Ministry of Higher Education and Scientific Research
Hassiba Benbouali University of Chlef
Faculty of Exact Sciences and Informatics
Informatics Department

THESIS

Presented for the award of the degree of

DOCTORAT

Field: Informatics

Specialty: Informatics

By
KARIMA MOULEY

Subject :

PARTITIONING AND BIG GRAPH ANALYSIS

Supported on 20/04/2025, before the jury composed of:

Mourad LOUKAM	MCA	University of Chlef	President
Sarah IBRI	MCA	University of Chlef	Examiner
Nassim DENNOUNI	MCA	Higher School of Management-Tlemcen	Examiner
Khadidja YACHBA	MCA	Graduate School of Economics-Oran	Examiner
Mohammed Amin TAHRAOUI	MCA	University of Chlef	Director
Abdelaziz KELLA	MCB	University of Chlef	Co-Director

Abstract

The rapid growth of large graphs has an important role in analysing and managing their structures. The capacity to effectively analyse these graphs is crucial for many applications, including social network analysis, communication networks, and sensor networks.

This thesis focuses on developing a graph partitioning based on Critical Nodes Problem (CNP) to solve several issues, including detecting communities in social networks, accurately locating the source of information in social networks, and evaluating the impact of CNP in Wireless Sensor Networks (WSNs).

The first part of this thesis describes the development of a scalable graph partitioning method that uses critical nodes to optimize vertex removal, therefore lowering network complexity while keeping critical structural characteristics and connectivity. The second part seeks to tackle the problem of community detection in social networks. We offer a unique technique that uses critical nodes to improve the quality of discovered communities. The third part emphasizes solving the problem of source detection with limited observation nodes in social networks. We propose a novel algorithm that employs critical nodes as observation nodes, it significantly improves source detection accuracy. The final part of this thesis explores the impact of critical nodes in WSNs, demonstrating improved efficiency, optimized performance, reduced energy consumption, and enhanced data transmission reliability.

We validate the proposed approaches, according to experimental results on real-world and synthetic datasets. The results show that the algorithms outperform existing methods in terms of various performance measures, including modularity and Normal Mutual Information (NMI), accuracy, and energy consumption for WSNs.

Keywords: *Graph Partitioning, Critical Nodes Detection Problem (CNDP), Community Detection, Source Detection, Wireless Sensor Networks (WSNs).*

Résumé

La croissance rapide des grands graphes joue un rôle important dans l'analyse et la gestion de leurs structures. La capacité à analyser efficacement ces graphes est cruciale pour de nombreuses applications, y compris l'analyse des réseaux sociaux, les réseaux de communication, et les réseaux de capteurs. Cette thèse se concentre sur le développement d'une partition de graphe basée sur le Problème des Nœuds Critiques (CNP) pour résoudre plusieurs problèmes, y compris la détection de communautés dans les réseaux sociaux, la localisation précise de la source d'information dans les réseaux sociaux et l'évaluation de l'impact du CNP dans les Réseaux de Capteurs Sans Fil (WSNs). La première partie de cette thèse décrit le développement d'une méthode de partitionnement de graphes évolutive qui utilise des nœuds critiques pour optimiser la suppression de sommets, réduisant ainsi la complexité du réseau tout en conservant les caractéristiques structurelles critiques et la connectivité. La deuxième partie cherche à aborder le problème de la détection de communautés dans les réseaux sociaux. Nous proposons une technique unique qui utilise des nœuds critiques pour améliorer la qualité des communautés découvertes. La troisième partie met l'accent sur la résolution du problème de détection de sources avec un nombre limité de nœuds d'observation dans les réseaux sociaux. Nous proposons un nouvel algorithme qui utilise des nœuds critiques comme nœuds d'observation, ce qui améliore considérablement la précision de la détection de source. La dernière partie de cette thèse explore l'impact des nœuds critiques dans les WSNs, démontrant une efficacité améliorée, des performances optimisées, une réduction de la consommation d'énergie et une fiabilité accrue de la transmission des données.

Nous validons les approches proposées, selon les résultats expérimentaux sur des ensembles de données réelles et synthétiques. Les résultats montrent que les algorithmes surpassent les méthodes existantes en termes de diverses mesures de performance, y compris la modularité et l'information mutuelle normale (NMI), la précision et la consommation d'énergie pour les WSNs.

Mots-clés : *Partitionnement de graphes, Problème de détection des nœuds critiques (CNDP), Détection de communautés, Détection de sources, Réseaux de capteurs sans fil (WSNs).*

ملخص

النمو السريع للرسوم البيانية الكبيرة يلعب دوراً مهماًا في تحليل وإدارة هياكلها. القدرة على تحليل هذه الرسوم البيانية بفعالية أمرتركز هذه الأطروحة على تطوير تقسيم الرسم البياني بناءً على مشكلة العقد الحرجة لحل عدة قضايا، بما في ذلك اكتشاف المجتمعات في الشبكات الاجتماعية، وتحديد موقع مصدر المعلومات بدقة في الشبكات الاجتماعية، وتقييم تأثير مشكلة العقد الحرجة في الشبكات الاستشعارية اللاسلكية.

الجزء الأول من هذه الأطروحة يصف تطوير طريقة تقسيم الرسوم البيانية القابلة للتوسع التي تستخدم العقد الحرجة لتحسين إزالة الرؤوس، وبالتالي تقليل تعقيد الشبكة مع الحفاظ على الخصائص الهيكلية الحرجة والاتصال.الجزء الثاني يسعى إلى معالجة مشكلة اكتشاف المجتمعات في الشبكات الاجتماعية.نقدم تقنية فريدة تستخدم العقد الحرجة لتحسين جودة المجتمعات المكتشفة.الجزء الثالث يركز على حل مشكلة اكتشاف المصدر باستخدام عقد المراقبة المحدودة في الشبكات الاجتماعية.نقترح خوارزمية جديدة تستخدم العقد الحرجة كعقد مراقبة، مما يحسن بشكل كبير دقة اكتشاف المصدر.الجزء الأخير من هذه الأطروحة يستكشف تأثير العقد الحرجة في شبكات الاستشعار اللاسلكية، موضحاًا تحسين الكفاءة، الأداء المحسن، تقليل استهلاك الطاقة، وزيادة موثوقية نقل البيانات.

نحن نتحقق من صحة الأساليب المقترحة، وفقاًا للنتائج التجريبية على مجموعات البيانات الحقيقية والصناعية.تظهر النتائج أن الخوارزميات تتفوق على الطرق الحالية من حيث مقاييس الأداء المختلفة، بما في ذلك التماسك والمعلومات المتبادلة الطبيعية، والدقة، واستهلاك الطاقة لشبكات الاستشعار اللاسلكية.

الكلمات المفتاحية: تقسيم الرسوم البيانية، مشكلة اكتشاف العقد الحرجة، كشف المجتمعات، كشف المصادر، شبكات الاستشعار اللاسلكية.

Acknowledgements

I would like to thank everyone who has helped and supported me in completing this thesis.

I would also like to thank my thesis supervisor, Mr. Mohamed Amin TAHRAOUI, for his great help, persistent support, and essential insights during my research. Their knowledge and support have helped structure this work, and I am really thankful for their patience and effort in guiding me.

I would also like to thank every member of my thesis committee for their helpful remarks, smart recommendations, and help in refining my study. Their experience and intelligent perspectives considerably improved the quality of this thesis.

My profound gratitude extends to my family: my mother, father, sister, brother, and my husband for their constant love, support, and understanding.

Contents

List of Figures

List of Tables

List of Algorithms

List of Equations

List of Abbreviations

BFS Breadth First Search. 60

CC-CNP Connected Components Critical Nodes Problem. 39

CCC-CNP Component Cardinality Constraint Critical Nodes Problem. 40, 41

CNCDA Critical Nodes Community Detection Approach. x, xi, 85, 92–94, 96–98

CNDP Critical Nodes Detection Problem. vii, 1, 4, 34, 35, 66, 108

CNEO Critical Node-Based Energy Optimization. xii, 110–114

CNP Critical Nodes Problem. vi, x, 21, 32, 34–36, 40, 42, 43, 63, 75, 80, 101, 108, 116–118

CNSD Critical Node-based Source Detection. x, 99, 100, 104

CPM Clique Percolation Method. 50

CPU Central Processing Unit. 68, 77, 107

DBSCAN Density-Based Spatial Clustering of Applications with Noise. 51

DFS Depth First Search. 36

FPGAs Field-Programmable Gate Arrays. 68

GNNs Graph Neural Networks. 50

GP Graph Partitioning. 30

GPS Global Positioning System. 12

HDI Hub-Depressed Index. 19

HPI Hub-Promoted Index. 19

ICM Independent Cascade Model. 61

ILP Integer Linear Programming. 61

IoTs Internet Of Things. 68, 115

IWSNs Industrial Wireless Sensor Networks. viii, 2, 3, 5, 77, 78, 107, 108, 112–115

LPA Label Propagation Algorithms. 44, 50, 52, 91, 93, 94

LTM Linear Threshold Model. 61

MaxNum Maximize the number of connected components. ix, 37

MCDS Minimum Connected Dominating Set. 77

MCMC Markov Chain Monte Carlo. 61

MinMaxC Minimize the size of the largest connected component. 37–39

MIS Maximal Independent Set. 37, 81, 108, 109, 115

MLE Maximum Likelihood Estimation. 61

MST Minimum Spanning Tree. 71

MVC Minimum Vertex Cover. 41

NMI Normal Mutual Information. x, xi, 2, 3, 49, 92–94, 97, 98

OSes Operating Systems. 68

PPINs Protein-Protein Interaction Networks. 1, 19, 24, 47, 63

QoS Quality of Service. 68–70

SI Susceptible-Infected. viii, 56, 102

SIR Susceptible-Infected-Recovered. 57, 61, 65

SIS Susceptible-Infected-Susceptible. 64

TPSN Timing-sync Protocol for Sensor Networks. 68

TSP Travelling Salesman Problem. 117

VLSI Very Large Scale Integration. 25, 28

VS Vertex Separator. 41

WSNs Wireless Sensor Networks. 2, 4, 5, 19, 24, 45, 66–70, 72–78, 83, 107, 113, 115, 117, 118

Introduction

Context and Motivation

Graphs are important tools for modelling and illustrating a variety of complex system topologies, including social and communications networks. These systems present interactions among their components and dependencies between their elements. Graph theory is a branch of mathematics that examines, analyses, and optimizes systems that may be represented as graphs. Graphs consist of nodes (or vertices), and the relations between nodes are edges (or links).

Graph partitioning is one of the methods that belongs to graph theory, which aims to divide a graph into smaller and more manageable partitions, it ensures that the original graph's structural features are retained. This is an important approach, it has an essential role in the optimization operations of numerous networks, such as communication systems and parallel algorithms.

As a known method of graph partitioning named the Critical Nodes Detection Problem (CNDP), which aims to search for important nodes in a network that have an impact of influence on the connectivity and structure of the network. Critical nodes are the core of this work, they have major importance in solving many problems that are complex in many domains, including cybersecurity. These nodes can detect susceptible nodes that can be attacked to ensure a more effective method of defence against threats. Similarly, critical nodes can also be used in communication networks, especially in optimizing problems, like enhancing routing algorithms and load balancing. The identification of these critical nodes can also be used in transportation networks, as hubs, for example, in order to manage the traffic flow. Furthermore, these nodes are also used in biological networks, such as in the known networks of Protein-Protein Interaction Networks (PPINs). These nodes can act as important proteins to solve disease problems and can improve drug research.

Problematic of the Subject

The identification of critical nodes whose presence or absence significantly influences the structure and behaviour of networks is a fundamental challenge. Specifically, it has been proven to be NP-complete [1], which means that no polynomial-time algorithm is currently known to provide an exact solution for this problem in large graphs.

The application of critical node detection in real world networks presents additional challenges:

- In community detection, detecting important nodes that separate communities is important. Most existing methods result in low modularity partitions.
- In source detection, for locating the origin of information diffusion, traditional approaches may not be feasible techniques, especially in observed networks.
- Finally, in WSNs, maintaining energy efficiency and connectivity is critical. The removal of specific nodes can degrade network performance, and there is a need for approaches that reduce energy usage while preserving connectivity.

Objectives

The main objectives of this thesis are:

- Develop an efficient heuristic for identifying critical nodes that significantly influence the structure and behaviour of large and complex networks.
- Apply these identified critical nodes to enhance the performance of three major tasks in complex network analysis:
    1. Community detection in social networks, by identifying meaningful group structures with good quality.
    2. Source detection in social networks, by ensuring more accurate and efficient tracing of information diffusion origin.
    3. Energy efficiency and connectivity in Industrial Wireless Sensor Networks (IWSNs), by optimizing communication paths for prolonged network operation.
    4. Design and implement simulation frameworks for evaluating the effectiveness of the proposed approaches using real datasets.
    5. Perform comparative analysis with existing techniques to validate improvements in performance metrics, including modularity, NMI, accuracy, and energy consumption.

Contributions

- An efficient heuristic-based approach for identifying critical nodes that significantly influence the structure and behaviour of complex networks.
- A novel community detection approach that uses critical nodes to improve partition quality, increase modularity, and better reflect real-world community structures.
- A novel source detection approach that employs critical nodes as observation nodes, enhancing accuracy in identifying the origin of information spread.
- A simulation framework for IWSNs that shows how critical nodes can reduce energy consumption while maintaining network connectivity.
- Comprehensive evaluation across multiple real benchmarks, showing competitive performance in terms of modularity, NMI, accuracy, and energy efficiency.

This thesis highlights the importance of critical nodes and how their identification can help solve many problems in graph theory and its applications. We will provide detailed analysis and propose novel algorithms for leveraging critical nodes in various domains that aim to optimize the efficiency, reliability, and scalability of complex networks. Additionally, a comprehensive evaluation on multiple benchmarks with competitive results in many metrics, including modularity, NMI, accuracy, and energy metric.

Scope of the Thesis

This thesis focuses only on undirected, unweighted, and static graphs, which represent many real-world networks. This study deals only with the use of critical nodes as a graph partitioning method, supported by their significant impact on connectivity and structure. By identifying and removing a minimal set of nodes, the graph can be partitioned into smaller, more manageable groups. This approach is suited for analysing large-scale networks where traditional partitioning methods may struggle with scalability. Additionally, incorporating node or edge weights can enhance the detection of community structures by adding essential information.

In the context of source detection, this study employs a probabilistic approach based on a Gaussian estimation model to identify the origin of information spread. Consider critical nodes as observation nodes to collect important information from the network. These observations are then used to estimate the most likely source using probabilistic inference. The analysis assumes that the network is undirected, unweighted, and static,

and the spreading process follows a Gaussian model, while more complex dynamics or time-varying topologies are not addressed.

This thesis also includes the application of critical nodes in WSNs, focusing specifically on their impact on energy efficiency and network connectivity. The study considers WSNs as undirected, unweighted, and static graphs, without considering real-time dynamics or packet-level routing protocols. While many existing approaches integrate detailed MAC/routing layers or use probabilistic models for energy consumption, this work simplifies the setting to emphasize topological partitioning using critical nodes. The sensors are assumed to have uniform energy at initialization, and communication energy is modelled based on fixed transmission and reception costs. The goal is to evaluate how removing a small set of nodes can affect the connectivity of the network and to have better energy.

Organization of the Thesis

This thesis is organized into ten main chapters; each one addresses an important part of the research.

- **Chapter 1: Preliminary in Graph Theory**

  This chapter represents the basic concepts of graph theory, definition, and basic properties of graphs, also discussing some families of graphs. It also provides essential metrics used in graphs. This chapter offers a wide range of applications in many domains, alongside the challenges that researchers encounter within graph theory.

- **Chapter 2: Graph Partitioning Problem**

  This chapter defines the problem of graph partitioning, gives a formal definition of the problem, provides the different applications, and gives an overview of existing literature on the subject. The chapter also presents various algorithms and methods for solving this issue and key challenges involved in the graph partitioning problem.

- **Part 1: State of the Art**

  The first part of this thesis contains the following chapters:

  1. **Chapter 3: Critical Nodes Detection Problem (CNDP)**

     This chapter introduces the CNDP, and it explains different variants with examples. This chapter also delves into the wide array of applications, it also examines various methods for identifying critical nodes within a network.

  2. **Chapter 4: Community Detection**

This chapter addresses the community detection problem. It provides a definition, characteristics, and applications of community detection. It also examines various algorithms for community detection.

3. **Chapter 5: Source Detection**

   This chapter addresses the problem of source detection. It provides a definition and required conditions to locate the source of information in a network. The chapter gives the challenges associated with source detection and its applications in many domains.

4. **Chapter 6: Impact of CNDP on Wireless Sensor Networks (WSNs)**

   This chapter defines WSNs, gives applications, and discusses key challenges in detail. This chapter sets the stage for employing graph theory in WSNs to improve performance, and evaluating the impact of using critical nodes to enhance network performance.

- **Part 2: Contributions**

  The second part of this thesis contains the following chapters:

1. **Chapter 7: Heuristic for Critical Nodes Detection Problem (CNDP)**

   This chapter discusses the role of critical nodes in maintaining network connectivity and facilitating efficient partitioning. We present our proposed heuristic for identifying critical nodes in detail. We explain the importance of these nodes in solving many problems in complex networks.

2. **Chapter 8: Community Detection using Critical Nodes**

   This chapter introduces our proposed approach called Critical Nodes Community Detection Algorithm (CNCDA). It lays the stage for using critical nodes for solving the community detection problem in social networks.

3. **Chapter 9: Source Detection using Critical Nodes**

   In this chapter, we present our proposed approach named Critical Node-based Source Detection (CNSD), which lays the stage for using critical nodes for solving source detection problems in social networks. The critical nodes are considered as observation nodes for more efficient source detection.

4. **Chapter 10: Energy Optimization in IWSNs using Critical Nodes**

   In this chapter, we present our proposed approach named Critical Node Energy Optimization in IWSNs (CNEO), which enhances network performance in IWSNs, including connectivity and energy efficiency .

- **General Conclusion**

At the end, the thesis concludes with a general conclusion of the key findings, a discussion of the main contributions of the research, and suggestions for potential future work directions in the area of graph partitioning in general.

Publications

- K. Mouley and M. A. Tahraoui, *"Locating the source of information in social networks using critical nodes", Engineering, Technology amp; Applied Science Research* 15 (Feb., 2025) 19136–19142.
- K. Mouley and M. A. Tahraoui, *"Locating information source in social networks based on betweenness centrality", in First National Conference on Artificial Intelligence Smart Technologies and Communications,* (Algeria, Chlef), November, 2023
- K. Mouley, M. A. Tahraoui, and A. kella, *"energy-efficiency using critical nodes detection problem in industrial wireless sensor networks (iwsns)", Revue Nature et Technologie* (2025).

Chapter 1

Preliminary in Graph Theory

1.1 Introduction

Graph theory is a fundamental branch of mathematics that provides tools for presenting, evaluating, and solving network and relationship problems. It is a powerful tool and flexible discipline with many applications, including computer science, biology, and science. In general, it models systems and relationships to understand the overall structure and behaviours of the system.

A graph is a mathematical representation of a collection of elements, called vertices or nodes, connected by relationships, called edges or links. Graphs serve as the foundation for evaluating network structures and are useful for comprehending real-world phenomena such as social network interactions and communication interaction systems. In graph theory, we investigate the features, structure, and behaviour of these graphs in order to uncover patterns, optimize perfectness, and solve specific issues in many applications.

In this chapter, we will give a definition of graph theory, its fundamental concepts, and notations. It covers the types of graphs, as well as basic properties of graphs. The chapter also discusses some parameters and families of graphs. It also contains several fundamental graph metrics. We explore the different applications and challenges of graph theory. We presume that the reader is familiar with fundamental topics in graph theory and theoretical computer science. We shall provide an outline of the terms utilized in this thesis. For further background information, see [2].

1.2 Graph Definition

A graph denoted by $G = (V, E)$ consists of two finite sets, the first set named vertices denoted by V, the second set named edges denoted by E. $|V|$ is the number of all vertices in graph G, and $|E|$ is the number of all edges in graph G.

The Figure 1.1 shows a graph that has five vertices ($|V|$) and seven edges ($|E|$).

The vertices V are $(1, 2, 3, 4, 5)$, and the edges E are $\{(1, 2), (1, 3), (2, 3), (2, 5), (3, 4), (3, 5), (4, 5)\}$.

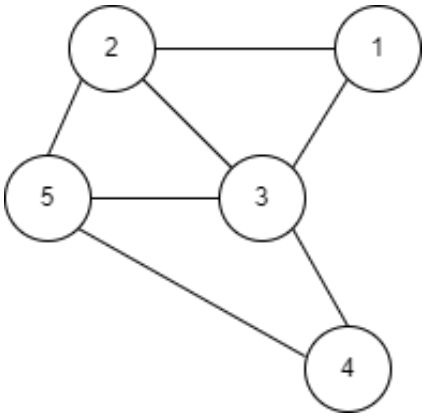


Figure 1.1: Graph

In graph theory, there are two principal types of graphs: directed and undirected graphs. In the following, we define and explore both types of graphs:

1. **Directed graph**: also called a *digraph*, it is a graph in which the edges have a direction (See Figure 1.2) from starting vertex u to destination another vertex v, denoted by (u,v). For example, the Figure 1.2 shows a directed graph, the edges are signed by arrows, which represent flows between vertices.

2. **Undirected graph**: the edges have no direction (See Figure 1.1), also denoted by (u,v), where the relation between vertices is bidirectional.

In this thesis, we work only on undirected graphs. The reason is that many real-world networks, including social networks and wireless sensor networks, contain bidirectional edges. It models interactions more precisely, which is crucial for solving many issues.

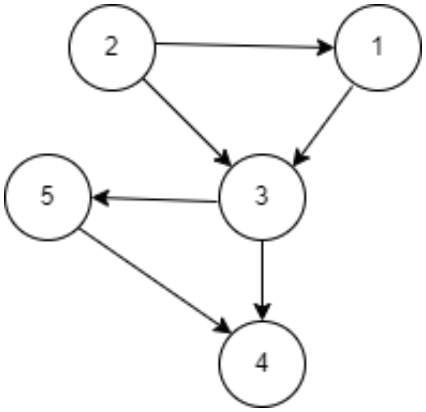


Figure 1.2: Directed Graph

1.3 Basic Properties of Graphs

There are basic properties in graph theory that describe their structure and behaviours. In this section, we discuss some basic properties of graphs.

1.3.1 Neighbours

Two vertices u and v are neighbours if there is an edge (u,v) connecting them, we can say that u and v are adjacent to each other (See Equation 1.3.1):

$$N(v) = u \in V : (u, v) \in E \tag{1.3.1}$$

$N(v)$ denotes the set of all neighbours of a vertex v in a graph G. This set includes all vertices that share an edge with vertex v. Neighbours are generally used in algorithms for searching and traversal.

1.3.2 Degree

Degree indicates the number of all neighbours of a vertex v, denoted by $d(v)$. Formally, in an undirected graph as in the following Equation 1.3.2:

$$d(v) = |(u, v) \in E : u \in V| \tag{1.3.2}$$

If $d(v) = 0$, that means that v is an isolated vertex, and it is not adjacent to any other vertex, and if $d(v) = 1$, it indicates a pendant vertex, related to only one vertex.

The lowest degree of a graph G is formulated as Equation 1.3.3:

$$\delta(G) = min\{d(v) : v \in V\} \tag{1.3.3}$$

The greatest degree of a graph G is formulated as Equation 1.3.4:

$$\Delta(G) = max\{d(v) : v \in V\} \tag{1.3.4}$$

1.3.3 Path

A sequence of vertices $(v_1, v_2, ..., v_k)$ such that there is an edge between each pair (v_i, v_{i+1}). A simple path with all its vertices distinct. Paths are used in finding the shortest path between two vertices or detecting cycles in the graph.

1.3.4 Distance

Distance is the minimal number of edges in a path that connects v_1 and v_2 as the shortest path problem, denoted by $dist(v_1, v_2)$ (See Equation 1.3.5 and Equation 1.3.6):

$$dist(v_1, v_2) = \text{min(length of all paths from } v_1 \text{ to } v_2) \tag{1.3.5}$$

$$dist(v_1, v_2) = \infty, \text{ If no path exists between } v_1 \text{ and } v_2. \tag{1.3.6}$$

Distance is crucial in algorithms like Dijkstra's and Bellman-Ford algorithms in order to calculate the shortest paths in graphs.

1.3.5 Connected Graph

A graph G is connected if there exists a path between any two different vertices (v_1 and v_2). Or we can say for any two vertices (v_1 and v_2), there exists a sequence of edges that relate v_1 to v_2. We say that a graph is not connected if there exist some vertices that are not reachable from each other.

1.3.6 K-edge-connected graph

A graph is a k-edge-connected graph if there are k edges-disjoint paths between each pair of distinct vertices v_1 and v_2 in G. In other words, it requires the removal of at least k edges to disconnect the graph.

1.3.7 Connected Components

Connected components are subgraphs where any two vertices are connected by paths, and no vertex is connected to a vertex outside the subgraph. We can say that a connected component is a maximal connected subgraph. In the case when the graph is disconnected, it is a set of connected components.

1.4 Some Graph Parameters

There exist various problems and concepts related to graph theory. These problems help in analysing properties of graphs. In this section, we detail these problems:

1.4.1 Independent Set

An independent set in a graph ($S \subseteq V$) is independent if no two vertices in S are adjacent. We can say that for all pairs of vertices u and v in S, there is no edge between them. Formally:

$$\forall (u, v) \in S, (u, v) \notin E \tag{1.4.1}$$

An independent set is considered maximal if it cannot be extended by including any other node. A maximum independent set refers to an independent set with the largest possible cardinality among all independent sets.

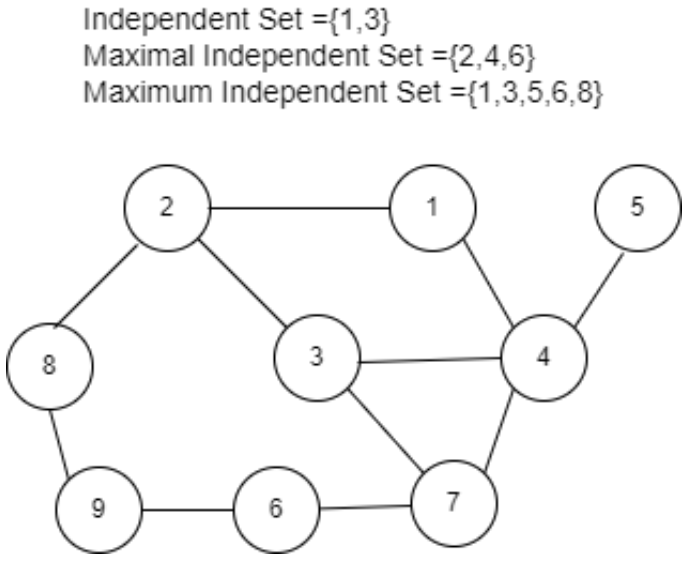


Figure 1.3: Independent Sets

A clique in a graph G is a subset C of $V(G)$ in which all two vertices in C are nearby in G. Formally:

$$\forall u, v \in C, (u, v) \in E. \tag{1.4.2}$$

Figure 1.4 shows the different possible cliques in black colour in the same graph.

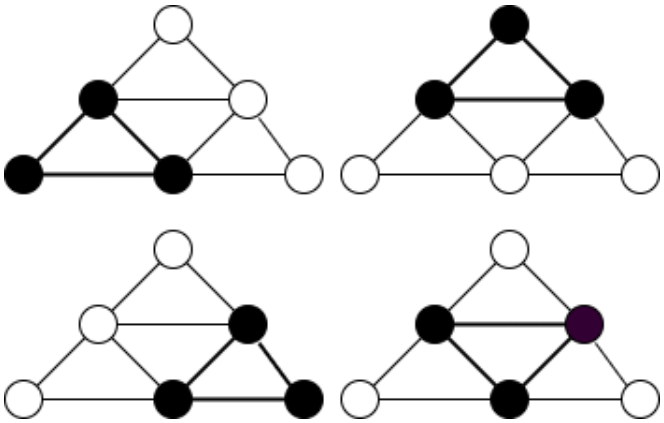


Figure 1.4: Clique

The clique number $\omega(G)$ represents the order of a maximal clique within G. Figure 1.5 represents the largest clique in black colour.

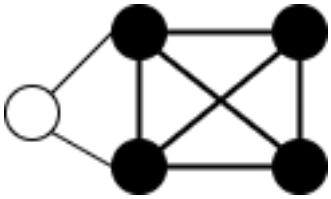


Figure 1.5: Maximum Clique Problem

1.4.2 Shortest Path Algorithms

Dijkstra's Algorithm is the most known algorithm to find the shortest paths from a source vertex to all other vertices in the graph, it is widely used in network routing and GPS navigation. This algorithm works by repeatedly picking the nearest vertex (with the least tentative distance) from the source, updating the shortest path estimates to its neighbours, and repeating the process until the shortest path to all vertices is found.

1.4.3 Planarity Testing

A graph is planar if it does not contain a subgraph homeomorphic to N_k for some $k > 1$, and it can be drawn on a plane without any edges crossing. The Kurato-Seymour theorem is the most known test for planarity.

1.4.4 Graph Isomorphism

If a bijection exists between two graphs $G = (V, E)$ and $G' = (V', E')$, then they are isomorphic.

$$F : v_1 \to v_2 \text{ such that } (u, v) \in E \iff (f(u), f(v)) \in E \tag{1.4.3}$$

In other words, two graphs are isomorphic if their vertices and edges have a one-to-one correspondence while preserving the graph's structure.

1.5 Some Families of Graphs

Graphs can be classified in various classes depending on their structure or properties. In this section, we will introduce different important families of graphs:

1.5.1 Bipartite Graph

A graph $G(V, E)$ is a bipartite graph, where the set of vertices V is divided into two different classes X and Y such that :

1. if $V = X \cup Y$ with $X \cap Y = \emptyset$.
2. $\forall (x, y) \in E$, $x \in X$ and $y \in Y$.

This type of graph is generally used in graph matching problems like marriage problems and graph colouring problems. The Figure 1.6 shows a bipartite graph, where all vertices are split into two classes, A (vertices in white colour) and B (vertices in black colour), where there exist edges between vertices from different classes.

1.5.2 Complete Graph

In this type of graph, there exists an edge between every pair of distinct vertices. A complete graph with n vertices is denoted by K_n, and it has the maximum number of edges

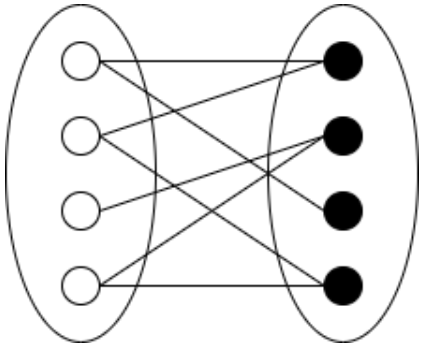


Figure 1.6: Bipartite Graph

possible, where n is the number of vertices, which is given by the formula :

$$|E| = \frac{n(n-1)}{2} \tag{1.5.1}$$

The following Figure 1.7 shows a complete graph representing highly connected structures where every vertex is connected to every other vertex.

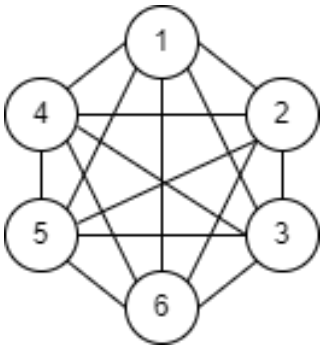


Figure 1.7: Complete Graph

1.5.3 Trees and Forests

A tree is a graph where there is one edge between any two vertices (there exists no cycle), as shown in Figure 1.8, it represents a hierarchical structure like charts.

A forest is a collection of disjoint trees. It is not necessarily connected, but it is acyclic.

A tree with n vertices has exactly $(n-1)$ edges. Removing any edge from a tree results in two disconnected components.

1.5.4 Planar Graph:

A graph is planar if it can be modelled in the plane such that no edges cross.
Euler's formula for planar graphs, which relates the number of vertices V, edges E, and

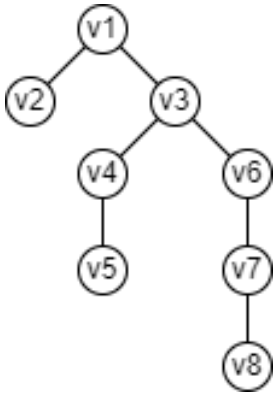


Figure 1.8: Tree

faces F:

$$V–E + F = 2 \tag{1.5.2}$$

V is the vertices set, E is the edges set, and F is faces. In the Figure below 1.9, the graphs represent planar graphs, where no edges cross each other; also, these graphs can be drawn on a plane surface without any intersection of edges.

