the network belongs to only one community. In this case, each node is exclusively assigned to one group, and there is no overlap between communities. This type of modularity is generally more suitable for networks where each node belongs to a single category or class.

$$Q = \sum_i \left(\frac{e_{ii}}{m} - \left(\frac{d_i}{2m} \right)^2 \right) \tag{3.8}$$

where:
- i represents a community index.
- e_{ii} is the number of edges within community i.
- d_i is the sum of degrees of nodes in community i.
- m is the total number of edges in the network.

3.11 Conclusion

In this chapter, we embarked on a comprehensive exploration of community detection, a pivotal aspect of network analysis. Commencing with an elucidation of its fundamental goals and overarching significance, we underscored its relevance across diverse domains, ranging from social networks to biological systems.

Our journey extended to a detailed examination of conventional methods employed in community detection, wherein we delved into their strengths and limitations. Furthermore, we introduced traditional machine learning approaches, emphasizing their capacity to enhance community detection methodologies through the integration of data-driven insights.

Looking ahead, the subsequent chapters will unveil our unique approach to community detection within social networks.

Part II

Methodology and Experimentation

Chapter 4

DBOCD: Density-Based Overlapping Community Detection

4.1 Introduction

Identifying cohesive groups or communities of nodes that share common characteristics or interactions is a central challenge in the field of network analysis. This challenge becomes more complex when dealing with overlapping communities. In this context, we introduce an innovative and flexible method aimed at addressing the intricate task of identifying communities within the complex web of social networks.

Our method is applicable to undirected and unweighted networks, capable of uncovering both overlapping and non-overlapping communities, and adaptable to a variety of network scenarios. To enhance its efficacy, we have enhanced our model by incorporating unsupervised machine learning techniques, with a particular focus on harnessing the potential of clustering algorithms.

We introduce a novel approach for detecting overlapping communities in social networks, which integrates a density-based measure of node significance. Our method first computes the Local density of each node in the network and subsequently employs a clustering algorithm to recognize both overlapping and distinct communities, based on the density of their constituent nodes. We assess the performance of our approach on diverse social network datasets. Our findings demonstrate that it effectively addresses the challenge of identifying overlapping communities (Louafi & Titouna, 2023a).

4.2 Contribution

In this section, we present our innovative algorithm, Density Based Overlapping Community Detection (DBOCD), designed for community detection. The DBOCD algorithm follows a three-stage process that includes Local density calculation, the creation of initial overlapping classes, and a series of iterative grouping steps. These stages are executed sequentially to effectively identify and refine overlapping or disjoint communities within the given network. Algorithm 4.1 provides a comprehensive overview of DBOCD.

Algorithm 4.1. DBOCD Algorithm.

Require: Database in the form of a graph G
Ensure: Overlapping Communities $C_1, C_2, \ldots, C_K$
1: Calculate Local density scores for each node using the formula 4.1
2: Sort the nodes according to their Local density
3: Select Initial Classes using Breadth-First Search (BFS)
4: Refine clusters based on dissimilarity until no more merging is possible
5: Group according to Maxdis as defined in formula 4.5

Figure 4.1 visually represents the conceptual framework of the Density Based Overlapping Community Detection (DBOCD) algorithm. In this figure, each element is carefully designed to convey specific aspects of the algorithm's functioning.

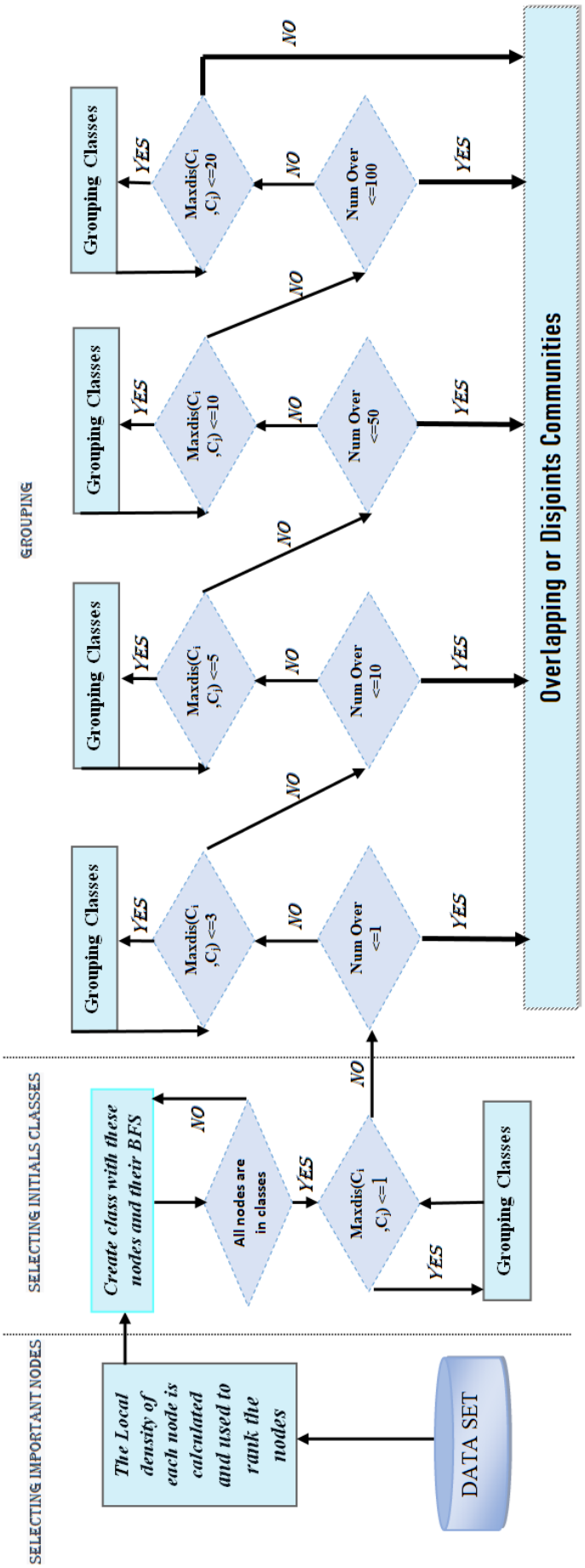


Figure 4.1: Flowchart of the DBOCD Algorithm.

The graphical representation provides a detailed insight into how the algorithm operates and the relationships between different components. In the following subsections, we provide a detailed breakdown of each stage in the DBOCD algorithm, highlighting the methodologies employed and the significance of each stage in the community detection process.

4.2.1 Selecting important nodes

After importing the database and convert it into a graph, we calculate the Local Node density with equation 4.1, then nodes ranked in descending order of their Local density.

This stage is divided into two distinct steps: firstly, the calculation of Local density for each node, and secondly, the storage of these density values in descending order.

4.2.1.1 Calculate the Local density of nodes

Local density serves as a metric for gauging the significance of individual nodes within the immediate vicinity of a graph. This localized neighborhood is precisely defined as the ensemble of nodes that can be reached within a two-hop distance from a given target node. The computation of this metric is facilitated through the application of a modified Breadth-First Search (BFS) algorithm, specifically configured with a two-hop traversal constraint. The resultant subgraph derived from this procedure exclusively encompasses vertices and edges existing within these two hops, providing a concise snapshot of the local structural aspects within the broader graph. This, in turn, empowers the assessment of a node's relevance and influence within its immediate context. At the core of this phase lies the fundamental calculation of Local density scores for each node in the graph.

The density is calculated using the formula 4.1:

$$D = \frac{2 * |E|}{|V|(|V| - 1)} \tag{4.1}$$

Where:
E: stands for the number of edges,
V: represents the count of nodes.

Here, we calculate the Local density D'; the density in the subgraph G' created by BFS $h = 2$:

$$D' = \frac{2 * |E'|}{|V'|(|V'| - 1)} \tag{4.2}$$

Where:
E': stands for the number of edges in G',
V': represents the count of nodes in G'.

We chose a distance of two hops in the BFS (Breadth-First Search) algorithm during the

calculation of Local density according to the context of community detection. Here are some of the reasons for selecting a two-hop distance:

- **Local Neighborhood Definition:** The two-hop distance is chosen to define a localized neighborhood around each node. A one-hop distance would only capture the immediate neighbors of a node, providing a limited view of its local structure. In contrast, a two-hop distance extends the neighborhood to nodes that are reachable within two hops, offering a more comprehensive snapshot of the immediate surroundings.
- **Balancing Local Detail and Computational Efficiency:** A two-hop distance strikes a balance between capturing local details and maintaining computational efficiency. By considering nodes within a two-hop radius, the algorithm can capture more relevant information about the node's local context without overly expanding the neighborhood, which could lead to increased computational complexity.
- **Enhancing Robustness:** Using a two-hop distance can enhance the robustness of the Local density calculation. It reduces sensitivity to outliers or isolated nodes by considering a broader set of neighboring nodes, making the algorithm more resilient to variations in the network structure.

4.2.1.2 Detection of important nodes by Local density

The arrangement of nodes according to their Local density scores creates a prioritized list, where nodes of greater significance are positioned at the forefront. In cases where nodes possess identical Local density scores, their ordering is established by their node numbers in ascending order. This sorted roster of nodes, accompanied by their corresponding Local density scores, is retained for use in the subsequent phases of the algorithm. To execute this stage, Algorithm 4.2 delineates the essential procedures.

Algorithm 4.2. Local density Calculation.

Require: Graph G
Ensure: List of nodes sorted by their Local density scores
1. Create subgraph G' using BFS with a hop distance of 2 from each node
2. Calculate Local density scores for every node in G' using equation 4.2
3. Store nodes along with their Local density scores in the list $Local$

In the first phase of our process, we use the concept of Local density scores as a fundamental tool to pinpoint and rank nodes according to their importance within their nearby environment. This evaluation of Local density scores assigns numerical values to nodes, which are then organized in a specific order. These Local density scores, coupled with the ordered nodes, form the fundamental building blocks for the ensuing steps within our DBOCD (Density-Based Overlapping Community Detection) algorithm.

4.2.2 Selecting Initial Classes

In the second phase of the DBOCD algorithm, the crucial task of creating initial overlapping classes is undertaken to establish a foundation for efficient community detection. This involves identifying nodes with the highest Local density and using Breadth-First Search (BFS) with a hop distance of 2 to combine these nodes with their neighbors and neighbors' neighbors, forming cohesive initial classes. This process iterates until all nodes are assigned to at least one class.

Following the establishment of initial classes, the algorithm proceeds with the first step of merging to optimize the number of classes. Two classes are merged if the DisMax value, calculated using equations 4.3, 4.4, and 4.5, is less than or equal to 1. This merging process is crucial for ensuring the cohesiveness and non-redundancy of the initial classes, prerequisites for effective community detection.

The formula to compute the similarity between two communities, denoted as C_i and C_j, is delineated by equations 4.3 and 4.4:

$$Dissimilarity(C_i, C_j) = |C_i| - |C_i \cap C_j| \tag{4.3}$$

$$Dissimilarity(C_j, C_i) = |C_j| - |C_i \cap C_j| \tag{4.4}$$

To encapsulate this essential dissimilarity measure, DBOCD introduces a composite measure called DisMax, computed as the maximum value between the dissimilarities of C_i and C_j (formula 4.5). The DisMax measure serves as a pivotal criterion for merging communities, emphasizing the importance of community overlap and node density in the merging process.

$$DisMax(C_i, C_j) = \max(Dissimilarity(C_i, C_j), Dissimilarity(C_j, C_i)) \tag{4.5}$$

In the Density-Based Overlapping Community Detection (DBOCD) algorithm, the dissimilarity between two communities, C_i and C_j, is crucially assessed through these measures. As communities with high DisMax values exhibit minimal overlap, this strategic merging leads to the formation of cohesive and non-redundant initial classes, optimizing the efficiency of community detection in complex networks.

