Thesis submitted to the

UNIVERSITY OF MOHAMED BOUDIAF - M'SILA, ALGERIA

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

in Partial Fulfillment of the Requirements for the Degree of:

Master's in Computer Science

Specialization: Business Intelligence and Optimization

By

Lebcir, Kheira

Menasri, Boutheyna

Entitled

Design of Fault-Tolerant Wireless Sensor Networks Using the GRASP Algorithm

Under the supervision of

Raouf Ouanis Lakehal Ayat

Jury Members

First name Last name	University of M'sila	President
Raouf Ouanis Lakehal Ayat	University of M'sila	Reporter
First name Last name	University of M'sila	Examiner

June, 2025

الملخص

تهدف هذه الدراسة إلى تطوير طريقة فعّالة لتصميم شبكات الاستشعار اللاسلكية بحيث تظل متماسكة وموثوقة حتى في حال تعرض بعض العقد للأعطال. تعتمد الدراسة على خوارزمية بحث ميتاهيوريستية تُعرف باسم "الإجراء التكيفي العشوائي الجشع" ، حيث يتم بناء حلول أولية تعتمد على ربط العقد القريبة بشكل جشع مع إدخال عنصر من العشوائية، ثم تحسين هذه الحلول تدريجيًا عبر تحسينات محلية لضمان أقصى قدر من الاتصال والتكرار. تم تقييم أداء الخوارزمية من خلال محاكاة سيناريوهات أعطال عشوائية وموجهة، وأظهرت النتائج قدرة الخوارزمية على الحفاظ على استمرارية الشبكة ورفع نسبة توصيل البيانات حتى في الظروف الصعبة. كما تم مقارنة الأداء مع أعمال بحثية سابقة، مما أبرز تفوق المنهجية المقترحة من حيث المرونة والموثوقية. تقدم المذكرة شرحًا مفصلًا للمنهجية والخطوات البرمجية المستخدمة، مع تحليل شامل للأداء والنتائج.

الكلمات المفتاحية: شبكات الاستشعار اللاسلكية، تحمل الأعطال، خوارزميات التحسين ,الإجراء التكيفي العشوائي الجشع، تصميم الشبكات.

Abstract

This thesis presents a fault-tolerant design approach for Wireless Sensor Networks (WSNs) using the Greedy Randomized Adaptive Search Procedure (GRASP) metaheuristic. It addresses the challenge of maintaining network connectivity and reliability in the face of random and targeted node failures. The WSN is modeled as an undirected graph, and GRASP is applied to iteratively construct and enhance network topologies via greedy initialization and local improvements. The approach integrates backup paths to ensure redundancy and robustness. Performance is evaluated through simulation under diverse failure scenarios, demonstrating GRASP's effectiveness in preserving high delivery rates and network resilience. Furthermore, the proposed method is compared with existing approaches in the literature, highlighting its superior adaptability and fault tolerance. The thesis presents a detailed methodology, implementation process, and comprehensive performance analysis.

Keywords: Wireless Sensor Networks, Fault Tolerance, GRASP, Metaheuristic, Network Optimization.

Dedication

*"To the soul of **my father**, may Allah have mercy on him. You accompanied me through the darkest moments of my life, so I dedicate this work to you, that you may share with me its brightest joy.To **my mother** my friend, my comfort, the light of my heart .No words can hold all the warmth you gave me. You were the peace of my storm, the harbor I returned to. This is yours before it is mine.To my steady side, to my mountain my brother **Mohamed**, whose name I carry in hope, just as the Prophet Muhammad (peace be upon him) guided a nation. When I lost my way, I turned to you.To **Ali**, my younger brother who outgrew me in height, and to the shoulder I lean on, my safest place, every Sunday morning on our way to college, and in truth, your shoulder carried not only my head, but also my moods and worries ...not just on Sundays, but through all days and years.To the two halves of my heart, **my beloved sisters**, I love you dearly, close in my heart despite the distance.To those whom I am proud to be their aunt, **I love you deeply**.To the **Majaz Cultural Association**, through its workshops and activities, I found happiness, a home of creativity, belonging, and the spark of thought.To my classmates, thank you for walking alongside me on this journey, And to my dear friend Bouthayna, you have been the best colleague and a true sister along the way. May life bless you with all the happiness you deserve ."*

LEBCIR,KHEIRA

إهـداء

إلى نفسي... إلى تلك التي قاومت التعب والعناء وتجاوزت الانكسارات وسارت في دروب العلم رغم كثرة العثرات... شكراً لي أنني لم أستسلم ....لأنني آمنت أن النهاية الحلوة تليق ببدايات مثقلة بالصبر والإنتظار. إلى منبع الحنان ورفيقة الدعاء وأجمل الحاضرين الغائبين... إلى أمي ملكة قلبي ، التي تعالج جسدها هناك وتداوي قلبي من هنا بدعائها وبحبها، بصورتها الراسخة في وجداني وآسرة كياني غيابك عن مناقشة المذكرة حضورٌ من نوع آخر.... أكثر دفئاً وصدقا..... فمكانك في الصف الأول من القلب لا يتغير ، لا يحول ولا يزول . وإلى السند الصامت والركن الشامخ... إلى أبي قرة عيني ... الذي علمني أن الرجولة موقف وأن العطاء لا يُشترط أن يكون بصوت عال شكراً لحضورك الثابت رغم كل الانشغالات. إلى من تقاسمتُ معهن الضحكات والدموع وسهر الليالي الثقيلة... إلى أخواتي .... حبكن زاد لا ينفد، وعطاؤكن ظل أستظل به حين تميل الشمس عن كتفي. وإلى أختي الكبرى.... التي كانت لي أما حين أثقلت الأمومة على أمي... كلمات الشكر تعجز عن وصف حنانك ووقوفك بجانبي ، حبيبتي أنت نسخة أخرى من قلب أمي. الى أخي.... شكرا على كل عناء و شغب الذي قدمته لي و عانيته معك حفظك الله إلى الأخت الصديقة و رفيقة دربي صليحة التي خففت عني بأحاديثها الكثير وكانت لي كتفا ثالثاً حين تعب الكتفين. فيك من الصفاء ما يكفي لعالم....وفي صداقتك ما يكفي لي. إلى خالتي و صديقتي أحلام التي كانت الدعم الخفي والجند المجهول والظل الذي لم يتخلى عني ولو لحظة...أنتِ الخير المتسلل إلى حياتي والرفق في الأيام. شكرااا وإلى كلّ من حضرني في هذه الرحلة من الأهل والزميلات والأوفياء... لكم مني عرفان لا يُكتب وامتنان لا ينسى. بكم كان الطريق أيسر..... دمتم لي خيرا لا ينتهي. وأخيرا بل أولا في القلب... إلى أرواح شهداء غزة الجريحة إلى من سقوا أرض العزة بدمهم وكتبوا بأجسادهم معنى الصمود... لكم كل ما يُكتب وكل ما يُنجز وكل ما سنكون عليه ما حيينا..

مناصري بثينة

Acknowledgements

First and foremost, we raise our hearts in gratitude to **God Almighty**, whose mercy embraced us, whose light guided us, and whose strength carried us through every challenge. Alhamdulillah for every step, every lesson, and every moment that brought us here.

To our supervisor, **Dr. Lakehal-Ayat Raouf Ouanis**, we extend our heartfelt thanks. Thank you for believing in us, for seeing potential even when we doubted ourselves, and for offering your guidance with patience and kindness. Despite your health challenges, your steady presence and thoughtful advice were a silent strength behind our progress. We are truly grateful.

To all the professors and staff at our university, we offer deep appreciation. Each of you left a trace on our journey in a word, a gesture, a lesson and together, those traces shaped the students we have become. Thank you for your dedication, your passion, and your impact, both seen and unseen.