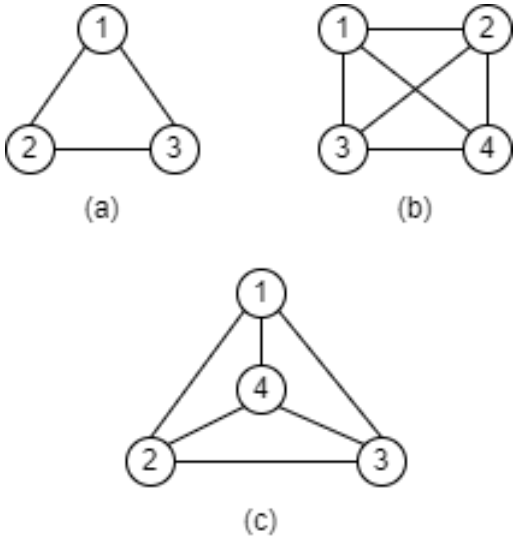


Figure 1.9: Planar Graphs

1.5.5 Unit Disk Graph

In this type of graph, vertices are points in 2D space. An edge exists between two vertices u and v if the distance between them (denoted by $dist(u, v)$) is less than or equal to r, where r is a fixed radius. Formally:

$$dist(u, v) \leq r \tag{1.5.3}$$

Unit disk graphs are particularly useful in wireless network modelling, where nodes (vertices) represent devices, and edges represent the communication links between devices that are within a certain range of each other. The following Figure 1.10 illustrates unit disk graph, where vertices are placed in a 2D plane, with edges drawn between vertices that are within the specified radius r of each other.

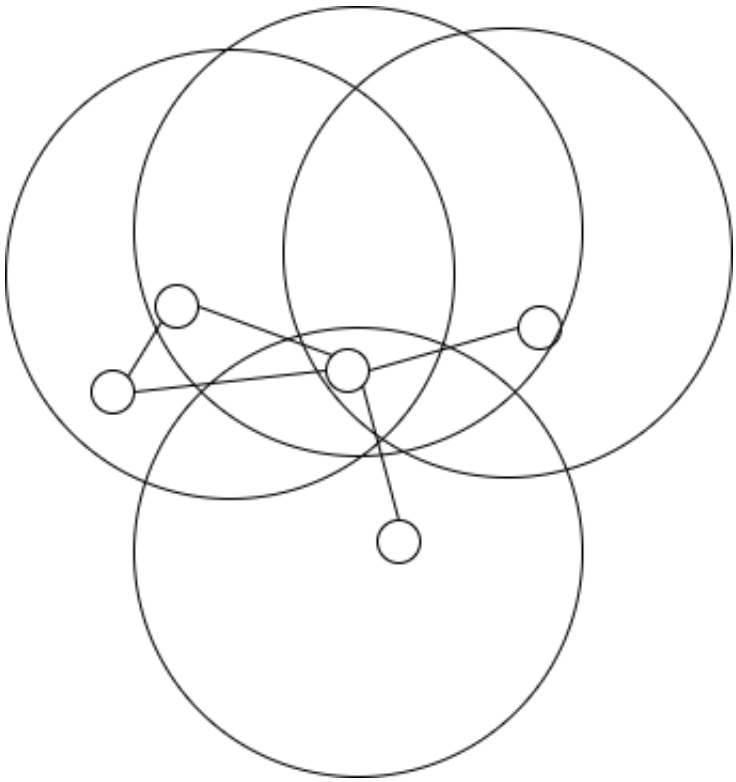


Figure 1.10: Unit Disk Graph

1.5.6 Random Graph

The most known random graph is the Erdős–Rényi model, denoted by $G(|V|, p)$.

p represents an edge probability between any two vertices, and as we know $|V|$ is the number of vertices. Every edge is included independently with a probability value p. This type of graph is generally used to model social networks.

1.6 Graph Metrics

Graph metrics are useful for assessing the structure and behaviour of a graph because they reveal information about the relationships and features of its vertices and edges. This section discusses several fundamental graph metrics:

1.6.1 Diameter

The diameter of a graph G denoted by diam(G) refers to the maximum distance between two unique vertices v_1 and v_2 in the graph. Formally:

$$diam(G) = \max_{v_1, v_2 \in V} dist(v_1, v_2) \tag{1.6.1}$$

1.6.2 Centrality Measures

Centrality measures the importance of each vertex in a graph based on their position, connectivity, or influence within the network. There exist various types of centrality measures:

1. **Degree Centrality**

   This type of centrality measures the importance of a node according to its number of edges.
   For a given vertex v, the degree centrality is calculated as follows:

   $$C_D(v) = \deg(v) = |N(v)| \tag{1.6.2}$$

   Normalized Form (for undirected graphs):

   $$C_D(v) = \frac{\deg(v)}{(n-1)} \tag{1.6.3}$$

   n is the number of vertices.

2. **Closeness Centrality**

   This type of centrality measures how close a vertex is to all other vertices in the graph (based on shortest path).
   For a given vertex v :

$$C_C(v) = \frac{1}{\sum_{u \in V} d(v,u)} \tag{1.6.4}$$

The normalized form is :

$$C_C(v) = \frac{n-1}{\sum_{u \in V} d(v,u)} \tag{1.6.5}$$

3. **Betweenness Centrality**

   Betweenness centrality measures the extent to which a node lies on the shortest paths between other nodes, the node is as a bridge.
   For a given vertex v:

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

(1.6.6)

   The total number of shortest paths between s and t is denoted by σ_{st}, while $\sigma_{st}(v)$ represents the number of shortest pathways that pass via v.

4. **Eigenvector Centrality**

   Calculates the total connection of nodes to other highly connected nodes.
   For a vertex v:

$$C_E(v) = \frac{1}{\lambda} \sum_{u \in N(v)} C_E(u) \tag{1.6.7}$$

   Where λ is the eigenvalue, and $N(v)$ represents the neighbours of vertex v.

1.6.3 Similarity Measures

Measure the resemblance between nodes, generally used to determine how similar two vertices are, based on their structure or features. Some common similarity measures are:

1. **Cosine Similarity:** Measures the similarity between two nodes based on their feature vectors:

$$S_{\cos}(v_1, v_2) = \frac{\mathbf{x}_{v_1} \cdot \mathbf{x}_{v_2}}{\|\mathbf{x}_{v_1}\| \|\mathbf{x}_{v_2}\|} \tag{1.6.8}$$

2. **Jaccard Similarity:** Compares the neighbourhood sets of two vertices, v_1 and v_2:

$$S_{jacc}(u, v) = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|} \tag{1.6.9}$$

3. **Hub-Promoted Index (HPI):** Determines the role of high-degree nodes (hubs). For two nodes u and v, the Hub-Promoted Index is defined as:

$$HPI(u, v) = \frac{|N(u) \cap N(v)|}{\min(\deg(u), \deg(v))} \tag{1.6.10}$$

   $N(u)$ and $N(v)$ are the neighbourhoods of nodes u and v, respectively.

   $deg(u)$ and $deg(v)$ represent the degrees of u and v, respectively.

4. **Hub-Depressed Index (HDI):** In contrast to HPI, it penalizes the similarity score when one or both of the nodes are hubs. This makes the measure more sensitive to low-degree nodes and their unique connections.
   For two nodes u and v, the Hub-Depressed Index is defined as:

$$HDI(u, v) = \frac{|N(u) \cap N(v)|}{\max(\deg(u), \deg(v))} \tag{1.6.11}$$

1.7 Applications and Challenges of Graph Theory

Graph theory has a significant role in many application domains, such as detecting communities of similar interest in social networks. Additionally in PPINs can aid in drug target research [3]. Another area of use is epidemics to model the spread of disease [4]. Graph theory is also used to reduce and optimize the topology of networks in communication systems, such as in WSNs [5]. It can also be used in network optimization using the graph partitioning technique to optimize traffic congestion [6]. It can also be used in image segmentation, which uses graph-cut algorithms to partition a picture into relevant sections [7]

Despite the various applications and relevance domains of graph theory, it faces several challenges that limit its effectiveness and practical use in real-world applications. One of the main issues is the lack of trusted datasets of appropriate scales that are necessary

for the rigorous evaluation of detection algorithms on graph data. The definitions of the background (normal patterns and noise) and foreground (target and anomalous patterns) are still being finalized. Additionally, even from a single dataset, a large number of graphs can be generated, making it difficult to identify the most effective graph representation for a given task [8].

Many applications also involve dynamic relationships, meaning that the graph structure evolves over time. Addressing large-scale dynamic graphs is a rapidly growing research area that requires substantial effort. Furthermore, in certain real-world applications, there is a need to quickly construct a variety of graphs from the same input or from different datasets. However, stockade, modelling, and graph data access techniques are often hard-coded, leading to processes that are time-consuming, error-prone, and difficult to maintain [8].

Moreover, many graph algorithms suffer from high computational complexity, and their performance degrades as the number of nodes and edges increases. As the size of the graph grows, the time and resources required to process it grow exponentially, posing a significant challenge to scalability [8].

1.8 Conclusion

Graph theory is a useful tool for representing, analysing, and solving problems of complex networks and systems in many applications, such as computer science, biology, and communication. In this chapter, we introduce graph theory and give definitions, notations, properties, and problems in order to understand network analysis as the fundamentals of modelling and studying complicated interactions among entities.

In the following chapter, we will talk about graph partitioning as an interesting subject in graph theory and how it is applied to real-world problems. Specifically, we will investigate the role of critical nodes whose removal has a major impact on network connectivity.

Chapter 2

Graph Partitioning Problem

2.1 Introduction

Graphs have an important role in understanding and evaluating the relationships in complex networks. As graphs grow in size and complexity, efficiently analysing and processing them becomes a challenge, particularly when working with large-scale networks.

Graph partitioning is considered an important method to solve this issue. It aims to partition a graph into smaller, manageable partitions while maximizing specified criteria, and making analysis and processing more flexible, such as minimizing edge cuts. The number of edges that cross the boundaries between partitions is an objective to find a balance between partitions, which should be as small as possible for efficiency, and the cost of cutting edges between these partitions, which should be minimized to maintain connectivity and reduce information loss. Additionally, graph partitioning is an effective method that reduces the computational complexity.

In this chapter, we give a thorough understanding of graph partitioning problem. We begin by outlining the notion, definition, and its use in various areas. We then look at the traditional and new graph partitioning methods. In addition, we discuss the challenges that come with graph partitioning, known as Critical Nodes Problem, which aims to detect important nodes within the graph, especially those that impact the performance of the partition graph and maintain connectivity.

2.2 Problem Definition and Notation

In this section, we aim to introduce and define the graph partitioning problem and related notations.

2.2.1 Notation

Graph partition is defined as the following [9]:

Let a graph $G = (V, E)$ be partitioned into k disjoint subgroups of equal size, with the goal of minimizing the number of edges between them.

Formally, it is to find a partition $(V_1, V_2) \in V$ (i.e.$\{V_1, V_2, ..., V_k\}$ such that:

$$V_i \cap V_j = \Phi \text{ for } i \neq j \text{ and } \cup V_i = V \tag{2.2.1}$$

This is proof that vertices are assigned to a single partition.

The objective of graph partitioning is to minimize the number of cut edges, which connect vertices from distinct partitions. Additionally, these partitions should be as balanced as possible, based on limitations of a problem (such as equal size or weight).

2.2.2 Problem Definition

Graph partitioning, often known as k-way partitioning, is a method for dividing a graph into k distinct subgraphs based on many criteria, depending on the partitioning method. Although the main objectives are known. It is an NP-hard problem, especially for large-scale graphs, there is no algorithm that provides an optimal solution in linear time. Thus, a faster algorithm might be the slowest for some graphs and vice versa.

We can classify graphs partitioning as [10]:

- Partitioning without any constraints:
  In this class of partitioning, it minimizes an objective function by identifying k partitions of graph G .
- Partitioning with constraints:
  1. The first constraint in graph partitioning is partition balancing. To discover a k-partition ($P_k = S_1, S_2, ..., S_k$) of G with a balanced $(k, 1 + \epsilon)$ partition, each subset $S_i \forall i \in [1, k]$ must be identified. Let $W_i = |V_i|$ be the weight of the ith partition of G.

     The average weight W_{avg} and balance $B(P_k)$ of the partition P_k are defined as follows:

$$W_{avg} = \frac{\sum_{i=1}^{k} w_i}{k} \tag{2.2.2}$$

$$B_{P_k} = \frac{max w_1 ... w_k}{w_{avg}} \tag{2.2.3}$$

If the requirement $B(P_k) < (1+\epsilon)$ is met, the partition P_k is properly balanced; this indicates that the partitioning is dispersed equally while taking an error ϵ into account.

2. The second constraint is finding a k-partition P_k that minimizes the cost of all exterior edges (s_i, s_j) joining two partitions S_i and S_j is the second restriction, known as the cut-edge.

   It enables high-performance computing by lowering communication overhead. The following formula is used to determine the cost of the cut-edges $cut(S_i, S_j)$ between two partitions, S_i and S_j:

$$cut(S_i, S_j) = \sum_{s_i \in S_i, s_j \in S_j} W(s_i, s_j) \tag{2.2.4}$$

   Where the weight of the edge (s_i, s_j) is represented by $W(s_i, s_j)$. In this instance, the cut-edges global cost between k-partitions is defined as follows:

$$cut(P_k) = \sum_{i,j<=k} cut(S_i, S_j) \tag{2.2.5}$$

   The objective is to reduce the overall cost of the cut edges while remaining within the partitioning limits.

   The following Figure 2.1 shows a graph partitioning example.

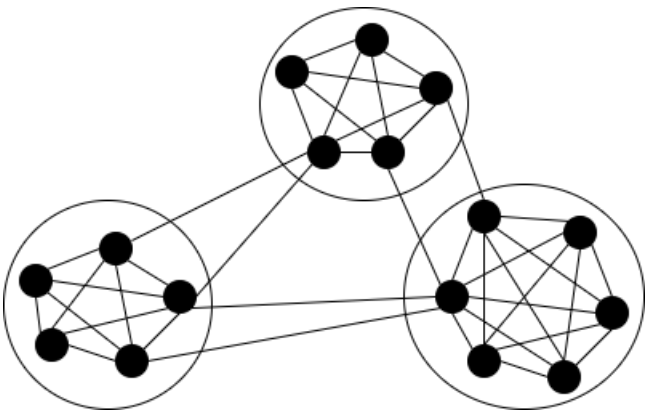


Figure 2.1: Graph Partitioning

2.3 Application

Graph partitioning has various applications across different fields. In the following we detail each application of the problem:

- **Parallel Computing and Load Balancing:**

  In parallel computing, the problem of partitioning a large task into smaller and more manageable subtasks arise. Graph partitioning is used to efficiently distribute these subtasks among different processors. The idea is to distribute the jobs so that each processor has roughly the same load while minimizing computation time [11].

- **Metropolitan Transportation Systems:**

  Graph partitioning is used to optimize fiber-optic cable networks and wireless sensor networks in metropolitan transportation systems. By dividing the transportation network into regions or subgraphs, the system may better manage data flow, and improve fault tolerance, and also optimize routing for less congestion. This is very beneficial for managing large-scale infrastructure [12].

- **Complex Networks:**

  1. **Social Networks:**

     Graph partitioning is used to detect groups or communities with similar interests, well-known as community detection, like in [13], which aims to identify communities in Facebook to detect influencer individuals. Graph partitioning simplifies the task of understanding network dynamics, such as tracing the origin of trends and information spread.

  2. **Biological Networks:**

     Biological networks such as PPINs[14], use graph partitioning to identify functional modules or pathways by clustering together proteins that are closely interconnected. This can assist in understanding biological functions and in identifying potential drug targets.

  3. **Power Grid Networks:**

     Partitioning is used in power grid networks that have two fundamental problems, including disturbances and cascading that can cause catastrophic blackouts. To avoid those issues, it is necessary to partition the power grid network into independent partitions [15].

  4. **Wireless Sensor networks (WSNs):**

     Graph partitioning aims to organize sensor nodes into partitions more efficiently, which can optimize data aggregation and minimize energy consumption, which is essential in WSNs [16].

- **Image Segmentation**

  In image segmentation, graph partitioning is applied to partition the image pixels into meaningful regions for identifying different objects or areas in an image, such as object recognition and scene understanding, like segmenting an image into regions for object recognition. Image pixels are modelled as a graph, and graph partitioning helps divide the image into coherent regions based on similarity [17].

- **Transportation Optimization**

  In transportation networks, graph partitioning is used to divide into subgraphs, enabling efficient routing within each partition and reducing the number of inter-partition connections that need to be computed. This improves the efficiency of path-finding algorithms. As an example [18], they optimize traffic flow by dividing road networks into regions.

- **Very Large Scale Integration (VLSI)**

  In VLSI physical design, graph partitioning is used to reduce the complexity due to the increase in computation time with billions of modules in the circuit, by partitioning circuits into smaller components [19].

- **Distributed Databases and Query Optimization:**

  Graph partitioning is used in distributed databases and query optimization in order to minimize the average number of shards in a query, optimize latency and processing time, and optimize SQL query execution in a distributed context, which involves minimizing data transfer between nodes. The query execution plan is depicted as a graph, where nodes represent operations such as joins and scans, and edges represent data dependencies [20].

In this research, we are particularly interested in complex networks since they are important to tackling a wide range of critical concerns in modern systems, like connectivity optimization, resource allocation, and data flow. Graph partitioning is an interesting strategy in analysing and optimizing complex networks like social networks and wireless sensor networks. An efficient partitioning method that divides the network can improve performance and reduce complexity while enhancing efficiency and resilience. These networks have sophisticated topologies, making them important elements for graph-based analysis and optimization to solve practical real-world issues, including community detection, source detection, and energy efficiency in wireless networks.

2.4 Graph Partitioning Algorithms

Graph partitioning can be classified into numerous forms depending on the methods employed and the objectives they seek to achieve [21]:

2.4.1 Edge Cut Partitioning

The basic purpose of edge cut partitioning is to divide the graph into disjoint subgraphs while reducing the number of edges connecting vertices from distinct partitions (See Figure 2.2). In [22], they developed METIS, which employs a multilevel edge-cut partitioning approach. Their objective is to minimize cut size and balance the partitions for high quality of partitioning with low computational cost. Another research [23] improves cut-edge partitioning by including a multilevel method to enhance the scalability and for high-quality partitioning.

Edge cut partitioning is especially essential for problems involving parallel computing, network design, and optimization when the goal is to balance workload among partitions or reduce communication overhead. By minimizing the edge cut, partitioning seeks to ensure that the subgraphs are as internally connected as feasible while remaining as separate as necessary, hence improving computational efficiency or system performance.

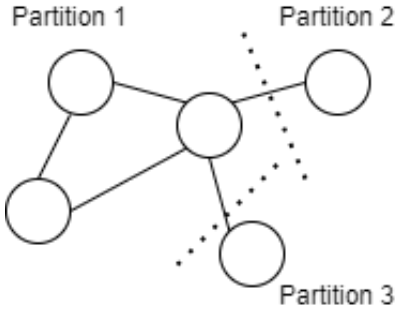


Figure 2.2: Edge Cut illustration

2.4.2 Vertex Cut Partitioning

Vertex cut partitioning aims to divide the graph into disjoint subgraphs such that the replication of vertices across partitions is minimized (See Figure 2.3). Authors in [24] offer a vertex-cut partitioning method for web-scale graphs, clustering vertices to minimize edge-cuts and assigning each cluster to partitions to balance load and computational expense.
The goal of this type is to reduce the number of essential nodes or vertices, which, when removed, divide the network into smaller, more manageable components. This method

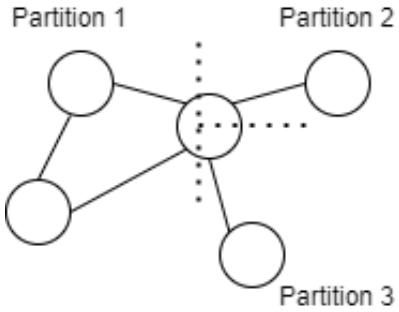


Figure 2.3: Vertex Cut example

is commonly used in fault tolerance, network resilience, and parallel computing, where identifying significant vertices is critical, as their removal can quickly split the graph or isolate components of a system for optimization.

2.4.3 Recursive Partitioning

Recursive partitioning divides the graph into two sections and then recursively partitions each subgraph. It enables easy implementation but may not always result in balanced partitions. Several researchers have concentrated on recursive partitioning for graph partitioning, as an example [25] combines multi-level recursive partitioning with high-performance computing contexts to break down complex tasks into smaller recursive phases, balancing computational demands and decreasing communication between partitions.

This class is beneficial for big graphs because it divides problems into smaller subproblems, which allows for efficient parallel computing and hierarchical clustering. It produces high-quality partitions for a lesser cost than previous algorithms.

2.4.4 Offline Partitioning

Offline graph partitioning partitions a graph based on its full structure and loads it into memory before applying partitioning methods [21]. The common techniques in offline partitioning, including spectral partitioning, multilevel partitioning, and geometric partitioning utilize geometric properties of nodes to create partitions.

1. **Multilevel Partitioning**

   This type aims to balance high-quality partitions with computational efficiency. It has three phases, including coarsening, initial partitioning, and un-coarsening, as shown in Figure 2.4 [26]. In [27], they used this technique. A study on [28] proposed a framework designed to scale across distributed systems by coarsening graph structures

while maintaining balance constraints. The new techniques prevent oversized clusters during the coarsening phase, allowing fine control over graph partition quality even in distributed environments. Others [29] introduce a multilevel approach that combines spectral methods with evolutionary algorithms. Which is important in the VLSI design domain for high-quality partition.

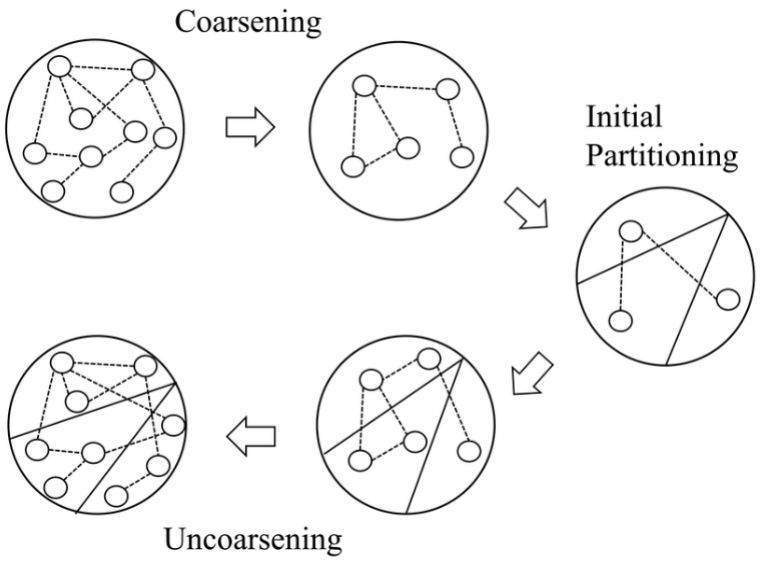


Figure 2.4: Multilevel Partitioning Illustration

2. **Spectral partitioning**

This partitioning method uses eigenvalues and eigenvectors of matrices, such as a graph's Laplacian matrix, to identify partitions. The Fiedler vector (the second-smallest eigenvalue) helps determine optimal cuts by categorizing nodes based on network architecture. Authors in [30] proposed a framework where they applied spectral coarsening techniques to hypergraphs, allowing for effective partitioning by reducing the complexity of the hypergraph without losing essential structural properties by incorporating hypergraph conductance and community detection. However, traditional graph methods might fail to capture high-order interactions. Spectral partitioning methods can also be used in community detection, like [31], they identify clusters within networks based on eigenvectors of the Laplacian. Newman et al. [31] demonstrate the efficiency of applying spectral approaches through the usage of the Laplacian eigenvectors in splitting networks into communities by minimizing edge cuts between clusters.

Offline partitioning involves adjusting partitions in response to real-time changes in the graph, it works with a static graph structure, making it more suitable for scenarios and a high-quality, optimized partitioning, which is required.

3. **Geometric Partitioning**

   In this partitioning method, the graph is divided into partitions using the physical or geometric coordinates of its nodes. Generally useful in case when the networks have geographic constraints.

   Geometric partitioning preserves locality and minimizes edge cuts and communication costs in distributed systems. It's effective in evenly distributed graphs but struggles with irregular or highly clustered ones. Recursive coordinate bisection and space-filling curves improve partitioning quality, but challenges remain for non-uniformly distributed data. Authors in [32] present a novel parallel approach for multidimensional jagged geometric partitioning that outperforms existing recursive coordinate bisection algorithms. This method is suitable for situations requiring dynamic load balancing and geometric data locality. It is discussed in [33] how geometric partitioning can improve fault tolerance in distributed systems. This study intends to enhance repair operations by reducing data redundancy and improving the effectiveness of approaches for single-node failures.

2.4.5 Online Partitioning

Online partitioning is designed for dynamic graphs as they evolve over time. Streaming partitioning algorithms assign nodes to partitions online, for high-quality partitioning with low memory overhead, it can adapt to changes without the need for a complete repartitioning, reducing the cost and disruption to the system. However, traditional offline systems necessitate repartitioning the entire graph for each update, which incurs enormous computational overhead. In [34] they discovered that using local views of a graph reduces processing costs.

In the table below, we compare research articles according to their graph partitioning method, advantages, and disadvantages.

Table 2.1: A Comparative Analysis based on Graph Partitioning Algorithms

Works	Type of GP	Benefits	Limitations
[22]	Offline, Multilevel (Edge Cut Partitioning)	Efficient for large unstructured graphs. Produces high-quality partitions with minimal edge cuts. Widely used for sparse matrices and scientific computing.	Assumes the entire graph is available in memory. Not suitable for streaming or dynamic graphs
[23]	Offline, Multilevel, Evolutionary	High-quality partitions for edge-cut minimization. Incorporates evolutionary strategies for refinement.	Higher computational cost due to evolutionary refinement. Not suitable for real-time applications.
[27]	Online, Streaming, Vertex Cut	Designed for resource-constrained edge devices. Scales efficiently to large graphs.	Sacrifices optimality for scalability. Higher memory usage due to vertex replication.
[24]	Offline, Clustering, Edge Cut	Efficient for web-scale graphs using community detection. Provides natural graph divisions for applications like crawling.	May struggle with overlapping communities. Clustering quality heavily affects partitioning quality.
[28]	Multilevel, Distributed, Offline	Combines multilevel strategies with distributed computation. Suitable for extremely large graphs.	Requires a distributed environment. Communication overhead between nodes.
[29]	Evolutionary, Hypergraph, Offline	Tailored for hypergraphs with differentiable approaches. High-quality partitions for complex graph structures.	High computation for evolutionary algorithms. Focused on hypergraphs, not general graphs.
[30]	Spectral, Offline	Leverages eigenvectors for partitioning. Produces highly balanced and quality partitions.	Computationally expensive due to eigenvector calculations. Requires access to the entire graph upfront.
[34]	Online, Streaming, Vertex Cut	Lightweight algorithms for dynamic graphs. Balances load effectively in streaming scenarios.	Limited optimization compared to offline methods. Higher memory for vertex replication.
[33]	Geometric, Offline	Utilizes spatial properties for efficient partitioning. Suitable for spatial datasets and sensor networks.	Assumes geometric relationships are relevant. Limited to graphs with spatial embeddings.

2.5 Challenge of Graph Partitioning

Graph partitioning has the following challenges due to the complexity of tasks:

1. **Minimizing Edge Cuts:** This challenge is ensuring the balance of the resulting divisions while minimizing the number of cut edges. Many researchers focus on this challenge [22], [23], [27], [24].

2. **Partition Balancing:** Ensuring that the partitioned subgraphs are of equal or nearly equal size, which is especially important in many applications where each partition should be balanced in parallel computing environments. Many algorithms require this challenge [22], [23], [27], [24], [28].

3. **Scalability:** As the massive graph with huge volumes of data, the storage back-ends is needed [10]. Many algorithms struggle with large-scale graphs, requiring efficient heuristics or approximation methods to find near-optimal solutions within reasonable time frames, as in [22], [23].

4. **Dynamic graph:** Many real-world graphs change over time. Most existing algorithms need to be updated to support dynamic graphs such as slowly evolving graphs (like road networks and co-authorship networks) and streaming networks (social networks and web networks). In addition, fast analysis of data in motion, which powers the graph is needed [10]. Authors in [35] proposed a dynamic partitioning algorithm that can dynamically add or remove partitions based on graph changes, it adapts balance computation across distributed systems and minimizes cut-edges. The survey [21] included methods tailored for dynamic graph processing.

5. **Computational complexity:** As we know that graph partitioning is NP-hard, most proposed approaches do not find the optimal solution for large-scale graphs and also can not be solved effectively in polynomial time [10]. To handle this challenge, many algorithms improve efficiency in computational processes and system performance [22], [27], [24].

2.6 Discussion

Graph partitioning aims to divide a graph into smaller, more manageable subgraphs. It is considered an important method for managing and analysing big, complex networks. Such as in the well-known problem of graph theory, the community detection problem, which seeks to identify set of nodes within a network that are more densely connected than others. Graph partitioning is generally used to reduce the complexity of a network while

maintaining a balance in the size of each subgraph. It also aims to minimize the number of edges connecting nodes from different partitions, known as cut edges. This process helps make the dynamics and structure of a network easier to understand.

In this regard, cut vertices are nodes whose removal can increase the number of connected components in the graph. In other words, a cut vertex is a critical node that, when removed, breaks the graph into disconnected subgraphs. It is an interesting method in graph partitioning. Critical nodes in most real-world networks, like social networks and communication networks represent nodes of great importance, or we can say influential nodes. If these nodes are removed, it affects the network's connectivity. Understanding these critical nodes helps in designing more robust and resilient networks.

The role of critical nodes detection can partition the graph more efficiently. If we remove these nodes, the graph can be divided into smaller, more manageable subgraphs. This makes it easier to understand the network's structure by simplifying the graph and reducing its complexity, especially for large-scale networks, but how to identify these nodes?

2.7 Conclusion

In this chapter, we defined the graph partitioning problem and highlighted its different application domains. This technique simplifies the analysis of large, complex networks. It aims to split a graph into smaller, more manageable partitions. This partition not only reduces computational cost and scalability but also improves resource management in many fields.

We presented numerous approaches of graph partitioning, we also highlighted its challenges. A particular discussion shows the role of critical nodes whose removal significantly impacts the network's connectivity. The fundamental idea behind these nodes is to optimize network partitioning by minimizing cut vertices and ensuring a balanced distribution of distinct divisions.

In the following chapter, we will go into detail regarding the Critical Nodes Problem. This well-known problem is useful for identifying nodes whose removal will affect the graph's overall connectivity. Furthermore, understanding this topic is critical for optimizing graph partitioning algorithms and improving network analytics.

Part I

State of the Art

Chapter 3

Critical Nodes Problem (CNP)

3.1 Introduction

Graph partitioning is an essential concept in graph theory, and its success is frequently determined by the existence of specific nodes that play a critical role in the network's structure and connection. These nodes are referred to as "Critical Nodes" and are regarded as an important partitioning approach due to their goal of splitting the network into several areas. The identification of these nodes and their removal can consistently lead to the division of the graph into several disconnected components. This property makes critical nodes beneficial for many tasks such as community detection, load balancing, source detection and optimizing the flow or efficiency of networks.

In this chapter, we focus on the Critical Nodes Detection Problem (CNDP), we will provide a comprehensive exploration of this concept. We will define this problem and give a formal definition of CNDP and outline its existing variants. Furthermore, we will investigate the broad range of applications of CNDP across different fields. These applications demonstrate the variety and importance of recognizing nodes in real-world networks. However, finding these nodes poses several obstacles. We will address the difficulties of correctly detecting such nodes, particularly in big, dynamic graphs. Furthermore, we will examine the proposed algorithms for resolving this issue, emphasizing their strengths and limitations.

3.2 Problem Definition

Critical Nodes Problem (CNP) aims to identify a subset of nodes within a graph, removal of which results in changing the structure of the residual graph and minimizes or maximizes a predefined metric on the remaining graph [1].

After removing critical nodes, it is crucial to consider how the network becomes disconnected, because disconnecting it in any way is often inefficient. Identifying which nodes are "critical" depends on the type and goal of the application at issue, such as using connectivity metric, which is considered a good measure to optimize the network structure after node removal [1].