The merging phase enhances the efficiency of community detection in networks by creating initial classes that are not only densely connected but also distinct from each other. The framework of the DBOCD algorithm is visually represented in Figure 4.1.

This phase can be bifurcated into two distinct stages. In the first stage, the algorithm focuses on the creation of the initial set of classes. In the second stage, attention shifts to optimizing the number of classes by identifying and eliminating similarities or redundancies. Algorithm 4.3 delineates the detailed procedures for this crucial phase.

Algorithm 4.3. Selecting Initial Classes.

Require: List of nodes sorted by their Local density scores (Local)
Ensure: Initial Classes

1. Select the node with the highest Local density score ▷ Initiate class formation
2. Add the selected node and its BFS to a community ▷ Expand community with neighbors
3. Adding nodes at least to a community ▷ Complete community assignment
4. Calculate DisMax (equation 4.5) ▷ Evaluate community dissimilarity
5. Merge two communities if their Maxdis ≤ 1 ▷ Ensure cohesive and non-redundant classes

4.2.3 Grouping communities

To facilitate the grouping of classes in the DBOCD algorithm, dissimilarity between communities is employed using the previously defined DisMax formula (formula 4.5). The process is iterated until the number of overlapping nodes decreases, as detailed in subsequent sections.

The specific conditions for each merging step are outlined as follows:

Merging Step 1:

The algorithm begins by calculating the number of overlapping nodes identified in the previous phase. If this count is less than or equal to 1, the process stops, and a choice is made between disjoint or overlapping community detection. However, if the count exceeds 1, regrouping is initiated with the condition that Maxdis is less than or equal to 3.

Merging Step 2:

The quantity of overlapping nodes identified in the prior phase is then assessed. If this count is at or below 10, the process concludes, and a decision is made between disjoint or overlapping community detection. If the count exceeds 10, the algorithm initiates the regrouping step, with the condition that Maxdis does not surpass 5.

Merging Step 3:

The algorithm evaluates the number of overlapping nodes identified in the previous phase. If this count is 50 or less, the process concludes, and a decision is made between disjoint or overlapping community detection. If the count surpasses 50, the algorithm proceeds with the regrouping step, ensuring that Maxdis remains at or below 10.

Merging Step 4:

In the final merging step, if the overlap list contains more than 100 elements, a similar merging process is carried out by comparing possible pairs of communities. The threshold between two communities for merging is set to be less than or equal to 20.

The overlap list, containing unique elements among the members of all communities, is updated. To operationalize this stage, Algorithm 4.4 provides a comprehensive, step-by-step delineation of the procedural framework.

Algorithm 4.4. Grouping Overlapping Classes.

Require: Overlapping Classes
Ensure: Disjoint or overlapping communities
Calculate Ov, the number of overlapping nodes.
if $Ov \leq 1$ **then**
Calculate the dissimilarity between each pair of communities using the DisMax measure in equation 4.5
Regroup classes with $Maxdis \leq 3$
else
if $Ov \leq 10$ **then**
Regroup classes with $Maxdis \leq 5$
else
if $Ov \leq 50$ **then**
Regroup classes with $Maxdis \leq 10$
else
if $Ov \leq 100$ **then**
Regroup classes with $Maxdis \leq 20$
end if
end if
end if
end if
Detect communities.

This algorithm aggregates classes exhibiting minimal dissimilarity, employing an overlapping hierarchical clustering method to refine nodes identified in the preceding step. The fundamental concept is to unite nodes with minimal dissimilarity based on shared non-common node patterns, aiming to unveil coherent communities within the graph characterized by shared nodes and connections. The algorithm initiates by selecting the node with the highest Local density score obtained during the initial stage, which becomes the core of a community. Its BFS nodes are progressively incorporated into this developing community until all graph nodes are assigned to at least one community. The iterative process involves assessing dissimilarity between each pair of communities and continues until the number of overlapping nodes decreases, aligning with the complexity and interconnection of communities.

The determination of overlap and dissimilarity values in our DBOCD algorithm is an iterative process grounded in rigorous testing and evaluation. We conducted a series of comprehensive tests on diverse network datasets to assess the algorithm's performance across a spectrum of scenarios. These tests involved systematically varying overlap and dissimilarity thresholds to observe their impact on the resulting community structures. Through this experimentation, we aimed to identify values that strike a balance between precision and generality in community detection. The tests also considered the algorithm's sensitivity to changes in these parameters, providing insights into the robustness of our approach. The final selection of overlap and dissimilarity values is thus informed by empirical evidence derived from these extensive tests, ensuring that our algorithm is well suited to adapt to the complexities inherent in diverse networks.

4.3 The time complexity

Complexity analysis plays a crucial role in understanding the computational demands of an algorithm. In the context of our algorithm, several factors come into play, such as the number of nodes (N), the number of edges (E), and the number of communities (K) within the graph. Let's break down the complexity of each step:

Calculating Local density for All Nodes: The complexity of this step depends on the specific algorithm used for calculating Local density. In general, it falls within the range of O(n^2). This is because, for each node, we might need to consider interactions with all other nodes, resulting in a quadratic relationship with the number of nodes (N).

Selecting initial classes: This step involves nested loops as we traverse nodes and communities to make initial class selections. The overall complexity is typically O(K * N^2), where K is the number of communities. The nested structure arises because, for each community, we evaluate the entire set of nodes (N).

Regrouping: This step also employs nested loops to traverse nodes and communities in order to remove overlapping nodes. Additionally, we need to assess the connectivity of the subgraphs formed by the communities. The complexity here is influenced by the number of sub-communities and the size of each sub-community, which can vary. Consequently, the complexity can be approximated as O(C), where C represents the combined influence of the number of sub-communities and their sizes.

The complexity of the algorithm can be roughly estimated as:

$$O(N^2 \cdot K \cdot C). \tag{4.6}$$

Where:
- N : the number of nodes in the graph
- K : is the number of communities
- C : represents the combined complexity influenced by sub-communities and their sizes.

This complexity estimation is useful for assessing the computational demands of the algorithm on graphs of varying sizes and community structures.

4.4 Applying DBOCD to network: a case study

To illustrate the functionality of our DBOCD algorithm, we showcase its application on a sample network depicted in Figure 4.2a, which comprises 11 nodes and 13 edges. The graph is unweighted and undirected. Refer to Table 4.1 for a comprehensive summary of Local density values within each node's subgraph, presented in descending order.

The second phase of our clustering approach involves establishing initial classes using the

top nodes from the array in Table 4.1, along with their respective Breadth-First Search (BFS) outcomes. Subsequently, we employ hierarchical clustering, a process capable of producing clusters with overlapping data points.

This phase entails grouping nodes based on their similarity, forming initial clusters that lay the groundwork for further refinement. The hierarchical clustering process, known for handling overlapping structures, contributes to the generation of more intricate and nuanced communities within the network.

By following these steps, we provide a detailed understanding of how the DBOCD algorithm operates in the context of a specific network, offering insights into the creation and refinement of clusters with overlapping characteristics.

Nodes according to Local density

N	’1’, ’6’, ’10’	’9’, ’11’	’2’, ’3’, ’5’, ’7’	’8’	’4’
Local density	0.667	0.5	0.333	0.25	0.24
Order	’1’, ’6’, ’10’, ’9’, ’11’, ’2’, ’3’, ’5’, ’7’, ’8’, ’4’				

Table 4.1: Nodes ordered by local node density.

Grouping with Dissimilarity We group nodes with their BFS:

> Node ’1’: C1 = $['1','2','3','4']$.

> Node ’6’: C2 = $['4','5','6','7']$.

> Node ’10’: C3 = $['8','9','10','11']$.

> Stop here because all nodes are at least in a class.

We find one overlapping community: $C1$ and $C2$.

The maximum dissimilarity (**Maxdis**$(C1, C2) = 3$) allows grouping in these two classes:

> $C1 = ['1','2','3','4']$ (Green).

> $C2 = ['4','5','6','7']$ (Yellow).

> $C3 = ['8','9','10','11']$ (Pink).

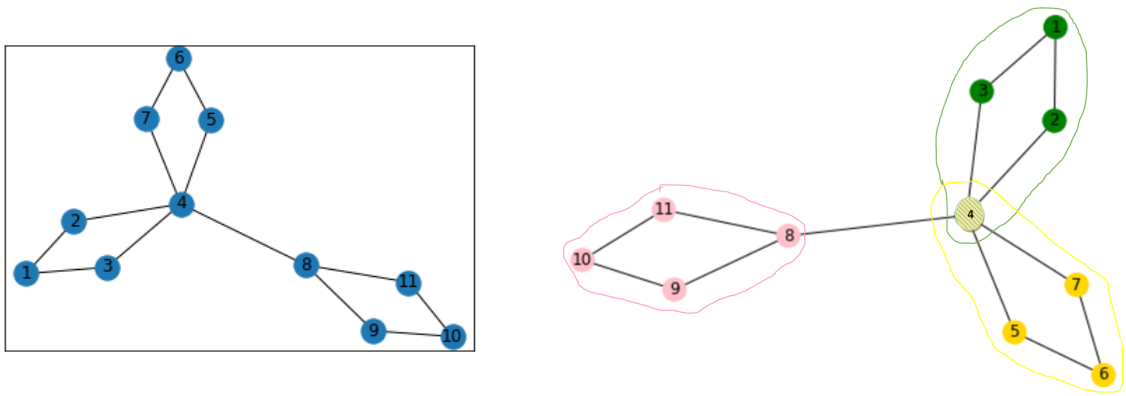
Grouping communities

The number of overlapping nodes is $Ov = 1$. Any further grouping will detect communities.

The Figure 4.2b present the overlapping community detection of this example.

4.5 Experimental Results

The DBOCD algorithm demonstrates its robustness as a valuable tool for uncovering both disjoint and overlapping communities within intricate networks. Nevertheless, it is crucial to acknowledge that the efficacy of the DBOCD algorithm can be contingent upon the distinct characteristics of the studied network and the selected parameter configurations. Consequently,



(a) Illustrative example. (b) Communities of example.

Figure 4.2: Example of community detection with DBOCD.

it is recommended to conduct experiments utilizing evaluation measures and modularity for overlapping communities to assess the algorithm's effectiveness.

To verify the effectiveness of DBOCD, extensive experiments were carried out using a Python implementation. These experiments were carried out on a computer featuring an Intel Core i7 processor with 16 GB of RAM, which ensured remarkable performance. DBOCD's performance was evaluated on four real network datasets and five synthetic networks from the LFR benchmark. The aim of this comprehensive evaluation was to confirm the effectiveness of the algorithm in identifying significant communities across a range of network scenarios.

4.5.1 Experiments on real-World Networks

The experimental datasets comprised real-world networks, namely 'Zachary's Karate Club' network (Zachary, 1977), 'Dolphins of Lusseau' (Lusseau et al., 2003), 'Books about Politics' (Krebs, 2008), and 'The American College Football Network' (Jiang & McQuay, 2012). A summary of the key information for these networks is presented in Table 4.2.

	Karaty	Dolphin	PolBooks	Football
Nodes	34	62	105	115
Edges	78	159	440	612
Number C	2	2	3	12

Table 4.2: Properties of Real Networks

The proposed DBOCD algorithm showcases exceptional performance across a diverse range of real-world networks. A notable strength of our algorithm lies in its stability, evident in consistent results across repeated runs. Table 4.3 provides a visual representation of the modularity values assigned to networks by DBOCD.