To our families and loved ones your love was our anchor. Thank you for your unwavering presence, your prayers whispered in the silence, and your belief in us even in our most uncertain moments. You were the warmth that carried us through long nights and heavy days.

This work is more than a project. It is a piece of our story, written with effort, grace, and the love of those who stood beside us.

Thank you, from the depths of our hearts.

Contents

List of Tables

List of Figures

General Introduction

Wireless Sensor Networks (WSNs) are among the most transformative technologies of the modern era. They have revolutionized how data is sensed from the physical environment, processed, and wirelessly transmitted to centralized systems. A typical WSN consists of a large number of small, autonomous sensor nodes, each incorporating three essential components: a sensing unit, a processing unit, and a wireless communication module.

WSNs have gained widespread adoption in numerous fields such as environmental monitoring, smart agriculture, healthcare, military surveillance, and smart city infrastructure. This is largely due to their advantages in terms of rapid deployment, energy efficiency, scalability, and ability to operate in harsh or inaccessible environments. However, alongside these benefits, WSNs face several technical challenges, the most critical of which include energy efficiency, network security, scalability, and **fault tolerance**.

Fault tolerance, in particular, is a fundamental concern. Since sensor nodes are often deployed in challenging environments, they are prone to failure due to environmental factors, hardware malfunctions, or malicious attacks. Therefore, ensuring the network remains operational even when individual nodes fail is crucial. Achieving this requires a resilient network architecture that maintains both **connectivity** and coverage, even in the face of node failures. To address this challenge, various fault tolerance techniques have been proposed, including:

- **Redundancy**;
- **Clustering**;
- **Intelligent Deployment of Nodes**;
- **Energy-efficient Routing**.

In this context, the present study aims to enhance the fault tolerance capabilities of WSNs by employing a metaheuristic approach based on the **Greedy Randomized Adaptive Search Procedure (GRASP)** algorithm. GRASP is a multi-start metaheuristic that constructs initial solutions using a randomized greedy process, followed by iterative local search refinements to achieve high-quality solutions.

In this work, the WSN is modeled as an undirected graph where vertices represent sensor nodes, and edges represent wireless communication links between nodes within range. The

primary objective is to ensure overall network **connectivity and robustness**, especially under node failure scenarios, by identifying critical nodes and optimizing their placement and roles in the network.
To assess the effectiveness of the proposed approach, two types of node failure scenarios are simulated:

- **Random failure:** resulting from unpredictable hardware faults or environmental conditions;
- **Targeted failure:** caused by intentional attacks targeting central (high-degree) nodes to maximize disruption.

This study is divided into four main chapters:

- **Chapter 1:** presents the theoretical background on WSNs, their architecture, characteristics, and major challenges, with a detailed focus on fault tolerance mechanisms;
- **Chapter 2:** provides an overview of the GRASP algorithm, explaining its core concepts, operational phases, and its application in network optimization problems, while comparing it with other metaheuristics;
- **Chapter 3:** outlines the methodology adopted in this research, including mathematical modeling, design of the proposed algorithm, and the technical implementation details;
- **Chapter 4:**introduces the simulation environment and failure scenarios, presents a performance evaluation of the results, and discusses observed limitations and constraints. Additionally, this chapter also includes **a comparative analysis with prior studies** in the field to highlight the improvements achieved by the GRASP-based approach in terms of fault resilience and network performance .

Chapter 1: Wireless Sensor Networks

1.1 Introduction

This chapter focuses on Wireless Sensor Networks (WSNs) from the perspective of fault-tolerant network design, a critical challenge impacting the efficiency and reliability of these networks in real-world environments. It begins with a comprehensive definition of WSNs, detailing their fundamental architecture, core components, and key operational characteristics. The chapter then addresses the various types of faults that commonly affect these networks, followed by an examination of fault tolerance mechanisms and protocols designed to ensure network robustness and continuous operation despite failures. Additionally, an overview of metaheuristic optimization techniques employed to enhance network performance and fault resilience is provided, including a comparative analysis of different approaches. This theoretical foundation serves as a basis for selecting the most appropriate strategy, which will be explored in depth in the subsequent chapters, with particular emphasis on the GRASP (Greedy Randomized Adaptive Search Procedure) algorithm as a primary tool for fault-tolerant network design.

1.2 Wireless Sensor Networks (WSNs)

1.2.1 Definition

Wireless Sensor Networks (WSNs) are networks composed of spatially distributed, autonomous sensor nodes that monitor and record physical or environmental conditions such as temperature, humidity, sound, or pressure,and wirelessly transmit this data to a central location for processing and analysis. WSNs are typically infrastructure-less, self-organizing, and designed for rapid, flexible deployment in a variety of environments [1], [2].

1.2.2 Architecture of WSNs

Wireless Sensor Networks (WSNs) rely on an integrated organization that combines different technical layers with the way sensor nodes connect to each other. To fully understand the architecture of these networks, it is essential to review the layers that compose them, along with the various forms the network connections between nodes can take.The WSN architecture is often modeled after the OSI (Open Systems Interconnection)reference model and typically consists of five main layers:

- **Physical Layer**:Handles the physical connection between sensor nodes and the base station using wireless technologies such as radio waves, infrared, or Bluetooth. It is responsible for frequency selection, modulation, signal detection, and data encryption;
- **Data Link Layer**: Ensures reliable communication between nodes and the base station, often using protocols like IEEE 802.15.4. It manages data framing, error control, and medium access control (MAC);
- **Network Layer**: Responsible for routing data between nodes and managing addressing. It handles path selection and deals with challenges such as power saving and self-organization of sensor nodes;
- **Transport Layer**: Provides end-to-end communication reliability, congestion avoidance, and flow control. It manages data segmentation and reassembly, ensuring complete data delivery;
- **Application Layer**: Facilitates the transfer of specific sensed data to the base station for further processing and user applications. It also manages traffic and provides software interfaces for various application domains such as military, medical, and environmental monitoring.

In addition to these five layers, WSNs often include three cross-layer planes to enhance network management and efficiency:

- **Power Management Plane**: Manages the power levels of sensor nodes to optimize sensing, processing, and communication energy consumption;
- **Mobility Management Plane**: Detects sensor node movement and maintains network connectivity by tracking neighbors and power levels;
- **Task Management Plane**: Distributes tasks among sensor nodes to improve energy efficiency and prolong network lifetime [3].

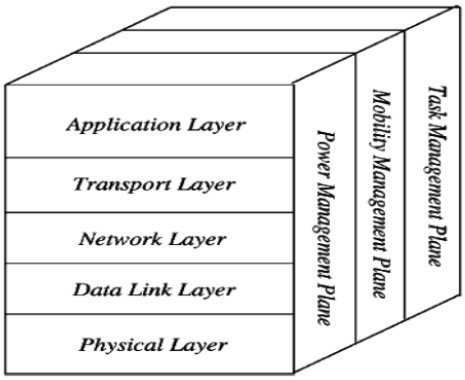

Figure 1.1: Wireless Sensor Network protocol stack.

Since the network consists of multiple interconnected nodes, the way these nodes are arranged and connected **the network topology** directly affects the network's efficiency and performance. Therefore, it is necessary to understand the different connection patterns that WSNs can adopt. Common topologies include:

- **Star Topology** :A central node connected directly to all other nodes;
- **Mesh Topology (Fully Connected)** : Every node connected to every other node;
- **Bus Topology** : Nodes connected in a linear sequence;
- **Tree Topology** : Hierarchical branching structure;
- **Ring Topology** : Nodes connected in a closed loop [4].