Given a graph $G = (V, E)$, where V is the set of nodes and E is the set of edges, and a specific graph connectivity metric σ, the CNP involves finding a subset of nodes $S \subseteq V$ so that $|S| \leq k$ such that [36]:

1. If we remove S from V , σ is optimized in the residual graph $G' = (V \setminus S, E')$.
2. $f(\sigma)$ represents an objective function in such a way that [1]:
   (a) Maximizing the number of connected components.
   (b) Minimizing pairwise connectivity.
   (c) Minimize the largest connected component size.
   (d) Limit the maximal component size to a given. bound
   (e) Bound pairwise connectivity to a given threshold.

Determining the concept of criticality in relation to the input problem is made more precise by defining the metric σ. This metric explains how the graph structure must be disconnected and ensures that, once the nodes are removed, the required network connectivity criteria are met, allowing for the correct identification of critical nodes [1]. For example, we suppose that in a distributed computing setting, a consensus requires communication between at least k servers. We must divide the network into smaller partitions, each with fewer than k servers, to keep it so that the attackers can't control the majority of the network and disrupt the consensus process. All partitions have $(k-1)$ servers are preferable to one where all servers are separated and just one group contains $(t+1)$ servers.

The CNDP is generally NP-complete depending on the class of graphs [1].

3.3 Variants of CNP

In this section, we describe each variant of CNDP in detail with example [1]:

3.3.0.1 CNP (Critical Node Problem)

This CNP variant aims to find a set of nodes containing at most k nodes in a graph, such that if these nodes are removed, it reduces pairwise connectivity in the remaining network. The fragmentation of the graph is maximized when pairwise connectivity is minimized.

Given a graph $G = (V, E)$ and an integer k, find a set of nodes $S \subseteq V$ containing at most k nodes, when removed, these nodes maximize the fragmentation and minimize pairwise connectivity of the remaining graph $(G \setminus S)$ such as; $|S| \leq k$,

The following function in Equation 3.3.1 to be minimized can be used to formulate the objective:

$$f(S) = \sum_{C_i \in G(V \setminus S)} \frac{\sigma(\sigma - 1)}{2} \tag{3.3.1}$$

C_i is the set of all connected components in $(G \setminus S)$ once the k nodes have been deleted, and σ is the size of the connected component C_i, it is may calculated efficiently in linear time using a Depth First Search (DFS) strategy.

The following Figure 3.1 represents the illustration of the CNP variant, the critical nodes are coloured in black (S $= \{v3, v7\}$), i.e., we have $k = 2$ critical nodes to detect if removed, it should minimize the pairwise connectivity formula:

$$f(S) = \sum_{C_i \in G(V \setminus S)} \frac{\sigma(\sigma - 1)}{2} = 1 + 3 = 4 \tag{3.3.2}$$

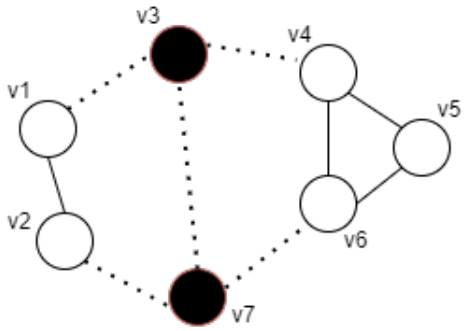


Figure 3.1: CNP Variant

3.3.1 MaxNum (Maximize the number of connected components)

In this variant, the objective is to minimize the variation in the cardinality of the connected components and increase the number of them. In [37] they proposed a novel variant

called MaxNum that maximizes the number of connected components while disconnecting the graph to the utmost. It aims to remove a set of nodes, and it is formulated as follows:

Given an undirected graph $G = (V, E)$ and an integer k, this variant aims to find a set of nodes $S \subseteq V$, such that:

$$\max(n \in G(V \setminus S)) \text{and} |S| \leqslant k \tag{3.3.3}$$

Where n is the number of connected components. There is no fragmentation of the graph if $n = 1$. In the case where each node is isolated, the maximum fragmentation of the graph happens, the result is the appearance of many connected components. This solution of MaxNum can be aligned with MIS in G, in the case of $|G(V \setminus S)| \leqslant k$. Similarly, with the problem of Edge Cut Partitioning (See Chapter 2 in Section 2.4.1), reducing the edge cut and making sure the subgraphs are as internally connected as possible while staying as separated as needed.

The following Figure 3.2, shows the MaxNum variant where there are two nodes in black that are considered critical, if removed, it maximizes the number of connected components to three.

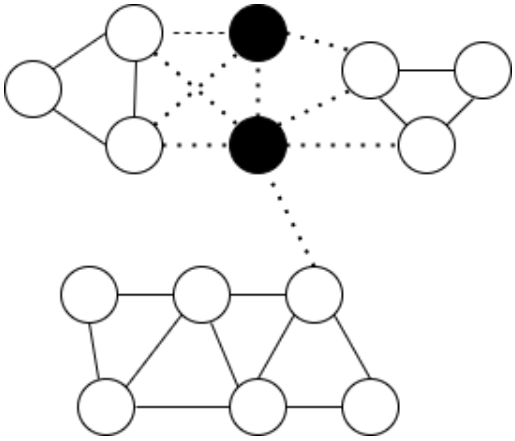


Figure 3.2: MaxNum Variant

3.3.2 MinMaxC (Minimize the size of the largest connected component)

In this variant, the size of connected components is considered as a connectivity metric, which the aim is to keep it to a minimum. In fact, the variant MinMaxC must remove at most k nodes from the remaining graph in order to minimize the size of the biggest connected component. It can be formulated in this way:

Let an undirected graph $G = (V, E)$ and an integer k. The objective of this variant is

to find a set of nodes $S \subseteq G$, such as:

$|S| \leqslant k$, and $\min \sigma_{h,h \in G(V \setminus S)}$ Where h is the biggest connected component.

The following Figure 3.3 represents the MinMaxC variant, where there are 3 critical nodes (in black colour), if removed, its minimize the size of the largest connected components.

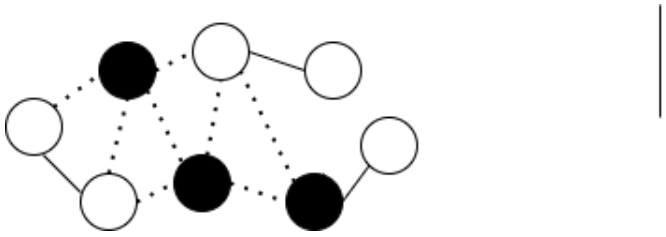


Figure 3.3: MinMaxC Variant

This variant MinMaxC is similar on determining the minimum vertex cover set of the graph if the objective is to limit the cardinality of the biggest connected component to 1.

A formulation of MinMaxC can be expressed as follows, where each node $i \in V$ has a weight such as $w \geqslant 0$ and a deletion cost, such as $c_i \geqslant 0$.

$$\min f(S) = \sum_{i \in h} C_i \tag{3.3.4}$$

In accordance with:

$$\sum_{j \in S} W_j \leqslant k \tag{3.3.5}$$

In this instance, we have two versions of this variant to consider $MinMaxCW_1$ and $MinMaxCW_2$.

In the first version $MinMaxCW_1$, $W_i = 1$, and $\forall i \in V$. Therefore, instead of deleting a set of finite cardinalities, we remove a set with bounded total node deletion cost.

In the second version $MinMaxCW_2$, $W_i > 1$ and $c_i > 1$, meaning that $i \in V$. Therefore, the objective is to eliminate a set of constrained total node deletion costs in order to minimize the maximum weighted component in the remaining graph.

3.3.3 CC-CNP (Connected Components Critical Nodes Problem)

The objective of this variant is to remove a set of nodes to minimize connectivity metrics while ensuring the remaining graph remains connected by limiting the maximal component

size to a given bound by deleting the minimal set of nodes [1].

Given a graph $G = (V, E)$ and an integer L. The objective of this variant is to reduce the number of nodes whose removal limits the pairwise connectivity of every connected element in the remaining network to a predetermined value. The following is a statement of its formulation

$$\arg\min S \subseteq V|S| \text{ , s.t. } \sum_{vi,vj\in \text{ h}} u_{ij} \leq \text{ L, for each connected component h } \subseteq G[V \setminus S].$$

$$\text{u}_{i,j} = \begin{cases} 1 & \text{if } v_i \text{ and } v_j \text{ are in the same component of } G(V \setminus S), \\ 0 & \text{otherwise.} \end{cases}$$

(3.3.6)

In order to ensure that the cardinality of every connected component in the residual graph is no more than a specified number L of nodes, in the case of $c_{ij} = 1$, it looks for a minimal set of nodes to be removed.

Figure 3.4 represents the CC-CNP variant, there are two nodes $v3$ and $v7$ (in black colour), if removed ensure that the pairwise connectivity of each connected component in the residual graph is no more than a threshold.

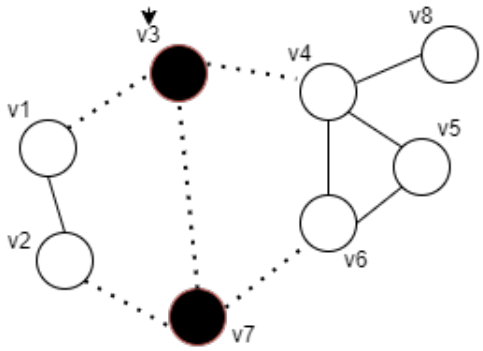


Figure 3.4: CC-CNP Variant

Finding the graphs with minimum vertex cover sets is the result of solving this variant when $L = 1$. It is also simple to see that the CC-CNP is MinMaxC in reverse. Actually, by eliminating a minimal set of nodes, the CC-CNP seeks to limit the connected component size in the remaining network to a specified threshold L.

3.3.4 β-vertex disruptor Problem

The objective of this variant is to minimize the number of nodes that, when deleted, minimize total pairwise connectivity to a certain level. A specified fraction β of the input graph's overall pairwise connectivity defines this level. The following is one way to formulate the problem.

Let a graph $G(V, E)$ where $|V| = n$, and a constant fraction $0 \leqslant \beta \leqslant 1$. This variant aims to find a minimum set of nodes $S \subseteq V$ when deleted, it limits pairwise connectivity in $G(V \setminus S)$ to $\beta \begin{pmatrix} n \\ 2 \end{pmatrix}$.

A range of complementary values can be handled in the remaining graph by using β as an input parameter. Furthermore, it is readily apparent that the CNP and the β-vertex disruptor problems are complimentary variants of the same problem. In both variants, we search for a set of nodes, no more than k, whose removal limits pairwise connectivity to a threshold L for the CNP , whilst a proportion β of the aggregate pairwise connectivity determines the threshold for the β-vertex disruptor problem. We remember that in a network with n nodes is $\begin{pmatrix} n \\ 2 \end{pmatrix} = \frac{n(n-1)}{2}$

The total pairwise connectivity is Consequently, the recognition version of the two variations is the same.

3.3.5 CCC-CNP (Component Cardinality Constraint Critical Nodes Problem)

[38]

The objective of previous traditional variants of Critical Nodes Problem is to identify a specific node whose removal leads to disconnecting the graph or fragmenting it. However, the CCC-CNP variant adds a novel constraint that the number of nodes in each resulting connected components must not exceed a defined threshold, denoted by L. This constraint forces a balance between connectivity and maintaining some level of overall connectivity in the graph.

Given a graph $G = (V, E)$, and a positive integer L as input, a weight w_i is defined for removing each node $v_i \in V$, and a positive cost c_{ij} is met for each edge $(v_i, v_j) \in E$. When we removed a set of nodes $A \subseteq V$, we obtain a residual graph $G(V \setminus A)$, whose connected components contain the node sets $h_1, h_2, h_3..., h_m$. The CCC-CNP aims to identify A so that:

$$\min \sum_{v_i \in 1} w_i$$

$\forall \sigma_k \in G[V \setminus A], \sum_{v_i, v_j \in h_k} c_{ij} \leqslant L.(3.3.7)$

Therefore, CCC-CNP aims to identify a set of nodes $A \subseteq V$ with a minimal total weight. So that, in the residual graph $G(V \setminus A)$, the total connection costs of each connected component is no greater than L.

CCC-CNP demands that as few nodes as possible be deleted in the unit case, where $w_i = c_{ij} = 1$. In such a way that the pairwise connectivity of each connected component is at most L. This means that the cardinality of A, not the total weight, should be minimized so that the pairwise connectivity, not the total connection cost, is no more than L.

Thus, CCC-CNP consists of finding a set of nodes $A \subseteq V$ of minimal total weight, such that the total connection costs of each connected component, in the induced graph $G[V \setminus A]$, is no more than L.

The Figure below 3.5 represents the CCC-CNP variant, the overall connection costs of each connected component, in the induced graph no more than 3.

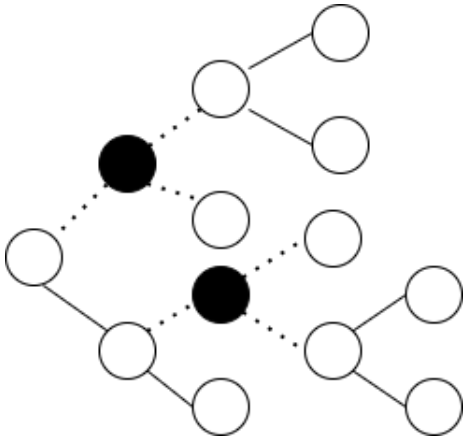


Figure 3.5: CCC-CNP Variant

Often regarded, CCC-CNP is considered a variant of the Graph Partitioning Problem, a classic Graph Theory Problem. This variant is similar to the Minimum Vertex Cover (MVC), which seeks to identify a minimum collection of nodes $A \subseteq V$ such that $G[V \setminus A]$ is an independent set when $L = 1$. Additionally, the Vertex Separator (VS) problem calls for a minimum number of nodes in order to divide the graph into two limited subsets. Moreover, the same as finding an ϵ-separator of G, where $\epsilon = \frac{L}{n}$.

In this research, we are interested in this variant CCC-CNP, so we propose a new heuristic to solve the problem for general graphs.

3.4 Application

The critical nodes detection problem has many applications in various fields, including social network analysis, biology, communication, and security networks. For instance, it can be used in epidemiology [39], such as the identification of critical nodes that can control and limit the spread of diseases and also isolate highly connected individuals to prevent outbreaks. Thus, this process has been considered for different objectives, including disruption diffusion [40], early detection of diffusion outbreaks [41] and locating sources of diffusion [42].

Another interesting application of CNP is social network analysis [43], like identifying influential people to maximize the spread of information or products. Other applications of CNP can be found in Network Security [44]. For instance, the detection of critical nodes can target key nodes in an enemy's communication network to disrupt operations. It can also be used in supply chain [45] , such as identifying vulnerabilities in distribution systems.

The importance of the CNP lies in reducing the computational complexity by partitioning large networks into partitions, well-known graph partitioning problems, such as identifying and segmenting critical nodes can balance workloads and ensure efficiency in networks [46].

Another important application is biological networks [47], in order to analyse and intervene in biological systems, such as metabolic or protein interaction networks, and identify critical proteins or genes as potential drug targets. Critical nodes can also be used in network resilience and robustness infrastructure under various networks, including power grids [48], and transportation networks [49] to help improve resilience by safeguarding against failures or targeted attacks. By identifying critical nodes and solving this problem, various domains ensure efficiency and minimize many important risks.

3.5 Related Works

Various algorithms have been proposed to address the CNP, these methods are classified as exact, heuristic, or hybrid:

Researchers propose solutions to identifying critical nodes in many networks, such as wireless sensor [50], ad hoc [51], planar complex [52] and undirected weighted networks [53]. Researchers focus on heuristic methods such as [53], [54], [55], [56], they aim to provide scalable solutions for many applications, including large or sparse graphs.

- Exact algorithms, such as [57] and [58] provide fantastic solutions, making it suitable for small or medium-sized networks, while simultaneously giving a clear and system-

atic method with theoretical guarantees. However, it is computationally demanding and therefore unsuitable for large networks.

- Heuristic approaches, like [36], [43],[53], [54], [55], [56]. As massive graphs grow, it ensures scalability and efficiency by requiring fewer resources, and it is suitable for many real-world applications on the following types of networks: planar, sparse, and weighted, but it gives an approximate solution.

- Hybrid approaches, including [59], [60] combine the strengths of accurate and heuristic methods to improve solution quality. In addition, it can handle larger graphs than completely accurate methods. However, it requires careful integration of many techniques, which increases computational time, and solutions may need precision of accurate methods.
  The following table shows a comparative analysis of CNP algorithms.

Table 3.1: A comparative Analysis of Critical Nodes Problem Algorithms

Works	Type	Limitations	Benefits
[57]	Exact algorithm	High computational complexity for large networks	Provides optimal solutions
[58]	Exact algorithm	Computationally expensive for large networks	Guarantees an optimal solution
[36]	Heuristic algorithm	May not provide optimal results in all cases	Efficient for sparse graphs, faster than exact methods.
[43]	Heuristic algorithm	Assumes specific network dynamics (complex contagions)	Effective in social network contagion scenarios
[53]	Hybrid algorithm	Complexity increases with network size and weight factors	Flexible and scalable to weighted undirected networks
[54]	Heuristic algorithm	May not always find the most critical nodes	Well-suited for real-world sparse graphs
[55]	Heuristic algorithm	Approximation of critical nodes, may miss some key nodes	Faster for large-scale networks, good for real-world scenarios
[56]	Heuristic algorithm	May be suboptimal in highly dynamic networks	Useful for infrastructure resilience analysis
[59]	Hybrid algorithm	The accuracy may vary based on the construction method	Balances speed and solution quality
[60]	Hybrid algorithm	May struggle with scalability for very large networks	Effective in large-scale networks

3.6 Discussion

Critical nodes in graph partitioning are crucial for dividing a graph into smaller subgraphs. Keeping or removing these nodes might have a substantial impact on the connectivity of graph.

Partitioning large networks into smaller communities or subgraphs allows for more efficient analysis, better scalability, and improved performance in various applications, including community detection and source location. In addition, it facilitates the partitioning process. These nodes, often considered "cut vertices", have unique positions within the graph where their removal leads to the disconnection of connectivity between subcomponents. As the subject of minimizing the size of subgraphs, ensuring efficient graph traversal, and improving graph robustness. In this way, critical nodes are therefore the connectors that control the flow of data or information propagating throughout the graph, and they serve as the keys to partitioning.

There are several methods for identifying critical nodes in the context of graph partitioning, depending on the type of network and the domain of the problems, including community detection methods, like modularity maximization and LPA, can be used to find clusters within the graph. Communities may merge or become detached if nodes on their boundaries are removed. Critical nodes in community detection are important in network analysis to find groups of nodes that are more densely connected than those outside the group. Critical nodes can also serve as boundary nodes.

Critical nodes in social networks serve as intermediaries between various groups. Identifying it as key influencers can aid in understanding information flow between communities and developing ways to prevent it.

By splitting the graph based on essential nodes, we may reduce the complexity of community discovery, making it more manageable. Instead of analysing the entire graph, smaller subgraphs formed by cutting along critical nodes can be analysed independently, which improves scalability.

Critical nodes are also essential for locating the source of information in a network, especially when trying to track the origin of rumours or diseases, placing observation nodes as critical nodes can increase the likelihood of accurately identifying the source. Since these nodes lie in crucial positions, their observation may yield more reliable insights into where the information originated and how it spread.

The identification of critical nodes is a key step in community detection and source location, providing insights into the structure of the graph and the most important nodes of the network. These nodes enable efficient partitioning by ensuring that the graph is split in a way that minimizes connectivity between subgraphs. Furthermore, they serve

an important role in detecting communities within networks and locating the source of information.

Additionally, critical nodes are important to optimize the performance of WSNs, using these nodes in WSNs can improve energy efficiency, connectivity, and overall network performance. Critical nodes detection is an important method in the design and operation of WSNs, especially in large-scale and real-time networks.

3.7 Conclusion

In this chapter, we have explored the critical nodes problem and its important role in graph partitioning. We define the critical nodes problem and explain its different variants. In addition, we present clarity on various applications in different fields. Additionally, this chapter also discusses approaches to solve the critical nodes problem. This makes it easy to understand the difficulties in effectively partitioning graphs and identifying significant nodes in these systems. Moreover, we draw attention to the importance of this problem in the way of comprehension of the structure and dynamics of large, complex networks in many domains.

Chapter 4

Community Detection

4.1 Introduction

In many different application fields, community detection is an essential task in the field of network analysis. Finding a group of nodes in a network that are more closely related to one another than to the rest of the graph is the goal of community detection. These groups, which are represented as communities, reflect the basic structure of the network and are crucial for understanding how system components interact.

In this chapter, we will define the concept of community detection and its numerous applications. Furthermore, we will examine several approaches and strategies for identifying communities in graphs and describe how they may be used to find significant patterns in large, complex networks.

4.2 Definition and Characteristics of Community Detection

4.2.1 Definition

Community detection aims to find groups of nodes in a graph that are more closely related to each other than to the rest of the network. [61].

In general groups of nodes within a network are known as communities. These communities have a modular structure in which some groups of nodes are strongly related. This means that they share similar interests, characteristics, and behaviours, such as social networks, biological networks, and wireless networks [62].

The Figure below 4.1 shows a community detection problem, which divides a graph into three communities (coloured in different colours).

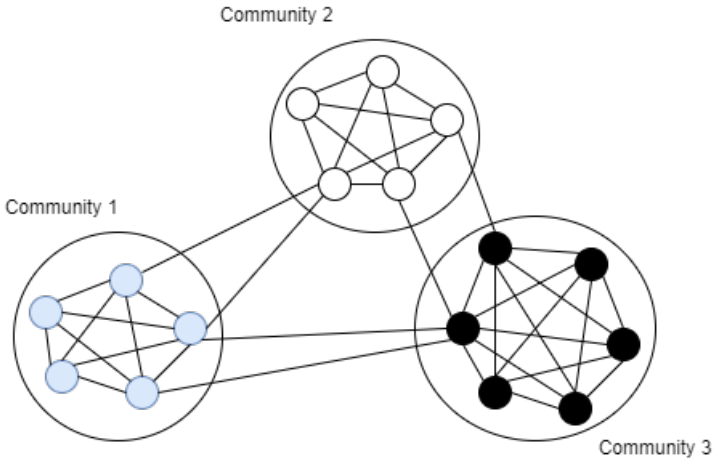


Figure 4.1: Community Detection

4.2.2 Characteristics of Communities

Nodes inside a detected community are densely connected subgraphs, and the edges between nodes within the same community should be longer than the edges connecting nodes to other communities. In addition, edges connecting nodes within a community to those outsides should be sparse. There are fewer edges than those within the community. Moreover, in certain networks, nodes can belong to many communities, whereas in others, communities are completely disjoint, meaning that each node only belongs to one [63].

4.2.3 Applications

Community detection has several applications across diverse fields because of its ability to identify hidden structures inside networks [64], including social network analysis, where community detection is widely used to identify groups of individuals with similar interests. Nodes are the users, and edges represent relationships. Like Facebook, community detection aims to help understand user behaviours, trends, and social influence patterns that can be used in viral marketing to create targeted campaigns for certain audiences and anticipate the flow of information to key influencers who can change opinions or behaviours within a group [62].

Another important application of community detection is communication networks, to find communities such as in the routing problem, where nodes represent people and links represent mobile communications. The identification of these communities can help optimize routing and resource allocation within networks [65]. It can also be used in biological networks, such as PPINs, to identify functional modules within each group [66].

Another interesting application of community detection is information and epidemic

spread. In order to understand how information spreads through networks, communities help to identify the origin source of information by isolating the spread within specific clusters [67]. It helps optimize complex technical systems like phone and transportation networks [62].

Community detection can be used in academic and citation networks. It aids in the identification of research trends and cooperation within scholarly networks. Like research areas or fields. Analysing co-authorship or citation networks exposes groups of researchers who operate in comparable subjects. Nodes represent research communities, and two communities share an edge if researchers travel between them [68].

Community detection is essential in improving user experience and functionality on digital platforms, like detecting user communities, helping platforms such as Amazon recommend relevant products. Authors in [69] proposed an e-commerce risky community detection platform to detect risky groups containing identified fraudsters and other closely related users.

4.3 Obstacles of Community Detection

Due to network complexity and diversity, community detection involves a variety of obstacles, including the dynamic change of networks over time, the need for algorithms that can adapt to changes, like in [70]. The measurements, such as modularity or normalized mutual information (NMI) have limitations and may not always reflect the quality of communities. Thus, [71] proposed methods for evaluating the quality of community detection algorithms using ground-truth data. This requires comparing algorithm-discovered communities to known ground-truth communities [71]. In addition, managing a large network can still become slow [72]. Additionally, real-world networks have nodes belonging to multiple communities, traditional algorithms generate disjoint communities [73]. However, fuzzy clustering is being developed to accommodate overlaps [74].

4.3.1 Evaluation Metrics for Community Detection

Evaluation metrics are commonly used to ensure the effectiveness of community detection [75]:

- **Modularity:** Measures how well a network is divided into communities. A high modularity indicates a strong community structure with nodes that are closely linked.

$$Q = \frac{1}{2m} \sum_{(} i, j)[A_{ij} - \frac{k_i k_j}{2m}]\Delta(ci, cj) \tag{4.3.1}$$

In this equation, A represents the adjacency matrix, m represents the total number of edges, k_i represents the degree of vertex i, and c_i represents the vertex i community.

- **Normalized Mutual Information (NMI):** Measures the similarity between the detected community structure and ground truth. Higher NMI evaluations mean a greater similarity to real communities.

$$NMI(p_1, p_2) = 2I(p_1, p_2)/H(p_1) + H(p_2) \tag{4.3.2}$$

$I(p_1, p_2)$ represents the mutual information of the two partitions p_1 and p_2, and $H(p_1)$ is the entropy of p_1.

4.4 Related Works

In the literature, community detection can be classified into:

1. **Clustering-Based Algorithms:**

   Use traditional clustering methods such as k-means to split graphs based on node similarity or distance. However, the number of clusters needs to be determined.
   Hierarchical Clustering: creates a hierarchy of communities. It can be divisive (top-down) or agglomerative (bottom-up). In [76] they based this method on computing the similarity between nodes or edges. The primary approach to network partitioning is clustering using hierarchical methods like [77] combines K-means clustering with node centrality to detect communities based on density and degree centrality, identifying core nodes and expanding communities [78].

2. **Modularity-Based Algorithms:**

   These algorithms aim to optimize the modularity measure, like the well-known Louvain Method, which ensures fast optimization for modularity of the network, and Newman-Girvan used hierarchical clustering with edge betweenness to iteratively remove edges and optimize the modularity measure of the resulting partitions [79], [80].

3. **Spectral Clustering:**

   This method uses eigenvectors and eigenvectors of the graph Laplacian to partition the graph and maximize modularity [81], [82], [72].

4. **Label Propagation Algorithms (LPA):**

   These algorithms assign labels to nodes and propagate them throughout the network until a stable community structure is achieved, like in [83] , LPA, with an emphasis on overlapping community detection. It adapts the traditional label propagation method to allow nodes to belong to multiple communities based on their importance, such as [84], it modifies the standard LPA by incorporating node influence as a key factor for community detection, and others [85], [86].

5. **Optimization-Based Algorithms:**

   Community detection as an optimization problem, e.g., minimizing conductance or maximizing density, like in [87], they used evolutionary algorithms, a class of optimization algorithms, to detect communities in social networks by focusing on node centrality for community identification, and others in [88], [89].

6. **Probabilistic and Statistical Methods:**

   These algorithms use a probabilistic process such as stochastic block models to partition the graph, where the probability of an edge existing between two nodes depends on their community membership [70], [90].

7. **Deep Learning-Based Methods:**

   Graph Neural Networks (GNNs):

   These methods use neural networks to learn representations of nodes and edges and can be trained to discover community patterns in graphs.

   Autoencoders for Graphics:

   Develop low-dimensional representations of the graph, which are subsequently clustered into communities [91], [92].

8. **Dynamic Community Detection:**

   Detect communities in evolving networks, that change over time. Most researchers [93], [94] adapt evolutionary community detection approach to adapt to changes in dynamic networks.

9. **Overlapping Community Detection:**

   Overlapping Label Propagation: In networks where communities are not strictly disjoint, an LPA variant that allows nodes to belong to multiple communities.

   Clique Percolation Method (CPM): Identify k-cliques to detect overlapping communities by evaluating crossings, as in [81], [72].

10. **Density-Based Algorithms:**

    Density-Based Spatial Clustering of Applications with Noise (DBSCAN): is a spatial data clustering approach that has been extended to locate communities by identifying and grouping dense clusters of nodes.

    Density Peak-Based Methods: The goal is to locate community centers depending on node density and grow the community to surrounding nodes with comparable interests. [95], [96].

11. **Centrality-Based Algorithms:**

    Node Centrality-based Methods: Community detection based on node centrality (e.g., degree centrality, betweenness centrality, or closeness centrality) helps to identify influential nodes that can define the boundaries of communities.

    Edge Centrality-based Methods: Similar to node centrality, but focusing on edges to identify important connections that indicate community boundaries, like [97] uses edge centrality to detect communities, where edges connecting more central nodes are prioritized in community formation, such as [98] , uses edge degree betweenness centrality to perform a divisive approach to community detection, focusing on the importance of edges for splitting the network, as [99] integrates influential node detection within the process of community detection, utilizing centrality measures to identify key nodes in communities[100], detects communities by focusing on rank centrality, a measure of node importance, to guide the community identification process, [101] detects local communities by considering nodes with greater centrality within a specific neighbourhood, [102] Centrality-based and optimization algorithm. It combines node importance and optimization to partition the network and identify communities, like in [87], [101].

12. **Hybrid Algorithms:**

    Combination of Clustering and Optimization: Combine clustering approaches with optimization techniques (such as genetic algorithms or simulated annealing) to improve the quality of community detection.

    Multi-objective Optimization: Combine multiple criteria, such as modularity, node centrality, edge density, and uses optimization techniques to identify communities like [103]

    Hybrid optimization and centrality-based algorithm. Combine local community detection with node importance rankings to identify communities effectively in large social networks, [104]

The Table below shows the comparison of many algorithms, according to methodology, advantage, limitations.

Table 4.1: A Comparative Analysis for Community Detection Algorithm

Works	Type	Benefits	Limitations
[87]	Optimization-based algorithm.	Utilizes evolutionary strategies to efficiently find communities based on central nodes, offering scalability to medium-sized networks.	High computational cost for large-scale networks; results depend on parameter tuning in evolutionary algorithms
[83]	LPA with overlapping detection.	Efficient for overlapping communities; computationally inexpensive for sparse graphs.	Relies on initial node labeling, which may cause inconsistent results; unsuitable for very large graphs.
[97]	Centrality-based algorithm.	Focuses on local edge centrality, which simplifies computations for detecting localized communities.	Struggles with detecting global communities; local optimization may overlook network-wide structures.
[103]	Hybrid of centrality and optimization.	Effective for localized community detection; adapts well to high-degree nodes.	Limited scalability; effectiveness decreases in sparse graphs.
[98]	Centrality-based, divisive clustering.	Precise community boundaries due to edge-level analysis; useful for dense graphs.	High complexity makes it unsuitable for large-scale networks.
[77]	Clustering-based algorithm and centrality.	Combines centrality with density-based clustering for better granularity; applicable to small networks.	Performance is highly dependent on initial K-means centroids; fails in sparse networks.
[105]	Hybrid (centrality and optimization).	Balances global and local perspectives; better at handling imbalanced community structures.	Sensitive to parameter tuning; increased computation time for dense networks.
[84]	LPA.	Fast and scalable for overlapping communities; simple implementation.	Randomness in label updates may lead to varying results; less effective for dense networks.
[102]	Centrality and optimization.	Combines node importance with modularity optimization, leading to meaningful partitions.	Limited adaptability for dynamic networks; computational overhead.
[104]	Multi-objective optimization.	Considers multiple criteria (e.g., modularity, importance); effective for attributed networks.	High computational cost; requires careful parameter configuration.
[78]	Clustering-based algorithm.	Core-node expansion simplifies detecting robust communities.	Ineffective for overlapping communities; heavily influenced by initial core node selection.
[106]	Probabilistic statistical model-based algorithm.	Captures both hierarchical and overlapping communities; provides statistical insight.	Computationally intensive for large networks; requires parameter estimation.
[107]	Modularity-based algorithm.	Scalable and efficient for large-scale networks; high modularity optimization.	Limited to disjoint communities; modularity optimization may fail

Girvan and Newman proposed the topic of community detection in 2002 [108]. It uses the concept of edge betweenness to break the network into smaller networks.