The proposed DBOCD algorithm shows remarkable performance on a spectrum of real-

	Karaty	Dolphin	PolBooks	Football
Number C of algo	2	2	3	10
Number overlapping nodes	2	2	3	10
Q_M	0.3582	0.5191	0.4285	0.5081

Table 4.3: Modularity of DBOCD with Real Networks.

world networks, highlighting its versatility and efficiency in capturing diverse structural patterns. The stability of the algorithm is a notable strength, as evidenced by the consistent results obtained in multiple runs. When examining specific real-world networks, such as ”Zachary’s Karate Club”, ”Dolphins of Lusseau”, ”Books about US Politics” and ”The American College Football Network”, the algorithm successfully identifies significant communities. The number of communities detected corresponds well to the expected structures in each domain, demonstrating the algorithm’s ability to adapt to different types of network. The inclusion of overlapping nodes adds a layer of complexity to the communities, enhancing the algorithm’s ability to reveal nuanced relationships within the networks. Quantitative evaluation by modularity values Q_M provides a robust measure of community quality, with higher modularity values indicating the effectiveness of the algorithm in forming cohesive and distinct communities. This comparative analysis of various real-world networks highlights the reliability of the algorithm and its potential for real-world application, providing valuable information for decision-making processes. While celebrating these successes, it is essential to recognise the limitations

4.5.2 Experiments on generated network ”LFR Benchmark”

The Lancichinetti Fortunato Radicchi (LFR) benchmark, introduced by Lancichinetti, Fortunato, and Radicchi (2008), stands as a widely adopted tool for assessing and comparing community detection algorithms on synthetic networks. This benchmark provides researchers with a versatile framework to generate networks featuring predefined community structures and tunable parameters, replicating diverse scenarios encountered in real-world networks. Its utility lies in enabling a comprehensive evaluation of algorithmic performance, scalability, and robustness. Through comparisons with ground-truth community structures, researchers gain valuable insights into the strengths and limitations of different methods. The LFR benchmark’s significance extends to its role in facilitating systematic exploration of algorithm behavior under various conditions. Researchers utilize this tool to refine their approaches, contributing to the ongoing advancement of community detection methods in the field of network analysis.

Our evaluation on LFR benchmark networks, ranging from 250 to 3000 nodes with an average degree between 5 to 10, reveals the DBOCD algorithm’s remarkable ability to identify overlapping community structures. By setting the degree distribution exponent t_1 to 1.5 and the community size distribution exponent t_2 to 3, we introduce variability simulating real-world scenarios. The exploration of two community size intervals (10 to 50 nodes and 20 to 100 nodes)

and two maximum degree values (20 or 50) adds granularity to our analysis, offering insights into the algorithm's behavior under different network structures. The mixing parameter nu, set to 0.1, captures network nuances, signifying the expected fraction of links through which a node connects with others in the same community.

In Table 4.4, we present the modularity values obtained from our evaluation. The increasing trend in modularity as the network size grows indicates the algorithm's scalability and effectiveness in handling larger, more complex networks.

Figure 4.3 visually represents the modularity overlap in artificial networks, showcasing the consistency and robustness of the DBOCD algorithm across varying synthetic network conditions. These results underscore the algorithm's practical applicability in uncovering overlapping community structures, offering valuable insights into network organization and connectivity.

N	200	400	600	800	1000	2000	3000
Q	0.7735	0.7817	0.7935	0.7812	0.7819	0.7799	0.7865

Table 4.4: Valeurs of overlapping modularity of networks.

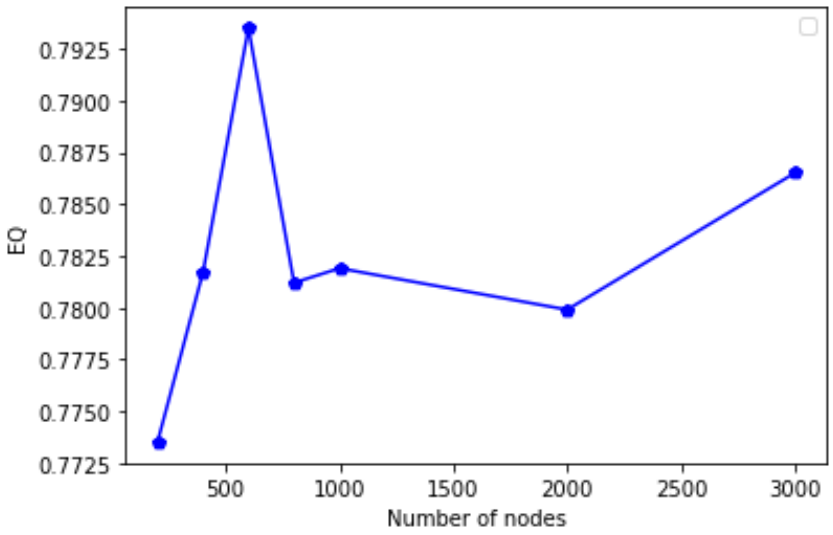


Figure 4.3: Modularity overlap in LFR benchmark networks.

4.6 Conclusion

In our investigation, we incorporated the concept of "Local density" into a community detection algorithm known as DBOCD, which leverages overlapping hierarchical clustering techniques. The DBOCD algorithm capitalizes on a hierarchical clustering approach to assess node similarity, resulting in a significant reduction in the number of iterations needed for community partitioning, thereby enhancing overall efficiency.

Through the strategic initiation of the population based on the concept of Local density and the implementation of an innovative neighbor-based clustering operator, DBOCD demonstrates

clear superiority over conventional methods that rely on random initialization. This assertion finds robust support in the outcomes of our extensive experiments, conducted on both actual and synthetic networks. Notably, our approach attains superior values in terms of modularity overlap, serving as reliable indicators closely mirroring real-world community structures. These results underscore not only efficiency but also stability.

In the upcoming chapter, our objective is to broaden the application of DBOCD by addressing the challenge of overlapping memes within exceedingly large and intricate networks. This line of inquiry holds significant promise for advancing the field of community detection in analytical research.

Chapter 5

PCMeans: Community Detection by PageRank and K-means

5.1 Introduction

Community detection in networks is a fundamental challenge in network analysis, particularly in the context of the intricate structures found in social networks. Addressing this challenge, this chapter introduces PCMeans, a novel approach designed for efficient community detection. PCMeans operates through three key phases: identifying crucial nodes using Local PageRank, partitioning data points through hierarchical clustering based on similarity, and refining clusters with k-means clustering to obtain disjoint clusters from the overlapping ones identified earlier.

PCMeans stands out for its simplicity, low time complexity, and ease of implementation. The method adopts an initial number of communities and nodes, the determination of which is elaborated later in this chapter. Experimental results and comparative analysis, as detailed in Louafi and Titouna (2023b), demonstrate that PCMeans consistently delivers high-quality results for both real-world and synthetic networks, rivaling other algorithms in terms of accuracy. Notably, PCMeans exhibits superior efficiency, addressing slow convergence through its streamlined processes.

This chapter provides an in-depth presentation of PCMeans, including a detailed discussion of results and thorough comparisons with existing algorithms in the literature.

5.2 Methodology

In this section, we introduce the PCMeans algorithm, a novel approach for community detection in graphs. The PCMeans algorithm employs a multi-stage process involving Local PageRank calculation, overlapping hierarchical clustering, and K-means clustering. These stages work sequentially to identify and refine communities within the graph. This algorithm consists of three successive stages that together form a comprehensive approach for community detection. The Figure 5.1 illustrates well the different stages of our approach. These stages are defined as follows:

- Local PageRank Calculation: The initial stage of the PCMeans algorithm involves computing the Local PageRank score for each node within the graph. Local PageRank measures the importance of nodes within their local neighborhoods and is calculated using a modified Breadth First Search approach. The nodes are then sorted in descending order based on their Local PageRank scores, forming a ranked list of nodes by importance within their local contexts.
- Overlapping Hierarchical Clustering: In the second stage, the algorithm initiates an overlapping hierarchical clustering process. Starting with the node exhibiting the highest Local PageRank score, a new cluster is formed by including this node and its neighbors. This clustering process iterates until every node is allocated to at least one cluster. To assess the cohesion between clusters, the algorithm evaluates the similarity based on the nodes they share. Specifically, the algorithm computes the overlap in nodes between any two

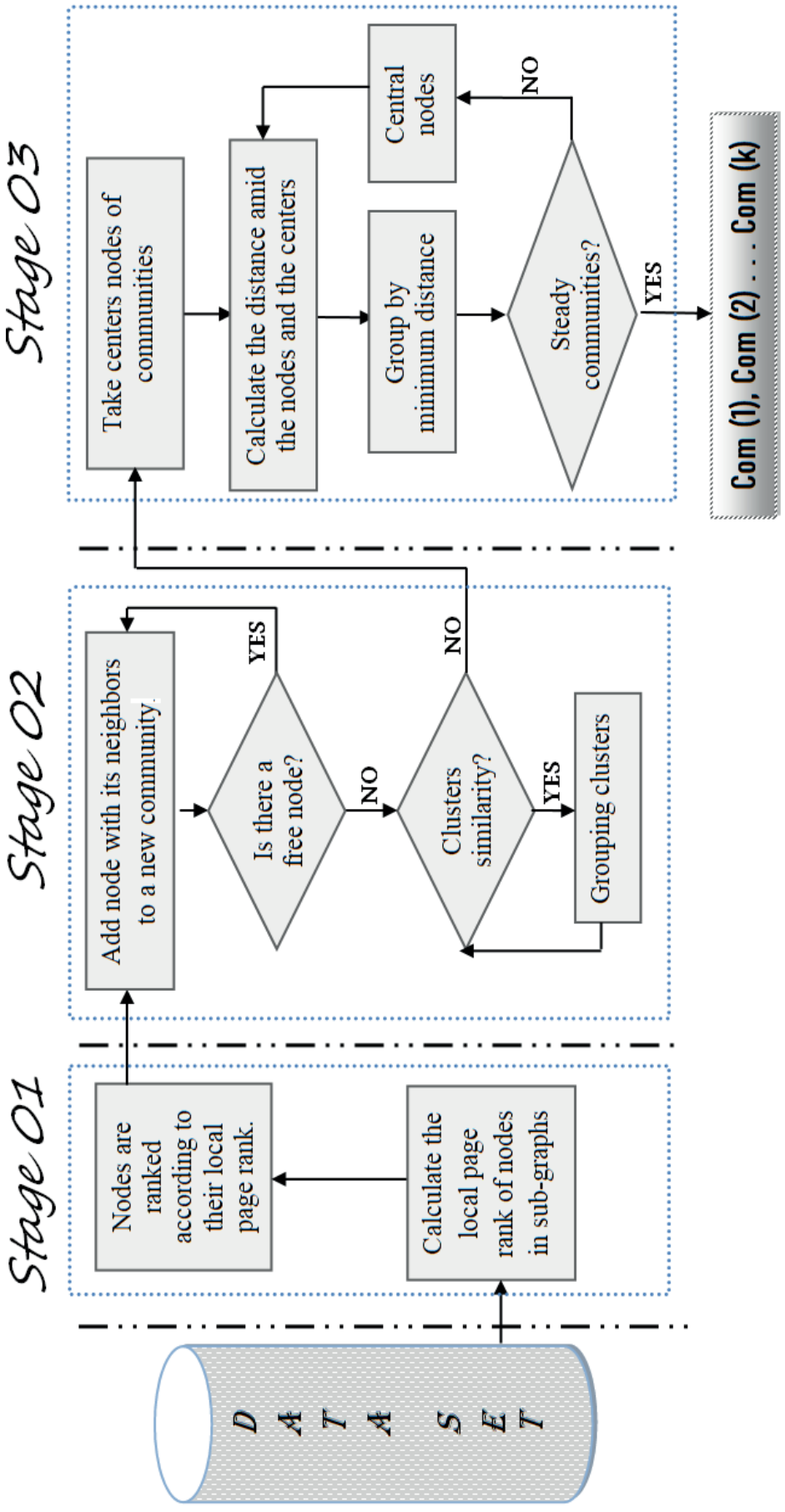


Figure 5.1: Flowchart of the PCMeans Algorithm.

clusters and calculates the similarity as a percentage. When the similarity between two clusters surpasses the predefined threshold of 50%, the algorithm merges these clusters. This merging process continues iteratively until no further merging is feasible, ultimately yielding a set of refined and non-overlapping communities.The 50% threshold ensures meaningful overlap between clusters, enhancing the accuracy of the identified community structure through multiple tests.