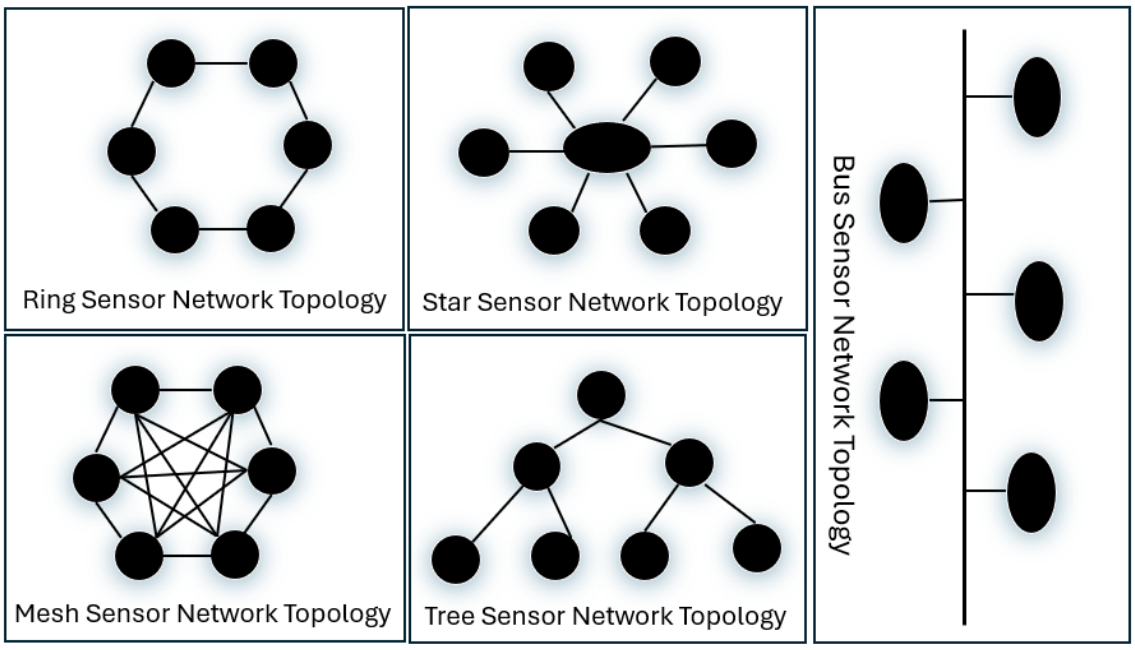

Figure 1.2: Network topologies.

1.2.3 Key Components

Wireless Sensor Networks (WSNs) are built upon a set of fundamental components that work together to enable efficient data collection, processing, and communication. To better understand the structure and operation of WSNs, it is essential to explore their key components, which include:

- **Sensor Nodes** :These serve as the basic unit of WSN. Each one is a small, battery-powered device, which consists of sensoring, processing, and transmitting units, and a power supply. They collect data from the environment and transmit it to the other nodes or the base station ;
- **Base Stations** :They act as gateways between the sensor network and external world (systems). They collect data from sensor nodes, and forward to the other networks or users. They are generally more powerful than sensor nodes and can have a continuous power supply;
- **Communication Protocols** : These are the guidelines and standards for data transmission within the network. ZigBee, Bluetooth, and LPWANs are a few common examples of protocols. These are designed to achieve utmost or to optimize (minimize, maximize) energy efficiency, reliability, and scalability in WSNs [3].

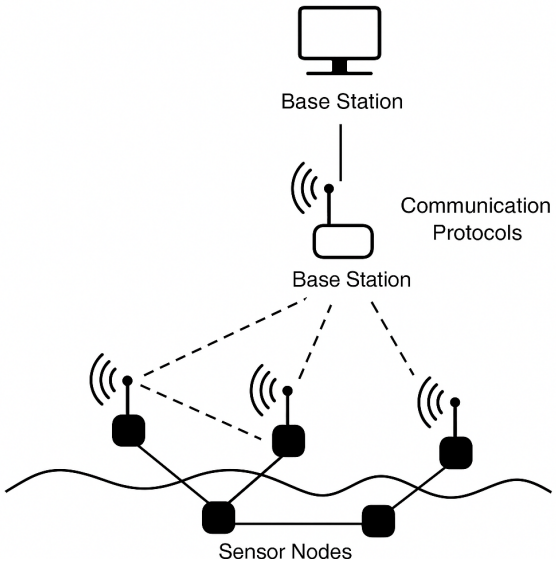

Figure 1.3: Key Components of WSNs.

1.2.4 Key Characteristics

In addition to their core components, Wireless Sensor Networks (WSNs) exhibit several distinctive features that set them apart from traditional networks. Understanding these characteristics is crucial for appreciating how WSNs operate in diverse and often challenging environments. Key features include

- **Resource Constraints** :Nodes have limited energy, memory, and processing capabilities;
- **Self-Organization** :Networks can autonomously configure and adapt to changes, including node failures or additions;
- **Scalability** : WSNs can support a large number of nodes, making them suitable for both small and large-scale deployments;
- **Fault Tolerance** :Designed to maintain functionality even when some nodes fail due to harsh environmental conditions or energy depletion;
- **Distributed and Dense Deployment** :Nodes are often deployed in large numbers and operate collaboratively to monitor the environment;
- **Data-Centric Communication** : Focus on collecting, processing, and transmitting relevant data rather than maintaining continuous connections;
- **Dynamic Topologies** : Network structure can change due to node mobility, failures, or environmental factors [3], [5].

1.2.5 Applications of WSNs

WSNs are widely used across various domains such as:

- **Environmental Monitoring** :Tracking weather, natural disasters, and ecosystems;
- **Healthcare** :Remote patient monitoring and assisted living;
- **Industrial Automation** :Machinery monitoring and infrastructure health;
- **Military and Security**:Surveillance, intrusion detection, and battlefield monitoring;
- **Agriculture** :Precision farming and livestock tracking;
- **Transportation**:Traffic control and asset tracking [1], [6]–[8].

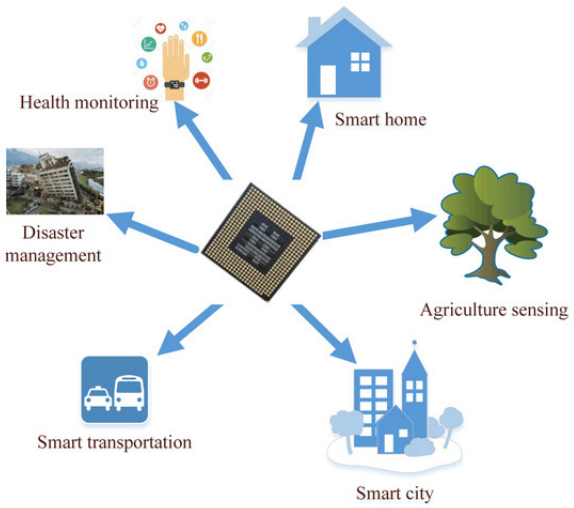

Figure 1.4: Applications of WSNs.

1.2.6 Challenges in WSNs

Wireless Sensor Networks (WSNs) face some important challenges:

- **Energy Efficiency** :Sensor nodes are typically battery-powered and are often deployed in inaccessible locations, making energy conservation crucial for maximizing network lifetime. Energy constraints impact all aspects of WSN design, from communication protocols to data processing ;
- **Security** :Unsecured wireless communication makes WSNs vulnerable to many threats, such as eavesdropping, data tampering, and node compromise. Ensuring data integrity, confidentiality, and secure node authentication is a continuous challenge ;
- **Scalability** :WSNs must be able to manage networks of a few to thousands of nodes, often in dynamic environments. Protocols and architectures must be able to adapt to network size and density variations without sacrificing performance ;
- **Fault Tolerance** :Node failures are common due to harsh environments, energy depletion, or hardware failures. WSNs require effective mechanisms for fault detection, isolation, and recovery to provide reliable data collection and network connectivity [3], [8].