Recent research has focused on clustering approaches [109], multi-objective evolutionary [110], label propagation [111], and local modularity density [112]. In contrast, some have demonstrated interest in identifying the major nodes and using them as first communities [103], [113],

In [103] , the authors emphasize the importance of selecting high-importance nodes to expand initial communities through local similarity. These nodes are then merged to establish final community structures, similar to [78] , but with a focus on local similarity between nodes.

Another study [114] found that the core node's cliques play a role in community building. In [115] , the authors suggested a two-stage approach for detecting local communities, utilizing core detection and community extension. Others [113] employed local degree central nodes and the Jaccard coefficient to find the seed nodes, while calculating the internal force using the fitness function between nodes to expand the community.

Others [105] computed node influence using the k-shell decomposition algorithm and k-shell entropy, using initial communities consisting of seeds and neighbours. They also proposed the concept of belonging as a way to quantify local information, and they integrate every two communities. Another study [116] calculated node influence using node convergence degree and combined network topology with node properties.

As we have observed in earlier studies, some nodes in networks are more essential than others depending on the dispersion of information. Centrality can be used to assess the significance of these nodes. Many researchers employed the core nodes to detect communities, as in [117], [118], [119]. In [118], [119] They prioritized nodes based on their eigenvector centrality rating and then identified the communities for each leader. [118] To construct a community, similar nodes are assigned to the discovered leader node based on a threshold k. In [119], neighbours are assigned depending on their role in building communities.

In another study [120], they look at leadership and attractive force, and each node connects to the local leader in its neighbourhood, producing a directed dependence relationship. Once all dependencies are resolved. All of these dependent trees form a forest. After merging tiny branches, each tree in the forest became a community, with the root node serving as the leader of the related cluster.

In [121], the key nodes are selected by a ranking function; removing these nodes disconnects the graph as much as feasible. Also in [103], they identify significant nodes, then build communities around the core nodes using a similarity measure and each node's strength, and then merge weak communities using a proposed relation and maximum modularity.

4.5 Conclusion

Community detection is an important task for understanding dynamics and complex networks. It finds communities that are closely connected inside a network, offering important information about the underlying connections and roles. We presented in this chapter an overview of the community detection method, its definition, characteristics, issues, and application. We also defined the different metrics used to evaluate its efficacy.
In this chapter, I also defined different classes of algorithms used in community detection. I discussed how many researchers have addressed the problem of community detection efficiently. However, none of these studies employ the notion of critical nodes to detect communities. While some researchers have used the centrality measures to identify influential nodes within networks, these approaches do not directly leverage the concept of critical nodes as central nodes in community identification.

Chapter 5

Source Detection

5.1 Introduction

One of the fundamental issues that comes up in many different fields, especially in social and communication networks, is determining the source of information within a network. Applications such as viral marketing, illness modelling, and locating significant people or system nodes in these networks depend on a knowledge of how information or influence spreads. The source detection problem has been the most important subject for many researchers; it aims to enhance network management, prediction, and also optimization.

This chapter will explain the source detection problem and explore the key role of observation nodes in enhancing the identification of the information sources. These observation nodes help in better tracking the dissemination of information and enable more accurate source identification because of their thoughtful positioning within the network. We will also discuss the applications of source detection. In addition, we will explore the importance of critical observation nodes and introduce Gaussian estimation as a technique to enhance the accuracy of source detection.

5.2 Definition Problem

Given a network $G = (V, E)$, where V represents the set of nodes and E represents the set of edges, locating the source of information in a network aims to determine the source node $s \in V$ that begins the spread of the diffusion process. [122]. It is considered a task of finding the original spreader or influencer in a network whose actions or characteristics result in the observed impacts on the network's structure or behaviour. such as influence maximization, epidemic spread, and information diffusion.

The following Figure 5.1, represents an illustration of the problem, $S*$ is the source, and O is the set of observation nodes.

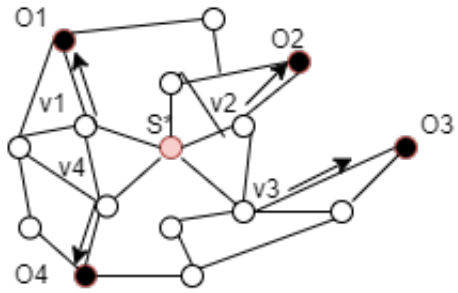


Figure 5.1: Source Detection Problem$S*$

5.3 Conditions

In the following subsections we explain the conditions to locate the source of information in a network.

5.3.1 Model of diffusion

Information spreads through the network following a predefined diffusion model, including [122]:

1. **SI Model (Susceptible-Infected):** One of the simplest diffusion models. Where each node in the network is susceptible (S) or infected (I). Once a node becomes infected, it remains infected forever and can spread the infection to its neighbours.

   - **Susceptible (S):** The node can be infected.
   - **Infectious (I):** The node is infected and spreads the infection to other nodes.

   At each time step, an infectious node has a probability of infecting other susceptible neighbour. The process continues until all nodes are infected.
   This model of diffusion has a large area in the real world such as the spread of a virus in the computer, and also early stages of epidemics. The main problem of this model is that it does not account for recovery.

2. **SIR Model (Susceptible-Infected-Recovered):** Extends the SI model by adding recovered (R), where nodes cannot pass the infection to neighbours.

   - **Susceptible (S):** The node can be infected.

- **Infectious (I):** The node is infected and can spread the infection to other nodes.
- **Recovered (R):** The node has recovered and is immune to further infection.

An infectious node u infects each susceptible neighbour v with probability p. After time t_i with probability p, the infectious node recovers and becomes recovered. The process stops when no infectious nodes remain.
As an example of this model, the COVID-19 epidemic. The main problem of this model is the increase of computational complexity due to tracking all the states.

3. **SIS Model (Susceptible-Infected- Susceptible):** Such as SIR model, the infected node (I), can be susceptible (S), but in this model, when a susceptible node becomes infected, they cannot be susceptible after some period p.

   - **Susceptible (S):** Node can be infected.
   - **Infectious (I):** Spreading the infection.
   - **Susceptible (S):** Node can be infected.

4. **SIRS (Susceptible-Infected-Recovered-Susceptible) :** Model, a recovered node can be a susceptible node with a probability p.

   - **Susceptible (S):** Node can be infected.
   - **Infectious (I):** Spreading the infection.
   - **Recovered (R):** Recovered and immune.
   - **Susceptible (S):** Nodes recover after or with probability p.

   Such as the disease of COVID-19. The main problem is the increase of computational complexity due to the requirement of many parameters,

5. **Independent Cascade Model:** Used to model the flow of information in the network, where a node has two states:

   - **Active (A):** A node who received the information and became infected
   - **Inactive (I):** A node has not adopted the information yet.

   The active node has a single chance to activate each inactive neighbour with probability p. If successful, it becomes active in t_i. The process stops when no further activations occur.

   This model is applicable in viral marketing, product adoption, and social media. The main problem is the well-chosen activation parameter probabilities.

Each model is suited for a specific application, it depends on the nature of the spread, network structure and available data.

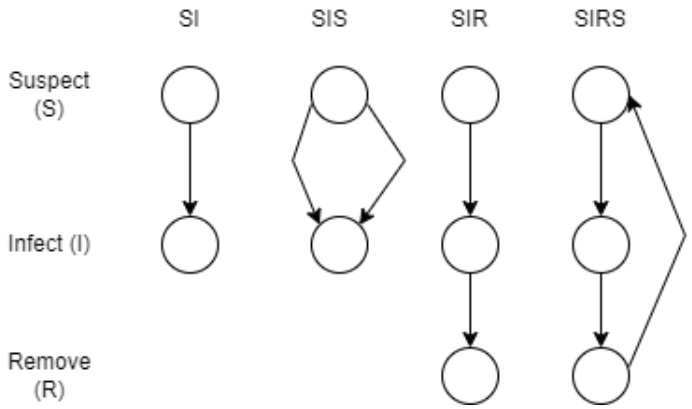


Figure 5.2: Models of diffusion

5.3.2 Observation Nodes

5.3.2.1 Definition

Observation nodes are strategically placed or selected nodes in a network used to monitor the spread of information. These nodes act as sensors, collecting data such as infection time, which is critical for detecting the source of diffusion [123].

5.3.2.2 Role of Observation Nodes

Observation nodes are a subset of nodes in a network, their role is to monitor and record the propagation data to detect the source of information. Observation nodes act as sensors within the network [124] gathering information about the nodes, such as their state during propagation, the time of infection, and direction of information passing through each node. Furthermore, these nodes can facilitate the identification of the source by providing the data required, like analysing the time at which observation nodes become infected. In addition, well-chosen observation nodes can improve the accuracy and computational efficiency of source detection, ensuring scalability (to focus on smaller, critical parts of the network), robustness (to minimize the impact of noise) and costs [125].

The placement of observation nodes plays an important role in diffusion processes, including strategies like high centrality nodes or critical nodes, boundary nodes, and random placement. Additionally, these nodes are crucial for tracking diffusion in real-time, enabling immediate intervention in dynamic changes of structure [126], such as in epidemic networks, monitor the spread of infectious diseases in real-time to identify patient zero, in cybersecurity to track malicious activity in real-time to detect the attack's origin, and in social media to identify the spread of rumours.

Observation nodes are indispensable in locating the source of information, like using a minimal number of nodes to reduce costs. limited data can reduce the accuracy, so great coverage is necessary, especially in large-scale and complex networks. By strategically selecting and monitoring a subset of nodes, they provide the critical data required to infer the source efficiently and accurately.

5.3.3 Network Topology

The structure of the network should affect the propagation of information [127].

5.4 Source of Information in a Network G

5.4.1 Definition

The source of information S refers to the specific node in a network $G(V, E)$, where V represents the set of nodes, and E is the set of edges that initiates the spread of information. Identifying this source node is critical for understanding the origin and dynamics of propagation processes [128].

The source node s from which the diffusion process begins in the network topology significantly affects the spreads reach and dynamics, it initiates the spread at time t_0. Observations of infected nodes at later times $t > t_0$ provide to locate s.

5.4.2 Formal Definition

Given a graph $G = (V, E)$, where V is the set of nodes and E is the set of edges between nodes. The source of information is the node $S \in V$ from which information or influence begins to propagate through the network, according to a diffusion process.

Let the information spread over time based on a diffusion model M, which defines the rules governing the propagation of information across the edges of the graph.

Let an observation node $O \subseteq V$, it records data when it receives the information.

Given the observed diffusion data $D(v_i, t_i | v_i \in O)$ where t_i is the time step at which observation node v_i receives the information, the objective is to identify the source node S based on the observed diffusion pattern.

5.4.3 Challenges in Identifying the Source Node s

Identifying the source of information in a network is a complex problem due to various challenges, including [122]:

- Most researchers considered a tree as a network structure for source detection, with no cycle. However, the source can receive the information from the same nodes.
- Locating the source is easier in a tree using Breadth First Search (BFS) algorithm where the graph is best suitable topology to represent the real-world network.
- Source detection in real-time is important to well control the spread of information and minimizing their negative impact.
- In source detection, real-world networks must be gathered in real-time, making it computationally more expensive.
- Most networks are dynamic, like social networks that change over time, it is important to choose an adaptive method while examining the network and selecting the suitable diffusion model.
- The variation of edge weights and node types, also hierarchical structures, can lead to a difficult analysis. Therefore, to improve the accuracy, it should take into consideration the heterogeneous infection probability.
- Many researchers assume that there is only one source to detect. However, in many scenarios, they consider multiple sources of information that initiate the diffusion. Which requires more efficient algorithms. There is scope for wide research to design a single model to detect source(s) irrespective of their number.
- In the case of identifying multiple sources, few researches based on community detection methods, where the partition of multiple sources forms their own community.
- Challenge, especially in asynchronous and stochastic diffusion processes in online social networks, to improve the adaptability and accuracy of source locating methods [129].
- Observing all nodes in the network in real-time is usually impossible and expensive [129].
- The well-chosen set of nodes as observers can affect the accuracy of source locating methods [129].

5.5 Algorithms for Identifying the Source Node s

Various algorithms are developed to estimate the origin of information dissemination, utilizing various approaches to identify the source with great accuracy and efficiency. The following Table 5.1 provides information on these algorithms:

Table 5.1: Algorithms for Source Detection

Algorithms	Types	References
Graph-based Algorithms	Reverse Propagation Algorithms	[130], [131]
	Likelihood-based Methods	[132], [133]
	Minimum Cut/Maximum Flow Algorithms	[134] [135]
	Centrality-based Algorithms	[136], [137]
Epidemic-based Algorithms	Susceptible-Infected-Recovered (SIR) Model	[138], [139]
	Linear Threshold Model (LTM)	[140], [141]
	Independent Cascade Model (ICM)	
Optimization-based Algorithms	Maximum Likelihood Estimation (MLE)	[142], [143]
	Integer Linear Programming (ILP)	[144], [145]
Machine Learning Algorithms	Supervised Learning Algorithms	[146], [147]
	Unsupervised Learning Algorithms	[148], [149]
	Reinforcement Learning	[150], [151]
Hybrid Algorithms	Combine different approaches :	[152], [153]
	Graph-based	
	Epidemic-based	
	Optimization-based	
	Machine learning	
Probabilistic Algorithms	Bayesian Inference	[154], [155]
	Markov Chain Monte Carlo (MCMC)	[156]

In this work, we used graph-based methods for source detection. These algorithms investigate graph features and the ways in which information flows across the network in order to determine the source of information.

Graph-based algorithms for source detection offer a strong collection of tools for determining the origin of information in a network based on network topology, diffusion process, and flow characteristics. Table 5.2 shows the different graph-based algorithms to solve the source detection problem.

Table 5.2: Graph-based algorithms for source detection

Algorithms	Concept	Type	Ref
Reverse Propagation	Simulating the reverse diffusion process 1. Identify the observation nodes 2. Tracing the spread of information	Breadth-First Search (BFS) is modified (traverse) backward, from observation nodes, until the source is found . Depth-First Search (DFS): LIKE BFS but will explore deeper paths first, applicable in hierarchical structure.	[130], [131], [157]
Likelihood-Based	Calculating the likelihood of nodes based on the observations and known diffusion patterns. The diffusion follows a predictable pattern. Identify which node as source, Maximizes the likelihood of the observed .	For each source node, compute the likelihood of the observed diffusion pattern based on known information. Assuming information spreads to specific diffusion model, it compares the observed spread pattern to the predicted The node that maximizes the function is most probable source.	[132], [158]
Centrality-Based	Assume that the source of information is likely a highly central node in network.	For each node in the network calculate centrality measures and rank them The node with the highest centrality value is considered the most likely source	[136], [159],
Min Cut/Max Flow	used to partition a network into two subsets such that the cut is minimized. Information spreads from a single source node and eventually reaches observation nodes. Identify the minimal set of nodes and edges whose removal would disconnect the network	Model the network as directed flow network The maximum flow between a source and a sink represents the strongest connection in the network. Minimum Cut Theorem: states that the maximum flow in a network is equal to the capacity of the minimum cut separating the source from the sink.	[134] [135],
Markov Chain-based	model the diffusion process as stochastic Each nodes state depends on previous state. The process is memoryless the current state only. The state of each node changes over time	Simulate random walks starting from each node to model the spread of information. Track the progression of the diffusion process and compare it to the observed spread	[160], [161]

5.6 Applications of Identifying the Source Node s

Identifying the source of information in a network is critical in various domains [122], including epidemiology to identify the patient zero or the origin of an infectious disease outbreak, like COVID-19. Another important application of source detection is cybersecurity to detect the origin of malicious activity in computer networks. Also used in social networks to locate the origin of fake news, rumours. It can be used in large networks such as power networks to identify the starting point of power grid failures or blackouts, in transportation networks to trace disruptions, such as train or air, in distribution networks to pinpoint the origin of water contamination.

Another interesting application of source detection is financial networks to detect the origin of fraudulent or illegal transactions. It can be used also in biological and ecological networks [162] to locate the essential PPINs.

Source detection can be applied in academic and knowledge networks [163] to detect plagiarism in academic documents and also used in criminal networks [164] to locate the origin of illegal activities in covert networks, like identifying the leader or key actor in drug trafficking.

5.7 Using Critical Nodes as Observation Nodes for Source Detection

Critical Nodes Problem aims to identify a set of nodes within a network; if we remove or keep them, it can impact the graph 's structure. In source detection, these nodes play an important role. By observing the information flow in networks, it can significantly enhance the source detection accuracy because these nodes provide data from key locations in the network, enabling better triangulation of the source.

Critical nodes methods are considered as cut vertices methods in graph partitioning, are influential nodes for monitoring the diffusion of information. Additionally, strategically placing observation nodes at critical nodes ensures wide coverage of the network. Moreover, to effectively achieve good monitoring and reduce the usage of resources, it is better to minimize the number of observation nodes that influence large portions of the network.

Observing them helps to understand the dynamics of information spread. Trace back the source with greater accuracy due to the high-quality data collected. We can also apply it in many real-world scenarios, such as epidemiology, critical nodes can be hospitals. In cybersecurity, critical nodes might represent routers.

5.8 Gaussian Estimation for Source Detection and Accuracy Enhancement

Gaussian estimation is a statistical strategy for improving accuracy in source detection problems. This strategy is especially beneficial when there is uncertainty in the data or when attempting to estimate parameters based on incomplete or noisy data. Gaussian estimation has an important role to enhance the source detection accuracy, such as denoising the data by assuming that the data follows a Gaussian distribution. It also minimizes ambiguity, making data points more typical of the underlying propagation pattern. Also, examine the network's topological structure. Changing the Gaussian model to consider network variables including node centrality, community structure, and edge weights. Adaptive Gaussian models dynamic networks with evolving diffusion processes [165] alter parameters (mean and variance) to monitor sources more accurately as the network evolves.

5.9 Related Works

Identifying sources and anomalies is crucial for controlling information flow in networks, which can be represented as graphs. Graph-based algorithms can be used to detect the source of information. Social networks are an effective tool for transmitting information between people, resulting in numerous interactions that must be managed to ensure accurate and dependable communication [166].

Several research studies have examined source detection using various diffusion models. The SI model is widely used in studies due to its simplicity [167], [168], [169]. In this study, we employed the observation nodes approach as described in [169], [143], [123], [170], [171].

Several studies use graph-based centrality measures, which consider central nodes as observation nodes. In [168], a new propagation centrality algorithm was introduced. In [169], the nodes with the highest and lowest degrees of centrality were combined as observers. In [167], it was stated that centrality metrics alone are not accurate estimators of the source. Therefore, eccentricity and proximity measures were combined to improve findings. In [143], the network was partitioned, with observation nodes chosen at random and based on centrality. In [172], nodes with the highest betweenness centrality score were selected as observers.

Other techniques, such as [173], discover various sources of information by identifying recovered and unobserved infected nodes and partitioning the network based on the source node with the highest likelihood estimation in infected clusters. In [174], the SIS model was utilized to create an estimator based purely on observed infected nodes.

The SIR model was employed in [175]. In [175], a minimal-radius collection of nodes, known as Jordan Cover, was chosen to cover all observed diseased nodes. This research assumes there is just one source to detect.

The main question is how to efficiently choose observers and estimate the source's localization using the information they gather.

Various methods are employed to identify the source of information, such as Gaussian estimates, shared messages, and distance-based approaches [143]. The Gaussian approach is vital for source localization as it models measurement uncertainty and allows for fast source detection estimation using observers. This research employed the same strategy as [123].

5.10 Conclusion

In this chapter, we define the problem of source detection, which is crucial in many fields, and identify several models of information diffusion. After that, we discussed the role of observation nodes, and the importance of critical nodes was emphasized, showing how these nodes can be influential in detecting the source with high accuracy. We also explored the different graph-based algorithms employed for source detection. We also explained the importance of applying the Gaussian model, which reflects the inherent uncertainty in the diffusion process, to identify the source of information and increase detection accuracy.

Chapter 6

Impact of CNDP on Wireless Sensor Networks (WSNs)

6.1 Introduction

Wireless Sensor Networks (WSNs) known as transformational technology, aim to gather, transmit, and analyse data in real-time from remote or inaccessible locations. It consists of several sensor nodes, each with its own sensing and processing, and they can communicate with each other, which can also provide solutions for a wide range of applications.

The significance of WSNs is based on their architecture, energy efficiency, dependability, and scalability. One of the most difficult concepts is ensuring that the network can manage diverse and dynamic systems while maintaining a balance of performance and resource capacity limits, such as energy consumption. Thus, there are numerous proposed approaches and methods used to improve the performance of these networks.

In this chapter, we will look at how critical nodes play an important role in WSNs performance and optimization. Critical nodes, such as gateways, cluster heads, or nodes with high connection, are essential for guaranteeing effective data transfer, network robustness, and energy efficiency. Furthermore, integrating graph theory into WSNs is regarded as a valuable tool for modelling and analysing network structure. WSNs may be tuned for energy economy, reliability, and scalability using graph-based approaches, resulting in improved performance in real-world applications.

This chapter will provide an in-depth understanding of the architecture, definition, characteristics, applications, and challenges of WSNs. We will give an overview of the application of graph theory in WSNs with a particular focus on how critical nodes can impact network design and performance. The discussion will also highlight the benefits,

limitations, and ongoing challenges in the effective deployment and management of WSNs in a variety of settings.

6.2 Definition of Wireless Sensor Networks (WSNs)

A Wireless Sensor Network (WSN) is a system composed of spatially distributed sensor nodes wirelessly connected to monitor physical or environmental conditions, such as temperature, pressure, cameras, humidity, and vibration. These nodes transmit the collected data to the base station for further analysis and decision-making [176].

WSNs are characterized by their self-organizing, computing capabilities, energy constraints, adaptability, and multi-hop communications, making them suitable for various applications, including environmental monitoring, health, military, and industrial automation [177].

6.3 Applications of WSNs

Wireless Sensor Networks (WSNs) are flexible systems utilized in a variety of applications, including environmental monitoring, such as in [178], agriculture [179], healthcare [180], smart cities [181], industrial automation [182] , security and surveillance [183] military and defence [184], home automation [185], transportation and logistics [186], energy management [187], underwater [188], structural health monitoring [180] disaster management [189].
These applications highlight WSNs adaptability and usefulness in developing technology and enhancing quality of life in many ranges of domains.

6.4 Challenges in WSNs

Wireless Sensor Networks (WSNs) provide distinct challenges due to design limits, operating circumstances, and application-specific requirements. These difficulties can be categorized into many groups [190]:

1. **Energy Constraints**
   Sensor nodes require significant power during data collection, processing, and transmission. Batteries of limited capacity should be changed after consumption. For this challenge, it is better to design, develop, and implement energy-efficient hardware and software protocols.

2. **Limited Memory and Storage**
   Sensor nodes are constrained in terms of processing power, memory, and storage. In order to build an effective method for security, researchers should limit the code size of the security approach.

3. **Security and Privacy**
   Security in WSNs is necessary for large applications of security threats, where the data should be private, such as in surveillance and building monitoring. The solutions for this challenge, such as authentication and encryption are not enough. The design of new technologies needs maintenance of data security, like mixing the Internet Of Things (IoTs) with WSNs.

4. **Fault Tolerance**
   When a node fails, it can disrupt network operations. The network should be able to not disconnect in case of any fault. In that case, an efficient routing algorithm is applied to change network configurations.

5. **Time Synchronization**
   Many WSNs applications require synchronization over extremely long lifetimes, the absence of synchronization leads to inconsistencies in data collection and processing. As a solution to this challenge, Timing-sync Protocol for Sensor Networks (TPSN). Quality of Service (QoS) Problem: Ensuring reliable data delivery with minimal latency, especially in real-time and critical applications, like health or disaster. The problem is when the network topology changes at times, the solution is prioritization mechanisms for an unbalanced QoS and QoS-aware protocols.

6. **Self-Management:**
   WSNs need to function without the need for human assistance.

7. **Hardware and Software:**
   Cost-effective nodes are essential to sensor networks, and because flash memory is inexpensive, it's frequently used. Nodes' CPU has a major influence on their energy usage and processing capacity. Central Processing Unit (CPU) design flexibility is provided by microcontrollers, microprocessors, and Field-Programmable Gate Arrays (FPGAs); nevertheless, FPGAs have many difficulties in reducing power consumption.

8. **Operating System**
   It should be simpler than general-purpose Operating Systems (OSes), with a straightforward programming style. It should allow developers to concentrate on application logic rather than low-level operations such as scheduling and networking.

9. **MAC Layer Issues**
   Has a direct impact on energy usage in sensor networks. Collisions, control packet overhead, and idle listening are all major sources of energy waste at this layer. Implementing power-saving forward error control is difficult due to its high computing requirements and the impracticality of extended packets.

10. **Architecture**
    Sensor network architecture is critical for integrating functionality and maintaining QoS in WSNs. It should be long-lasting, scalable, and also adaptable in many application domains. It should be addressed through constant developments in hardware, protocols, and algorithms.

11. **Data Collection and Transmission**
    Sensor nodes gather data from the environment, process it, and transmit it to a base station or sink. Careful data collection and transmission are important, especially when redundant samples are collected, as it consumes energy.

12. **Calibration**
    Calibration is the process of adjusting sensor readings to corrected values, but manual calibration is time-consuming and costly due to sensor node failure and random noise.

13. **Deployment**
    Deployment of wireless sensor networks in real-world locations is laborious and depends on the application's demographic. Challenges include energy management, network congestion, and low data yield due to multiple transmission attempts and insufficient information delivery. Locations may require helicopter drops or topology placement.

14. **In-network Processing**
    In-network processing algorithms reduce communication costs by removing redundant sensor information from nodes and avoiding unnecessary forwarding. This technique, inspired by spatial correlation between sensor observations, allows for data aggregation and mining in sensing and monitoring applications.

15. **Decentralized Management**
    Large-scale and energy constraints in WSNs make centralized algorithms ineffective for network management solutions. Sensor nodes collaborate with neighbours for localized decisions, potentially leading to less optimal results but potentially more energy-efficient and reduced management overhead.

16. **Robustness**
    To support lifetime requirements, each node must be designed to be as robust as

possible, allowing the system to tolerate and adapt to individual node failure. System modularity helps in dividing functionality into isolated sub-pieces, allowing for isolation testing. System components should be independent and have narrow interfaces to prevent unexpected interactions. Wireless sensor networks must also be robust to external interference, using multi-channel and spread spectrum radios to avoid congested frequencies and ensure successful deployment.

17. **Data Interpretation and Knowledge Formation**
    Such as handling noisy data and developing inference techniques to ensure user trust and efficient conversion of raw data.

18. **Heterogeneity**
    As a group where nodes are not identical and have different capabilities, such as cluster architecture. It arises when two different WSNs need to communicate, requiring unified communication interfaces.

19. **Communication**
    Sensor networks collect and communicate multimedia information, including snapshot and streaming content, affecting QoS and requiring high bandwidth for transmission.

20. **Secure Localization**
    Sensor networks rely on accurate location information for fault detection, but attackers can manipulate non-secured location information through false signal strengths and replaying signals.

6.5 Applications of Graph Theory in WSNs

1. **Network Representation and Modelling**
    WSNs can be presented as a graph, $G = (V, E)$, where V is the set of vertices or sensor nodes, and E is the set of edges or communication links between nodes.
    Graphs can simplify the visualization of network analysis, such as finding the best modelling of WSNs, which is necessary and has been the topic of many researchers [191], [192].
    There are different types of graphs used to solve WSNs problems, including general graphs (undirected, directed) and others like fuzzy graphs and random graphs.

2. **Critical Node and Edge Identification**
    To identify nodes or edges crucial for network functionality, such as data flow or connectivity. In graph theory using centrality measures: Identifies critical nodes using metrics like betweenness, degree, and closeness centrality. Cut Sets: Locates

edges or nodes whose removal partitions the graph, disrupting network connectivity [193].

3. **Connectivity and Coverage Analysis**
   It ensures that the entire network is connected and that all areas of interest are covered. In graph theory, the concept of connected graphs, determines the minimum set of nodes that can monitor the entire area, optimizing node deployment using dominate set. Additionally, it ensures fault tolerance by maintaining connectivity even after the failure of k nodes (k-Connectivity concept) [194], [195]

4. **Routing Optimization**
   In order to find the most efficient paths for data transmission while minimizing energy consumption. In graph theory, using shortest path algorithms, like Dijkstra's and Bellman-Ford, Minimum Spanning Tree (MST): Reduces redundant communication paths and conserves energy [196], [197].

5. **Clustering and Hierarchical Design**
   To partition the network into smaller, manageable clusters for ensuring scalability and energy efficiency.
   In graph theory, using graph partitioning: Divides the network graph into clusters to minimize intra-cluster distances and balance workload.
   Domination Algorithms: Select cluster heads from a dominating set of nodes to act as communication hubs [198], [199].

6. **Energy-Efficient Network Design**
   To minimize energy consumption while maintaining network functionality. In graph theory using energy-aware
   Spanning Trees: Select edges based on energy costs to conserve power.
   Load Balancing: Distributes data transmission evenly using flow algorithms [200].

7. **Fault Tolerance and Reliability**
   To ensure network resilience against node or link failures. In graph theory, the k-edge connectivity concept is used, it ensures alternative paths are available if links fail.
   Cut Vertices and Bridges: Identifies critical nodes or links whose failure would partition the network [201], [192].

8. **Localization and Positioning**
   To locate sensor nodes for purposes like tracking and monitoring.
   Graph theory Concepts using geometric graphs: To model node connectedness, use edges based on distance.
   Trilateration versus Multilateration: Uses graph distances for node localization. [202].

9. **Sensor Deployment Optimization**
   To discover the best sensor node location for maximum coverage while minimizing expenditures. In Graph Theory with Voronoi Diagrams: Organizes the area into areas depending on proximity to sensor nodes for optimal placement. Set Cover Problem: Ensures that all targets are covered with a minimum number of nodes [203].

The integration of graph theory with WSNs enhances the design, analysis, and performance of these networks. By leveraging graph-based models, researchers can address key challenges in WSNs, such as energy efficiency, routing, fault tolerance, and scalability. However, real-world constraints and dynamic environments necessitate further adaptations of graph-based methods for practical deployment.

6.6 The impact of using CNP in WSN

The use of critical nodes in WSNs can have a significant impact on network performance, reliability, and efficiency. Like in [204], [205], [50], [206], [207], [208], [209], [210], as shown on the following table 6.1:

Table 6.1: Impact of CNP in WSNs

Objective in WSN	Role of CNP	Impact	Example
Enhanced Efficiency	Strategically chosen sensors that are important in data aggregation ,routing ,network control	Centralizing communication tasks, critical nodes reduce redundant data transmission, lowering energy consumption across the network	In a cluster-based WSNs, critical nodes serve as cluster heads, collecting data from neighbouring sensors and transmitting aggregated data to the base station.
Improved Fault Tolerance	Ensure the network stays operational even when nodes fail.	Safeguarding these nodes or ensuring redundancy, the network can tolerate faults without a complete breakdown	In a forest monitoring WSNs, backup nodes can replace critical nodes if they fail due to harsh environmental conditions.
Optimal Energy Usage	balance energy consumption by handling high communication loads while ensuring other nodes conserve power	Prolongs network lifespan by avoiding premature energy depletion in non-critical nodes.	In energy-aware routing protocols like LEACH, cluster heads (critical nodes) optimize energy use by aggregating and compressing data before forwarding it.
Enhanced Data Reliability	Acting as key intermediaries, ensure reliable data routing and minimize communication delays.	Improves the accuracy and timeliness of data delivery, critical for applications like disaster response	In a smart city WSNs, prioritize emergency signals, such as fire alarms, for faster routing to the control center.
Effective Partitioning	reduces complexity and improves manageability	Enhances the scalability of large-scale WSNs deployments	In an industrial monitoring setup table nodes divide a factory to isolate floor WSNs into smaller zones faults and optimize monitoring
Security, Attack Resilience	Protecting critical nodes from attacks ensures secure data transmission prevents network collapse	Increases resistance to malicious activities such as denial-of-service attacks targeting critical nodes.	Encryption protocols can be applied specifically to critical nodes in a military WSNs to safeguard sensitive information

6.7 Comparison

In the literature, researchers who use critical nodes in WSNs based on specific objectives, such as energy efficiency, fault tolerance, and energy consumption, but the method used is different on application, these studies highlight the core of critical nodes in ensuring good performance in WSNs (See Table 6.2 and Table 6.3).