> K-means Clustering: The final stage of PCMeans involves applying K-means clustering to the communities identified in the second stage. The initial centroids for K-means are set as the centers of the communities obtained from overlapping hierarchical clustering. The number of clusters is set equal to the number of communities identified in the previous stage. K-means clustering further groups nodes within the same community while addressing overlapping data points in the clusters.

Algorithm 5.1 provides a global overview of the PCMeans algorithm, detailing the steps involved in each stage.

Algorithm 5.1. PCMeans Algorithm.

Require: Graph G
Ensure: Communities $C_1, C_2, \ldots, C_K$
1: Calculate Local PageRank scores for each node
2: Sort the nodes according to Local PageRank
3: Realize overlapping hierarchical clustering
4: Refining clusters based on similarity until no more merging is possible
5: Set k equal to the number of communities
6: Calculate centres of communities
7: Use this centers as initial centroids for K-means
8: Apply K-means clustering on the communities

In the following subsections, we provide a detailed breakdown of each stage in the PCMeans algorithm, highlighting the methodologies employed and the significance of each stage in the community detection process.

5.2.1 Detection of important nodes by local PageRank

The PCMeans algorithm initiates its community detection process by leveraging the concept of Local PageRank to identify influential nodes within a graph (Brin & Page, 1998; Bar-Yossef & Mashiach, 2008). Local PageRank serves as a metric for quantifying the significance of a node within its immediate neighborhood. This local neighborhood is defined as the set of nodes that are reachable within a distance of two hops from the target node. This calculation is facilitated by a modified Breadth First Search algorithm with a hop distance of 2. The resulting subgraph obtained from this algorithm is confined to vertices and edges that are within these two hops. This approach provides valuable insights into the local structure of a larger graph, enabling the assessment of a node's importance within its immediate surroundings.

However, it's important to emphasize that while the Local PageRank focuses on local neighborhoods, the PCMeans algorithm operates as a global community detection method. This implies that it considers the entire graph's structure rather than just localized regions. While Local PageRank contributes to the understanding of node significance within a specific vicinity, it is only one facet of the multifaceted factors considered by PCMeans. The algorithm also accounts for edge weights connecting nodes and their corresponding cluster assignments to establish the definitive community structure of the graph.

The foundation of this stage lies in the calculation of Local PageRank scores for each node within the graph. The calculated scores are then employed to sort the nodes in descending order based on their Local PageRank values. The Local PageRank calculation for a node v_i is formulated as follows in equation 5.1:

$$PR(v_i) = \frac{1-d}{N} + d \sum_{v_j \in M(v_i)} \frac{PR(v_j)}{L(v_j)} \tag{5.1}$$

Where:
d denotes the damping factor,
v_i and v_j represent the nodes under consideration within graph G',
$M(v_i)$ constitutes the set of nodes linked to node v_i within graph G',
$L(v_i)$ denotes the count of links within subgraph G' associated with node v_i,
N signifies the total number of nodes within graph G'.

The ordering of nodes based on their Local PageRank values establishes a ranked list, with nodes of higher importance appearing at the top. In situations where nodes share the same Local PageRank value, their arrangement is determined by their node numbers in ascending order. This ranked list of nodes, along with their corresponding Local PageRank scores, is preserved for subsequent stages of the algorithm. Algorithm 5.2 outlines the key steps of this stage.

This initial stage employs Local PageRank as a mechanism to identify and rank nodes based on their significance within their local environments. These Local PageRank scores, along with the sorted nodes, form the groundwork for subsequent stages of the PCMeans algorithm.

Algorithm 5.2. Local PageRank Calculation.

Require: Graph G
Ensure: List of nodes sorted by their Local PageRank scores ($Local$)
1: Create subgraph G' using Breadth First Search with a hop distance of 2 from each node
2: Calculate Local PageRank for every node in G' using equation 5.1
3: Store nodes along with their Local PageRank scores in the list $Local$

5.2.2 Overlapping Hierarchical Clustering

In this stage, the PCMeans algorithm employs an innovative overlapping hierarchical clustering technique to further refine the nodes identified in the previous stage. The fundamental

premise of this technique is to group nodes that exhibit significant similarity in terms of their local connectivity patterns. By building upon the Local PageRank scores obtained earlier, this stage endeavors to uncover cohesive communities within the graph that share common nodes and connections.

The algorithm's initiation involves selecting the node with the highest Local PageRank score achieved in the initial stage. This high-ranking node becomes the nucleus of a community, and its neighboring nodes are incrementally incorporated into this developing community.

This process is continued iteratively until all nodes within the graph are assigned to at least one community. As this iterative process unfolds, the algorithm assesses the degree of similarity between each pair of communities.

To quantify the similarity between communities, PCMeans employs a similarity measure based on the percentage of common nodes between two communities. This measure serves as a pivotal component in deciding whether two communities should be combined. The formula to compute the similarity between two communities, denoted as C_i and C_j, is delineated by equations 5.2 and 5.3:

$$Similarity(C_i, C_j) = \frac{|C_i \cap C_j|}{|C_i|} \tag{5.2}$$

$$Similarity(C_j, C_i) = \frac{|C_i \cap C_j|}{|C_j|} \tag{5.3}$$

To encapsulate the essential similarity mesure, PCMeans introduces a composite measure called SimMax, computed as the maximum value between the similarities of C_i and C_j:

$$SimMax(C_i, C_j) = \max(Similarity(C_i, C_j), Similarity(C_j, C_i)) \tag{5.4}$$

One of the notable strengths of this approach is its inherent capacity to accommodate nodes belonging to multiple communities, a trait that resonates with the intricate and interconnected nature of real-world networks. By harnessing this technique, PCMeans facilitates the formation of a hierarchical structure of interconnected communities, offering insights into the interplay between these communities at various levels of granularity.

However, the algorithm acknowledges that while overlapping communities have their merits, there's value in employing disjoint communities to attain more precise results. In anticipation of the subsequent stage, PCMeans plans to use the number and central characteristics of these communities, further refining the insights extracted from the clustering process.

This stage plays a pivotal role in refining the communities detected based on their local prominence, fostering the creation of comprehensive and cohesive groupings with varying degrees of overlap. The resulting overlapping communities serve as a foundation for subsequent

analysis and clustering steps in the PCMeans algorithm.

To implement this stage, Algorithm 5.3 outlines the step-by-step procedure:

Algorithm 5.3. Overlapping Hierarchical Clustering.

Require: List of nodes sorted by their Local PageRank scores (Local)
Ensure: Overlapping community detection

1: Select the node with the highest Local PageRank score
2: Add the selected node along with its neighbors to a community
3: Continue adding nodes to the community until all nodes are assigned to at least one community
4: Calculate the similarity between each pair of communities using the SimMax measure in equation 5.4
5: Merge two communities if their similarity exceeds the half
6: Repeat steps 4 and 5 until further merging is infeasible

5.2.3 K-means Clustering

K-means clustering stands as a widely adopted methodology employed to segment a collection of data points into distinct clusters, with the primary objective of grouping similar points together. In the context of community detection within complex networks, K-means serves as a valuable tool for categorizing nodes into distinct communities based on their inherent similarities or distances. It's imperative to recognize that the efficacy of the community detection process is heavily contingent on the judicious selection of the parameter K, which signifies the number of communities sought. This selection often involves a balance between empirical trial and error and the use of external validation metrics.

It's notable that K-means clustering is not immune to the nuances of its initialization. The initial placement of centroids plays a crucial role in determining the final clustering outcome. At times, K-means may also converge to suboptimal solutions due to its sensitivity to initial conditions. In order to mitigate this issue, it is recommended to execute the algorithm multiple times with varying initializations and subsequently select the most promising solution based on established criteria, such as the minimization of the sum of squared distances between nodes and their respective centroids.

Within the PCMeans algorithm, the choice of K is informed by the number of communities identified during the previous stage of overlapping hierarchical clustering. These communities' centers are then designated as the initial centroids for the ensuing K-means clustering operation. This approach not only harnesses the insights gained from the overlapping hierarchical clustering stage but also accelerates the convergence of K-means due to the strategic initial placement of centroids.

The application of K-means clustering serves as a post-processing step subsequent to the hierarchical clustering phase. This strategic sequencing aligns with the goal of refining the

community structure identified through hierarchical clustering. K-means is particularly well-suited for this task due to its efficiency in partitioning data points into non-overlapping clusters, further delineating community boundaries with precision. The procedure underlying the K-means clustering stage is encapsulated in Algorithm 5.4.

Algorithm 5.4. K-means Clustering Algorithm.

Require: Identified communities, k: the desired number of communities.
Ensure: Communities $C_1, C_2, \ldots, C_K$
1: Set the value of k equal to the number of communities identified in the previous stage
2: Utilize the centers of the identified communities as the initial centroids
3: Assign each data point to the nearest centroid of its corresponding cluster
4: Recalculate the centroids of the newly formed clusters
5: Iterate steps 3 to 4 until the algorithm converges

This stage represents the culmination of the PCMeans algorithm's community detection journey. The K-means clustering technique lends structure to the communities discovered through previous stages, offering a clear delineation of individual community memberships. The refined community structure produced by this stage forms a valuable basis for further analysis and insights into the network's underlying structure.

5.3 The time complexity

The time complexity analysis of the PCMeans algorithm is a crucial aspect to assess its computational efficiency. PCMeans exhibits favorable time complexity in comparison to some alternative community detection algorithms. This advantage can be attributed to several key factors within the algorithm's design.

Firstly, the time complexity of PCMeans depends on its various stages. The initial stage involves calculating the Local PageRank score for each node, denoted as T1, and its time complexity is $O(NlogN)$, where N is the number of nodes.

The second stage, which involves Overlapping Hierarchical Clustering and is denoted as T2, has a time complexity of $O(K^2 * NlogN)$, where K represents the number of communities.

The third stage of the algorithm entails running K-means clustering on the communities to group nodes into K non-overlapping clusters, and it is denoted as T3. The time complexity of this stage is $O(it * K * n)$, where t signifies the number of iterations required for convergence.

The overall time complexity of the PCMeans algorithm can be expressed as follows:

$$O(nlogn + K^2 nlogn + it * K * n). \tag{5.5}$$

PCMeans stands out as an efficient method for community detection in large networks due to its low time complexity. This efficiency is primarily achieved through the incorporation of

Local PageRank, the use of overlapping hierarchical clustering, and the effectiveness of K-means clustering. These factors collectively make PCMeans a practical and powerful tool for analyzing complex network structures. The table 5.1 compares between the complexity of our algorithm with other well known community detection methods.

Method	Complexity
Clauset and Newman	$O(n^2 d \log n)$
Girvan and Newman	$O(n^5)$
Walktrap	$O(n^4)$
Fortunato	$O(n^7)$
Louvain	$O(mn^3)$
DBOCD	$O(n^2 \cdot K \cdot C)$
PCMeans	$O(n \log n + K^2 n \log n + it \cdot K \cdot n)$

Table 5.1: Comparison of complexity for various community detection algorithms.

The reasons behind PCMeans' low time complexity:

Local PageRank: PCMeans employs Local PageRank to identify influential nodes. This approach significantly reduces the computational burden and search space, contributing to its efficiency.