1.3 Fault Tolerance in WSNs

1.3.1 Definition and Importance

Fault tolerance in Wireless Sensor Networks (WSNs) is the system's ability to maintain reliable operation and deliver accurate data even when some nodes or links fail due to hardware issues, energy depletion, or environmental factors. This is achieved through mechanisms such as error detection, diagnosis, and recovery, allowing the network to identify and correct faults before they impact overall performance . Fault tolerance is critical in WSNs because these networks are often deployed in harsh or inaccessible environments, making them highly susceptible to failures. Without effective fault tolerance, failures can lead to data loss, reduced coverage, and compromised system reliability, which is especially risky in applications like environmental monitoring, healthcare, and military operations.Therefore, robust fault tolerance is essential to ensure the availability, reliability, and dependability of WSNs in real-world, mission-critical scenarios [9].

1.3.2 Types of Faults in WSNs

Wireless Sensor Networks face various faults that directly impact their performance and reliability. To better understand the challenges these networks encounter, it is important to identify the different types of faults, which include:

- **Node Failures** :Sensor nodes can stop functioning due to hardware damage, environmental stress, or battery exhaustion, leading to loss of data and network connectivity;
- **Communication Failures** :Issues such as wireless interference, link breakages, or packet loss disrupt data transmission between nodes, causing routing failures or network partitioning;
- **Energy Depletion** :Nodes often fail when their batteries are drained, which is a common cause of node and network failures in WSNs;
- **Hardware/Software Malfunctions**: Faults can arise from defective sensors, transceivers, or corrupted software, resulting in incorrect data, node crashes, or erratic behavior[3], [10].

1.3.3 Fault Tolerance Mechanisms in WSNs

They are four main fault tolerance mechanisms commonly used in WSNs [5]:

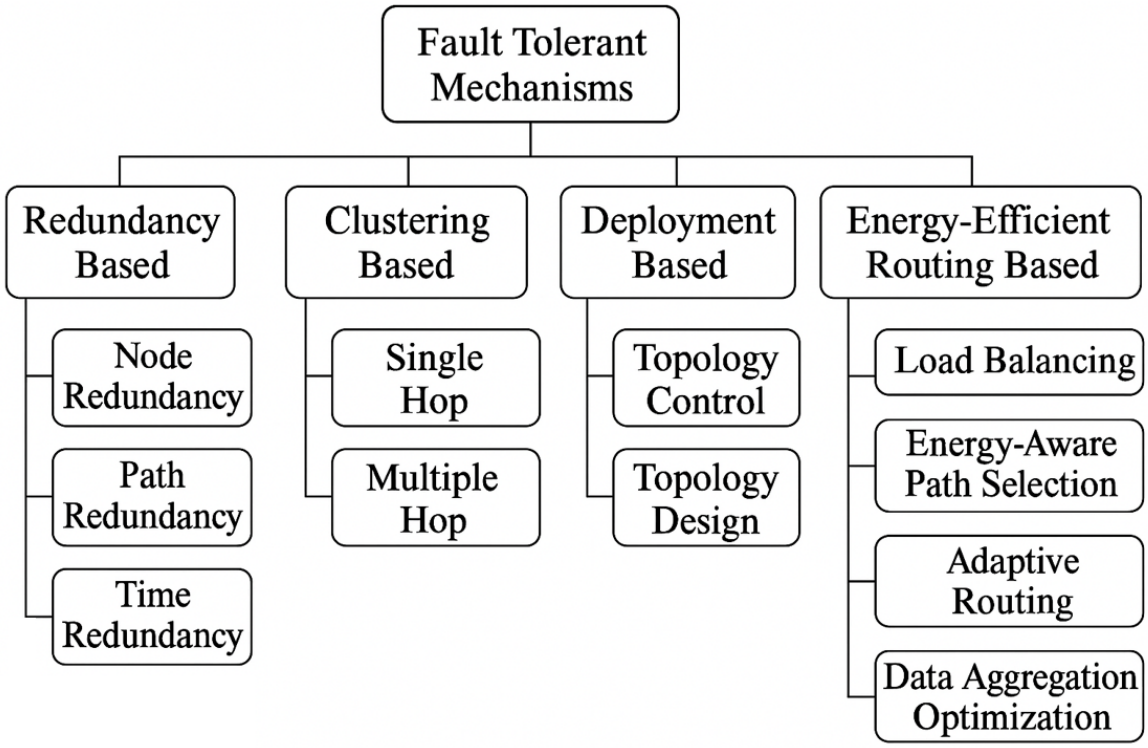

Figure 1.5: Fault Tolerance Mechanisms in WSNs.

1.3.4 Fault Tolerance Protocols in WSNs

Wireless Sensor Networks (WSNs) encompass a wide range of protocols designed to ensure data reliability, energy efficiency, and robustness in dynamic and failure-prone environments. Numerous fault tolerance protocols have been proposed in the literature, each adopting a distinct strategy to enhance network resilience. Among the most prominent and widely adopted protocols in this field are:
SPIN, **LEACH**, **PEGASIS**, **TEEN**, **APTEEN**, **Rumor Routing**, **GAF**, and **HEED**.
From these, the following table presents a comparative overview of **SPIN** and **LEACH**, focusing on their approach, fault tolerance features, and key differences [3], [11].

Protocol	Approach	Fault Tolerance Features	Key Differences
SPIN	Data-centric	Uses meta-data negotiation to reduce redundant data transmission and energy use. Fault tolerance is limited to data dissemination efficiency.	Focuses on efficient data dissemination, not cluster formation.
LEACH	Clustering	Periodically rotates cluster heads to balance energy consumption and uses re-clustering to recover from CH failures, improving fault tolerance.	Emphasizes cluster-based management and energy balancing.

Table 1.1: Comparison of SPIN and LEACH Protocols.

1.4 Optimization Techniques in WSNs

Wireless Sensor Networks face numerous challenges, such as energy efficiency, fault tolerance, scalability, and resource management. To address these challenges, optimization techniques are employed to enhance network performance, extend lifetime, and ensure reliable operation. Among these techniques, metaheuristic optimization methods have gained significant attention due to their ability to solve complex, non-linear, and large-scale optimization problems efficiently[12].

1.4.1 Metaheuristic Optimization for WSNs

Metaheuristic optimization techniques are inspired by natural phenomena, biological processes, or physical systems.They are particularly useful for solving optimization problems in WSNs, where traditional methods may struggle due to the complexity and dynamic nature of the network.Some of the most widely used metaheuristic techniques include [12], [13]:

1. **Genetic Algorithms(GA) :** Genetic Algorithms are inspired by the process of natural selection and evolution. They use operations such as selection, crossover, and mutation to evolve a population of candidate solutions over generations .
   Applications of GA in WSNs
   - Optimizing energy-efficient routing protocols;
   - Solving node placement and coverage problems;
   - Enhancing fault tolerance and network lifetime[14], [15].

2. **Particle Swarm Optimization (PSO):** PSO is inspired by the social behavior of birds and fish. It uses a population of particles (candidate solutions) that move through the solution space, guided by their own best-known position and the global best-known position.
   Applications of PSO in WSNs

   - Optimizing energy-efficient routing protocols;
   - Improving energy-efficient routing and load balancing;
   - Enhancing network coverage and connectivity.

3. **Ant Colony Optimization (ACO):** ACO is inspired by the foraging behavior of ants. It uses artificial ants to explore the solution space, depositing pheromones to mark promising paths. Over time, the pheromone trails guide the search toward optimal solutions .
   Applications of ACO in WSNs

   - Solving routing problems (energy-efficient and fault-tolerant routing);
   - Optimizing data aggregation and clustering;
   - Enhancing network lifetime and coverage.