Table 6.2: Algorithms based on CNP in WSNs (1)

ReF	Objective	Benefits	Limitation	Complexity
[204]	develop energy-efficient algorithms for identifying all critical nodes in , WSNs ensuring reliable operation while minimizing energy consumption.	It provides comprehensive CNP, enhancing reliability and fault. tolerance Energy efficiency is prioritized to prolong the network's lifespan	The approach may face challenges in dynamic topologies and can have high computational overhead for large-scale networks	Computational complexity is due to the focus on complete critical node determination.
[50]	leverages geometric relationships between nodes to efficiently identify critical nodes in industrial WSNs	High accuracy, scalability, real -time detection make it well-suited for industrial environments, it reduces energy consumption by focusing on geometry-based computation	It relies on accurate node localization, which can be challenging in industrial environments with interference, The algorithm may struggle with irregular topologies.	reduces computational complexity compared to other approaches but may still require frequent updates in dynamic environments
[205]	Investigate how the removal of the most critical node affects the network's operational lifespan lifespan	Highlights the importance of protecting critical nodes to maximize network lifetime and performance	Focused only on the most critical node, it doesn't provide a comprehensive strategy for multiple critical	Moderate complexity, as it primarily involves evaluating the impact of one node's removal
[207]	Propose a distributed algorithm based on the connected dominating set to identify critical nodes while maintaining connectivity	Distributed detection reduces need for centralized control improving scalability and fault tolerance.	Dependence on the connected dominating set may lead to inefficiencies in dense or dynamic networks.	Lower computational complexity due to the distributed approach but may be higher in dense networks.

Table 6.3: Algorithms based on CNP in WSNs (2)

Ref	Objective	Benefits	Limitation	Complexity
[209]	Use ant colony optimization to balance energy consumption and optimize critical node utilization in WSNs	Enhances energy efficiency and network robustness by balancing load across critical nodes	May require high computation and convergence time particularly for large networks or real-time applications	High complexity due to the nature of ant colony optimization algorithms
[206]	Analyse how the failure of multiple critical nodes affects network reliability and lifespan	Provides insights into the cascading effects of critical node failures, helping in proactive fault management	Focuses on impact analysis rather than solutions to prevent failures, limiting its applicability	Moderate complexity, primarily dependent on the network's size and structure
[208]	Examine how the removal of critical nodes affects latency in WSNs	Helps understand the importance of maintaining critical nodes for low-latency applications	Limited to latency analysis and does not address energy efficiency or node management strategies	Relatively low complexity focusing on latency measurements.
[211]	Propose methods to detect critical nodes in IoT networks to ensure reliable communication	Tailored to IoT networks this approach addresses reliability and scalability across diverse IoT applications	May require additional computation and energy for large-scale IoT deployments	Moderate to high depending on IoT network size and heterogeneity
[210]	Identify key nodes in WSNs to prevent cascading failures that can disrupt the entire network	: Improves fault tolerance and ensures network stability by preventing cascading node failures. node management	Focused on failure analysis, it does not provide energy-efficient or proactive critical	High complexity due to cascade failure modelling
[212]	Study critical nodes in scale-free topologies while considering energy consumption	Provides insights into the unique challenges of of scale-free networks types of networks	Limited to scale-free topologies, reducing applicability to other	Moderate, focused on weighted scale-free network analysis.

6.8 Related Works

By 2023 and 2033, the global industrial automation market is expected to reach 213.53 billion dollars and 529.16 billion dollars, respectively [213]. As a subclass of WSNs, IWSNs contribute significantly to this growth by providing real-time monitoring, control, and optimization solutions for increased operational efficiency and productivity in industrial systems [214]. There are three primary uses for IWSNs [215]: environmental sensing, condition monitoring, and process automation. However, humidity and vibration, sensor node failure, and limited resources such as CPU, memory, and energy all pose problems to the network topology and connectivity of IWSNs [216]. Communication is the most energy-intensive activity in industrial settings, hence the energy efficiency of sensor nodes is critical for network performance [217].

To overcome these concerns, researchers concentrated on recognizing cut-vertex, boundary nodes, and dominated sets. Many researchers utilize the cut vertex approach, as shown in [218], to identify the most critical nodes in the network, increase security and energy consumption, and ensure that the energy-efficient link monitoring strategy concentrates on the right locations. The authors [219], [201] employed cut vertex in connection estimate to ensure that the network can continue to communicate even if some nodes fail or the network topology changes, hence improving network fault tolerance and dependability. Another study [220] employed cut vertex to ensure k-connectivity even if a node fails. Therefore, they detected and protected some special nodes while adding non-critical nodes to fill gaps. Cut vertex ensures connectivity while reducing energy consumption. However, it has limits in terms of scalability and failure tolerance. A dominant set, a subset of nodes inside a network, restricts the number of active nodes that actively participate in communication while also limiting scalability, complexity, and network stability.

Others [221] use the dominant set to maximize coverage, reduce energy consumption, and increase energy efficiency by selecting a limited selection of nodes for complete connection. In [222], they use the Minimum Connected Dominating Set (MCDS) with a bi-partite inspired approach to successfully pick cluster heads, hence optimizing network energy usage while keeping connection and performance. The authors of [223] propose a protocol that uses centralized nodes to provide efficient energy usage, network performance optimization, network lifespan extension, and connection augmentation. [224] also uses centralized nodes to improve routing and energy usage. Others [225] rely on border nodes, which are critical for maximizing coverage and energy efficiency in sensor networks because they define coverage zones in the Voronoi diagram. In [226], they used these nodes to optimize energy-efficient sensor network architectures. Boundary nodes are deliberately placed at cluster borders to recognize and report on perimeter occurrences. Their suggested protocol is successful in reducing energy usage and controlling data transmission, especially in

large-scale networks.

A hybrid technique [50], that combines cut vertex detection with boundary node detection can increase IWSNs reliability, resource usage, and scalability, but it may face challenges with exact placement, static assumptions, and network condition adaptation.

6.9 Discussion

None of these studies have particularly addressed critical node identification, they just combine two algorithms and consider them to be critical nodes. Leveraging critical nodes in WSNs can significantly ensure good performance in these networks, especially in energy efficiency, fault tolerance, and communication tasks.

These nodes are essential for collecting data and transmitting aggregated data in routing protocols to reduce communication delays. Additionally, these nodes maintain connectivity that can improve fault tolerance.

However, using critical nodes in WSNs may face challenges, like the dynamic topology and high computational time for large networks. In addition, these nodes can fail, so the whole network will be broken.

6.10 Conclusion

In this chapter, we have defined WSNs, which are considered networks consisting of distributed sensor nodes that communicate wirelessly to collect and relay data from many environments, making them indispensable in many domains. We represent the different application domains of WSNs.

We highlighted the problems of WSNs and how graph theory might help ensure reliable and economical functioning in large-scale systems. One of the most successful approaches for resolving these difficulties is critical node identification.

Part II

Contributions

Chapter 7

Heuristic for Critical Nodes Detection Problem (CNDP)

7.1 Introduction

Some specific nodes play an important role in the structure and connectivity of the network. These nodes are called critical nodes, if these nodes are removed, it is regarded as a partitioning method because of their objectives in dividing the network into many partitions. The identification of these nodes and their removal has a significant importance in many domains, such as community detection, load balancing, and source detection.

In this chapter, we propose a novel heuristic to deal with the Critical Nodes Detection Problem (CNDP). Identifying the optimal set of critical nodes is a computationally challenging task. In fact, the Critical Node Detection Problem (CNDP) has been proven to be NP-complete, as shown in [38], making exact solutions computationally infeasible for large-scale networks.

7.2 The Proposed Heuristic for Critical Node Identification

In our approach, we search to delete as few nodes from G, such that no component of this disconnected graph has more than a given number of vertices. Formally, this variant can be stated as follows:

Input: An undirected graph $G = (V, E)$ and an integer l.
Output: A subset $S \subseteq V$ of nodes subject to all components of $G[V \setminus S]$ are of cardinality at most l.

This variant of CNP has been proven to be NP-complete in [38]. Therefore, we pro-

pose a new heuristic to address this problem, as presented in Algorithm 1. For a graph G, let M_S be the set of all connected components in induced subgraph $G[S]$ and let $LargestComp(M_S)$ be the function that returns the number of nodes of the largest connected components of M_S.

As demonstrated in Algorithm 1, we propose a novel heuristic to address this problem. Let M_S be the set of all connected components in the induced subgraph $G[S]$ for a graph G, and $LargestComp(M_S)$ be the function that returns the number of nodes of the largest connected components of M_S, and

Algorithm 1 Heuristic Critical Nodes Problem

Require: G : Graph $G(V, E)$ and an integer l

Ensure: A subset $S \subseteq V$ of critical nodes such that for every connected component $h \in G[V \setminus S]$, $|V(h)| < l$

```
MIS = MaxIndSet(G);
B=true
while B do
    x = argmin{LargestComp(M_{MIS∪{j}}), j ∈ V \ MIS}
    if LargestComp(M_x) > l then
        B= false;
    else
        MIS = MIS ∪ {x} ;
    end if
end while
return S;
```

Our algorithm starts by calculating the MIS. This set is selected randomly using the following function:

Algorithm 2 MaxIndSet($G = (V, E)$)

```
S = ∅;
S' = V
while S' ≠ ∅ do
    randomly select a vertex v from S'
    S = S ∪ {v}
    S' = S' \ (N(v) ∪ {v})
end while
return S;
```

This pseudo-code uses a greedy approach, selecting vertices randomly and removing

their neighbours from consideration to ensure independence. The process continues until no more vertices can be added to the independent set.

To make our method clear to readers, we give the network in Fig. 8.1 as an example to explain our proposed heuristic, this network consists of 17 nodes.

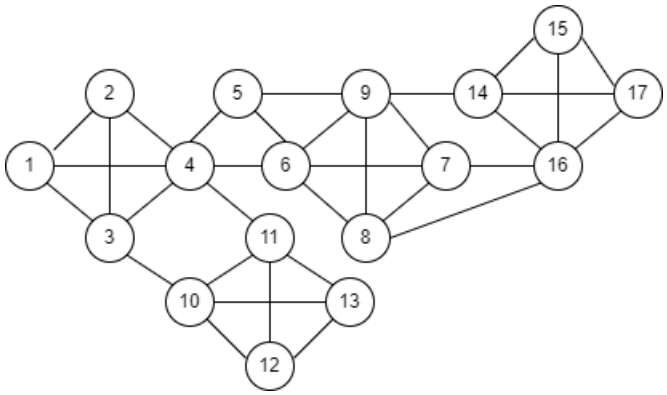


Figure 7.1: Example of our Proposed Method

Fig. 7.2 shows an example of a maximal independent set of the graph in Fig. 8.1.

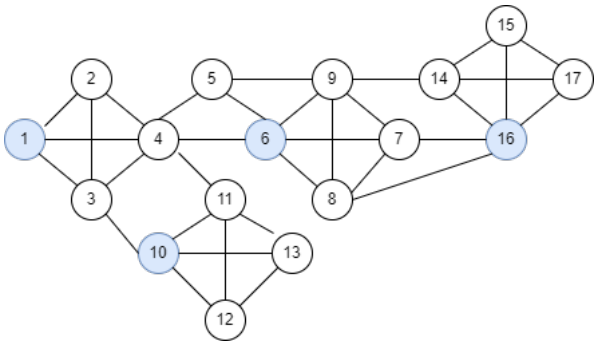


Figure 7.2: An Example of Maximal Independent Set

After the initial MIS is computed, the heuristic greedily selects the node $x \in V$ not currently in the MIS which returns the minimum objective function $LargestComp$ (see Line 4). The set MIS is augmented to include node x, and the computations will be repeated until the following condition holds $\forall v \in V \setminus MIS, LargestComp(M_v) > l$. At this time, the method terminates and the set of critical nodes to be deleted $V \setminus MIS$ is returned. By applying this step on the graph in Fig. 8.1 with $l = 3$ and $MIS = \{1, 10, 6, 16\}$, we obtain five critical nodes $CNP = \{4, 6, 10, 9, 16\}$ whose removal mostly partitions the graph into five connected components of size at least 3 as shown in Fig. 8.4.

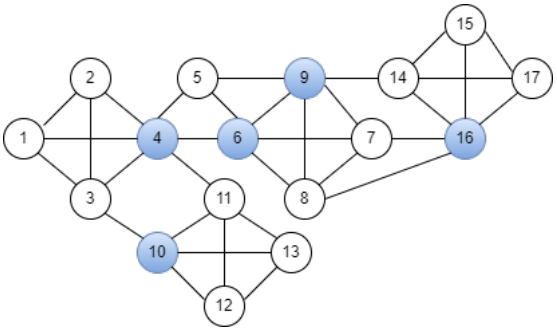


Figure 7.3: Example of Critical Node Identification

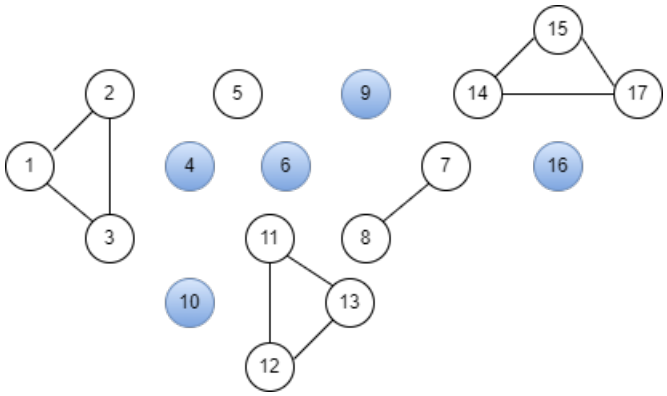


Figure 7.4: Deletion of Critical Node

7.3 Discussion

In Figure 7.5, we present our contributions in this research, as shown that we employ the critical nodes detection problem to solve many issues in various complex networks. In the first problem of community detection, critical nodes are nodes with a significant importance that can influence the network. After removing these nodes, the graph will be partitioned into several subgraphs, which has the greatest impact on network connection over the remainder graph. In the second problem of source detection, we begin by selecting critical nodes as observation nodes as an efficient set of nodes in order to ensure accurate source detection. Selecting nodes that are well-distributed and appropriately located improves source localization accuracy. The third problem of the evaluation of the impact of critical nodes in WSNs, if these nodes are removed, it improves the energy efficiency and structural optimization of WSNs ensuring a balance between connectivity and energy consumption.

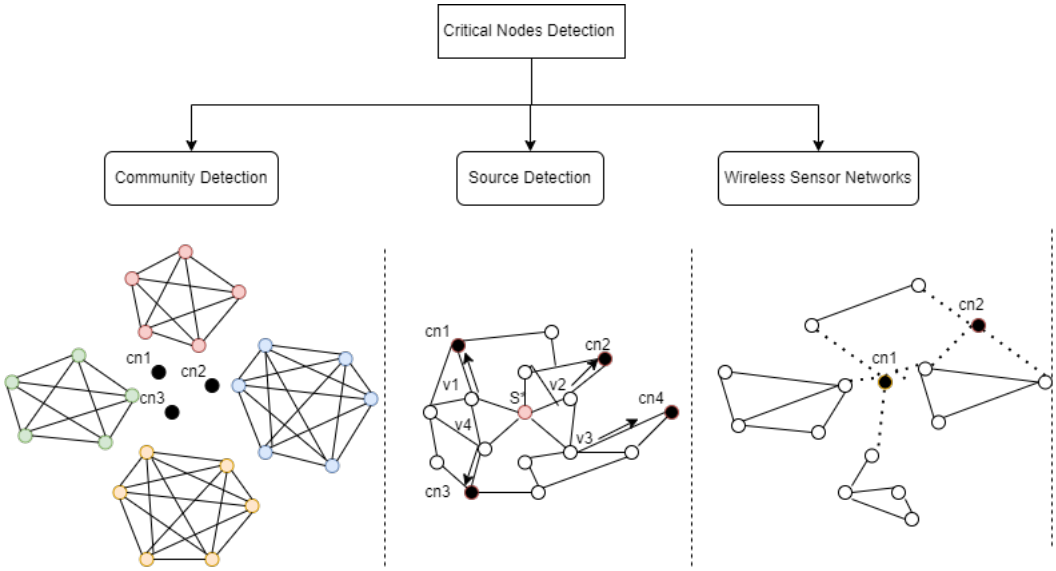


Figure 7.5: Our contributions

7.4 Conclusion

In this chapter, we proposed a heuristic method designed to efficiently approximate solutions to this problem. Our approach aims to identify critical nodes while maintaining reasonable accuracy in complex real-world networks.

The critical nodes problem must be addressed, especially for big and complex networks, as this chapter emphasizes. The knowledge acquired here will serve as a basis for the upcoming chapters, in which we will provide a new implementation that expands upon these fundamental ideas. This method seeks to solve a number of significant issues in network analysis and graph theory, such as detecting communities, enhancing network efficiency in general.

Chapter 8

Community Detection using Critical Nodes

8.1 Introduction

Community identification is important for network analysis activity in a wide range of application domains. The objective of community identification is to identify a set of nodes in a network that are more closely associated with each other than with the rest of the graph. These communities serve as a representation of the network's fundamental structure and are essential to comprehending the interactions between system components. Critical nodes are important for detecting communities, their removal can have a big effect on the network's structure and connection as well as upend already existing communities. These nodes are essential to understanding the current community structure and the dynamic process of community identification.

An extensive examination of how critical nodes influence community structures will be part of this. We will concentrate on how these nodes act as the structural foundations that support communities and how the overall connectivity of the network may be significantly altered by the loss of critical nodes. Lastly, we will examine how community detection proposed algorithm can enhance the modularity and effectiveness of community identification by utilizing the idea of these nodes.

8.2 Critical Nodes Community Detection Approach (CNCDA)

Our proposed approach CNCDA aims to identify significant nodes. The goal is to locate a small fraction of nodes whose elimination has the greatest impact on the network connection over the remainder graph. We use these nodes to identify communities in social

networks. Our approach identifies critical nodes and creates communities based on their similarity, then merges them sequentially to maximize modularity.

Our approach is based on an iterative process (Algorithm 3). In each iteration, we begin by identifying a set of critical nodes, denoted as S, from G (Line 3). This CNP set is determined by the criterion that the size of the largest component in the residual graph $G[V \setminus S]$ is at most l. Then, in (Line 4) an initial community solution is generated using the connected components in the graph $G[V \setminus S]$. This also includes the merging of unassigned nodes. Finally, we merge the discovered communities to form the final community structures in the current iteration (Line 5).

Algorithm 3 Greedy Framework

Require: G : Graph $G(V, E)$ and an integer l

Ensure: C_{best}: Final assignment of nodes to communities

```
C_best = ∅
while ! termination condition do
    S=Critical-Node-Identification(G,l);
    C'=Initial-Communities-Construction(G,S);
    C=Communities-Merging(G,C');
    if Q(C) > Q(C_best) then
        C_best = C;
    end if
end while
return C_best ;
```

To make our method clear to readers, we give the network in Figure 8.1 as an example to explain our algorithm, this network consists of 17 nodes.

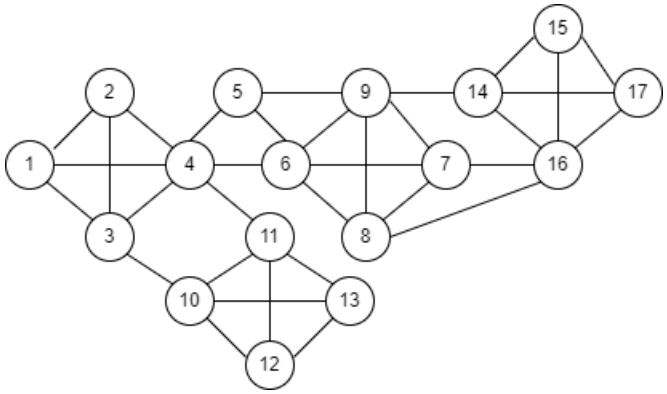


Figure 8.1: Example of our Proposed Approach

Our proposed approach has three steps. In the following section, we explain each step of our proposed algorithm (See figure 8.2) and apply it to the example shown in 8.1 and

draw the resulting graph.

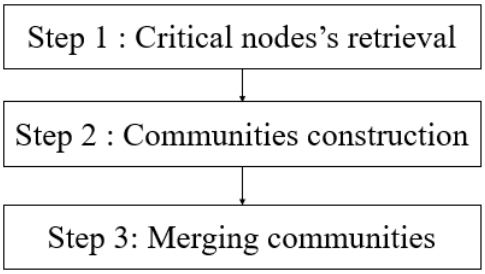


Figure 8.2: Proposed Approach Steps

8.2.1 Critical Node's Retrieval

In our approach, we seek to delete as few nodes as possible from G, such that no component of this disconnected graph has more than a given number of vertices. Formally, this variant can be stated as follows:

Input: An undirected graph $G = (V, E)$ and an integer l.
Output: A subset $S \subseteq V$ of nodes subject to all components of $G[V \setminus S]$ are of cardinality at most l.

This variant of CNP has been proven to be NP-complete in [38]. We used the proposed heuristic explained in Chapter 7 to address this problem and apply it to the graph in Fig. 8.1 with $l = 3$ and $MIS = \{1, 10, 6, 16\}$, we obtain five critical nodes $CNP = \{4, 6, 10, 9, 16\}$ whose removal mostly partitions the graph into five connected components of size at least 3, as shown in Figure 8.4.

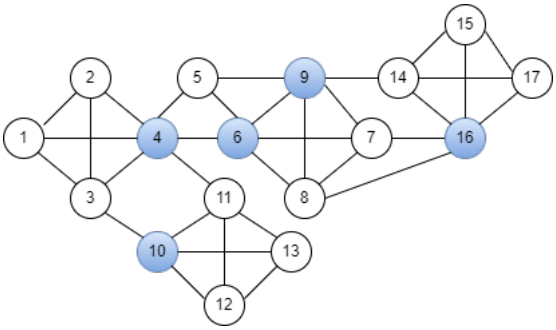


Figure 8.3: Example of Critical Node Identification

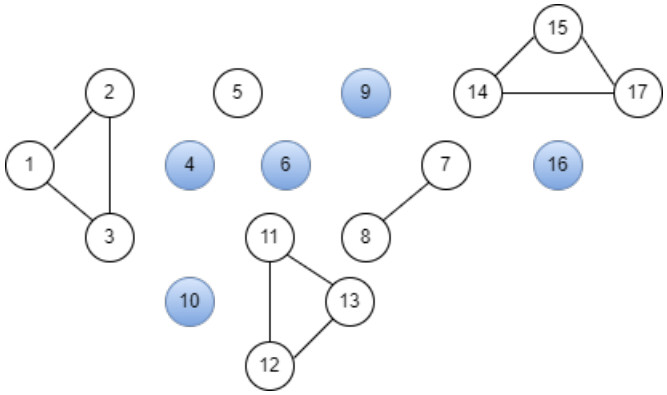


Figure 8.4: Deletion of Critical Node

8.2.2 Communities Construction

The pseudo-code for this step is outlined in Algorithm 4, which takes as input a graph $G = (V, E)$ and a set of critical nodes S. In this step, communities are formed based on the connected components resulting from the removal of critical nodes, where each connected component can be regarded as a potential community. Let C be a set of connected components defined as $C = \{C_1, C_2, \cdots C_m\}$ on $G[V \setminus S]$. We assign the set of critical nodes S to the community that is most similar to them based on the proposed similarity (Define in the following Section 8.2.2.1) between a node and a community as indicated in lines 5 to 13 in Algorithm 4.

Algorithm 4 Communities Construction

Require: G : Graph $G(V, E)$ and a set of critical nodes S

Ensure: community structure $C = C_1 \cup C_2 \cup \cdots \cup C_m$

1: let $C = \{C_1, C_2, \cdots, C_m\}$ the set of connected components of $G[V \setminus S]$;
2: **for** every $v \in S$ **do**
3: $index_{max} = 1$;
4: $sim_{max} = 0$;
5: **for** every $C_i \in C$ **do**
6: **if** $Sim_{measure}(v, C_i) > sim_{max}$ **then** ;
7: $sim_{max} = Sim(v, C_i)$;
8: $index_{max} = i$;
9: **end if**
10: **end for**
11: $C_{index_{max}} = C_{index_{max}} \cup \{v\}$;
12: **end for**
13: return C ;

The outcome of this stage in our example is illustrated in Figure 8.5 and Figure 8.6 .

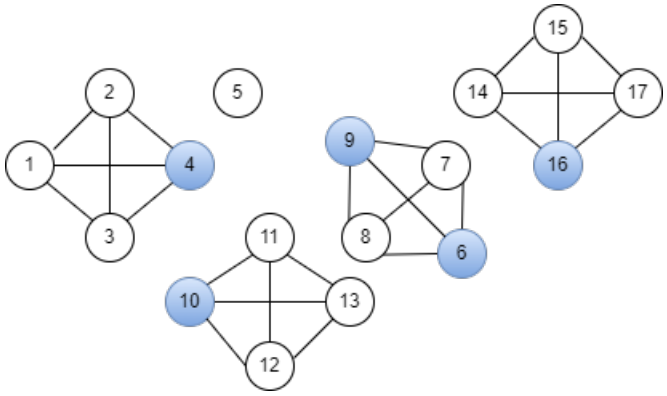


Figure 8.5: Example of Construction of Communities

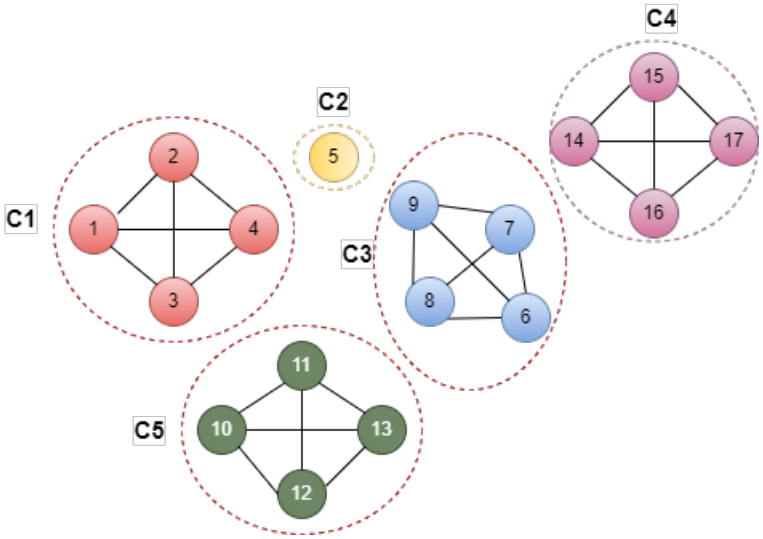


Figure 8.6: Initial Communities

8.2.2.1 A Proposed Similarity Measure

To assign each critical node to its most similar potential community, we propose a novel similarity measure inspired by the Jaccard similarity. This measure represents the similarity between a critical node and a potential community by comparing the critical node to all nodes within the connected component resulting after the elimination of critical nodes. Specifically, it calculates the sum of the Jaccard similarity values between the critical node and every node in the potential community.

We should assign the critical nodes to the connected component that maximizes this similarity measure.

Let cn be a critical node, and let $C = \{C_1, C_2, \ldots, C_n\}$ be potential communities represented as connected components.

The Jaccard similarity between node cn and any node $C_i \in C$ as defined in 1.6.9.

Then, the similarity between the critical node cn and the community C is computed as:

$$\text{Sim}(cn, C) = \sum_{C_i \in C} \text{Jaccard}(cn, C_i) \tag{8.2.1}$$

To assign each critical node cn to the most appropriate community, we select the community C^* that maximizes this similarity:

$$C^* = \arg\max_{Ci \in C} \text{Sim}(cn, Ci)$$

This measure can assign critical nodes to the communities where they are most closely associated in a more precise and structured way.

8.2.3 Merging Communities

The main objective of this third step is to enhance the current partition by merging small communities into larger ones. Let $\triangle Q_{c_i c_j}$ denote the modularity change for merging c_i and c_j. We consider all existing communities by defining a symmetric modularity matrix change $M[m, m]$, whose element is:

$$M_{ij} = \begin{cases} \triangle Q_{c_i c_j} & \text{if } i \neq j \\ 0 & \text{Otherwise} \end{cases} \tag{8.2.2}$$

The outline of the merging step is shown in Algorithm 5. The input is a network and a community structure C, and the output is a set of final communities after merging. At each merging iteration of this algorithm, it is essential to update the modularity matrix change, denoted as M, which arises from the combination of any two currently existing communities, c_i and c_j. This updating is crucial for determining the optimal merger. This merging procedure repeats iteratively and stops when no further increase in Q can be attained.

Algorithm 5 Communities Merging

Require: Graph $G = (V, E)$, community structure $C = C_1 \cup C_2 \cup \cdots \cup C_m$

Ensure: The final community set $C = C_1 \cup C_2 \cup \cdots \cup C_m$

repeat

Compute the matrix modularity change, denoted by M_{ij}.

Select the largest element M_{ij}.

$C_j = C_j \cup C_i$.

Update the number of communities m and the set C.

until the modularity is no longer increased.

Return The final community set $C = C_1 \cup C_2 \cup \cdots \cup C_m$

By applying this step on the graph shown in Fig. 8.1, we merged the two communities C_1C_5, then we merge C_2 C_3. After that, we merge C_2 C_3 with C_4 (See Figure 8.6). So, we have as a result two final communities $C_1 = \{1, 2, 3, 4, 10, 11, 12, 13\}$ and $C_2 = \{5, 6, 7, 8, 9, 14, 15, 16, 17\}$ as shown in Figure 8.7

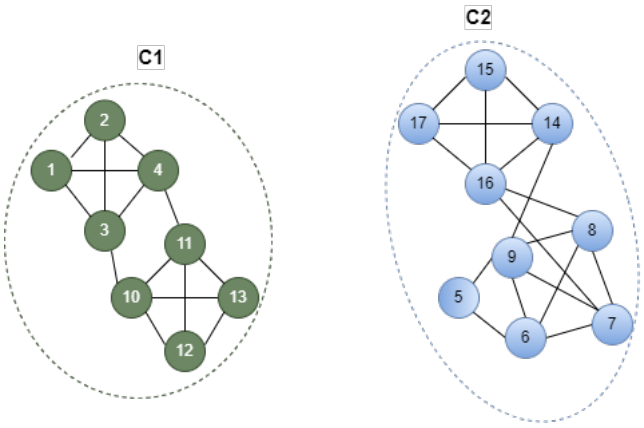


Figure 8.7: Example of Communities Merging

8.3 Results and Analysis

We evaluate the performances of the proposed algorithm on real networks with and without the ground-truth community structures and compare the results with six community detection algorithms, including Louvain [227], LPA [228], WalkTrap [229], Eigenvector [230], Fastgreedy [231] and TS [103].

8.3.1 Real Networks with Ground Truth

Four real networks with ground truth shown in Table 8.1 are used in order to test the efficiency of our proposed algorithm.

Table 8.1: Real networks with ground truth

Networks	n	m	C	Description
Karate	34	78	2	Zachary's karate club network
Dolphins	62	159	2	Dolphin social network
Polbooks	105	441	3	Books about US politics network
Football	115	613	12	American college football network

All these networks are available online:
http://networkrepository.com/networks.php
http://snap.stanford.edu/data/index.html.

- **Zachary's Karate Club network** : describes the friendships between 34 members in a karate club. It contains 34 vertices and 78 edges, separating into two communities.
- **Dolphin social network** : describes the social relationship of 62 nodes. nodes representing dolphins living in Doubtful Sound, New Zealand, it contains 159 edges, separating into two communities.
- **American College Football network** : describes the game schedules between 115 American college football teams during the 2000 season. It consists of 115 vertices and 613 edges, separating into 12 communities.
- **The Pol Books network** : describes the political books' data on US politics recorded in 2005 by Adamic and Glance. The network contains 105 books (nodes) and 882 edges between them, separating into three communities.

We compare the quality of detected communities using NMI measure on the previous four real networks. The corresponding results are shown in Table 8.2.
As we see from this table, our proposed algorithm, CNCDA outperforms many methods. Especially for the karate and football networks, the communities detected are coordinated with the ground truth.

We also use modularity measure to compare CNCDA with other methods, it obtains the results shown in Table 8.3) where C is the number of communities and Q represents the modularity value obtained.

Table 8.2: The NMI results of our proposed approach CNCDA on real networks with ground truth compared with other algorithms.

DataSets	**Karate**	**Dolphins**	**PolBooks**	**Football**
CNCDA	0.71	0.69	0.61	0.93
Louvain	0.59	0.48	0.51	0.88
LPA	0.70	0.69	0.57	0.92
Walktrap	0.50	0.54	0.54	0.90
Eigenvector	0.68	0.45	0.71	0.52
Fastgreedy	0.69	0.61	0.53	0.70
TS	0.71	0.89	0.55	0.90

Table 8.3: Modularity results of our proposed approach CNCDA on real networks with ground truth compared with other algorithms.