Overlapping Hierarchical Clustering: The algorithm's strategy of overlapping hierarchical clustering aids in quickly identifying the optimal number of clusters. This feature can reduce the number of iterations needed for convergence.

K-means Clustering: The use of the K-means algorithm in PCMeans is advantageous for its rapid convergence and effectiveness in large-scale datasets, further enhancing the algorithm's efficiency.

5.4 Applying PCMeans to network: a case study

Figure 5.2 depicts a sample network (unweighted, undirected graph with 20 nodes and 19 edges).

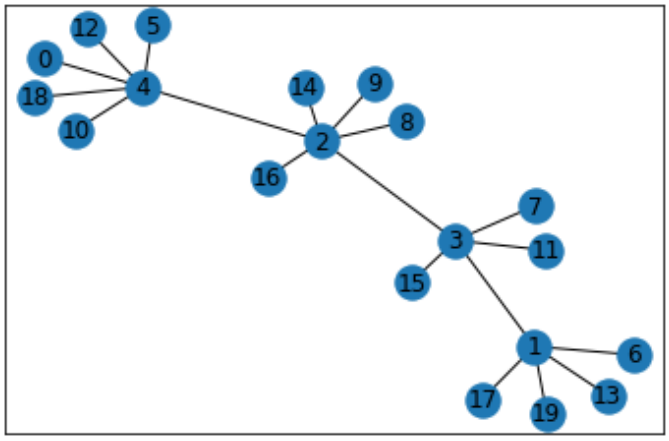


Figure 5.2: Simple network with 20 nodes.

Meanwhile Table 5.2 presents a summary of the PageRank values for nodes within the two-hop subgraph of each node, including the node order based on their respective PageRank values.

N	1	4	2	3	6,7,11,13,15, 17,19	0,5,8,9,10,12,14,16
LocalPagerank	0.26	0.25	0.18	0.10	0.16	0.08
Ordre	’1’,’4’,’2’,’3’,’6’,’7’,’11’,’13’,’15’,’17’,’19’,0’,’5’,8’,’9’,’10’,’12’,’14’,’16’,’18’					

Table 5.2: PageRank values of nodes.

The second phase of our clustering approach involves hierarchical clustering, capable of generating clusters with overlapping data points. We cluster each node based on the order established in the table 5.2 and assign it to a cluster along with its neighbors until all nodes are used in the cluster lists. The Table 5.4 displays these clusters, with overlapping nodes highlighted in bold. Figure 5.3 this steps, overlapping nodes are in white.

Node	Neighbors
’1’	[**’1’**, **’3’**,′ 6′,′ 13′,′ 17′,′ 19′]
’4’	[′0′, **’4’**,′ 5′,′ 10′,′ 12′,′ 18′]
’2’	[**’2’**, **’3’**, **’4’**,′ 8′,′ 9′,′ 14′,′ 16′]
’3’	[**’1’**, **’2’**, **’3’**,′ 7′,′ 11′,′ 15′]

Table 5.3: Overlapping communities.

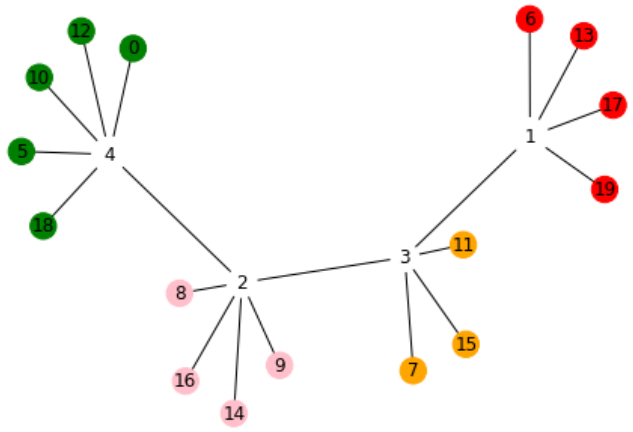


Figure 5.3: Communities in second stage of PCMeans.

To address the issue of overlap, we employ k-means clustering as a post-processing step following hierarchical clustering. Given its speed and capability to create non-overlapping groups, we apply k-means to the centers of the clusters obtained during hierarchical clustering, specifically the centroids of the communities denoted as [′1′,′ 2′,′ 3′,′ 4′].

The K-means method is then applied with k=4, resulting in a final partition of the data into distinct clusters. The goal of this third phase is to achieve a clustering solution that is more efficient and easier to interpret compared to overlapping clusters. We have four disjoint communities presented in Table 5.3 and Figure 5.4 illustrates the results.

Community	Nodes
'Red'	['1','6','13','17','19']
'Green'	['0','4','5','10','12','18']
'Pink'	['2','8','9','14','16']
'Yellow'	['3','7','11','15']

Table 5.4: Communities after applying k-means in PCMeans.

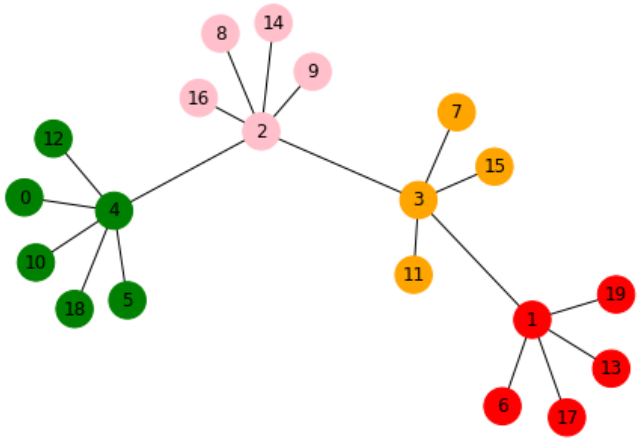


Figure 5.4: Communities in the third stage of PCMeans.

5.5 Experimental Results

The PCMeans algorithm emerges as a robust tool for uncovering communities within intricate networks. Its performance has been rigorously assessed, pitting it against other cutting-edge algorithms such as the Louvain Algorithm, Girven-Newman, INFOMAP Fluid Communities Algorithm, and Label Propagation Algorithm (LPA). In these comparative evaluations, PCMeans has consistently demonstrated competitive accuracy and efficiency. his optimisation can vary depending on the unique characteristics of the network under investigation and the chosen parameter configurations. Therefore, it is advisable to conduct experiments with diverse parameter settings and benchmark the outcomes against other algorithms to gauge PCMeans' suitability for a specific application.

To ascertain the effectiveness of PCMeans, extensive experimentation was carried out using a Python implementation. These experiments encompassed both synthetic and authentic social networks. The computational environment featured an Intel Core i7 processor and 16GB of RAM, ensuring robust performance.

The assessment of PCMeans' performance encompassed four real-world network datasets: Zachary Karate, Dolphin, Political Books, and Football, complemented by five synthetic networks sourced from the LFR benchmark. This comprehensive evaluation aimed to validate the algorithm's efficacy in capturing meaningful communities across a diverse range of network scenarios.

5.5.1 Experiments on real-World Networks

	Karaty	Dolphin	PolBooks	Football
Louvain (Blondel et al., 2008)	3	5	5	10
Newman(Newman & Girvan, 2004)	4	5	4	8
INFOMAP(Rosvall & Bergstrom, 2008)	3	5	6	12
Fast-Greedy(Parés et al., 2018)	4	4	4	7
LPA(Raghavan et al., 2007)	3	4	3	8
PCMeans	2	2	3	10

Table 5.5: Number of communities of different algorithms in real networks.

We calculate the Modularity Q for different algorithms. Each result is an average of 30 repetitions, with the best result highlighted. The descriptions of these networks and all results can be found in Table 5.6, and presented in Figure 5.5.

	Karaty	Dolphin	PolBooks	Football
Louvain (Blondel et al., 2008)	**0.4151**	0.5176	**0.5267**	**0.6045**
Newman(Newman & Girvan, 2004)	0.3943	0.4912	0.4671	0.4926
INFOMAP(Rosvall & Bergstrom, 2008)	0.4020	**0.5247**	0.5228	0.6005
Fast-Greedy(Parés et al., 2018)	0.3806	0.4954	0.5018	0.5784
LPA(Raghavan et al., 2007)	0.3990	0.4939	0.4565	0.5944
PCMeans	0.3582	0.5191	0.4285	0.5081

Table 5.6: Modularity of different algorithms with real networks.

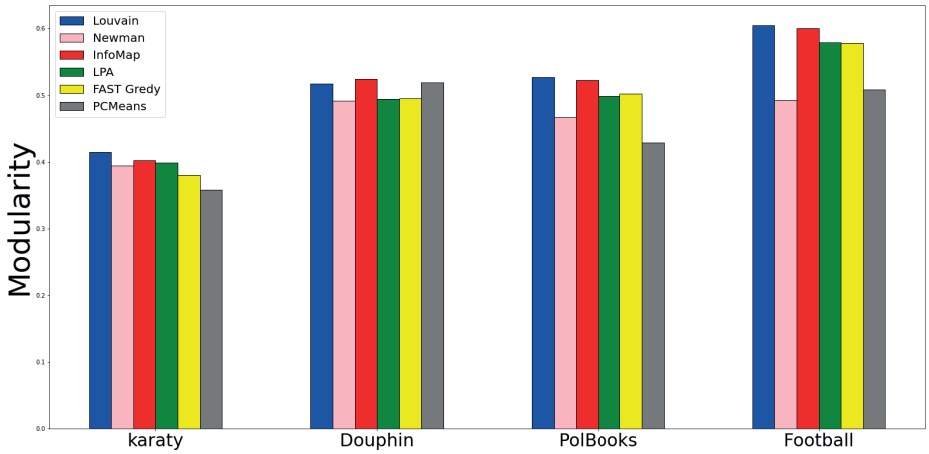


Figure 5.5: Modularity of real networks.

Moreover, while several algorithms exhibit unstable results, our algorithm remains stable. In summary, PCMeans produces similar partitions as compared to the other algorithms, while maintaining stability in its results.

The proposed algorithm delivers outstanding outcomes across diverse real-world networks. In the case of Zachary's karate club network, the algorithm exhibits convergence to the global

	Karaty	Dolphin	PolBooks	Football
Louvain (Blondel et al., 2008)	0.4899	0.5176	0.5368	0.8849
Newman(Newman & Girvan, 2004)	0.5791	0.4912	0.4671	0.6986
INFOMAP (Rosvall & Bergstrom, 2008)	0.5683	0.5247	0.5228	**0.9241**
Fast-Greedy(Parés et al., 2018)	0.5398	0.4954	0.5018	0.7623
LPA(Raghavan et al., 2007)	0.5683	0.4939	0.4986	0.8507
PCMeans	**1.0000**	**0.8140**	**0.6750**	0.9160

Table 5.7: NMI of different algorithms with real networks.

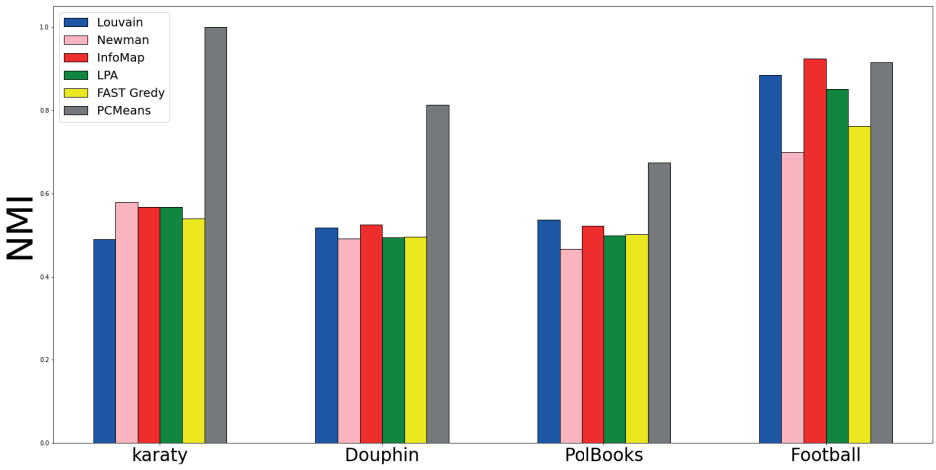


Figure 5.6: NMI of real networks.

optimum, evident from the NMI score of 1. This signifies an exact match between the algorithm-identified communities and the ground truth. Similar highly favorable results are also witnessed across other networks, including the Football club network, the Political book network, and the Dolphin network, with average NMIs of 0.9160, 0.6750, and 0.8140, respectively.