4. **Greedy Randomized Adaptive Search Procedure (GRASP):** GRASP is a multi-phase metaheuristic that combines greedy construction and local search. In the construction phase, a feasible solution is built using a greedy randomized approach. In the local search phase, the solution is refined by exploring its neighborhood .
   Applications of GRASP in WSNs

   - Optimizing network routing and clustering;
   - Enhancing fault tolerance and energy efficiency;
   - Solving node placement and coverage problems.

5. **Tabu Search**: Tabu Search is a metaheuristic optimization method that iteratively explores solution neighborhoods while avoiding revisits using adaptive memory. It balances exploitation (refining current solutions) and exploration (escaping local optima) through short-term and long-term memory strategies.
   Applications of TS in WSNs

   - Optimizing energy-efficient routing protocols and data aggregation;
   - Enhancing node localization accuracy and network coverage;
   - Solving dynamic topology control and resource allocation problems[12].

1.4.2 Comparison of Approaches (strengths and weaknesses)

Technique	Strengths	Weaknesses
Genetic Algorithms (GA)	- Effective for large solution spaces. - Handles multi-objective optimization.	- Computationally expensive. - Slow convergence. - Parameter tuning required.
Particle Swarm Optimization (PSO)	- Simple and fast. - Good for continuous problems. - Fast convergence.	- Prone to local optima. - Less effective for discrete problems. - Parameter tuning required.
Ant Colony Optimization (ACO)	- Excellent for combinatorial problems. - Handles dynamic environments.	- Computationally intensive. - Slow convergence. - Parameter tuning required.
GRASP	- Balances exploration and exploitation. - Flexible and adaptable.	- Limited exploration. - Requires hybrid approaches for improvement.
Tabu Search	- Escapes local optima effectively. - Handles complex constraints.	- Computationally intensive. - Parameter tuning required.

Table 1.2: Comparison of Optimization Techniques.

1.5 Conclusion

In conclusion, this chapter has established a comprehensive theoretical framework for understanding Wireless Sensor Networks (WSNs), covering their architecture, essential components, operational characteristics, and the inherent challenges they face particularly fault tolerance. Various fault tolerance mechanisms and protocols were reviewed, along with an in-depth comparative evaluation of metaheuristic optimization techniques such as Genetic Algorithms and Ant Colony Optimization, highlighting their strengths and limitations. Based on this analysis, the GRASP (Greedy Randomized Adaptive Search Procedure) algorithm was selected as the most suitable approach to designing fault-tolerant networks that meet both reliability and efficiency requirements. The following chapter will delve into the detailed theoretical background of the GRASP methodology and its practical applications in fault-tolerant network design.

Chapter 2:Greedy Randomized Adaptive Search Procedure

2.1 Introduction

After presenting in Chapter 1 the main problem addressed in this work—fault tolerance in Wireless Sensor Networks (WSNs),we provided a scientific overview of WSNs and discussed several existing algorithms commonly used to enhance network reliability and fault management.

In this chapter, we shift our focus to the metaheuristic algorithm selected to address this challenge: the Greedy Randomized Adaptive Search Procedure (GRASP). The objective is to provide a comprehensive theoretical background on GRASP. We begin by tracing its historical development, followed by a detailed explanation of its core structure and the two main phases that characterize its search process. We then examine its application to well-known combinatorial optimization problems such as the Traveling Salesman Problem (TSP), and finally explore its role in network optimization. The chapter concludes by highlighting GRASP's advantages over other metaheuristic methods, reinforcing its suitability for our fault-tolerant network design goals.

2.2 Greedy Randomized Adaptive Search Procedure(GRASP)

2.2.1 Background and Historical Development

The Greedy Randomized Adaptive Search Procedure was introduced (first introduced) in 1989 by Thomas A. Feo and Mauricio G.C. Resende [16]. It was developed to address combinatorial optimization problems by combining greedy algorithms, randomization, and local search. Over the years, GRASP has evolved with various enhancements, such as: reactive GRASP, parallel GRASP, and hybrid GRASP, which combine GRASP with other metaheuristics like tabu search or genetic algorithms, and it applied to various domains (like network design e.g. wireless sensor networks) due to its flexibility and effectiveness in solving complex optimization problems [17], [18].

2.2.2 Overview of GRASP Algorithm

GRASP is a metaheuristic algorithm designed for solving combinatorial optimization problems. It can be considered an enhanced or updated version of traditional greedy algorithms,

incorporating additional elements to improve performance and avoid getting trapped in local optima. GRASP combines greedy algorithms, randomization, and local search to effectively explore the solution space. Unlike traditional greedy algorithms, which make deterministic choices at each step, GRASP introduces randomness by selecting solutions from a candidate list of high-quality options. This randomization helps diversify the search and avoid premature convergence to suboptimal solutions. After constructing an initial solution using this greedy randomized approach, GRASP further refines the solution through a local search phase, where neighboring solutions are explored to improve the current solution. The algorithm operates iteratively, repeating the construction and local search phases for a fixed number of iterations or until a stopping criterion is met. The best solution found across all iterations is returned as the final result. GRASP strikes a balance between exploration diversification and exploitation intensification of the solution space, making it a powerful and flexible tool for solving complex optimization problems. Its ability to combine the simplicity of greedy algorithms with the robustness of randomization and local search has led to its widespread application in various domains, including scheduling, routing, facility location, and fault-tolerant wireless sensor networks [17], [18].

2.2.3 Phases of GRASP Algorithm

GRASP consists of 2 phases:

2.2.3.1 Construction Phase: Generating a Solution

The first phase consists of three parts:
Greedy Algorithm: The greedy algorithm builds a solution step-by-step, making the best local choice at each step. For example, in a network design problem, the greedy algorithm might prioritize connecting nodes with the highest connectivity or lowest energy consumption.A greedy algorithm always makes the choice that looks best at the moment. It makes a locally optimal choice in the hope that this choice will lead to a globally optimal solution.
Greedy algorithms do not always yield optimal solutions (eg.0-1-knapsack), but in some cases it does (eg.Minimum spanning tree).
Probabilistic selection: In the greedy algorithm the selection of the next element to add to the solution is (often) deterministic.The probabilistic selection process selects candidates among which we choose the next element to be added to the presently partial solution. The list of candidates is formally denoted the Restricted Candidate List (RCL) :

1. **Construction**
    - **Select** the β best elements (β = 1: purely greedy, β = n completely random);
    - **Selects** the elements that are not more than α away from the best element;

- **Select** elements above an absolute threshold of γ.

2. **Selection**

   - Equally probability;
   - Weighted probability.

Adaptive function: Modify the function which guides the greedy algorithm based on the element selected for the solution we are constructing.
The construction is called dynamic in contrast to the static approach which assigns a score to elements only before starting the construction (example: TSP links).

2.2.3.2 Local Search Phase: Finding a Local Minimum

The constructed solution is improved by exploring its neighborhood making small changes and accepting those that improve the objective until a local optimum is reached. This phase helps refine the solution and escape poor-quality regions[17], [19], [20].