DataSets	**Karate**		**Dolphins**		**PolBooks**		**Football**	
	C	Q	C	Q	C	Q	C	Q
Ground truth	**2**	**0.37**	**2**	**0.38**	**3**	**0.41**	**12**	**0.55**
CNCDA	3	0,43	4	0.54	3	0.53	12	0.62
Louvain	4	0.42	6	0.53	4	0.53	10	0.60
LPA	3	0.42	6	0.52	8	0.53	11	0.60
Walktrap	5	0.35	4	0.49	4	0.51	10	0.60
Eigenvector	2	0.39	2	0.49	3	0.49	12	0.47
Fastgreedy	3	0.38	4	0.50	4	0.50	6	0.55
TS	4	0.42	2	0.38	4	0.52	10	0.60

From this table, CNCDA is better than other methods, it detects communities with high quality. In addition, in Polbooks network the value of modularity is like Louvain and LPA, but the number of detected communities is three, such as the ground truth.
Figure 8.8 and Figure 8.9 show the results of the NMI and modularity values for the proposed algorithm CNCDA in the previous four real networks, respectively. Figure 8.8 shows the NMI values of the proposed algorithm CNCDA are good compared to other methods on the previous four real networks, especially for karate and football networks.

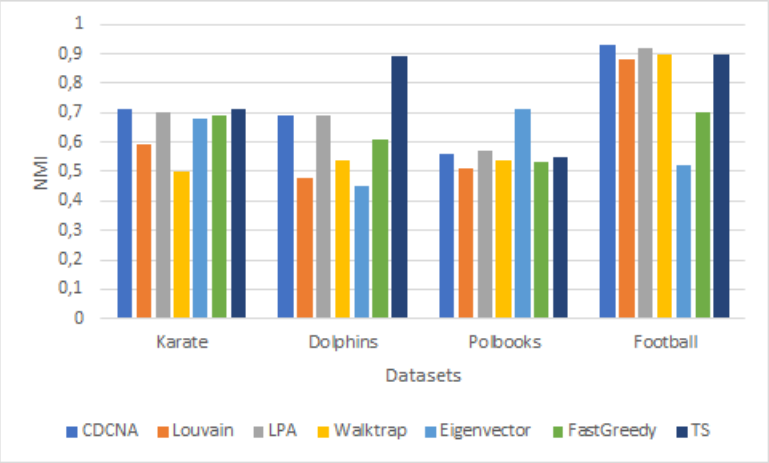


Figure 8.8: NMI results of our proposed approach CNCDA on real networks with ground truth compared with other algorithms

Fig. 8.9 shows the value of modularity, CNCDA has better results than other methods, karate, dolphins and football networks. For the Polbooks network, CNCDA has the same result with Louvain and LPA.

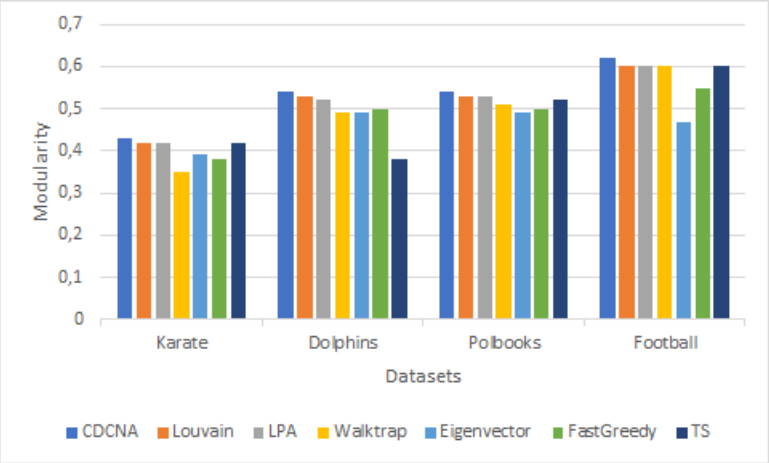
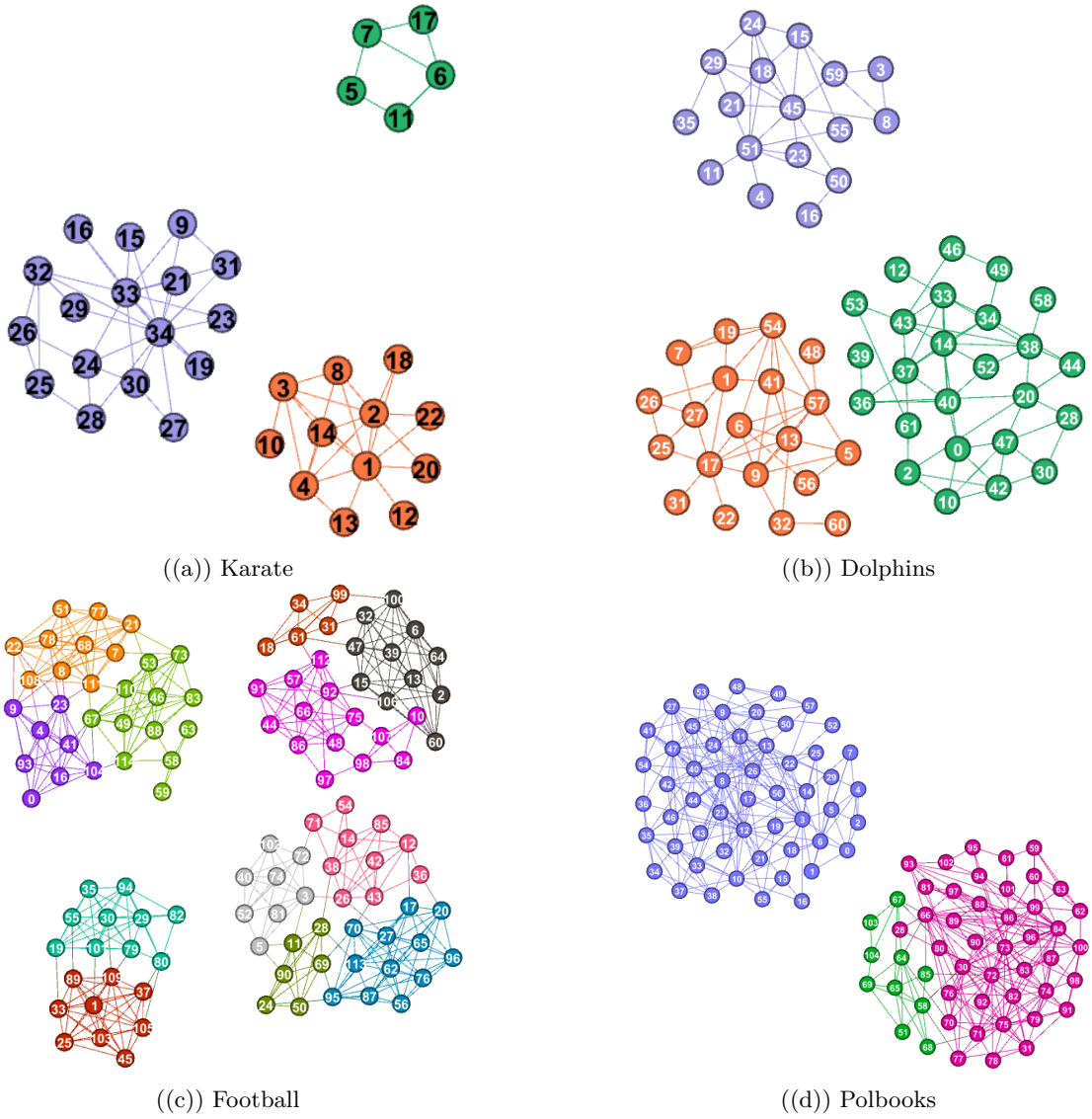


Figure 8.9: Modularity results of our proposed approach CNCDA on real networks with ground truth compared with other algorithms

There are three communities detected by CNCDA on karate network (see Figure 8.10(a)) coloured in blue, red and green. Figure 8.10(b) represents three communities detected by CNCDA on Dolphin network, coloured in blue, orange and green. Figure 8.10(c) shows that there are twelve communities detected by CNCDA in the Football network. CNCDA obtains better performance in terms of modularity and NMI measures than other methods as shown in Table 8.2 and Table 8.3. For the Polbooks network (Fig. 8.10(d)). There are three communities detected by CNCDA like the ground truth.



((a)) Karate ((b)) Dolphins

((c)) Football ((d)) Polbooks

Figure 8.10: The structures of detected communities on the four real networks by the proposed algorithm

8.3.1.1 Real Networks Without Ground Truth

Four real networks without ground truth are used in order to test the efficiency of our proposed algorithm with other methods (See Table 8.4).

- Lesmis network : describes the co-occurrences of characters in Victor Hugo's novel Les Miserables.
- Adjnoun network : describes the common adjective and noun adjacent in the novel David Copperfield.

- Jazz network : describes a jazz musician's network where the nodes are jazz musicians and edges are the cooperation of two musicians in a band.
- E-mail network : describes an email interaction between users in the university Rovira i Virgili in Tarragona in the south of Catalonia in Spain, the edges represent that at least one email was sent (These networks also are available online).

Table 8.4: Real-world Networks without ground truth

Networks	**n**	**m**	**Description**
Lesmis	77	660	Characters network
Adjnoun	112	759	Common adjective and noun adjacent network
Jazz	198	2770	Jazz musicians network
Email	1133	962	E-mail interactions network

We run our approach CNCDA, on these real networks without ground truth and compared it with other methods.

Table 8.5 shows the results of modularity Q and the number of communities C.

Table 8.5: Modularity results of our proposed approach CNCDA on real networks with and without ground truth compared with other algorithms.

DataSets	**Lesmis**		**Adjnoun**		**Jazz**		**Email**	
	C	Q	C	Q	C	Q	C	Q
CNCDA	6	0.60	7	0.35	4	0.45	10	0.56
Louvain	6	0.56	7	0.29	4	0.44	12	0.54
LPA	8	0.53	10	0.24	2	0.28	8	0.28
Walktrap	8	0.52	25	0.22	11	0.44	49	0.53
Eigenvector	6	0.55	1	0	3	0.39	7	0.49
Fastgreedy	5	0.50	7	0.29	4	0.44	16	0.51
TS	6	0.54	7	0.29	3	0.44	9	0.55

As we see from this table, CNCDA has the higher value of modularity compared to other methods in all networks.

In the Lesmis network, CNCDA and most other methods have good results in detecting the number of communities. However, in the Adjnoun network, the number of detected communities is seven, like Louvain, Fastgreedy, and TS. In Jazz network CNCDA obtains

four communities as Louvain and Fastgreedy. For the email network, CNCDA detects ten communities, and it is different from other methods.

Fig. 8.11 shows the results of modularity values for the proposed algorithm 'CNCDA compared with other methods.

Fig. 8.11 shows the results of modularity values for the proposed algorithm CNCDA compared with other methods.

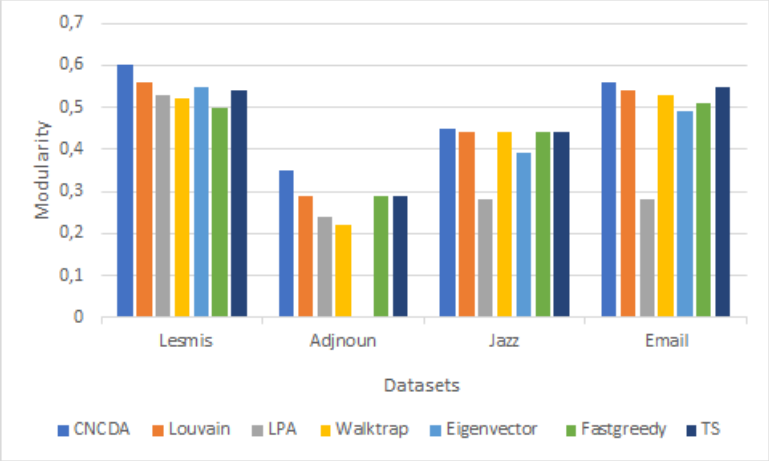


Figure 8.11: Modularity results of our proposed approach CNCDA on real networks without ground truth compared with other algorithms

8.4 Discussion

Unlike previous studies, no existing work has employed critical nodes for solving the community detection problem. In this chapter, after presenting and explaining our proposed community detection approach step by step using a detailed example, we compared its performance with other well-known algorithms using real-world networks. The evaluation was based on two key measures: Normal Mutual Information and modularity.

Our results show that incorporating critical nodes aims at a more effective division of networks, enabling efficient detection of communities even in complex structures. These nodes simplify the detection process, as their removal can disconnect network components. Building communities around these critical nodes provides a strong foundation for grouping similar nodes together. So, the CNCDA aims to assign each node to the group it is most closely similar to, assuming that members of the same community typically share similar interests.

The experimental results confirm the effectiveness of CNCDA. In particular, the modularity values obtained are consistently high compared to the real community structures.

Moreover, when compared to other known methods, CNCDA shows strong performance on both modularity and NMI measures on most datasets.

Overall, these findings demonstrate the superiority of our proposed approach: CNCDA not only detects high-quality communities but also performs competitively in real-world network analysis, as well as outperforming existing methods.

8.5 Conclusion

In this chapter, we proposed a new approach for community detection in social networks using critical nodes. These nodes are essential for preserving network structure and have the power to markedly influence the community structure.

To evaluate the effectiveness of our proposed approach, we conducted experiments using modularity and NMI as performance metrics, comparing our algorithm against several well-known methods. The results demonstrate that our approach achieves good performance in accurately identifying communities.

Chapter 9

Source Detection using Critical Nodes

9.1 Introduction

Determining the source of information is a fundamental issue that comes up in many different fields, especially in social networks. The source detection problem has been the most important subject of many researchers, it aims to enhance network management, prediction, and also optimization.

This chapter focuses on using critical nodes in a network to improve the source detection process. Critical nodes are crucial to preserving the network's connectivity and integrity. Due to their significant and influential function within the network structure, these nodes can serve as observation nodes. We can get comprehensive statistics on the information flow throughout the network by carefully employing these observation nodes to ensure scalability and efficiency while also increasing the accuracy of source detection.

9.2 Proposed Method

We propose a novel algorithm to locate the source of information using critical nodes. The objective is to select the most efficient set of observation nodes, known as critical nodes, and use Gaussian estimation to determine the most likely source of the information.

Our proposed approach Critical Node-based Source Detection (CNSD) follows a greedy framework, as outlined in Algorithm 6. Given a graph $G(V, E)$, the objective is to estimate the most likely source of information spread. The algorithm begins by identifying a set of critical nodes using a proposed detection method (Refer to Chapter 7). These nodes,

acting as observation nodes, are sorted based on their infection times to establish temporal reference data.

A delay vector d is then computed relative to the earliest infected observer. For each candidate node $v \in V \setminus O$, the algorithm starts by computing the delay covariance λ, the observed delay vector d, and the determinate delay μ. Using these values, the Gaussian likelihood is calculated, as defined in Equation 9.2.2, to estimate the probability of each candidate being the source. The node with the highest estimation value is then selected and returned as the most probable source S.

Algorithm 6 Greedy Framework for our proposed algorithm CNSD

Require: G : Graph $G(V, E)$

Ensure: S: Source of information

1: O = CriticalNodeDetection(G, k);
2: Sort all observers ($o \in O$) by time of infection (t_1);
3: Calculate the delay vector d relative to t_1.
4: for every $v \in G[V \setminus O]$
5: Calculate delay covariance λ;
6: Calculate observed delay d;
7: Calculate Determinate Delay μ;
8: Determine the Gaussian value of v;
9: Maximize the estimation value S;
10: return S;

CNSD considers all nodes except the observation nodes as candidate source nodes. After creating the network's propagation model, the Gaussian estimation value for each candidate source node is generated using observed data from the observation nodes. The node with the greatest estimation value is chosen as the source of information.

CNSD can be structured into three main steps, as illustrated in the following Figure 9.1:

The following subsections provide a detailed explanation with an example of each step of this process:

9.2.1 Critical Nodes Detection

Let a graph G be as shown in Figure 9.2:

- Nodes are persons in the network classified as A, B, C, D, E, F, G, and H.
- Edges represent connections between persons.

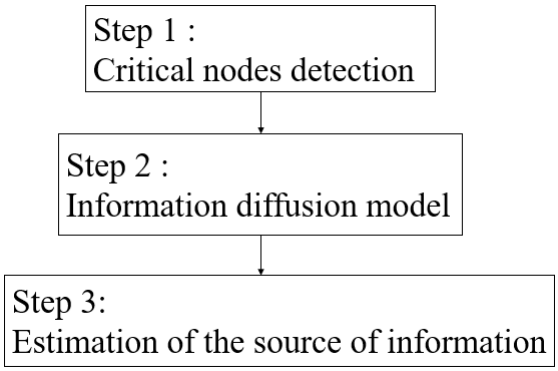


Figure 9.1: Step Process for Source Detection

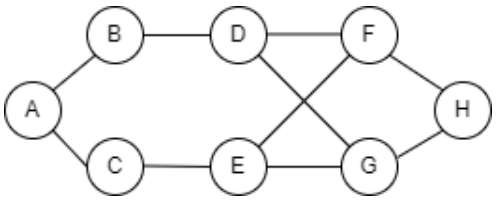


Figure 9.2: Example Graph

As shown in Figure 9.3 critical nodes are considered as observation nodes, D and E that prevent the network from splitting into connected components with a maximum size of three.

This technique treats crucial nodes as observation nodes. Removing these nodes from

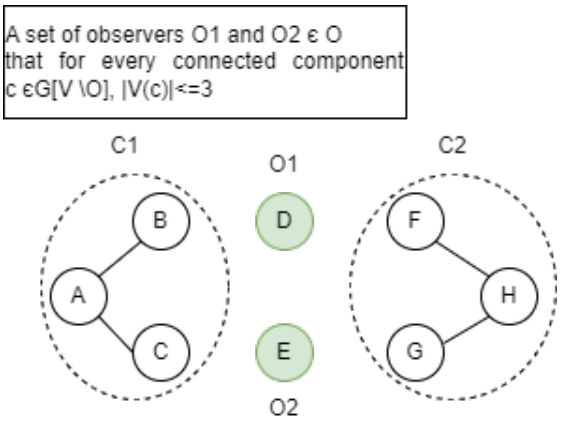


Figure 9.3: Illustration ofCritical Nodes Problem

the network creates a disconnected graph with connected components that have a limited number of vertices.

9.2.2 SI Diffusion Model

Assume node A is the source of information (S) and initially has the following information: Node A becomes infected, making other nodes susceptible. Node A sends information to nodes B and C. Node B, connected to D, sends information to D. D then sends it to G and F. Node D, an observation node, collects information from infected nodes that acquired it from A (refer to Figure 9.4).

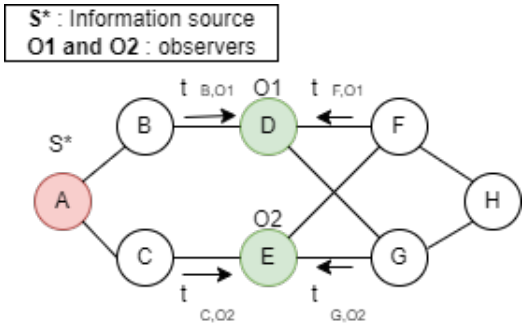


Figure 9.4: Information Source Detection

Identifying observation nodes before using the diffusion model is crucial for understanding the network's resilience and susceptibility to spreading information. The SI model illustrates how information propagates through a network and identifies a certain node as the source.

9.2.3 Gaussian Estimation Value

The third step of our approach considers the source node $v \in G[V \setminus O]$ to be the source if it maximizes the following Gaussian value:

$$S = \max Gaussian(v) \tag{9.2.1}$$

The Gaussian value of each source node, $v \in G[V \setminus O]$, is determined as shown in [123]:

$$Gaussian(v) = \mu_v^T \lambda^{-1} (d - \frac{1}{2}\mu_v) \tag{9.2.2}$$

In this equation, d represents the observable delay, μ the deterministic delay, and λ the delay covariance.

- **Observed delay** $d = (d1, d2, ..., d_n)$ represents the infection time difference among observers. The propagation delay between observers follows a Gaussian distribution that is both independent and identical.

- **Deterministic delay** $\mu = (\mu 1, \mu 2, , \mu_n)$ represents the average difference in infection times between the first and i th observers.
- **Covariance delay** λ refers to the correlation of infection time differences between i^{th} and j^{th} observers.

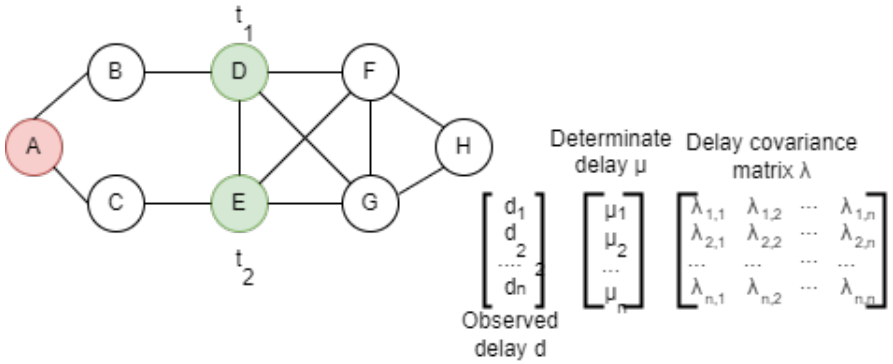


Figure 9.5: Source Estimation

9.3 Results and Analysis

9.3.1 Measure of Performance

Evaluating source identification algorithms requires comparing their accuracy to real-world networks. Accuracy refers to the chance of correctly recognizing the information source. Higher accuracy suggests improved performance. The definition of accuracy is as defined by [143]:

$$Accuracy = \frac{rightcorrect(S^* = S)}{Iterations} \tag{9.3.1}$$

S^* is the detected source node, S is the real source node, $Iterations$ is the total number of simulations, and $rightcorrect$ is the number of simulations where the detected source node matches the actual source node.

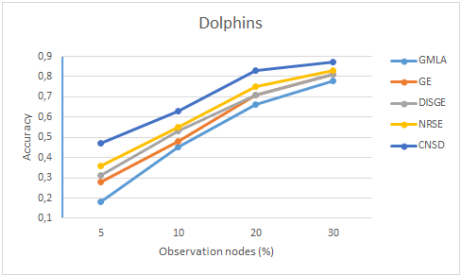
9.3.2 Datasets and Methods of Comparison

The proposed approach was assessed on real networks and compared to other methods such as NRSE [143], GE [123], DISGE [170], and GMLA [171], on the following datasets: Dolphins [232], Celegans [233], Netscience [234], and Email [235]. See Table 9.1 for results.

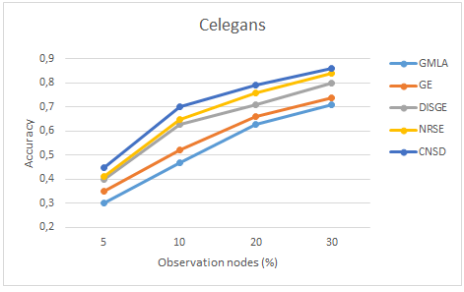
Table 9.1: Real Networks

Networks	n	m	k	Description
Dolphins	62	159	5.13	Dolphin network
Celegans	453	2025	8.94	Celegans network
Netscience	379	914	4.82	Netscience network
Email	1133	5451	9.62	E-mail network

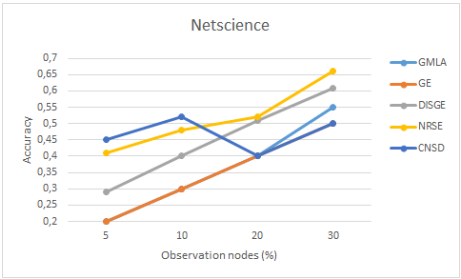
The numbers n, m, and k represent the number of vertices, edges, and average degree of the network, respectively. Figures 9.6 display accuracy results with different percentages of observation nodes (5, 10, 20, and 30%). Observation node variation evaluates the impact of node proportion on source detection performance. By increasing the number of observation nodes, we may analyse accuracy trends and determine thresholds when further observations no longer increase detection accuracy.



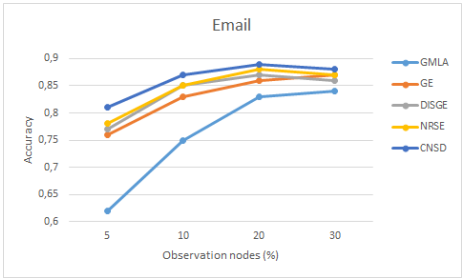
((a)) Dolphins network.



((b)) Celegans network.



((c)) Netscience network.



((d)) Email network.

Figure 9.6: Accuracy results of CNSD on different networks

Using a randomly selected set of observation nodes improves source detection and network structure analysis. The average degree of nodes in the Netscience network is modest, but random selection may result in nodes with a higher degree. The diffusion information collected by these nodes may have accurately identified the source of the information (see Figure 9.6). The amount of information acquired by observation nodes is crucial for accurate source detection. Increasing the number of observation nodes enhances detec-

tion accuracy compared to other approaches, except for the Netscience network, where performance varies.

9.4 Discussion

We have mentioned the importance of choosing an efficient set of observation nodes, we consider these nodes as critical nodes. To our knowledge there are no existing works that employ critical nodes in previous studies to solve source detection problem.

The suggested approach relies heavily on carefully selecting observation nodes to ensure accurate source detection. Selecting well-distributed and appropriately located nodes improves source localization accuracy. To improve source localization accuracy, we use a Gaussian estimation approach. This methodology finds sources in real-world networks, beating previous methods and displaying superior detection performance.

The proposed approach prioritizes connection within certain zones while selecting observation nodes. This method enables effective detection of the source in vast networks. However, the number of observation nodes can impact scalability.

In big networks, having too many observation nodes might increase computing expenses. Optimizing the number of observation nodes is critical for balancing efficiency and computational feasibility. Choosing the right diffusion model is crucial for scalability.

This strategy includes graph partitioning, which improves scalability by breaking down enormous networks into smaller, more manageable segments. This split optimizes the source detection process by reducing computational complexity and improving reaction times.

The suggested approach for source detection in large-scale networks leverages optimal node selection, graph partitioning, and diffusion modelling to give a scalable and reliable solution.

9.5 Conclusion

We described a three-step strategy for finding the information sources in social networks via observation nodes. The heuristic method identifies critical nodes and employs them as observation nodes for optimal detection efficiency. Then, to model the diffusion process, we adopt the Susceptible-Infected (SI) model in the second step of our approach. Finally, the Gaussian estimation technique is then used to determine the source's localization. The method outperforms other existing methods with up to 89% accuracy. This approach is useful to many fields, such as social network analysis, epidemiology, and strategic infor-

mation distribution in big networks. It sets a standard for future research on scalable and high-accuracy source detection methods.

Chapter 10

Energy Optimization in IWSNs using Critical Nodes

10.1 Introduction

By 2023 and 2033, the global industrial automation market is expected to reach 213.53 billion dollars and 529.16 billion dollars, respectively [213]. As a subclass of WSNs, IWSNs contribute significantly to this growth by providing real-time monitoring, control, and optimization solutions for increased operational efficiency and productivity in industrial systems [214]. There are three primary uses for IWSNs [215]: environmental sensing, condition monitoring, and process automation. However, humidity and vibration, sensor node failure, and limited resources such as CPU, memory, and energy all pose problems to the network topology and connectivity of IWSNs [216]. Communication is the most energy-intensive activity in industrial settings, hence the energy efficiency of sensor nodes is critical for network performance [217].

Critical nodes in a network are those whose failure or removal will drastically impair network performance. These nodes play a crucial role in ensuring effective data transmission and communication while consuming the least amount of energy. It is a well-known method in graph partitioning that can divide the network into smaller, controllable divisions while decreasing energy efficiency and communication costs.

Our suggested technique focuses on identifying critical nodes, making it a more effective strategy for enhancing network performance, energy efficiency, and reliability in real-time industrial applications.

10.1.1 Our Proposed Approach Critical Node-Based Energy Optimization in IWSNs (CNEO)

This section describes our innovative recommended technique for solving the problem of energy consumption and assuring connectivity in IWSNs. The suggested technique is based on CNDP, which aims to improve network performance by detecting specific nodes based on their relevance in the network's structure.

Instead of typical approaches such as cut vertices, dominant sets, boundary nodes, and centralized nodes, we offer a unique method that takes into account the Critical Nodes Detection Problem to improve energy efficiency and connectivity in IWSNs. To further identify these nodes, we use graph theory concepts, including connected component size and MIS.

Our technique consists of three phases:

1. Critical Node Detection

   The Critical Nodes Problem seeks to identify essential nodes that play a critical role in sustaining network connectivity and guaranteeing efficient data flow in IWSNs based on network architecture. Deleting these nodes has a major impact on connectivity [36]. These nodes are chosen via creating MIS in order to devise an effective partitioning technique. It chooses nodes such that no two nodes in the MIS are near, decreasing communication interference, then determines if the network would disconnect if the selected essential nodes are removed, and finally verifies the size of each connected component. In other words, critical nodes are those that, if eliminated, would cause the network to split into different partitions. This ensures that the chosen nodes are critical to maintaining the network connectivity.

   Our suggested technique addresses the primary difficulty of selecting nodes that are critical to sustaining network connectivity while consuming the least amount of energy. To address this issue, we use an advanced technique that considers both the size of connected components and MIS.

2. Graph Partitioning

   After identifying and removing critical nodes from the network, it is divided into different partitions. These partitions are typically connected components, which can significantly reduce communication overhead by limiting interactions between distant nodes while also conserving energy. Critical nodes are classified according to their relevance in sustaining network connection and eliminating interference. These

nodes typically play an important role in ensuring communication between nodes in the network. Once identified, these nodes are deleted, and their absence determines how the network responds.

Following the removal of these nodes, the network is separated into connected components. The extent to which the network is broken into distinct components provides important information about the importance of the deleted nodes. If the network breaks into a considerable number of components after removing a few nodes, it means that these nodes were truly crucial to sustaining overall connectedness, cutting communication costs. This happens when nodes within the same component communicate more often, whereas interactions between nodes in other components are minimized. Furthermore, it minimizes interactions between distant nodes, which reduces the need for long-distance communication by needing less energy for data transit. This is particularly valuable in energy-constrained situations. Because nodes in smaller components are closer together, data transmission is more localized, resulting in shorter communication flows that need less power. As a result, energy saving and network efficiency are improved. If the number of components does not rise, it indicates that the eliminated nodes are not vital to the network structure or connection.

Our goal is to decrease energy usage while still ensuring network connectivity. When we achieve a balance between minimizing communication overhead and avoiding severe network fragmentation, we can improve energy efficiency by partitioning the network into smaller components. Nodes within the same component can continue to communicate efficiently without the need for long-distance communication, which requires more energy. Furthermore, reducing communication between components can lower the overall energy used for data transfer.

3. Energy Consumption

The elimination of critical nodes saves energy by reducing duplicate communication paths. Energy consumption is estimated repeatedly, and the results are analysed to guarantee that the suggested method increases energy efficiency while maintaining network connectivity. Each node's energy consumption is computed based on its CPU time utilization, resulting in a simplified model.

In Algorithm 1, we provide three steps as shown in Figure 10.1. In the first step, we identify critical nodes based on their function in network connectivity using graph theory methods, including connected component and MIS (The proposed heuristic refers to Chapter 7), and then we delete these nodes from the network.

In the second step, the graph partitioning step will determine the connected components of the remaining network after removing the critical nodes from the original graph. If the number of components grows greatly, it indicates that the eliminated nodes are critical certainly. Furthermore, the constraint size of each connected component is less than or equal to a threshold L, preventing network fragmentation (no component becomes too large). Otherwise, if the size of the components exceeds L, reconsider the decision on which nodes are critical. Additional nodes may need to be changed to achieve effective connectivity.

In the final step, the energy consumption step, the algorithm simulates the energy usage of each node over a number of iterations. Initially, every node is assigned a fixed amount of energy. During each iteration, if a node still has energy remaining, it randomly performs a CPU task characterized by a random processing time and power consumption. The energy consumed for each task is calculated as shown in Line 28. This simulation models real-world energy depletion over time due to computational tasks. After all iterations, the algorithm sums the remaining energy across all nodes to compute the total energy consumption. Finally, it outputs the total energy consumed and the number of connected components formed after partitioning. The CNEO algorithm, therefore, offers a method to control network connectivity while considering the concept of energy efficiency, which is particularly important in distributed and resource-constrained networks.

Algorithm 7 CNEO

Require: G : Graph $G(V, E)$, L: The amount of connected components after partitioning, $E\_initial$: Initial energy for every node, $Iterations$: The number of iterations used to average the results.