	Karaty	Dolphin	PolBooks	Football
Louvain (Blondel et al., 2008)	0.3922	0.2708	0.6421	0.8034
Newman(Newman & Girvan, 2004)	0.4351	0.3901	0.5466	0.4640
INFOMAP (Rosvall & Bergstrom, 2008)	0.5905	0.2830	0.5360	**0.8966**
Fast-Greedy(Parés et al., 2018)	0.5351	0.4954	0.6378	0.5363
LPA(Raghavan et al., 2007)	0.5905	0.4647	0.6745	0.6839
PCMeans	**1.0000**	**0.7659**	**0.7623**	0.8543

Table 5.8: ARI of different algorithms with real networks.

In a comparative analysis with other experimental algorithms, the proposed algorithm emerges as the frontrunner, consistently outperforming them in terms of both NMI and ARI indicators across all real-world networks. It additionally secures the highest ARI score across all experiments. It's worth noting that, specifically on the Political book network, the algorithm achieves the second-best modularity score of 0.5191, closely resembling INFOMAP's performance. While it doesn't clinch the top modularity score, the NMI and ARI scores affirm the

algorithm's exceptional proximity to the true results. These outcomes underscore the depend-

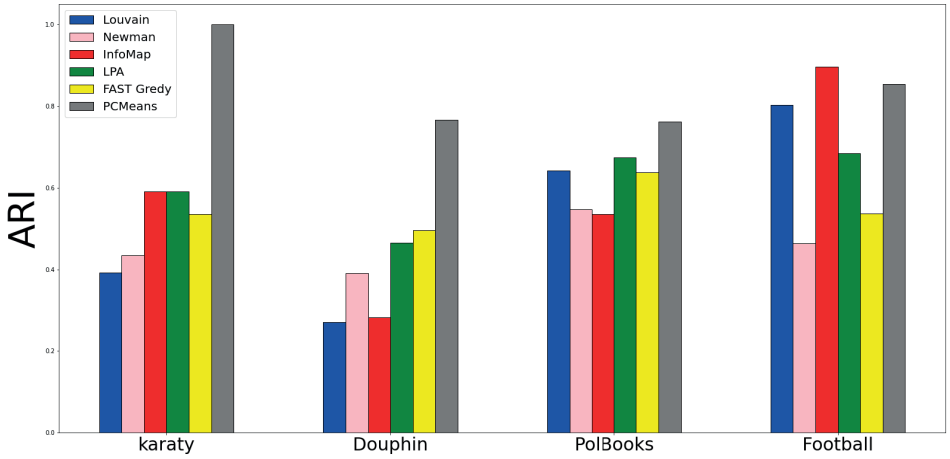


Figure 5.7: ARI of real networks.

ability and efficacy of the proposed algorithm in effectively discerning community structures within real-world networks.

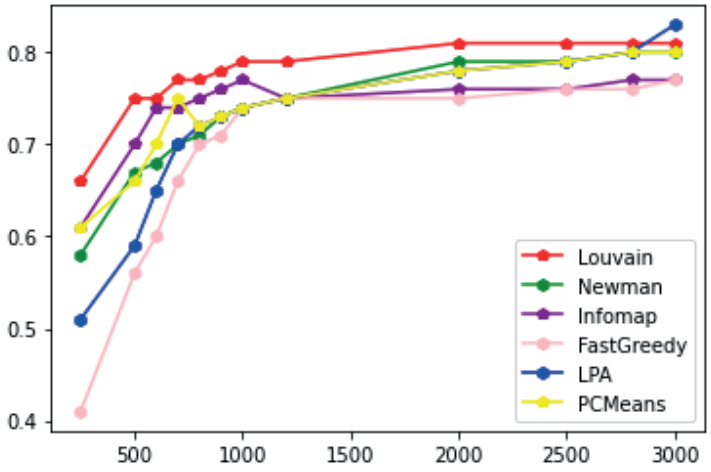
5.5.2 Experiments on Generated Network "LFR Benchmark"

To evaluate the precision of community detection algorithms, synthetic networks are commonly employed because they provide the advantage of controlling network characteristics and obtaining a known ground truth community structure by adjusting tunable parameters. One of the frequently used synthetic network models is the LFR benchmark network, developed by Lancichinetti, Fortunato, and Radicchi. This model mimics properties akin to those found in real-world networks.

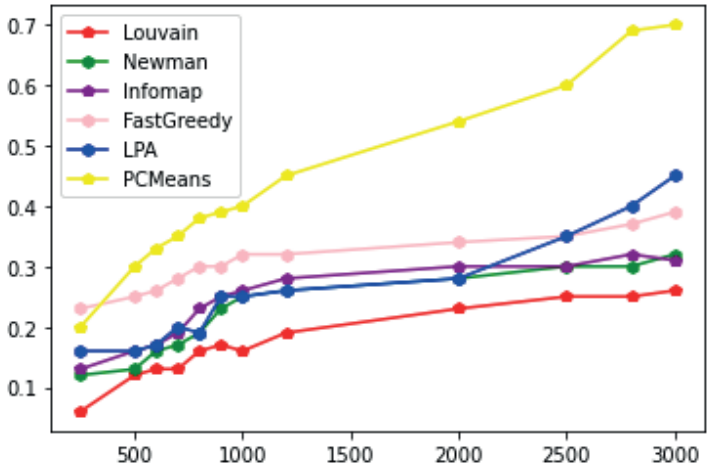
In our research, we concentrate on LFR benchmark networks that span from 250 to 3000 nodes, with an average degree falling within the range of 5 to 10. To emulate realistic scenarios, we set the degree distribution exponent (t1) to 1.5 and the community size distribution exponent (t2) to 3. Furthermore, we explore two community size intervals: small communities ranging from 10 to 50 nodes and larger communities spanning from 20 to 100 nodes. We also consider two maximum degree values, either 20 or 50. To capture network structure nuances, we introduce the mixing parameter (nu), signifying the expected fraction of links through which a node connects with others in the same community. This parameter is set to either 0.1 or 0.3.

Our evaluation process involves the assessment of various algorithms on these synthetic networks, gauging their performance using metrics such as Modularity, NMI, and ARI.

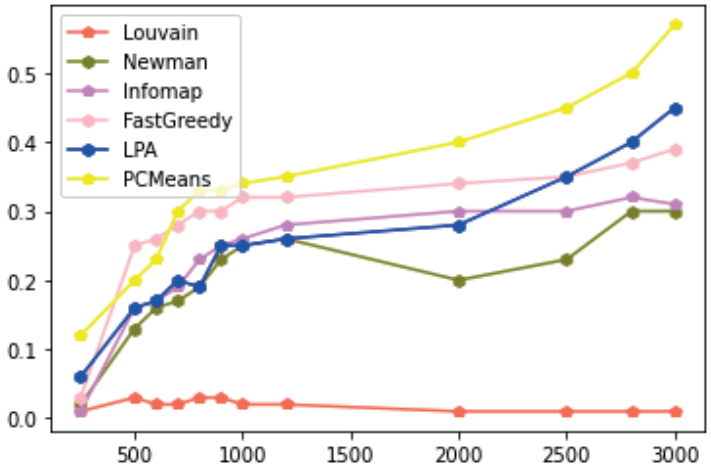
PCMeans outperformed other algorithms on LFR artificial networks, demonstrating the best NMI and ARI results and showcasing superior performance. While its modularity results were not the top-ranking, they remained satisfactory.



(a) Modularity of artificial networks.



(b) NMI of artificial networks.



(c) ARI of artificial networks.

Figure 5.8: Performance metrics on artificial networks.

5.6 Conclusion

In this research, we have introduced a novel concept termed as ”Local PageRank” by integrating it into a community detection algorithm. This algorithm, named PCMeans, effectively

combines K-means clustering with overlapping hierarchical clustering techniques. Our proposed PCMeans algorithm takes advantage of the hierarchical clustering approach to assess node similarity and significantly reduces the number of iterations required for community division, thereby enhancing overall efficiency.

By initializing the population based on the Local PageRank concept and employing the innovative neighbor-based clustering operator, PCMeans demonstrates clear superiority over conventional methods that rely on random initialization. This conclusion is strongly supported by the outcomes of our comprehensive experiments, conducted on both actual and synthetic networks. Notably, our approach yields superior values in terms of Normalized Mutual Information (NMI) and Adjusted Rand Index (ARI), which serve as reliable indicators of closely mimicking real-world community structures. These results exhibit not only effectiveness but also stability.

In the trajectory ahead, our intention is to extend the application of PCMeans to tackle the challenges posed by extremely large dynamic complex networks. This direction of exploration holds promising potential for further advancing the field of community detection in complex network analysis.

General Conclusion & Perspectives

In conclusion, this thesis delved into the intricate task of uncovering hidden communities within the complex fabric of social networks. The investigation revealed that clustering nodes into cohesive communities is an often overlooked dimension of social network dynamics. In response, a range of algorithms, including notable competitors such as Louvain and Girvan, has been proposed in academic discourse to unearth these latent structures. Two groundbreaking community detection paradigms, referred to as DBOCD and PCMeans, were introduced in this research.

These innovative models consider the importance of the most influential nodes, chosen by density and PageRank, calculated on a subgraph rather than the entire graph. A series of groupings follow, improving the quality of detected communities with each iteration. Importantly, DBOCD generates overlapping communities, while PCMeans produces disjoint communities, making machine learning methods essential for community identification.

The architecture of DBOCD and PCMeans is supported by three fundamental pillars, meticulously crafted to align with specific stages of the community detection process. DBOCD initiates by discerningly identifying pivotal nodes, leveraging the potency of local density. Subsequently, it constructs overlapping communities by associating nodes with their Breadth-First Search (BFS). In contrast, PCMeans begins with the astute identification of crucial nodes using the potency of local PageRank and constructs disjoint communities by linking nodes to their primary neighbors.

Remarkably, our approaches present a multifaceted array of advantages that distinguish them in the landscape. Their ease of implementation and capacity to produce unsupervised results facilitate the accessible detection of both disjointed and overlapping communities. Swift and scalable, they surpass the constraints of scale, adapting seamlessly to large-scale social networks and imbuing a practical dimension to their capabilities. Moreover, this research delves into diverse configurations of communities, addressing both disjoint and overlapping variations.

In an empirical application, the proposed algorithms underwent rigorous testing on authentic social networks, comparing their performance against several algorithms in the literature. The comprehensive evaluation encompassed parameters such as execution time, number of communities, stability, and the quality of detected community distributions. The study was anchored in a dual objective: an intellectual contribution through the development of more powerful al-

gorithms, and a conceptual exploration of the mathematical, graphical, and sociological foundations of communities.

In our ongoing journey of exploration, we envision exciting opportunities for growth and improvement. Our roadmap outlines plans to apply this to dynamic, weighted, and directed graphs, anticipating the transformation of our model into a recommendation system that leverages its powerful algorithms to provide valuable suggestions.

This study not only sheds light on the complexities of community detection within social networks but also proposes new methodologies, creating disjointed communities from overlapping ones through the use of machine learning methods. The potential impact of these methodologies extends beyond the current landscape, paving the way for advancements in community detection methodologies and their practical applications.