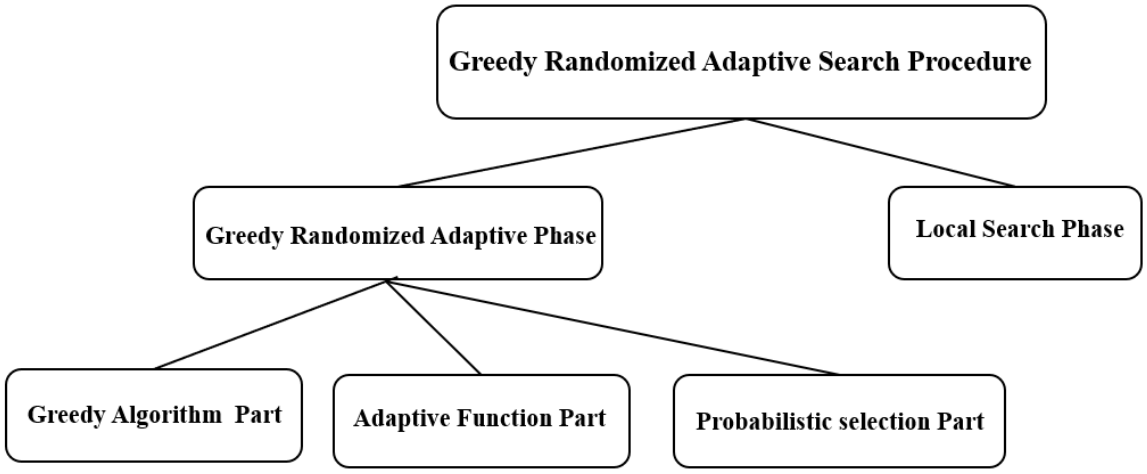

Figure 2.1: GRASP Phases.

2.2.4 Applications of GRASP in Combinatorial Optimization: TSP

2.2.4.1 Mathematical Formulation of The TSP

The Traveling Salesman Problem (TSP) [21] is a classic combinatorial optimization problem:

1. **Given:**
   - A set of cities $V = \{1, 2, \ldots, n\}$;
   - A distance (or cost) matrix $D = [d_{ij}]$, where d_{ij} is the distance from city i to city j.
2. **Objective:**

   Find a tour (a permutation of the cities) that starts and ends at the same city and visits each city exactly once, with the minimum total distance.

   Mathematical Model:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij}$$

   Subject to:

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \forall i = 1, \ldots, n$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad \forall j = 1, \ldots, n$$

$$x_{ij} \in \{0, 1\} \quad \forall i, j = 1, \ldots, n$$

$$u_i - u_j + n x_{ij} \leq n - 1 \quad \forall i \neq j,\ i, j = 2, \ldots, n$$

$$1 \leq u_i \leq n - 1 \quad \forall i = 2, \ldots, n$$

Where:

$$x_{ij} = \begin{cases} 1 & \text{if the tour goes from city } i \text{ to city } j \\ 0 & \text{otherwise} \end{cases}$$

The variables u_i are auxiliary variables used to eliminate subtours.

2.2.4.2 Solving the TSP Using GRASP

GRASP operates as follows for TSP:

Phase 1: Greedy Randomized Construction

- Start from a random city;

- At each step, create a Restricted Candidate List (RCL) of the nearest unvisited cities;
- Randomly select the next city to visit from the RCL;
- Repeat until all cities are visited.

Phase 2: Local Search

- Take the constructed tour and apply local search (e.g. 2-opt swaps);
- Try to improve the solution by reducing the total tour length;
- Repeat until no further improvement is possible;
- Repeat both phases multiple times, and retain the best solution found.

2.2.4.3 Numerical Example: Applying GRASP to the TSP

Given Distance Matrix (4 cities):

cities	**1**	**2**	**3**	**4**
1	0	10	15	20
2	10	0	35	25
3	15	35	0	30
4	20	25	30	0

Table 2.1: Distance matrix for the TSP example solved using GRASP .

Step 1 : Greedy Randomized Construction

- Start at city 1;
- Unvisited cities: 2 (10), 3 (15), 4 (20);
- RCL (let's use top 2 nearest): cities 2 (10) and 3 (15);
- Randomly pick city 3;
- Next, from city 3: unvisited cities 2 (35), 4 (30). RCL: city 4 (30);
- Only city 4 in RCL, so pick city 4;
- Next, from city 4: only city 2 left (25);
- Tour: $1 \rightarrow 3 \rightarrow 4 \rightarrow 2 \rightarrow 1$.

Step 2: Calculate Total Distance

- 1→3: 15, 3→4: 30, 4→2: 25, 2→1: 10;
- Total = 15 + 30 + 25 + 10 = 80.

Step 3: Local Search (2-opt)

- Try swapping segments to see if a shorter tour exists;
- e.g :**try 1 → 2 → 4 → 3 → 1:**
  1→2: 10, 2→4: 25, 4→3: 30, 3→1: 15 = 10 + 25 + 30 + 15 = 80;
- e.g :**try 1 → 2 → 3 → 4 → 1:**
  1→2: 10, 2→3: 35, 3→4: 30, 4→1: 20 = 10 + 35 + 30 + 20 = 95;
- Best tour found: 1 → 3 → 4 → 2 → 1 or 1 → 2 → 4 → 3 → 1, both with distance 80.

Step 4: Repeat Repeat the process with different random choices in the RCL for diversification [22] .

2.3 Use of GRASP in Network Optimization

2.3.1 Examples of GRASP Applications

Network Routing

1. Vehicle routing;
2. Aircraft routing;
3. SDH mesh-restorable network design.

Clustering

1. Graph partitioning;
2. p-hub location problem;
3. Node grouping for load balancing and minimizing intra-cluster distances [22].

2.3.2 Advantages of GRASP Compared to Other Metaheuristics Algorithms

- Simplicity and ease of implementation ;
- Flexible adaptation to various network problems;

- Balanced exploration (randomized construction) and exploitation (greedy +local search);
- Multi-start approach reduces risk of local optima;
- High parallelizability (independent iterations);
- Strong empirical performance in routing, clustering, and fault tolerance;
- Often outperforms or matches complex metaheuristics like genetic algorithms and tabu search [12].

2.4 Conclusion

In this chapter, we presented a detailed theoretical overview of the Greedy Randomized Adaptive Search Procedure (GRASP), the metaheuristic algorithm chosen for our study. We began with its historical context and structural framework, followed by an in-depth description of its core phases and notable applications in both combinatorial and network optimization problems.

Moreover, we identified key advantages that distinguish GRASP from other metaheuristic techniques, supporting its selection for our target problem. This theoretical foundation equips us with the necessary insights to move forward to the practical stage of our research, where we will implement GRASP to design a fault-tolerant architecture for Wireless Sensor Networks, as detailed in the next chapter.

Chapter 3: Fault-Tolerant Network Design Using GRASP

3.1 Introduction

This chapter presents the implementation phase of the GRASP algorithm for improving the fault tolerance of Wireless Sensor Networks (WSNs). It details the steps taken to adapt the algorithm to the network topology, including reconnection mechanisms, inter-node link optimization, and integration of fault-handling strategies. Furthermore, it discusses the design parameters and procedures involved in applying GRASP within WSN environments.

3.2 Problem Statement

The problem aims to design a wireless communication network consisting of a set of nodes distributed within a certain area. These nodes are connected to each other based on several criteria, including the maximum number of neighbors per node and the maximum communication range between nodes. The problem focuses on finding an efficient way to build a network topology that ensures:

- Each node is connected to a limited number of neighboring nodes (not exceeding a specified maximum);
- Provision of backup paths to guarantee continuous communication between nodes in case of failure in some nodes or links;
- Assignment of backup nodes for each node, which take over communication duties if the original node fails;
- Optimization of the network structure to improve its reliability and connectivity by adjusting node positions within the designated area, ensuring the network remains connected;
- Analysis of the network's resilience against random or targeted node failures, with the ability to evaluate network performance in terms of the number of active nodes and available communication paths among critical nodes.

3.3 Optimization Modeling

This modeling aims to enhance the robustness of a randomly distributed wireless sensor network by ensuring connectivity between nodes using the minimal number of links, while also guaranteeing the presence of backup paths for network nodes in case of failures.

The model focuses on intelligently selecting connections between nodes considering distance, the maximum number of neighbors, and the minimum required backup paths posing a challenge in balancing efficiency and reliability.

The modeling relies on a gradual improvement approach using the **GRASP** algorithm, which performs minor adjustments to node positions to find a better-connected network. The performance indicator (such as the ratio of actual to possible links) is continuously evaluated.