Ensure: E: Total energy consumption after removing critical nodes, C: Number of connected components after partitioning.

Step 1: Critical Nodes Detection
Initialize MIS (Maximal Independent Set) as null;
Shuffle nodes randomly;
for each node $v \in G$: **do**
    v is not in Visited;
    Add v to MIS;
    Mark v and its neighbours as visited;
end for
Calculate $Removed\_Nodes = Nodes(G) - MIS$;
Create subgraph G_sub using MIS;
for each node u in $Removed\_Nodes$: **do**
    Add u to G_sub and connect it to its neighbours;
    if all connected components in G_sub are less or equal than L: **then**
        retain u in G_sub;
    end if
end for
Return $Critical\_Nodes = Nodes(G) - Nodes(G\_sub)$;
Step 2: Graph Partitioning
Delete $Critical\_Nodes$ from G to create $G\_reduced$;
Calculate the number of connected components in $G\_reduced$;
Save component sizes and the number of components;
Phase 3: Energy Consumption
for each node $v \in G$: **do** Let $Energy[v] = E\_initial$;
    for each iteration: **do**
        if $Energy[v] > 0$: **then**
            Randomly select T_CPU;
            Compute energy consumed;
            $E\_CPU = P\_CPU * T\_CPU.$;
        end if
    end for
    Sum the remaining energy across all nodes;
end for
Return energy consumption, and the number of partitions;

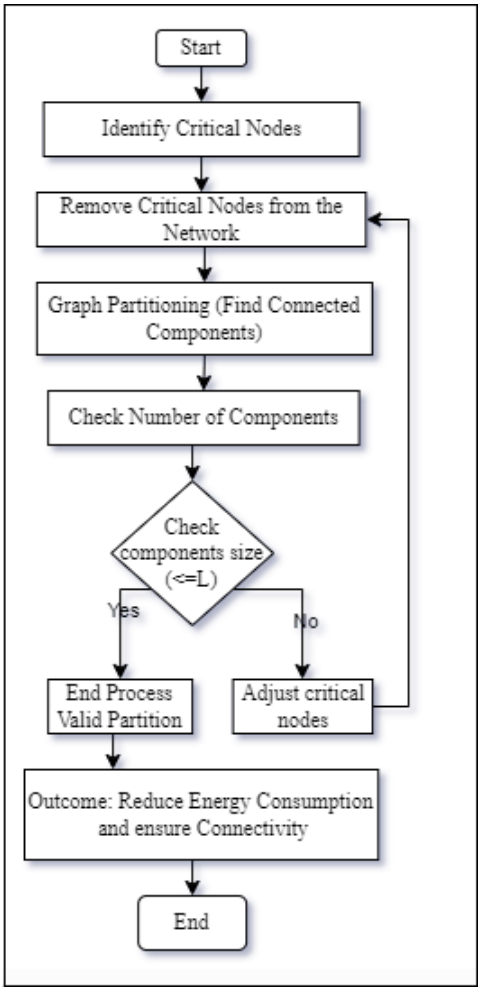


Figure 10.1: The flowchart of the different steps of our proposed approach

10.2 Results and Analysis

Our proposed approach, CNEO, is compared to CDSCUT & D-LPCN [204], ABCND [50], and ABIDE & D-LPCN [236].

IWSNs's nodes were randomly distributed throughout the networks. We selected a single sink node to evaluate the network configuration.

We used NetworkX and Python to run the simulation. To investigate power usage, nodes are gradually enlarged inside a $1000 * 1000\ m^2$ region. The transmission range is 50 milliseconds.

The parameters for assessment and comparison include energy usage and the proportion of critical nodes that are accurately detected. We run the simulation across 20 iterations

1. **Power Consumption as the number of nodes rise**

One key consideration with WSNs is power consumption. For different numbers of nodes, we compare our method to ABCND, ABIDE & D-LPCN, and E-CDSCUT D-LPCN algorithms (Figure 10.2). For all approaches, the figure shows that power consumption grows with the number of nodes. When there are more nodes, we have more communication between them, which leads to increased energy use. Our CNEO method utilizes far less energy than existing algorithms, including the ABCND algorithms. So, our suggested CNEO algorithm can optimize the network's energy usage.

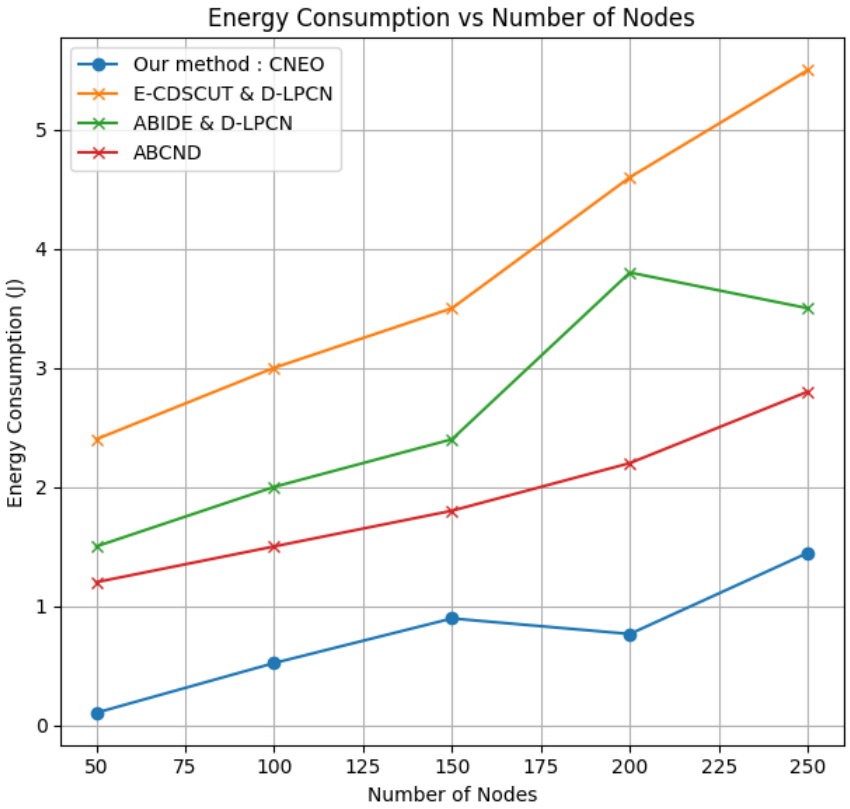


Figure 10.2: Comparison between energy consumption and the number of nodes

We look at the impact of deleting critical nodes from IWSNs. It minimizes energy consumption in a network. Energy consumption normally rises as the number of nodes grows. Greater nodes indicate greater communication and higher overall energy consumption. The energy consumption after deleting critical nodes is compared to alternative strategies. As we can see, CNEO consumes less energy than other methods, demonstrating that it is effective at reducing energy. When critical nodes are eliminated, the amount of energy and communication in the network is reduced.

2. **The number of partitions as the number of node rise**

After removing the critical nodes to reduce energy consumption, we observe a significant increase in the number of partitions (or connected components). As the number of remaining nodes increases, the number of partitions also increases. This makes sense because the removal of critical nodes, which often play a central role in maintaining network connectivity, leads to the fragmentation of the network. As a result, the network breaks into smaller, more disconnected components. Figure 10.3 shows that as the number of nodes increases from 50 to 250, the number of partitions also increases, which means that the fragmentation of networks is maximized. By systematically identifying and eliminating critical nodes. It separates the network into manageable components, which might be beneficial in a variety of applications.

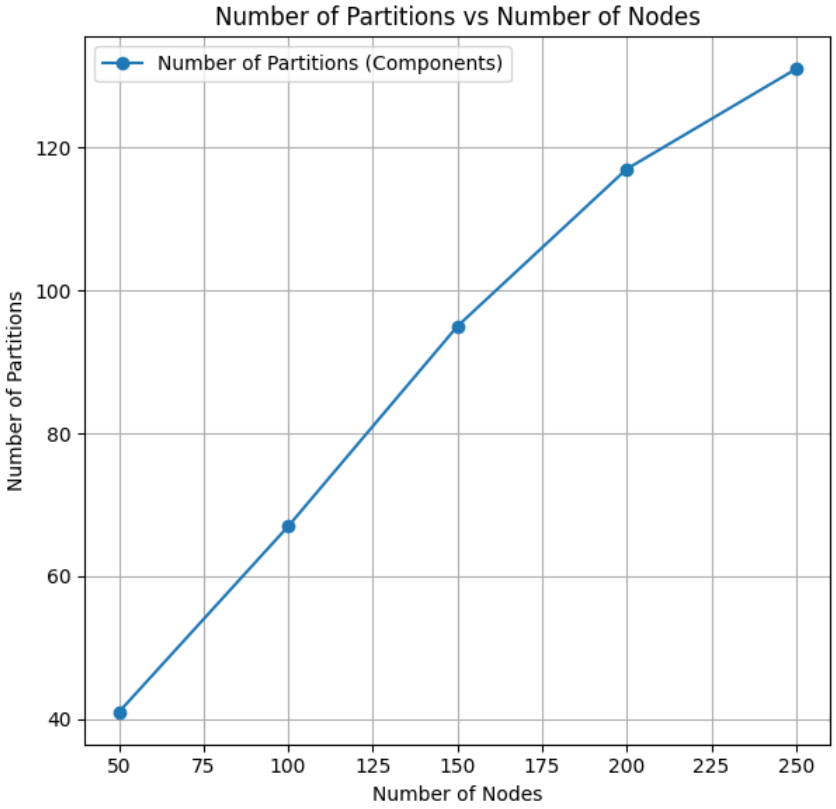


Figure 10.3: Number of partitions after removing critical nodes and the number of nodes

10.3 Discussion

In this chapter, our contribution aims to improve the energy efficiency and structural optimization of IWSNs. It realizes a balance between energy consumption and connectivity by identifying and removing critical nodes. Our proposed approach Critical Node-Based En-

ergy Optimization (CNEO). It outperforms other existing compared methods and decreases energy consumption by eliminating critical nodes, while it really preserves connectivity.

The adoption of graph-based concepts, such as the MIS and connected components, ensures that the network is successfully partitioned into smaller, and more manageable partitions, hence increasing fault tolerance.

The approach adapts to many different networks and provides constant performance and scalability. This is appropriate for numerous application domains, such as IoTs, environmental monitoring, and disaster recovery networks.

10.4 Conclusion

The identification and removal of critical nodes can significantly improve WSNs performance for real-time applications, especially in industrial applications, which enhance connectivity and reduce the risk of network fragmentation, also ensuring that the usage of energy is more efficient, extending the network lifetime, and improving its overall robustness. Furthermore, we outlined our proposed contribution through three steps in IWSNs . We demonstrated that by prioritizing critical nodes, it is possible to reduce energy consumption while simultaneously maintaining network connectivity.

General Conclusion

This thesis investigated several graph theory techniques for handling complex network issues by leveraging the concept of graph partitioning and critical nodes method to solve graph analysis problems. At first, we start by defining the basic notations and concepts of graph theory, we provide a good basis for comprehending its challenges, principles, and applications. After that, we address the problem of graph partitioning, in which we explore how partitioning the graphs into smaller and more manageable partitions, we detail and explain the different graph partitioning algorithms, highlighting the importance of dividing networks to enhance their analysis. This follows the identification of critical nodes as an essential aspect in graph partitioning, which has a significant role in managing the connectivity of networks. We introduced the concept of critical nodes, nodes within a network the removal of which can significantly impact the network's overall functionality and connectivity. We show how the identification and removal of these nodes can be crucial for maintaining the connectivity and efficiency of networks. Critical nodes, considered as important elements in any network, play a fundamental role in solving many problems in network analysis, which makes their study and application a key part of this thesis.

The major contribution of this thesis is based on employing the critical nodes to solve three problems:

1. **Community Detection:**

   Identifying communities, which are groups of nodes that are more densely connected than the rest of the network, is critical for understanding the network's structure and behaviour. We use the CNP to find these communities in social networks, and we improve the process inside the network as well as the modularity and measures. The purpose is to identify a small number of nodes that have the largest influence on network connectivity in the rest graph. Our proposed approach finds critical nodes, develops communities based on similarity, and merges them in order to optimize modularity. We demonstrated how the identification and prioritization of critical nodes can improve the detection of communities, the improved measures are essential for evaluating the quality of detected communities. This strategy improves the efficiency

of identifying relevant communities in social networks.

2. **Source Detection:**

   Locating the source of information is an important task. We proposed a novel approach, which employs the CNP as observation nodes. We developed a more effective strategy for identifying the source of information within social networks. The objective is to select the most efficient set of observation nodes, which aim to observe the flow of information and also enable source detection more accurately, and also using Gaussian estimation aims to determine the source of the information more accurately.

3. **Increase of efficiency in Wireless Sensor Networks (WSNs):**

   WSNs consist of distributed sensor nodes that communicate to monitor environmental conditions and are highly sensitive to the positioning and functioning of critical nodes. We showed that the application of critical nodes can significantly improve network performance. Our proposed approach based on Critical Nodes Problem addressed real-world challenges such as energy efficiency, connectivity, and network resilience. Critical nodes make it more effective for enhancing the performance of networks, energy efficiency, and reliability in real-time industrial applications.

After evaluating the proposed algorithms, demonstrating the effectiveness of utilizing the graph partitioning method by employing critical nodes.

This thesis highlights the importance of critical nodes in graph analysis and their important role in solving diverse network issues in many domains.

Finally, we present numerous viewpoints and perspectives based on the different subjects of this thesis as future research that might look into each chapter in order to present opportunities for further exploration:

1. **Formalization of a Unified Model for Multi-Objective CNP**

   Develop a formal framework to aggregate multiple objectives in the Travelling Salesman Problem (TSP): minimize connectivity and maximize fragmentation while respecting topological constraints (distance, centrality..). This would allow for the modelling of complex real-world situations through a multi-objective approach and for the study of the possible trade-offs. This would allow for modelling complex real-life situations through a multi-objective approach and studying the possible trade-offs.

2. **Identification of Multi-Context Influencer Nodes**

Another approach is to combine CNP with social indicators (centrality, activity, engagement) to identify strategic nodes in real social networks (Facebook, Twitter, etc.). These nodes could play a key role in the dissemination of information, but also in its regulation (combating fake news).

3. **Analysis of Influence Propagation with Intervention Constraints**

   Analysis of influence propagation with intervention constraints Extending the work on source detection to include intervention strategies: once the source is detected, how to minimize the impact of the spread (e.g., rumours, misinformation), under budget or speed constraints.

4. **Deployment of Parallel Algorithms for the Processing of Massive Graphs**

   Deployment of parallel algorithms for massive graph processing A technical perspective would be to develop a parallel and distributed implementation of algorithms related to CNP (on Spark, Hadoop, or CUDA), capable of processing massive graphs in a reasonable time. This would allow for the industrialization of the approach and the consideration of large-scale applications, for example, for streaming data analysis or social networks with millions of nodes.

5. **Adaptive and Learning-Based Approaches for Dynamic Networks**

   Future study might concentrate on adaptive graph partitioning algorithms that respond dynamically to changes in network topology over time. Another promising field is the use of deep learning and reinforcement learning to improve partitioning algorithms in large-scale networks.

6. **Dynamic and Query-Driven Community Detection with Deep Learning Integration**

   Future work in dynamic community detection, where these communities can change and evolve over time. Another future research lies in community search instead of detecting all communities in graph, only based on specific communities of interest. Incorporating deep learning into community search methods can enable more efficient and accurate query processing, especially when dealing with large-scale networks.

7. **Critical Node Partitioning and Energy Harvesting for Heterogeneous WSNs**

   Future work might look at the impact of critical node-based partitioning in heterogeneous WSNs where sensors have varying capacities. Another important possibility is the application of energy harvesting techniques to extend network lifespan in resource-constrained contexts.

Bibliography

[1] M. Lalou, M. A. Tahraoui, and H. Kheddouci, *The critical node detection problem in networks: A survey*, *Computer Science Review* **28** (2018) 92–117.

[2] F. Harary, *Graph theory (addison-weslay publ. comp. reading massachusets)*, .

[3] H. Kong, I. Kim, and B.-T. Zhang, *Autotarget: Disease-associated druggable target identification via node representation learning in ppi networks*, *Current Research in Biotechnology* **8** (2024) 100260.

[4] R. Alguliyev, R. Aliguliyev, and F. Yusifov, *Graph modelling for tracking the covid-19 pandemic spread*, *Infectious disease modelling* **6** (2021) 112–122.

[5] A. Hanane, A. Ullah, and S. Raghay, *Enhanced gaf protocol based on graph theory to optimize energy efficiency and lifetime in wsn technology*, *International Journal of Intelligent Unmanned Systems* **11** (2023), no. 2 214–225.

[6] C. Mylonas, E. Mitsakis, and K. Kepaptsoglou, *Criticality analysis in road networks with graph-theoretic measures, traffic assignment, and simulation*, *Physica A: Statistical Mechanics and its Applications* **629** (2023) 129197.

[7] M. Sato, H. Aomori, and T. Otake, *Automation and acceleration of graph cut based image segmentation utilizing u-net*, *Nonlinear Theory and Its Applications, IEICE* **15** (2024), no. 1 54–71.

[8] N. T. Bliss and M. C. Schmidt, *Confronting the challenges of graphs and networks*, *Lincoln laboratory journal* **20** (2013), no. 1 4–9.

[9] U. Elsner, *Graph partitioning-a survey*, .

[10] H. W. Y. Adoni, T. Nahhal, M. Krichen, B. Aghezzaf, and A. Elbyed, *A survey of current challenges in partitioning and processing of graph-structured data in parallel and distributed systems*, *Distributed and Parallel Databases* **38** (2020) 495–530.

[11] K. Schloegel, G. Karypis, and V. Kumar, *Graph partitioning for high performance scientific simulations*, .

[12] J. Li, R. K. Kapania, and W. B. Spillman Jr, *Placement optimization of distributed-sensing fiber optic sensors using genetic algorithms*, *AIAA journal* **46** (2008), no. 4 824–836.

[13] M. E. Newman, *Modularity and community structure in networks*, *Proceedings of the national academy of sciences* **103** (2006), no. 23 8577–8582.

[14] M. Girvan and M. E. Newman, *Community structure in social and biological networks*, *Proceedings of the national academy of sciences* **99** (2002), no. 12 7821–7826.

[15] A. Zocca, C. Liang, L. Guo, S. H. Low, and A. Wierman, *A spectral representation of power systems with applications to adaptive grid partitioning and cascading failure localization*, *arXiv preprint arXiv:2105.05234* (2021).

[16] L. Xiang, J. Luo, and A. Vasilakos, *Compressed data aggregation for energy efficient wireless sensor networks*, in *2011 8th annual IEEE communications society conference on sensor, mesh and ad hoc communications and networks*, pp. 46–54, IEEE, 2011.

[17] J. Shi and J. Malik, *Normalized cuts and image segmentation*, *IEEE Transactions on pattern analysis and machine intelligence* **22** (2000), no. 8 888–905.

[18] C. Ding, L. Zhu, L. Shen, Z. Li, Y. Li, and Q. Liang, *The intelligent traffic flow control system based on 6g and optimized genetic algorithm*, *IEEE Transactions on Intelligent Transportation Systems* (2024).

[19] S. Roy and S. Banerjee, *Performance analysis of meta-heuristic optimization techniques for multi-objective vlsi circuit partitioning*, *Engineering Research Express* **6** (2024), no. 4 045208.

[20] Ü. Çatalyürek, K. Devine, M. Faraj, L. Gottesbüren, T. Heuer, H. Meyerhenke, P. Sanders, S. Schlag, C. Schulz, D. Seemaier, et al., *More recent advances in (hyper) graph partitioning*, *ACM Computing Surveys* **55** (2023), no. 12 1–38.

[21] T. A. Ayall, H. Liu, C. Zhou, A. M. Seid, F. B. Gereme, H. N. Abishu, and Y. H. Yacob, *Graph computing systems and partitioning techniques: A survey*, *IEEE Access* **10** (2022) 118523–118550.

[22] G. Karypis and V. Kumar, *Metis: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices*, .

[23] P. Sanders and C. Schulz, *Kahip–karlsruhe high quality partitioning*, *URL http://algo2. iti. kit. edu/kahip* **31** (2016).

[24] D. Kong, X. Xie, and Z. Zhang, *Clustering-based partitioning for large web graphs*, in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pp. 593–606, IEEE, 2022.

[25] G.-P. B. Γκουντρουμάνης, *Multi-level partitioning methodologies and their applications in modern ic design*, B.S. thesis, 2023.

[26] M. Bae, M. Jeong, and S. Oh, *Label propagation-based parallel graph partitioning for large-scale graph data*, *IEEE Access* **8** (2020) 72801–72813.

[27] A. Chhabra, F. Kurpicz, C. Schulz, D. Schweisgut, and D. Seemaier, *Partitioning trillion edge graphs on edge devices*, *arXiv preprint arXiv:2410.07732* (2024).

[28] P. Sanders and D. Seemaier, *Distributed deep multilevel graph partitioning*, in *European conference on parallel processing*, pp. 443–457, Springer, 2023.

[29] R. Liang, A. Agnesina, and H. Ren, *Medpart: A multi-level evolutionary differentiable hypergraph partitioner*, in *Proceedings of the 2024 International Symposium on Physical Design*, pp. 3–11, 2024.

[30] A. Aghdaei, Z. Zhao, and Z. Feng, *Hypersf: Spectral hypergraph coarsening via flow-based local clustering*, in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 1–9, IEEE, 2021.

[31] M. E. Newman, *Spectral methods for community detection and graph partitioning*, *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics* **88** (2013), no. 4 042822.

[32] M. Deveci, S. Rajamanickam, K. D. Devine, and Ü. V. Çatalyürek, *Multi-jagged: A scalable parallel spatial partitioning algorithm*, *IEEE Transactions on Parallel and Distributed Systems* **27** (2015), no. 3 803–817.

[33] Y. Shan, K. Chen, T. Gong, L. Zhou, T. Zhou, and Y. Wu, *Geometric partitioning: Explore the boundary of optimal erasure code repair*, in *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles*, pp. 457–471, 2021.

[34] Z. Wang, Z. Yang, N. Wang, Y. Du, J. Nie, Z. Wei, Y. Gu, and G. Yu, *Lightweight streaming graph partitioning by fully utilizing knowledge from local view*, in *2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS)*, pp. 614–625, IEEE, 2023.

[35] M. A. K. Patwary, S. Garg, S. K. Battula, and B. Kang, *Sdp: scalable real-time dynamic graph partitioner*, *IEEE Transactions on Services Computing* **16** (2021), no. 1 564–574.

[36] A. Arulselvan, C. W. Commander, L. Elefteriadou, and P. M. Pardalos, *Detecting critical nodes in sparse graphs*, *Computers & Operations Research* **36** (2009), no. 7 2193–2200.

[37] S. Shen and J. C. Smith, *Polynomial-time algorithms for solving a class of critical node problems on trees and series-parallel graphs*, *Networks* **60** (2012), no. 2 103–119.

[38] M. Lalou, M. A. Tahraoui, and H. Kheddouci, *Component-cardinality-constrained critical node problem in graphs*, *Discrete applied mathematics* **210** (2016) 150–163.

[39] A. Arulselvan, C. W. Commander, P. M. Pardalos, and O. Shylo, *Managing network risk via critical node identification*, *Risk management in telecommunication networks, Springer* (2007) 79–92.

[40] R. G. Kumar, P. Bhardwaj, S. Basu, and A. Pavan, *Disrupting diffusion: Critical nodes in network*, in *2020 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, pp. 397–404, IEEE, 2020.

[41] R. C. Guinovart Gutierrez, *The Critical Node Detection Problem applied to COVID-19 diffusion.* PhD thesis, Politecnico di Torino, 2022.

[42] H. Moumen, B. Benreguia, L. Saadi, and A. Bounceur, *Locating the diffusion source in networks by critical observers*, in *2023 IEEE Symposium on Computers and Communications (ISCC)*, pp. 543–548, IEEE, 2023.

[43] C. J. Kuhlman, V. Anil Kumar, M. V. Marathe, S. Ravi, and D. J. Rosenkrantz, *Finding critical nodes for inhibiting diffusion of complex contagions in social networks*, in *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2010, Barcelona, Spain, September 20-24, 2010, Proceedings, Part II 21*, pp. 111–127, Springer, 2010.

[44] E. Hamouda, *A critical node-centric approach to enhancing network security*, in *International Conference on the Dynamics of Information Systems*, pp. 116–130, Springer, 2023.

[45] H. Wang and Z. Zhou, *Identification of key risk nodes and invulnerability analysis of construction supply chain networks*, *Buildings* **14** (2024), no. 7 1997.

[46] B. Wu, Z. Xiao, P. Lin, Z. Tang, and K. Li, *Critical path awareness techniques for large-scale graph partitioning*, *IEEE Transactions on Sustainable Computing* **8** (2023), no. 3 412–422.

[47] X. Liu, Z. Hong, J. Liu, Y. Lin, A. Rodríguez-Patón, Q. Zou, and X. Zeng, *Computational methods for identifying the critical nodes in biological networks*, *Briefings in bioinformatics* **21** (2020), no. 2 486–497.

[48] B. Fan, N. Shu, Z. Li, and F. Li, *Critical nodes identification for power grid based on electrical topology and power flow distribution*, *IEEE Systems Journal* **17** (2022), no. 3 4874–4884.

[49] L. Wang, S. Zheng, Y. Wang, and L. Wang, *Identification of critical nodes in multimodal transportation network*, *Physica A: Statistical mechanics and its applications* **580** (2021) 126170.

[50] S. Shukla, *Angle based critical nodes detection (abcnd) for reliable industrial wireless sensor networks*, *Wireless Personal Communications* **130** (2023), no. 2 757–775.

[51] M. Jorgic, M. Hauspie, D. Simplot-Ryl, and I. Stojmenovic, *Localized algorithms for detection of critical nodes and links for connectivity in ad hoc networks*, in *Mediterranean Ad Hoc Networking Workshop*, p. 12, 2004.

[52] D. Purevsuren and G. Cui, *Efficient heuristic algorithm for identifying critical nodes in planar networks*, *Computers & Operations Research* **106** (2019) 143–153.

[53] W. Chen, M. Jiang, C. Jiang, and J. Zhang, *Critical node detection problem for complex network in undirected weighted networks*, *Physica A: Statistical Mechanics and its Applications* **538** (2020) 122862.

[54] W. Pullan, *Heuristic identification of critical nodes in sparse real-world graphs*, *Journal of Heuristics* **21** (2015) 577–598.

[55] G. U. Alozie, A. Arulselvan, K. Akartunalı, and E. L. Pasiliao Jr, *A heuristic approach for the distance-based critical node detection problem in complex networks*, *Journal of the Operational Research Society* **73** (2022), no. 6 1347–1361.

[56] L. Faramondi, R. Setola, S. Panzieri, F. Pascucci, and G. Oliva, *Finding critical nodes in infrastructure networks*, *International Journal of Critical Infrastructure Protection* **20** (2018) 3–15.

[57] J. Rezaei, F. Zare-Mirakabad, S. A. MirHassani, and S.-A. Marashi, *Eia-cndp: An exact iterative algorithm for critical node detection problem*, *Computers & Operations Research* **127** (2021) 105138.

[58] C. Jiang, Z. Liu, J. Wang, H. Yu, and X. Guo, *An optimal approach for the critical node problem using semidefinite programming*, *Physica A: Statistical Mechanics and its Applications* **471** (2017) 315–324.

[59] B. Addis, R. Aringhieri, A. Grosso, and P. Hosteins, *Hybrid constructive heuristics for the critical node problem*, *Annals of Operations Research* **238** (2016) 637–649.

[60] M. Ventresca and D. Aleman, *Efficiently identifying critical nodes in large complex networks*, *Computational Social Networks* **2** (2015) 1–16.

[61] S. Harenberg, G. Bello, L. Gjeltema, S. Ranshous, J. Harlalka, R. Seay, K. Padmanabhan, and N. Samatova, *Community detection in large-scale networks: a survey and empirical evaluation*, *Wiley Interdisciplinary Reviews: Computational Statistics* **6** (2014), no. 6 426–439.

[62] P. Bedi and C. Sharma, *Community detection in social networks*, *Wiley interdisciplinary reviews: Data mining and knowledge discovery* **6** (2016), no. 3 115–135.

[63] Z. Lu, X. Sun, Y. Wen, G. Cao, and T. La Porta, *Algorithms and applications for community detection in weighted networks*, *IEEE Transactions on Parallel and Distributed Systems* **26** (2014), no. 11 2916–2926.

[64] A. Karataş and S. Şahin, *Application areas of community detection: A review*, in *2018 International congress on big data, deep learning and fighting cyber terrorism (IBIGDELFT)*, pp. 65–70, IEEE, 2018.

[65] N. P. Nguyen, T. N. Dinh, Y. Shen, and M. T. Thai, *Dynamic social community detection and its applications*, *PloS one* **9** (2014), no. 4 e91431.

[66] Y. Atay, I. Koc, I. Babaoglu, and H. Kodaz, *Community detection from biological and social networks: A comparative analysis of metaheuristic algorithms*, *Applied Soft Computing* **50** (2017) 194–211.

[67] P. Zhu, X. Wang, Q. Zhi, J. Ma, and Y. Guo, *Analysis of epidemic spreading process in multi-communities*, *Chaos, Solitons & Fractals* **109** (2018) 231–237.

[68] Y. Wang and X. Han, *Attractive community detection in academic social network*, *Journal of Computational Science* **51** (2021) 101331.

[69] F. Liu, Z. Li, B. Wang, J. Wu, J. Yang, J. Huang, Y. Zhang, W. Wang, S. Xue, S. Nepal, et al., *eriskcom: an e-commerce risky community detection platform*, *The VLDB Journal* **31** (2022), no. 5 1085–1101.

[70] S. Bahadori, H. Zare, and P. Moradi, *Podcd: Probabilistic overlapping dynamic community detection*, *Expert Systems with Applications* **174** (2021) 114650.

[71] J. Yang and J. Leskovec, *Defining and evaluating network communities based on ground-truth*, in *Proceedings of the ACM SIGKDD workshop on mining data semantics*, pp. 1–8, 2012.

[72] H. Van Lierde, T. W. Chow, and G. Chen, *Scalable spectral clustering for overlapping community detection in large-scale networks*, *IEEE Transactions on Knowledge and Data Engineering* **32** (2019), no. 4 754–767.

[73] T. Chakraborty, S. Ghosh, and N. Park, *Ensemble-based overlapping community detection using disjoint community structures*, *Knowledge-Based Systems* **163** (2019) 241–251.

[74] M. N. Al-Andoli, J. A. Alsayaydeh, I. M. Alwayle, C. K. N. C. K. Mohd, F. Abuhoureyah, et al., *Robust overlapping community detection in complex networks with graph convolutional networks and fuzzy c-means*, *IEEE Access* (2024).

[75] D. Lai and C. Nardini, *A corrected normalized mutual information for performance evaluation of community detection*, *Journal of Statistical Mechanics: Theory and Experiment* **2016** (2016), no. 9 093403.

[76] X. You, Y. Ma, and Z. Liu, *A three-stage algorithm on community detection in social networks*, *Knowledge-Based Systems* **187** (2020) 104822.

[77] B. Cai, L. Zeng, Y. Wang, H. Li, and Y. Hu, *Community detection method based on node density, degree centrality, and k-means clustering in complex network*, *Entropy* **21** (2019), no. 12 1145.

[78] K. Berahmand, A. Bouyer, and M. Vasighi, *Community detection in complex networks by detecting and expanding core nodes through extended local similarity of nodes*, *IEEE Transactions on Computational Social Systems* **5** (2018), no. 4 1021–1033.

[79] J. Zhang and Y. Chen, *Modularity based community detection in heterogeneous networks*, *Statistica Sinica* **30** (2020), no. 2 601–629.

[80] M. Mazza, G. Cola, and M. Tesconi, *Modularity-based approach for tracking communities in dynamic social networks*, *Knowledge-Based Systems* **281** (2023) 111067.

[81] Y. Li, K. He, K. Kloster, D. Bindel, and J. Hopcroft, *Local spectral clustering for overlapping community detection*, *ACM Transactions on Knowledge Discovery from Data (TKDD)* **12** (2018), no. 2 1–27.

[82] F. Hu, J. Liu, L. Li, and J. Liang, *Community detection in complex networks using node2vec with spectral clustering*, *Physica A: Statistical Mechanics and its Applications* **545** (2020) 123633.

[83] I. B. El Kouni, W. Karoui, and L. B. Romdhane, *Node importance based label propagation algorithm for overlapping community detection in networks*, *Expert Systems with Applications* **162** (2020) 113020.