References

Aftab, H., Shuja, J., Alasmary, W., & Alanazi, E. (2021). Hybrid dbscan based community detection for edge caching in social media applications. In *2021 international wireless communications and mobile computing (iwcmc)* (pp. 2038–2043).

Ahmed, M., Seraj, R., & Islam, S. M. S. (2020). The k-means algorithm: A comprehensive survey and performance evaluation. *Electronics*, *9*(8), 1295.

Akbar, Z., Liu, J., & Latif, Z. (2021). Mining social applications network from business perspective using modularity maximization for community detection. *Social Network Analysis and Mining*, *11*, 1–19.

Albert, R., Jeong, H., & Barabási, A.-L. (1999). Diameter of the world-wide web. *nature*, *401*(6749), 130–131.

Alexanderson, G. (2006). About the cover: Euler and königsberg's bridges: A historical view. *Bulletin of the american mathematical society*, *43*(4), 567–573.

Alloghani, M., Al-Jumeily, D., Mustafina, J., Hussain, A., & Aljaaf, A. J. (2020). A systematic review on supervised and unsupervised machine learning algorithms for data science. *Supervised and unsupervised learning for data science*, 3–21.

Alotaibi, N., & Rhouma, D. (2022). A review on community structures detection in time evolving social networks. *Journal of King Saud University-Computer and Information Sciences*, *34*(8), 5646–5662.

Amin, D. B., Lawless, I. M., Sommerfeld, D., Stanley, R. M., Ding, B., & Costi, J. J. (2015). Effect of potting technique on the measurement of six degree-of-freedom viscoelastic properties of human lumbar spine segments. *Journal of biomechanical engineering*, *137*(5), 054501.

Bader, D. A., Kintali, S., Madduri, K., & Mihail, M. (2007). Approximating betweenness centrality. In *Algorithms and models for the web-graph: 5th international workshop, san diego, ca, usa, proceedings 5* (pp. 124–137).

Barto, A. G., & Sutton, R. S. (1998). Reinforcement learning: An introduction (adaptive computation and machine learning).

Bar-Yossef, Z., & Mashiach, L.-T. (2008). Local approximation of pagerank and reverse pagerank. In *Proceedings of the 17th acm conference on information and knowledge management* (pp. 279–288).

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, *2008*(10), P10008.

Bonaccorso, G. (2017). *Machine learning algorithms*. Packt Publishing Ltd.

Bothorel, C., Brisson, L., & Lyubareva, I. (2021). How to choose community detection methods in complex networks. *Series Computational Social Science*, pp–16.

Boubou, M. (2007). *Contribution aux méthodes de classification non supervisée via des approches prétopologiques et d'agrégation d'opinions* (Unpublished doctoral dissertation). Université Claude Bernard-Lyon I.

Boyd, D. M., & Ellison, N. B. (2007). Social network sites: Definition, history, and scholarship. *Journal of computer-mediated Communication*, *13*(1), 210–230.

Bramer, M. (2007). *Clustering*. Springer.

Brin, S., & Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, *30*(1-7), 107–117.

Browet, A. (2014). *Algorithms for community and role detection in networks.* (Unpublished doctoral dissertation). Catholic University of Louvain, Louvain-la-Neuve, Belgium.

Bunke, H., & Messmer, B. T. (1993). Similarity measures for structured representations. In *European workshop on case-based reasoning* (pp. 106–118).

Butler, K. T., Davies, D. W., Cartwright, H., Isayev, O., & Walsh, A. (2018). Machine learning for molecular and materials science. *Nature*, *559*(7715), 547–555.

Cai, B., Zeng, L., Wang, Y., Li, H., & Hu, Y. (2019). Community detection method based on node density, degree centrality, and k-means clustering in complex network. *Entropy*, *21*(12), 1145.

Chaudhary, L., & Singh, B. (2021). Community detection using unsupervised machine learning techniques on covid-19 dataset. *Social Network Analysis and Mining*, *11*, 1–9.

Chen, M., Kuzmin, K., & Szymanski, B. K. (2014). Extension of modularity density for overlapping community structure. In (pp. 856–863).

Chen, Z., Hendrix, W., & Samatova, N. F. (2012). Community-based anomaly detection in evolutionary networks. *Journal of Intelligent Information Systems*, *39*(1), 59–85.

Clauset, A., Newman, M. E., & Moore, C. (2004). Finding community structure in very large networks. *Physical review E*, *70*(6), 066111.

Cornfield, J. (1967). Bayes theorem. *Revue de l'Institut International de Statistique*, 34–49.

Crane, K., Livesu, M., Puppo, E., & Qin, Y. (2020). A survey of algorithms for geodesic paths and distances. *arXiv preprint arXiv:2007.10430*.

Cunningham, P., Cord, M., & Delany, S. J. (2008). Supervised learning. In *Machine learning techniques for multimedia* (pp. 21–49). Springer.

Danielsson, P.-E. (1980). Euclidean distance mapping. *Computer Graphics and image processing*, *14*(3), 227–248.

Danon, L., Díaz-Guilera, A., Duch, J., & Arenas, A. (2005). Comparing community struc-

ture identification. *Journal of Statistical Mechanics: Theory and Experiment*, *2005*(09), P09008–P09008.

Eballe, R., & Cabahug, I. (2021). Closeness centrality of some graph families. *International Journal of Contemporary Mathematical Sciences*, *16*(4), 127–134.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd* (pp. 226–231).

Euler, L. (1741). Solutio problematis ad geometriam situs pertinentis. *Commentarii academiae scientiarum Petropolitanae*, 128–140.

Euler, L. (1956). The seven bridges of königsberg. *The world of mathematics*, *1*, 573–580.

Everitt, T., & Hutter, M. (2015). Analytical results on the bfs vs. dfs algorithm selection problem: Part ii: Graph search. In (pp. 166–178).

Fatima, S., & Maurya, K. C. (2023). A review article on upgrading shortest path problems. *International Journal Of Engineering And Management Research*, *13*(3), 221–226.

Fortunato, S. (2010). Community detection in graphs. *Physics reports*, *486*(3-5), 75–174.

Froehlich, D. E., & Gegenfurtner, A. (2019). Social support in transitioning from training to the workplace: A social network perspective. *Beziehungen in pädagogischen Arbeitsfeldern und ihren Transitionen über die Lebensalter*, 208–222.

Gao, T., Pan, R., Wang, S., Yang, Y., & Zhang, Y. (2021). Community detection for statistical citation network by d-score. *Statistics and Its Interface*, *14*(3), 279–294.

Geng, J., Bhattacharya, A., & Pati, D. (2019). Probabilistic community detection with unknown number of communities. *Journal of the American Statistical Association*, *114*(526), 893–905.

Ghouchan Nezhad Noor Nia, R., Jalali, M., Mail, M., Ivanisenko, Y., & Kübel, C. (2022). Machine learning approach to community detection in a high-entropy alloy interaction network. *ACS omega*, *7*(15), 12978–12992.

Gorripati, S. K., & Vatsavayi, V. K. (2017). A community based content recommender systems. *International journal of applied engineering research*, *12*(22), 12989–12996.

Gregory, S. (2009). Finding overlapping communities using disjoint community detection algorithms. In *Complex networks: Results of the 2009 international workshop on complex networks (complenet 2009)* (pp. 47–61).

Grolmusz, V. (2015). A note on the pagerank of undirected graphs. *Information Processing Letters*, *115*(6-8), 633–634.

Gujral, E., Papalexakis, E. E., Theocharous, G., & Rao, A. (2019). Hacd: Hierarchical agglomerative community detection in social networks. In (pp. 1–6).

Gupta, N., Singh, A., & Cherifi, H. (2016). Centrality measures for networks with community structure. *Physica A: Statistical Mechanics and its Applications*, *452*, 46–59.

Hahs-Vaughn, D. L. (2023). Foundational methods: descriptive statistics: bivariate and multivariate data (correlations, associations).

Hajij, M., Said, E., & Todd, R. (2020). Pagerank and the k-means clustering algorithm. *arXiv*

preprint arXiv:2005.04774.

Hamming, R. W. (1950). Error detecting and error correcting codes. *The Bell system technical journal*, *29*(2), 147–160.

Hardy, C. (2019). *Contribution au développement de l'apprentissage profond dans les systèmes distribués* (Unpublished doctoral dissertation). Rennes 1.

Henni, K., Mezghani, N., & Gouin-Vallerand, C. (2018). Unsupervised graph-based feature selection via subspace and pagerank centrality. *Expert Systems with Applications*, *114*, 46–53.

Jaccard, P. (1901). Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bull Soc Vaudoise Sci Nat*, *37*, 547–579.

Jakkula, V. (2006). Tutorial on support vector machine (svm). *School of EECS, Washington State University*, *37*(2.5), 3.

Jiang, J. Q., & McQuay, L. J. (2012). Modularity functions maximization with non negative relaxation facilitates community detection in networks. *Physica A: Statistical Mechanics and its Applications*, *391*(3), 854–865.

Jiawei Han, J. P., Micheline Kamber. (2012). *Data mining (third edition)*.

Jin, D., Yu, Z., Jiao, P., Pan, S., He, D., Wu, J., … Zhang, W. (2021). A survey of community detection approaches: From statistical modeling to deep learning. *IEEE Transactions on Knowledge and Data Engineering*.

Jonathan, K. B., Goldstein, J., Ramakrishnan, R., & Shaft, U. (1999). When is" nearest neighbor" meaningful? In *In int. conf. on database theory.*

Kadavankandy, A., Avrachenkov, K., Prokhorenkova, L. O., & Raigorodskii, A. M. (2015). Pagerank in undirected random graphs. *CoRR, abs/1511.04925*.

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of artificial intelligence research*, *4*, 237–285.

Kempf-Leonard, K. (2004). Encyclopedia of social measurement.

Khatir, N., & Nait-bahloul, S. (2018). Multi-criteria-based fusion for clustering texts and images case study on flickr. *Kybernetes*.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological cybernetics*, *43*(1), 59–69.

Krebs, V. (2008). A network of co-purchased books about us politics. *October*, *20*(1), 0–03.

Kumar, A., Barman, D., Sarkar, R., & Chowdhury, N. (2020). Overlapping community detection using multiobjective genetic algorithm. *IEEE Transactions on Computational Social Systems*, *7*(3), 802–817.

Lancichinetti, A., Fortunato, S., & Radicchi, F. (2008). Benchmark graphs for testing community detection algorithms. *Physical review E*, *78*(4), 046110.

Learned-Miller, E. G. (2014). Introduction to supervised learning. *I: Department of Computer Science, University of Massachusetts*, 3.

Li, H., Zhang, R., Zhao, Z., & Liu, X. (2021). Lpa-mni: an improved label propagation algo-

rithm based on modularity and node importance for community detection. *Entropy*, *23*(5), 497.

Li, L., Fan, K., Zhangy, Z., & Xia, Z. (2016). Community detection algorithm based on local expansion k-means. *Neural network world*, *26*(6), 589.

Liu, B. (2011). Supervised learning. In *Web data mining* (pp. 63–132). Springer.

Liu, B., Zhou, Q., Ding, R.-X., Palomares, I., & Herrera, F. (2019). Large-scale group decision making model based on social network analysis: Trust relationship-based conflict detection and elimination. *European Journal of Operational Research*, *275*(2), 737–754.

Liu, X., Fu, L., Wang, X., & Zhou, C. (2022). On the similarity between von neumann graph entropy and structural information: Interpretation, computation, and applications. *IEEE Transactions on Information Theory*.

Louafi, W., & Titouna, F. (2019). Community detection in social networks. In *The 2nd international conference on informatics and mathematics (iam '2019)*.