3.4 Mathematical Formulation

3.4.1 Decision Variables

- $x_{ij} \in \{0, 1\}$: A binary variable indicating whether there is a direct connection between node i and node j;
- $b_i \in \{0, 1\}$: A binary variable indicating whether node i has been assigned as a backup node for others;
- pdr $\in [0, 1]$: The ratio of realized links to the total possible links, used as a quality metric in the final evaluation.

3.4.2 Input Parameters

- **Number of nodes** n : (random within the range $15, 50$)
  This value represents the total number of sensor nodes randomly distributed in the network area. Selecting a random number of nodes between 15 and 50 simulates different network sizes, from small to moderate;
- **Coordinates of the nodes** (x_i, y_i)
  Each node is assigned a coordinate in a 2D plane, representing its physical location. These coordinates are critical for calculating distances between nodes, which affect connectivity;
- **Maximum number of neighbors per node** $N_{\max} = 3$
  Limits each node to have at most 3 direct neighbors, controlling network complexity and interference;

- **Minimum number of backup paths required** $R_{\min} = 2$
  Each node must have at least 2 backup paths for reliability and fault tolerance;

- **Maximum transmission range** $d_{\max} = 20$
  Defines the maximum distance two nodes can be apart to form a direct wireless connection.

Parameter	Value	Comment
Number of nodes (n)	Random between 15 and 50	Allows for testing scalability across various network sizes.
Coordinates of the nodes (x_i, y_i)	Random in 2D space	Used to model realistic spatial distribution of sensors.
Max neighbors per node (N_{max})	3	Restricts node degree to avoid excessive link density.
Min backup paths per node (R_{min})	2	Ensures that alternate routing options are available for each node.
Maximum transmission range (d_{max})	units 20	Limits connection range to reflect wireless transmission constraints.

Table 3.1: Input Parameters.

3.4.3 Objective Function

Maximize the effective connectivity in the network (PDR):

$$\max \left(\frac{\sum_{i=1}^{n} \sum_{j=i+1}^{n} x_{ij}}{\frac{n(n-1)}{2}} \right)$$

3.4.3.1 In-Depth Analysis of the Objective Function

This objective function aims to **maximize the actual connectivity ratio** within the network, which measures how well the nodes are connected using the **minimum** necessary number of links.

Numerator :

$$\sum_{i=1}^{n} \sum_{j=i+1}^{n} x_{ij}$$

This represents the **number of links actually established** between pairs of nodes (where each $x_{ij} = 1$ indicates a link exists between nodes i and j).

Denominator:

$$\frac{n(n-1)}{2}$$

This is the **maximum possible number of links** in a fully connected network of n nodes (every node connected to every other node).

3.4.3.2 Purpose of the Objective Function

It provides a **ratio between 0 and 1**:

- Close to **1** means the network is highly connected;
- Low values indicate poor connectivity or isolated parts within the network.

3.4.3.3 Interpretation of the Objective Function

- It offers a **clear quantitative measure** of network connectivity;
- Enables **comparison of solution quality** when using optimization algorithms;
- Helps **balance between:**
  - The number of links used (cost and complexity);
  - And the effectiveness of network connectivity (performance and reliability).

3.4.3.4 Note on Indirect Energy Conservation

By **minimizing the number of links** needed to maintain the required connectivity, the objective function also **indirectly helps conserve energy** in the network.

Each additional link involves communication overhead, which consumes energy for transmission and reception. Reducing unnecessary links leads to **lower energy consumption**, improving the network's overall energy efficiency and extending the lifetime of battery-powered nodes.

3.4.4 Constraints

- **Bidirectional Connectivity Constraint**
  - *Formula:*

$$x_{ij} = x_{ji} \quad \forall i, j$$

  - *Meaning:* If node i is connected to node j, then node j must also be connected to node i;

  - *Purpose:* To ensure bidirectional communication between nodes in the network.

- **Maximum Number of Neighbors per Node**
  - *Formula:*
$$\sum_{j=1}^{n} x_{ij} \leq N_{\text{max}} \quad \forall i$$
  - *Meaning:* Each node may have at most N_{max} direct neighbors;
  - *Purpose:* To control energy consumption, reduce communication interference, and prevent network overload.

- **Maximum Distance Between Connected Nodes**
  - *Formula:*
$$x_{ij} = 0 \quad \text{if } d_{ij} > d_{\text{max}}$$
  - *Meaning:* A connection is only allowed between nodes i and j if the distance d_{ij} is less than or equal to d_{max};
  - *Purpose:* To respect the communication range limitations of the nodes.

- **Backup Path Requirement**
  - *Formula:*
$$\sum_{j=1}^{n} x_{ij} \geq R_{\text{min}} \quad \text{(after adding backups)}$$
  - *Meaning:* After including backup links, each node must maintain at least R_{min} connections;
  - *Purpose:* To ensure the availability of alternate paths and maintain network resilience.

- **Node Failure and Backup Activation Constraint**
  - *Formula:*
$$x_{fb} = 1 \quad \text{if } b \in \text{backup\_map}(f)$$
  - *Meaning:* If node f fails, a backup node b (from its predefined backup map) must take its place and restore its connections;
  - *Purpose:* To preserve connectivity and continue normal operation after node failures.

3.5 GRASP Implementation and Algorithmic Workflow

3.5.1 Overview of Algorithmic Phases

This section presents the implementation phases of the GRASP (Greedy Randomized Adaptive Search Procedure) algorithm applied to the problem of designing a fault-tolerant and connected wireless sensor network. The implementation is structured into a systematic sequence that considers both the initial solution construction and its iterative improvement to maximize connectivity and availability, even in the presence of node or link failures:

Phase 1: Construction Phase

In this phase, an initial solution is built by connecting nodes using a greedy randomized approach based on a Restricted Candidate List (RCL). The RCL consists of nearby candidate nodes that can be connected without exceeding the maximum number of neighbors per node. Candidates are sorted by distance, and typically the closest node is chosen greedily. Some randomness is introduced later during solution improvement. This combination of greediness and controlled randomness forms the core of the GRASP method for iteratively enhancing the network.

Steps:

- Randomly generate node positions within the coverage area;
- Compute distances between all pairs of nodes;
- Identify candidate neighbors for each node based on distance and constraints;
- Select neighbors greedily but with a random choice within the RCL to build a preliminary network topology;
- Introduce backup links to enhance redundancy.

Phase 2:Local Search Optimization

Once the initial solution is constructed, it undergoes local improvements aimed at increasing the Packet Delivery Ratio (PDR) and strengthening fault tolerance. Small perturbations, such as rewiring connections or adjusting backup links, are explored. Changes are accepted only if they improve the objective function, i.e., enhance the network's reliability or connectivity.

Strategies include

- Reassigning neighbors with weak connectivity;

- Swapping neighbors to optimize link distribution;
- Adjusting backup connections based on reliability and distance;
- Evaluating improvements based on PDR and expected node failure rates.

Iteration Process

Each iteration consists of:

- Constructing an initial solution (Construction Phase);
- Performing local search improvements (Local Search Phase);
- Comparing the newly obtained solution with the best solution found so far;
- Retaining the best solution across all iterations.