[84] Y. Xing, F. Meng, Y. Zhou, M. Zhu, M. Shi, and G. Sun, *A node influence based label propagation algorithm for community detection in networks*, *The Scientific World Journal* **2014** (2014), no. 1 627581.

[85] E. Jokar and M. Mosleh, *Community detection in social networks based on improved label propagation algorithm and balanced link density*, *Physics Letters A* **383** (2019), no. 8 718–727.

[86] H. Li, R. Zhang, Z. Zhao, and X. Liu, *Lpa-mni: an improved label propagation algorithm based on modularity and node importance for community detection*, *Entropy* **23** (2021), no. 5 497.

[87] K. Rizman Žalik, *Evolution algorithm for community detection in social networks using node centrality*, *Intelligent methods and big data in industrial applications* (2019) 73–87.

[88] Y. Zhang, Y. Liu, J. Li, J. Zhu, C. Yang, W. Yang, and C. Wen, *Wocda: A whale optimization based community detection algorithm*, *Physica A: Statistical Mechanics and its Applications* **539** (2020) 122937.

[89] D. A. Abduljabbar, S. Z. M. Hashim, and R. Sallehuddin, *Nature-inspired optimization algorithms for community detection in complex networks: a review and future trends*, *Telecommunication Systems* **74** (2020) 225–252.

[90] S. Souravlas, S. D. Anastasiadou, T. Economides, and S. Katsavounis, *Probabilistic community detection in social networks*, *IEEE Access* **11** (2023) 25629–25641.

[91] M. Dhilber and S. D. Bhavani, *Community detection in social networks using deep learning*, in *Distributed Computing and Internet Technology: 16th International Conference, ICDCIT 2020, Bhubaneswar, India, January 9–12, 2020, Proceedings 16*, pp. 241–250, Springer, 2020.

[92] A. Ferraro, V. Moscato, and G. Sperlì, *Deep learning-based community detection approach on multimedia social networks*, *Applied Sciences* **11** (2021), no. 23 11447.

[93] Y. Yin, Y. Zhao, H. Li, and X. Dong, *Multi-objective evolutionary clustering for large-scale dynamic community detection*, *Information Sciences* **549** (2021) 269–287.

[94] X. Li, X. Zhen, X. Qi, H. Han, L. Zhang, and Z. Han, *Dynamic community detection based on graph convolutional networks and contrastive learning*, *Chaos, Solitons & Fractals* **176** (2023) 114157.

[95] H. Lu, Q. Zhao, X. Sang, and J. Lu, *Community detection in complex networks using nonnegative matrix factorization and density-based clustering algorithm*, *Neural Processing Letters* **51** (2020) 1731–1748.

[96] S. M. Saif, M. E. Samie, and A. Hamzeh, *A subgraphs-density based overlapping community detection algorithm for large-scale complex networks*, *Computing* **105** (2023), no. 1 151–185.

[97] X. Li, S. Zhou, J. Liu, G. Lian, G. Chen, and C.-W. Lin, *Communities detection in social network based on local edge centrality*, *Physica A: Statistical Mechanics and its Applications* **531** (2019) 121552.

[98] M. Arasteh and S. Alizadeh, *A fast divisive community detection algorithm based on edge degree betweenness centrality*, *Applied Intelligence* **49** (2019) 689–702.

[99] M. Ahsan, T. Singh, and M. Kumari, *Influential node detection in social network during community detection*, in *2015 International Conference on Cognitive Computing and Information Processing (CCIP)*, pp. 1–6, IEEE, 2015.

[100] Y. Jiang, C. Jia, and J. Yu, *An efficient community detection method based on rank centrality*, *Physica A: statistical mechanics and its applications* **392** (2013), no. 9 2182–2194.

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

[102] A. Mester, A. Pop, B.-E.-M. Mursa, H. Greblă, L. Dioşan, and C. Chira, *Network analysis based on important node selection and community detection*, *Mathematics* **9** (2021), no. 18 2294.

[103] S. Aghaalizadeh, S. T. Afshord, A. Bouyer, and B. Anari, *A three-stage algorithm for local community detection based on the high node importance ranking in social networks*, *Physica A: Statistical Mechanics and its Applications* **563** (2021) 125420.

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

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

[106] B. Karrer and M. E. Newman, *Stochastic blockmodels and community structure in networks*, *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics* **83** (2011), no. 1 016107.

[107] X. Que, F. Checconi, F. Petrini, and J. A. Gunnels, *Scalable community detection with the louvain algorithm*, in *2015 IEEE international parallel and distributed processing symposium*, pp. 28–37, IEEE, 2015.

[108] M. E. Newman and M. Girvan, *Finding and evaluating community structure in networks*, *Physical review E* **69** (2004), no. 2 026113.

[109] W. Lin, X. Kong, P. S. Yu, Q. Wu, Y. Jia, and C. Li, *Community detection in incomplete information networks*, in *Proceedings of the 21st international conference on World Wide Web*, pp. 341–350, 2012.

[110] L. Zhang, B. Li, H. Yang, F. Cheng, C. Zhang, and R. Cao, *Multi-objective optimization of local overlapping community detection: A formal model and novel evolutionary algorithm*, *IEEE Transactions on Network Science and Engineering* **10** (2023), no. 4 2124–2140.

[111] W. Li, J. Wang, and J. Cai, *New label propagation algorithms based on the law of universal gravitation for community detection*, *Physica A: Statistical Mechanics and its Applications* **627** (2023) 129140.

[112] K. Guo, X. Huang, L. Wu, and Y. Chen, *Local community detection algorithm based on local modularity density*, *Applied Intelligence* **52** (2022), no. 2 1238–1253.

[113] K. Guo, L. He, Y. Chen, W. Guo, and J. Zheng, *A local community detection algorithm based on internal force between nodes*, *Applied Intelligence* **50** (2020) 328–340.

[114] T. Zhang and B. Wu, *A method for local community detection by finding core nodes*, in *2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pp. 1171–1176, IEEE, 2012.

[115] X. Ding, J. Zhang, and J. Yang, *A robust two-stage algorithm for local community detection*, *Knowledge-Based Systems* **152** (2018) 188–199.

[116] W. Li, S. Jiang, and Q. Jin, *Overlap community detection using spectral algorithm based on node convergence degree*, *Future Generation Computer Systems* **79** (2018) 408–416.

[117] R. J. Boroujeni and S. Soleimani, *The role of influential nodes and their influence domain in community detection: An approximate method for maximizing modularity*, *Expert Systems with Applications* **202** (2022) 117452.

[118] S. Ahajjam, M. El Haddad, and H. Badir, *A new scalable leader-community detection approach for community detection in social networks*, *Social Networks* **54** (2018) 41–49.

[119] S. Ahajjam, M. El Haddad, and H. Badir, *Leadersrank: Towards a new approach for community detection in social networks*, in *2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA)*, pp. 1–8, IEEE, 2015.

[120] H. Sun, H. Du, J. Huang, Y. Li, Z. Sun, L. He, X. Jia, and Z. Zhao, *Leader-aware community detection in complex networks*, *Knowledge and Information Systems* **62** (2020) 639–668.

[121] J. Baumes, M. K. Goldberg, M. S. Krishnamoorthy, M. Magdon-Ismail, and N. Preston, *Finding communities by clustering a graph into overlapping subgraphs.*, *IADIS AC* **5** (2005) 97–104.

[122] S. Shelke and V. Attar, *Source detection of rumor in social network–a review*, *Online Social Networks and Media* **9** (2019) 30–42.

[123] P. C. Pinto, P. Thiran, and M. Vetterli, *Locating the source of diffusion in large-scale networks*, *Physical review letters* **109** (2012), no. 6 068702.

[124] F. Viti, M. Rinaldi, F. Corman, and C. M. Tampère, *Assessing partial observability in network sensor location problems*, *Transportation research part B: methodological* **70** (2014) 65–89.

[125] C. Ranasinghe and C. Kray, *Location information quality: A review*, *Sensors* **18** (2018), no. 11.

[126] Y. Chai, Y. Wang, and L. Zhu, *Information sources estimation in time-varying networks*, *IEEE Transactions on Information Forensics and Security* **16** (2021) 2621–2636.

[127] Y. Zhou, C. Wu, Q. Zhu, Y. Xiang, and S. W. Loke, *Rumor source detection in networks based on the seir model*, *IEEE access* **7** (2019) 45240–45258.

[128] J. Jiang, S. Wen, S. Yu, Y. Xiang, and W. Zhou, *Identifying propagation sources in networks: State-of-the-art and comparative studies*, *IEEE Communications Surveys & Tutorials* **19** (2016), no. 1 465–481.

[129] M. Fang, P. Shi, W. Shang, X. Yu, T. Wu, and Y. Liu, *Locating the source of asynchronous diffusion process in online social networks*, *IEEE Access* **6** (2018) 17699–17710.

[130] D. Frąszczak, *Rpasdt—rumor propagation and source detection toolkit*, *SoftwareX* **17** (2022) 100988.

[131] Z. Wu, D. Pi, J. Chen, M. Xie, and J. Cao, *Rumor detection based on propagation graph neural network with attention mechanism*, *Expert systems with applications* **158** (2020) 113595.

[132] P.-D. Yu, L. Zheng, and C. W. Tan, *Contagion source detection by maximum likelihood estimation and starlike graph approximation*, in *2024 58th Annual Conference on Information Sciences and Systems (CISS)*, pp. 1–6, IEEE, 2024.

[133] H. Alasmary, A. Khormali, A. Anwar, J. Park, J. Choi, A. Abusnaina, A. Awad, D. Nyang, and A. Mohaisen, *Analyzing and detecting emerging internet of things malware: A graph-based approach*, *IEEE Internet of Things Journal* **6** (2019), no. 5 8977–8988.

[134] M. Yang, Y. Feng, A. S. Rao, S. Rajasegarar, S. Tian, and Z. Zhou, *Evolving graph-based video crowd anomaly detection*, *The Visual Computer* **40** (2024), no. 1 303–318.

[135] W. Zhang, *Graph based approach to real-time metro passenger flow anomaly detection*, in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pp. 2744–2749, IEEE, 2021.

[136] N. Rastogi, *Exploring information centrality for intrusion detection in large networks*, *arXiv preprint arXiv:1904.12138* (2019).

[137] K. Shinan, K. Alsubhi, and M. U. Ashraf, *Botsward: Centrality measures for graph-based bot detection using machine learning.*, *Computers, Materials & Continua* **75** (2023), no. 1.

[138] W. Zhang, G. Zhu, M. Xing, J. Yang, H. Yu, and Z. Zhu, *A new diffusion strategy using an epidemic spreading model for encryption*, *Entropy* **26** (2024), no. 9 760.

[139] A. Reshetnikov, V. Berdutin, A. Zaporozhtsev, S. Romanov, O. Abaeva, N. Prisyazhnaya, and N. Vyatkina, *Predictive algorithm for the regional spread of coronavirus infection across the russian federation*, *BMC Medical Informatics and Decision Making* **23** (2023), no. 1 48.

[140] S. Wen, M. S. Haghighi, C. Chen, Y. Xiang, W. Zhou, and W. Jia, *A sword with two edges: Propagation studies on both positive and negative information in online social networks*, *IEEE Transactions on Computers* **64** (2014), no. 3 640–653.

[141] Y. Shao, L. Chen, Y. Chen, and W. Liu, *Social influence source locating based on network sparsification and stratification*, *Expert Systems with Applications* **208** (2022) 118087.

[142] J. Wang, S. Zhang, W. Chen, D. Kong, and Z. Yu, *Convex optimization–based multi-user detection in underwater acoustic sensor networks*, *International Journal of Distributed Sensor Networks* **14** (2018), no. 2 1550147718757665.

[143] W. Li, C. Guo, Y. Liu, X. Zhou, Q. Jin, and M. Xin, *Rumor source localization in social networks based on infection potential energy*, *Information Sciences* **634** (2023) 172–188.

[144] J. Pedro and N. Costa, *Hybrid amplifier placement in mesh dwdm networks using integer linear programming models*, *Journal of Optical Communications and Networking* **15** (2023), no. 10 E29–E39.

[145] N. P. Theodorakatos, M. D. Lytras, K. T. Kantoutsis, A. P. Moschoudis, and C. A. Theodoridis, *Optimization-based optimal pmu placement for power state estimation and fault observability*, in *AIP Conference Proceedings*, vol. 2872, AIP Publishing, 2023.

[146] K. Golmohammadi, O. R. Zaiane, and D. Díaz, *Detecting stock market manipulation using supervised learning algorithms*, in *2014 International Conference on Data Science and Advanced Analytics (DSAA)*, pp. 435–441, IEEE, 2014.

[147] S. Hosseini and M. Azizi, *The hybrid technique for ddos detection with supervised learning algorithms*, *Computer Networks* **158** (2019) 35–45.

[148] S. Sawant, A. Sethi, S. Banerjee, and S. Tallur, *Unsupervised learning framework for temperature compensated damage identification and localization in ultrasonic guided wave shm with transfer learning*, *Ultrasonics* **130** (2023) 106931.

[149] Y. Mohammadi, S. M. Miraftabzadeh, M. H. Bollen, and M. Longo, *Voltage-sag source detection: Developing supervised methods and proposing a new unsupervised learning*, *Sustainable Energy, Grids and Networks* **32** (2022) 100855.

[150] Q. Feng, L. Han, B. Pan, and B. Zhao, *Microseismic source location using deep reinforcement learning*, *IEEE Transactions on Geoscience and Remote Sensing* **60** (2022) 1–9.

[151] Y. Wu, J. Wei, J. Pan, and P. Chen, *Research on microseismic source locations based on deep reinforcement learning*, *IEEE Access* **7** (2019) 39962–39973.

[152] K. Gaurav, A. Kumar, and R. Singh, *Single and multiple odor source localization using hybrid nature-inspired algorithm*, *Sādhanā* **45** (2020) 1–19.

[153] J. Kim, J. Park, and E. Lee, *A new hybrid algorithm for software fault localization*, in *Proceedings of the 9th International Conference on Ubiquitous Information Management and Communication*, pp. 1–8, 2015.

[154] R. Frisch, *Stochastic machines dedicated to Bayesian inference for source localization and separation.* PhD thesis, Université Grenoble Alpes, 2019.

[155] R. B. Anderson, C. Pehlivantürk, and M. Pryor, *Optimization strategies for bayesian source localization algorithms*, *IEEE Transactions on Automation Science and Engineering* **20** (2022), no. 1 394–403.

[156] Y. Bai, W. Lu, J. Li, Z. Chang, and H. Wang, *Groundwater contamination source identification using improved differential evolution markov chain algorithm*, *Environmental Science and Pollution Research* (2022) 1–14.

[157] X. Wu, J. Guo, K. Xian, and X. Zhou, *Hierarchical travel demand estimation using multiple data sources: A forward and backward propagation algorithmic framework on a layered computational graph*, *Transportation Research Part C: Emerging Technologies* **96** (2018) 321–346.

[158] S. Lim, J. Hao, Z. Lu, X. Zhang, and Z. Zhang, *Approximating the k-minimum distance rumor source detection in online social networks*, in *2018 27th International Conference on Computer Communication and Networks (ICCCN)*, pp. 1–9, IEEE, 2018.

[159] P.-Y. Chen, S. Choudhury, and A. O. Hero, *Multi-centrality graph spectral decompositions and their application to cyber intrusion detection*, in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4553–4557, IEEE, 2016.

[160] J. Sun, H. Lu, and X. Liu, *Saliency region detection based on markov absorption probabilities*, *IEEE Transactions on Image Processing* **24** (2015), no. 5 1639–1649.

[161] J. Wu, G. Han, P. Liu, H. Yang, H. Luo, and Q. Li, *Saliency detection with bilateral absorbing markov chain guided by depth information*, *Sensors* **21** (2021), no. 3 838.

[162] W. Liu, L. Ma, L. Chen, B. Chen, B. Jeon, and J. Qiang, *A novel scheme for essential protein discovery based on multi-source biological information*, *Journal of Theoretical Biology* **504** (2020) 110414.

[163] T. Foltỳnek, N. Meuschke, and B. Gipp, *Academic plagiarism detection: a systematic literature review*, *ACM Computing Surveys (CSUR)* **52** (2019), no. 6 1–42.

[164] W. Zang, P. Zhang, C. Zhou, and L. Guo, *Locating multiple sources in social networks under the sir model: A divide-and-conquer approach*, *Journal of Computational Science* **10** (2015) 278–287.

[165] H. Soleimani, J. Hensman, and S. Saria, *Scalable joint models for reliable uncertainty-aware event prediction*, *IEEE transactions on pattern analysis and machine intelligence* **40** (2017), no. 8 1948–1963.

[166] E. Yoo, W. Rand, M. Eftekhar, and E. Rabinovich, *Evaluating information diffusion speed and its determinants in social media networks during humanitarian crises*, *Journal of operations management* **45** (2016) 123–133.

[167] S. S. Ali, T. Anwar, and S. A. M. Rizvi, *A revisit to the infection source identification problem under classical graph centrality measures*, *Online Social Networks and Media* **17** (2020) 100061.

[168] F. Yang, R. Zhang, Y. Yao, and Y. Yuan, *Locating the propagation source on complex networks with propagation centrality algorithm*, *Knowledge-Based Systems* **100** (2016) 112–123.

[169] S. Xu, C. Teng, Y. Zhou, J. Peng, Y. Zhang, and Z.-K. Zhang, *Identifying the diffusion source in complex networks with limited observers*, *Physica A: Statistical Mechanics and its Applications* **527** (2019) 121267.

[170] F. Yang, S. Yang, Y. Peng, Y. Yao, Z. Wang, H. Li, J. Liu, R. Zhang, and C. Li, *Locating the propagation source in complex networks with a direction-induced search based gaussian estimator*, *Knowledge-Based Systems* **195** (2020) 105674.

[171] R. Paluch, X. Lu, K. Suchecki, B. K. Szymański, and J. A. Hołyst, *Fast and accurate detection of spread source in large complex networks*, *Scientific reports* **8** (2018), no. 1 2508.

[172] W. Xu and H. Chen, *Scalable rumor source detection under independent cascade model in online social networks*, in *2015 11th international conference on mobile ad-hoc and sensor networks (MSN)*, pp. 236–242, IEEE, 2015.

[173] W. Zang, P. Zhang, C. Zhou, and L. Guo, *Discovering multiple diffusion source nodes in social networks*, *Procedia Computer Science* **29** (2014) 443–452.

[174] W. Luo and W. P. Tay, *Finding an infection source under the sis model*, in *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 2930–2934, IEEE, 2013.

[175] K. Zhu, Z. Chen, and L. Ying, *Catch'em all: Locating multiple diffusion sources in networks with partial observations*, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, 2017.

[176] J. Yick, B. Mukherjee, and D. Ghosal, *Wireless sensor network survey*, *Computer networks* **52** (2008), no. 12 2292–2330.

[177] L. Yong-Min, W. Shu-Ci, and N. Xiao-Hong, *The architecture and characteristics of wireless sensor network*, in *2009 International Conference on Computer Technology and Development*, vol. 1, pp. 561–565, IEEE, 2009.

[178] A. Lanzolla and M. Spadavecchia, *Wireless sensor networks for environmental monitoring*, 2021.

[179] D. Thakur, Y. Kumar, A. Kumar, and P. K. Singh, *Applicability of wireless sensor networks in precision agriculture: A review*, *Wireless Personal Communications* **107** (2019) 471–512.

[180] M. Abdulkarem, K. Samsudin, F. Z. Rokhani, and M. F. A Rasid, *Wireless sensor network for structural health monitoring: A contemporary review of technologies, challenges, and future direction*, *Structural health monitoring* **19** (2020), no. 3 693–735.

[181] A. Hilmani, A. Maizate, and L. Hassouni, *Automated real-time intelligent traffic control system for smart cities using wireless sensor networks*, *Wireless Communications and mobile computing* **2020** (2020), no. 1 8841893.

[182] F. C. Obodoeze, F. E. Ozioko, C. N. Mba, F. A. Okoye, and S. C. Asogwa, *Wireless sensor networks (wsns) in industrial automation: Case study of nigeria oil and gas industry*, *International Journal of Engineering Research and Technology* **2** (2013), no. 3 1–7.

[183] N. Bhadwal, V. Madaan, P. Agrawal, A. Shukla, and A. Kakran, *Smart border surveillance system using wireless sensor network and computer vision*, in *2019 international conference on Automation, Computational and Technology Management (ICACTM)*, pp. 183–190, IEEE, 2019.

[184] S. H. Lee, S. Lee, H. Song, and H. S. Lee, *Wireless sensor network design for tactical military applications: Remote large-scale environments*, in *MILCOM 2009-2009 IEEE Military communications conference*, pp. 1–7, IEEE, 2009.

[185] C. S. Abella, S. Bonina, A. Cucuccio, S. D'Angelo, G. Giustolisi, A. D. Grasso, A. Imbruglia, G. S. Mauro, G. A. Nastasi, G. Palumbo, et al., *Autonomous*

energy-efficient wireless sensor network platform for home/office automation, *IEEE Sensors Journal* **19** (2019), no. 9 3501–3512.

[186] J. Jiang, H. Wang, X. Mu, and S. Guan, *Logistics industry monitoring system based on wireless sensor network platform*, *Computer Communications* **155** (2020) 58–65.

[187] X. Jiang, J. Taneja, J. Ortiz, A. Tavakoli, P. Dutta, J. Jeong, D. Culler, P. Levis, and S. Shenker, *An architecture for energy management in wireless sensor networks*, *ACM SIGBED Review* **4** (2007), no. 3 31–36.

[188] D. Sandeep and V. Kumar, *Review on clustering, coverage and connectivity in underwater wireless sensor networks: A communication techniques perspective*, *IEEE Access* **5** (2017) 11176–11199.

[189] N. A. A. Aziz and K. A. Aziz, *Managing disaster with wireless sensor networks*, in *13th international conference on advanced communication technology (ICACT2011)*, pp. 202–207, IEEE, 2011.

[190] S. Sharma, R. K. Bansal, and S. Bansal, *Issues and challenges in wireless sensor networks*, in *2013 international conference on machine intelligence and research advancement*, pp. 58–62, IEEE, 2013.

[191] J. S. Manoharan, *A novel load balancing aware graph theory based node deployment in wireless sensor networks*, *Wireless Personal Communications* **128** (2023), no. 2 1171–1192.

[192] J.-W. Lin, P. R. Chelliah, M.-C. Hsu, and J.-X. Hou, *Efficient fault-tolerant routing in iot wireless sensor networks based on bipartite-flow graph modeling*, *IEEE access* **7** (2019) 14022–14034.

[193] R. R. Swain, T. Dash, and P. M. Khilar, *An effective graph-theoretic approach towards simultaneous detection of fault (s) and cut (s) in wireless sensor networks*, *International Journal of Communication Systems* **30** (2017), no. 13 e3273.

[194] T. Shi, S. Cheng, J. Li, H. Gao, and Z. Cai, *Dominating sets construction in rf-based battery-free sensor networks with full coverage guarantee*, *ACM Transactions on Sensor Networks (TOSN)* **15** (2019), no. 4 1–29.

[195] V. K. Akram, O. Dagdeviren, and B. Tavli, *A coverage-aware distributed k-connectivity maintenance algorithm for arbitrarily large k in mobile sensor networks*, *IEEE/ACM Transactions on Networking* **30** (2021), no. 1 62–75.

[196] M. Z. Ghawy, G. A. Amran, H. AlSalman, E. Ghaleb, J. Khan, A. A. Al-Bakhrani, A. M. Alziadi, A. Ali, and S. S. Ullah, *An effective wireless sensor network routing*

protocol based on particle swarm optimization algorithm, *Wireless Communications and Mobile Computing* **2022** (2022), no. 1 8455065.

[197] D. Lin, Z. Lin, L. Kong, and Y. L. Guan, *Cmstr: A constrained minimum spanning tree based routing protocol for wireless sensor networks*, *Ad Hoc Networks* **146** (2023) 103160.

[198] A. M. Jubair, R. Hassan, A. H. M. Aman, H. Sallehudin, Z. G. Al-Mekhlafi, B. A. Mohammed, and M. S. Alsaffar, *Optimization of clustering in wireless sensor networks: techniques and protocols*, *Applied Sciences* **11** (2021), no. 23 11448.

[199] J.-S. Pan, L. Kong, T.-W. Sung, P.-W. Tsai, and V. Snášel, *A clustering scheme for wireless sensor networks based on genetic algorithm and dominating set*, *Journal of Internet Technology* **19** (2018), no. 4 1111–1118.

[200] S. Messous, N. Liouane, A. Pegatoquet, and M. Auguin, *Hop-based routing protocol based on energy efficient minimum spanning tree for wireless sensor network*, in *2018 International Conference on Advanced Systems and Electric Technologies (IC_ASET)*, pp. 421–426, IEEE, 2018.

[201] M. Z. U. Haq, M. Z. Khan, H. U. Rehman, G. Mehmood, A. Binmahfoudh, M. Krichen, and R. Alroobaea, *An adaptive topology management scheme to maintain network connectivity in wireless sensor networks*, *Sensors* **22** (2022), no. 8 2855.

[202] M. I. M. Ismail, R. A. Dzyauddin, S. Samsul, N. A. Azmi, Y. Yamada, M. F. M. Yakub, and N. A. B. A. Salleh, *An rssi-based wireless sensor node localisation using trilateration and multilateration methods for outdoor environment*, *arXiv preprint arXiv:1912.07801* (2019).

[203] W. Wu, Z. Zhang, W. Lee, D. Du, et al., *Optimal coverage in wireless sensor networks*, .

[204] O. Dagdeviren, V. K. Akram, and B. Tavli, *Design and evaluation of algorithms for energy efficient and complete determination of critical nodes for wireless sensor network reliability*, *IEEE Transactions on Reliability* **68** (2018), no. 1 280–290.

[205] A. Yuksel, E. Uzun, and B. Tavli, *The impact of elimination of the most critical node on wireless sensor network lifetime*, in *2015 IEEE Sensors Applications Symposium (SAS)*, pp. 1–5, IEEE, 2015.

[206] H. U. Yildiz, B. Tavli, B. O. Kahjogh, and E. Dogdu, *The impact of incapacitation of multiple critical sensor nodes on wireless sensor network lifetime*, *IEEE Wireless Communications Letters* **6** (2017), no. 3 306–309.

[207] O. Dagdeviren, V. K. Akram, B. Tavli, H. U. Yildiz, and C. Atilgan, *Distributed detection of critical nodes in wireless sensor networks using connected dominating set*, in *2016 IEEE SENSORS*, pp. 1–3, IEEE, 2016.

[208] B. O. Kahjogh, I. Demirkol, D. Careglio, and J. D. Pascual, *The impact of critical node elimination on the latency of wireless sensor networks*, in *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)*, pp. 182–187, IEEE, 2017.

[209] M. Umadevi and M. Devapriya, *An enhanced ant colony based approach to optimize the usage of critical node in wireless sensor networks*, *Procedia Computer Science* **47** (2015) 452–459.

[210] X. Wang, J. Du, R. Zou, and Z. Zhou, *Key node identification of wireless sensor networks based on cascade failure*, *Modern Physics Letters B* **34** (2020), no. 34 2050394.

[211] S. Shukla, *Reliable critical nodes detection for internet of things (iot)*, *Wireless Networks* **27** (2021), no. 4 2931–2946.

[212] L. Han, X. Chen, M. Leng, Y. Deng, T. Hao, and H. Yang, *Research on energy consumption and identifying critical nodes of weighted scale-free topology in wireless sensor network*, *IEEE Access* (2024).

[213] K. Takki, "Future of Industrial Automation: Market Trends, Key Players, and Growth Projections — statzon.com." https://statzon.com/insights/industrial-automation-market, 2024. [Accessed 01-02-2025].

[214] M. Raza, N. Aslam, H. Le-Minh, S. Hussain, Y. Cao, and N. M. Khan, *A critical analysis of research potential, challenges, and future directives in industrial wireless sensor networks*, *IEEE Communications Surveys & Tutorials* **20** (2017), no. 1 39–95.

[215] F. A. Silva, *Industrial wireless sensor networks: Applications, protocols, and standards [book news]*, *IEEE Industrial Electronics Magazine* **8** (2014), no. 4 67–68.

[216] V. C. Gungor and G. P. Hancke, *Industrial wireless sensor networks: Challenges, design principles, and technical approaches*, *IEEE Transactions on industrial electronics* **56** (2009), no. 10 4258–4265.

[217] F. H. El-Fouly and R. A. Ramadan, *Real-time energy-efficient reliable traffic aware routing for industrial wireless sensor networks*, *IEEE Access* **8** (2020) 58130–58145.

[218] Z. A. Dagdeviren, *Weighted connected vertex cover based energy-efficient link monitoring for wireless sensor networks towards secure internet of things*, *IEEE Access* **9** (2021) 10107–10119.

[219] Z. A. Dagdeviren and V. Akram, *Connectivity estimation approaches for internet of things-enabled wireless sensor networks*, in *Emerging Trends in IoT and Integration with Data Science, Cloud Computing, and Big Data Analytics*, pp. 104–122. IGI Global, 2022.

[220] V. Khalilpour Akram, Z. Akusta Dagdeviren, O. Dagdeviren, and M. Challenger, *Pinc: Pickup non-critical node based k-connectivity restoration in wireless sensor networks*, *Sensors* **21** (2021), no. 19 6418.

[221] U. Baroudi, M. Aldarwbi, and M. Younis, *Energy-aware connectivity restoration mechanism for cyber-physical systems of networked sensors and robots*, *IEEE Systems Journal* **14** (2020), no. 3 3093–3104.

[222] R. R. Priyadarshini and N. Sivakumar, *Cluster head selection based on minimum connected dominating set and bi-partite inspired methodology for energy conservation in wsns*, *Journal of King Saud University-Computer and Information Sciences* **33** (2021), no. 9 1132–1144.

[223] Y. Gong, X. Guo, and G. Lai, *A centralized energy-efficient clustering protocol for wireless sensor networks*, *IEEE Sensors Journal* **23** (2022), no. 2 1623–1634.

[224] A. Siddiq and Y. J. Ghazwani, *Hybrid optimized deep neural network based intrusion node detection and modified energy efficient centralized clustering routing protocol for wireless sensor network*, *IEEE Transactions on Consumer Electronics* (2024).

[225] A. Chowdhury and D. De, *Energy-efficient coverage optimization in wireless sensor networks based on voronoi-glowworm swarm optimization-k-means algorithm*, *Ad Hoc Networks* **122** (2021) 102660.

[226] K.-V. Nguyen, C.-H. Nguyen, P. Le Nguyen, T. Van Do, and I. Chlamtac, *Energy-efficient routing in the proximity of a complicated hole in wireless sensor networks*, *Wireless Networks* **27** (2021), no. 4 3073–3089.

[227] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, *Fast unfolding of communities in large networks*, *Journal of statistical mechanics: theory and experiment* **2008** (2008), no. 10 P10008.

[228] U. N. Raghavan, R. Albert, and S. Kumara, *Near linear time algorithm to detect community structures in large-scale networks*, *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics* **76** (2007), no. 3 036106.

[229] L. Zhou, P. Yang, K. Lü, L. Wang, and H. Chen, *A fast approach for detecting overlapping communities in social networks based on game theory*, in *Data Science: 30th British International Conference on Databases, BICOD 2015, Edinburgh, UK, July 6-8, 2015, Proceedings 30*, pp. 62–73, Springer, 2015.

[230] M. E. Newman, *Finding community structure in networks using the eigenvectors of matrices*, *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics* **74** (2006), no. 3 036104.

[231] A. Clauset, M. E. Newman, and C. Moore, *Finding community structure in very large networks*, *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics* **70** (2004), no. 6 066111.

[232] D. Lusseau and L. Conradt, *The emergence of unshared consensus decisions in bottlenose dolphins*, *Behavioral Ecology and Sociobiology* **63** (2009) 1067–1077.

[233] Z. Wang, Y. Wang, J. Ma, W. Li, N. Chen, and X. Zhu, *Link prediction based on weighted synthetical influence of degree and h-index on complex networks*, *Physica A: Statistical Mechanics and its Applications* **527** (2019) 121184.

[234] J. Zhu and L. Wang, *Identifying influential nodes in complex networks based on node itself and neighbor layer information*, *Symmetry* **13** (2021), no. 9 1570.

[235] C. Gao, J. Liu, and N. Zhong, *Network immunization and virus propagation in email networks: experimental evaluation and analysis*, *Knowledge and information systems* **27** (2011) 253–279.

[236] V. K. Akram and O. Dagdeviren, *Breadth-first search-based single-phase algorithms for bridge detection in wireless sensor networks*, *Sensors* **13** (2013), no. 7 8786–8813.