Louafi, W., & Titouna, F. (2023a). Density-based overlapping community detection. In *The 1st national conference on new educational technologies and informatics ncneti'23*.

Louafi, W., & Titouna, F. (2023b). Pcmeans: Community detection using localpagerank, clustering, and k-means. *Social Network Analysis and Mining*, *13*(103). doi: 10.1007/s13278-023-01109-5

Luo, W., Lu, N., Ni, L., Zhu, W., & Ding, W. (2020). Local community detection by the nearest nodes with greater centrality. *Information Sciences*, *517*, 377–392.

Lusseau, D., Schneider, K., Boisseau, O. J., Haase, P., Slooten, E., & Dawson, S. M. (2003). The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, *54*(4), 396–405.

Ma, T., Liu, Q., Cao, J., Tian, Y., Al-Dhelaan, A., & Al-Rodhaan, M. (2020). Lgiem: Global and local node influence based community detection. *Future Generation Computer Systems*, *105*, 533–546.

Ma, T., Wang, Y., Tang, M., Cao, J., Tian, Y., Al-Dhelaan, A., & Al-Rodhaan, M. (2016). Led: A fast overlapping communities detection algorithm based on structural clustering. *Neurocomputing*, *207*, 488–500.

MacQueen, J., et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth berkeley symposium on mathematical statistics and probability* (pp. 281–297).

Madhulatha, T. S. (2012). An overview on clustering methods. *arXiv preprint arXiv:1205.1117*.

Mahdavinejad, M. S., Rezvan, M., Barekatain, M., Adibi, P., Barnaghi, P., & Sheth, A. P. (2018). Machine learning for internet of things data analysis: A survey. *Digital Communications and Networks*, *4*(3), 161–175.

Majeed, A., & Rauf, I. (2020). Graph theory: A comprehensive survey about graph theory applications in computer science and social networks. *Inventions*, *5*(1), 10.

Maulud, D., & Abdulazeez, A. M. (2020). A review on linear regression comprehensive in

machine learning. *Journal of Applied Science and Technology Trends*, *1*(4), 140–147.

Mawloud, M. (2017). *Mesures de distances dans le contexte de la recherche d'images par le contenu (cbir)* (Unpublished doctoral dissertation). Université du 20 Août 1955.

Moayedikia, A. (2018). Multi-objective community detection algorithm with node importance analysis in attributed networks. *Applied Soft Computing*, *67*, 434–451.

Moosavi, S. A., Jalali, M., Misaghian, N., Shamshirband, S., & Anisi, M. H. (2017). Community detection in social networks using user frequent pattern mining. *Knowledge and Information Systems*, *51*(1), 159–186.

Murtagh, F., & Contreras, P. (2012). Algorithms for hierarchical clustering: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *2*(1), 86–97.

Musdar, I. A., & Azhari, S. (2015). Metode rce-kmeans untuk clustering data. *IJCCS (Indonesian Journal of Computing and Cybernetics Systems)*, *9*(2), 157–166.

Nasteski, V. (2017). An overview of the supervised machine learning methods. *Horizons. b*, *4*, 51–62.

Newman, M. E. (2004). Fast algorithm for detecting community structure in networks. *Physical review E*, *69*(6), 066133.

Newman, M. E., & Girvan, M. (2004). Finding and evaluating community structure in networks. *Physical review E*, *69*(2), 026113.

Newman, M. E., Strogatz, S. H., & Watts, D. J. (2001). Random graphs with arbitrary degree distributions and their applications. *Physical review E*, *64*(2), 026118.

Noble, W. S. (2006). What is a support vector machine? *Nature biotechnology*, *24*(12), 1565–1567.

Orman, G. K., & Labatut, V. (2009). A comparison of community detection algorithms on artificial networks. In *International conference on discovery science* (pp. 242–256).

Palla, G., Derényi, I., Farkas, I., & Vicsek, T. (2005). Uncovering the overlapping community structure of complex networks in nature and society. *nature*, *435*(7043), 814–818.

Parés, F., Gasulla, D. G., Vilalta, A., Moreno, J., Ayguadé, E., Labarta, J., … Suzumura, T. (2018). Fluid communities: A competitive, scalable and diverse community detection algorithm. In (pp. 229–240).

Philip, S. Y., Han, J., & Faloutsos, C. (2010). *Link mining: Models, algorithms, and applications*. Springer.

Pons, P., & Latapy, M. (2005). Computing communities in large networks using random walks. In *Computer and information sciences-iscis 2005: 20th international symposium, istanbul, turkey, october 26-28, 2005. proceedings 20* (pp. 284–293).

Prasetya, D. D., Wibawa, A. P., & Hirashima, T. (2018). The performance of text similarity algorithms. *International Journal of Advances in Intelligent Informatics*, *4*(1), 63–69.

Raghavan, U. N., Albert, R., & Kumara, S. (2007). Near linear time algorithm to detect community structures in large-scale networks. *Physical review E*, *76*(3), 036106.

Rios, S. A., Aguilera, F., Nuñez-Gonzalez, J. D., & Graña, M. (2019). Semantically enhanced

network analysis for influencer identification in online social networks. *Neurocomputing*, *326*, 71–81.

Rokach, L., & Maimon, O. (2005). Decision trees. *Data mining and knowledge discovery handbook*, 165–192.

Rosvall, M., & Bergstrom, C. T. (2008). Maps of random walks on complex networks reveal community structure. *Proceedings of the national academy of sciences*, *105*(4), 1118–1123.

Roudiere, G. (2018). *Détection d'attaques sur les équipements d'accès à internet* (Unpublished doctoral dissertation). INSA de Toulouse.

Russell, S. (2016). *Artificial intelligence: A modern approach, ebook, global edition*. Pearson Education, Limited.

Russell, S. J., & Norvig, P. (2010). *Artificial intelligence a modern approach*.

Sah, P., Singh, L. O., Clauset, A., & Bansal, S. (2014). Exploring community structure in biological networks with random graphs. *BMC bioinformatics*, *15*(1), 1–14.

Samba, A. (2018). *Science des données au service des réseaux d'opérateur: proposition de cas d'utilisation, d'outils et de moyens de déploiement* (Unpublished doctoral dissertation). Ecole nationale supérieure Mines-Télécom Atlantique.

Samet, H. (2007). K-nearest neighbor finding using maxnearestdist. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *30*(2), 243–252.

Sarma, S. V. M. (2012). Applications of graph theory in human life. *International Journal of Computer Application*, *1*(2), 21–30.

Scicluna, N., & Bouganis, C.-S. (2015). Arc 2014: a multidimensional fpga-based parallel dbscan architecture. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, *9*(1), 1–15.

Sheng, J., Liu, C., Chen, L., Wang, B., & Zhang, J. (2020). Research on community detection in complex networks based on internode attraction. *Entropy*, *22*(12), 1383.

Shi, C., Yan, Z., Cai, Y., & Wu, B. (2012). Multi-objective community detection in complex networks. *Applied Soft Computing*, *12*(2), 850–859.

Shi, C., Yu, P. S., Cai, Y., Yan, Z., & Wu, B. (2011). On selection of objective functions in multi-objective community detection. In *Proceedings of the 20th acm international conference on information and knowledge management* (pp. 2301–2304).

Shu-Xi, W. (2012). The improved dijkstra's shortest path algorithm and its application. *Procedia Engineering*, *29*, 1186–1190.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine learning*, *3*(1), 9–44.

Syarif, I., Prugel-Bennett, A., & Wills, G. (2012). Unsupervised clustering approach for network anomaly detection. In *International conference on networked digital technologies* (pp. 135–145).

Tackx, R. (2018). *Analyse de la structure communautaire des réseaux bipartis* (Unpublished

doctoral dissertation). Sorbonne université.

Teletin, M., Czibula, G., Albert, S., & Bocicor, M.-I. (2018). Using unsupervised learning methods for enhancing protein structure insight. *Procedia Computer Science*, *126*, 19–28.

Toch, E., Lerner, B., Ben-Zion, E., & Ben-Gal, I. (2019). Analyzing large-scale human mobility data: a survey of machine learning methods and applications. *Knowledge and Information Systems*, *58*, 501–523.

Tutte, W. T. (2001). *Graph theory* (Vol. 21). Cambridge university press.

Ubaldi, E., Burioni, R., Loreto, V., & Tria, F. (2021). Emergence and evolution of social networks through exploration of the adjacent possible space. *Communications Physics*, *4*(1), 28.

Uddin, S., Khan, A., Hossain, M. E., & Moni, M. A. (2019). Comparing different supervised machine learning algorithms for disease prediction. *BMC medical informatics and decision making*, *19*(1), 1–16.

Utz, S., & Breuer, J. (2019). The relationship between networking, linkedin use, and retrieving informational benefits. *Cyberpsychology, Behavior, and Social Networking*, *22*(3), 180–185.

Vandaele, R., Saeys, Y., & De Bie, T. (2021). Graph approximations to geodesics on metric graphs. In (pp. 7328–7334).

Van den Heuvel, M. P., & Sporns, O. (2013). Network hubs in the human brain. *Trends in cognitive sciences*, *17*(12), 683–696.

Van Laarhoven, T., & Marchiori, E. (2016). Local network community detection with continuous optimization of conductance and weighted kernel k-means. *The Journal of Machine Learning Research*, *17*(1), 5148–5175.

Vilcek, A. (2014). Deep learning with k-means applied to community detection in networks. *CS224W Project Report*.

Wallis, W. D. (2007). *A beginner's guide to graph theory*. Springer Science & Business Media.

Watkins, C. J., & Dayan, P. (1992). Q-learning. *Machine learning*, *8*(3), 279–292.

West, D. B., et al. (2001). *Introduction to graph theory* (Vol. 2). Prentice hall Upper Saddle River.

Wicker, S., & Kim, S. (2002). *Codes, graphs, and iterative decoding*. Boston: Kluwer, to appear.

Wills, P., & Meyer, F. G. (2020). Metrics for graph comparison: a practitioner's guide. *Plos one*, *15*(2), e0228728.

Wu, Z., Wang, X., Fang, W., Liu, L., Tang, S., Zheng, H., & Zheng, Z. (2021). Community detection based on first passage probabilities. *Physics Letters A*, *390*, 127099.

Xu, M., Watanachaturaporn, P., Varshney, P. K., & Arora, M. K. (2005). Decision tree regression for soft classification of remote sensing data. *Remote Sensing of Environment*, *97*(3), 322–336.

Yahi, A. (2019). *Clustering des données de puces à adn* (Unpublished doctoral dissertation). University Mohamed BOUDIAF-Msila.

Yuan, Y., Chen, B., Yu, Y., & Jin, Y. (2020). An influence maximisation algorithm based on community detection. *International Journal of Computational Science and Engineering*, *22*(1), 1–14.

Zachary, W. W. (1977). An information flow model for conflict and fission in small groups. *Journal of anthropological research*, *33*(4), 452–473.

Žalik, K. R. (2008). An efficient k′-means clustering algorithm. *Pattern Recognition Letters*, *29*(9), 1385–1391.

Zhang, J., & Luo, Y. (2017). Degree centrality, betweenness centrality, and closeness centrality in social network. In *Proceedings of the 2017 2nd international conference on modelling, simulation and applied mathematics (msam2017)* (Vol. 132, pp. 300–303).

Zhao, Y. (2023). *Graph theory and additive combinatorics: Exploring structure and randomness*. Cambridge University Press.

Zhou, L., Wang, T., Qu, H., Huang, L., & Liu, Y. (2020). A weighted gcn with logical adjacency matrix for relation extraction. In *Ecai 2020* (pp. 2314–2321). IOS Press.

Zhou, X., Su, L., Li, X., Zhao, Z., & Li, C. (2023). Community detection based on unsupervised attributed network embedding. *Expert Systems with Applications*, *213*, 118937.