3.5.2 Implementation Details and Pseudocode

In this section, we present the core algorithms used to build and optimize a fault-tolerant WSNs using the GRASP **Figure 3.1**:

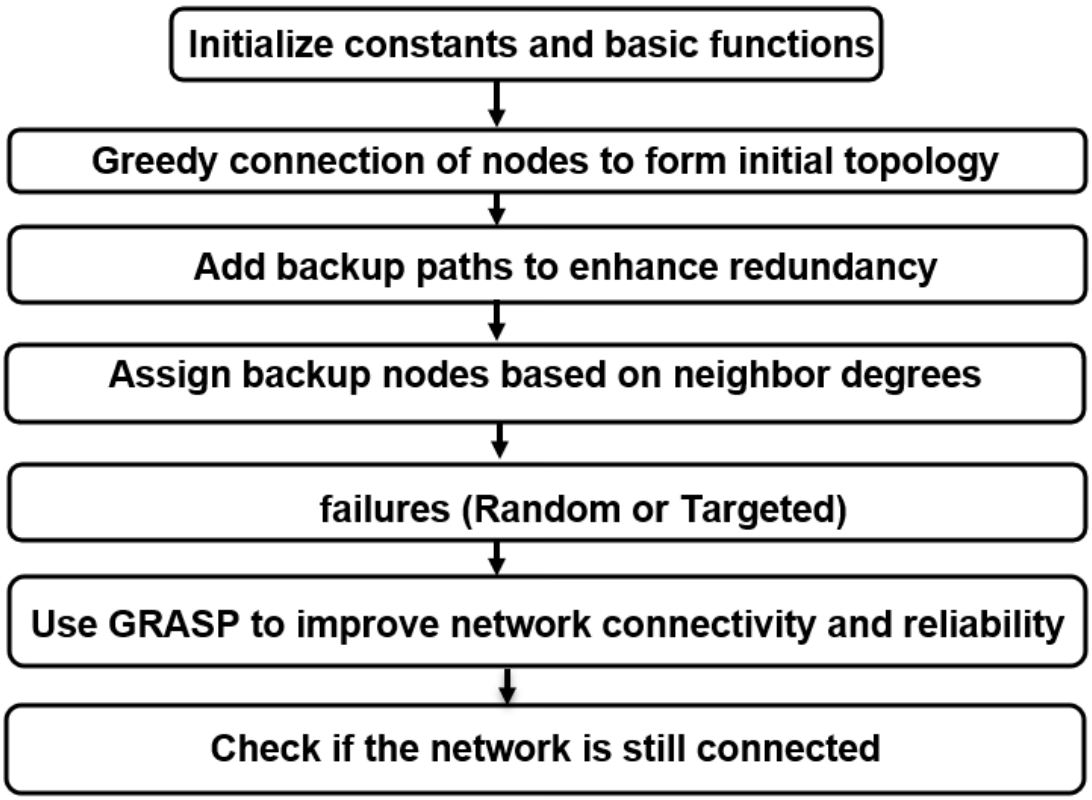

Figure 3.1: Algorithm Implementation Flowchart .

Variables and Basic Functions Initialization

Algorithm 1 Initialize Constants and Distance Function

Initialize constants:
AREA_SIZE ← 100
MAX_TRANSMISSION_RANGE ← 20
MAX_NEIGHBORS ← 3
MIN_REDUNDANT_PATHS ← 2
Define function *calculate_distance*(*node1*, *node2*):
return Euclidean distance between *node1* and *node2*

Explanation:

- **AREA SIZE:** Defines the size of the deployment area (e.g., 100x100 units);
- **MAX TRANSMISSION RANGE:** Maximum distance a node can directly communicate with another node;
- **MAX NEIGHBORS:** Limits the maximum number of neighbors a node can connect to;
- **MIN REDUNDANT PATHS:** Minimum number of backup paths (neighbors) each node should have to improve reliability;
- **calculate distance :** Computes the straight-line distance between two nodes using the Euclidean formula.

Initial Node Connection Using Greedy Heuristic

Algorithm 2 Greedy Node Connection

Input: *nodes*, *max_neighbors*
Output: *connectivity matrix*

```
Create empty connectivity matrix  Initialize neighbor count for each node to 0  for each node i in nodes do
    Generate list of candidate neighbors j not yet connected and within max_neighbors
     Sort candidates by distance  for each candidate j do
        if both i and j can accept more neighbors then
            Connect i and j in connectivity matrix  Increment neighbor counts for i and j
             break
return connectivity matrix
```

Explanation:

- Connect each node to its closest neighbors while respecting the max neighbor limit;
- Ensures efficient network by linking nodes to nearby nodes;
- Uses a connectivity matrix to track connections between nodes (1 = connected, 0 = not connected);
- Stops adding neighbors once the node reaches the max allowed neighbors.

Addition of Backup Paths for Reliability

Algorithm 3 Add Backup Paths

Input: *connectivity*, *nodes*, *min_redundant_paths*
Output: Updated *connectivity matrix*

```
for each node do
    while its number of neighbors < min_redundant_paths do
        Find possible nodes to connect (not already connected and within neighbor limits)
        Sort candidates by distance  for each candidate do
            if both nodes can accept more neighbors then
                Connect them in connectivity matrix
                Update neighbor counts
                break
        if no candidates found then
            break
return updated connectivity matrix
```

Explanation:

- After initial connections, add extra links to nodes that don't meet the minimum backup path requirement;
- Backup paths help the network survive node failures;
- Connections are still made preferentially with closest nodes to maintain efficiency.

Backup Node Assignment for Each Primary Node

Algorithm 4 Assign Backup Nodes

Input: *connectivity*
Output: *backup map*
for *each node* **do**
    Get its neighbors
    Choose neighbor with the highest degree as backup node
    Store backup node in backup map
return backup map

Explanation:

- Assign a backup node for each node from among its neighbors;
- The neighbor with the highest number of connections is typically chosen because it's more reliable;
- The backup map is used to reroute traffic if the primary node fails.

Network Connectivity Check

Algorithm 5 Check Network Connectivity

Input: *connectivity*
Output: True or False
Create graph from connectivity matrix Check if the graph is connected using a graph library
return True or False

Explanation:

- Verifies if every node is reachable from every other node after failures or changes;
- Connectivity is critical for network functionality;
- Uses graph traversal algorithms internally.

Network Optimization via GRASP

Algorithm 6 GRASP Improve Connectivity

Input: *nodes*, *connectivity*, *iterations*
Output: Best connectivity and node positions

```
Initialize best PDR and best connectivity
for number of iterations do
    Slightly perturb node positions within area bounds
    Generate new connectivity using greedy_connect_nodes
    Add backup paths
    if network is connected then
        Calculate new PDR
        if new PDR > best PDR then
            Update best connectivity, nodes, and PDR
return best connectivity and node positions
```

Explanation

1. Motivation for Using GRASP

   - **Escaping Local Optima:**After the initial network is built using greedy connection and backup links, the topology may be stuck in a "local optimum"a solution that is good but not the best possible. Small changes might not improve it;
   - **Handling Failures:**When random or targeted failures occur, the network may lose connectivity or reliability;
   - **Main Goal:**The aim is to intelligently reconfigure the network to maximize the Packet Delivery Ratio (PDR) and ensure robust connectivity.

2. How GRASP Improves the Network

   - **Iterative Process**
     - The GRASP algorithm performs several improvement iterations (typically 10 by default);
     - In each iteration, a new candidate topology is generated and evaluated.
   - **Random Perturbation** of Node Positions:Each node's position is slightly perturbed within the allowed area (AREA SIZE),This helps explore new network layouts that might offer better connectivity or redundancy.
   - **Greedy Reconnection**
     - The function greedy connect nodes is called to rebuild the connectivity matrix based on the new node positions;

  - Each node connects to its closest neighbors, respecting the maximum allowed number of neighbors.

- **Adding Backup Paths**
  - The function add backup paths enhanced is used to ensure each node has at least the minimum number of redundant (backup) connections;
  - This step enhances reliability and fault-tolerance.
- **Checking Network Connectivity**
  - The function is network connected checks if the new network is fully connected;
  - If the network is not connected, this candidate solution is discarded.
- **Evaluating and Retaining the Best Solution**
  - The new solution is evaluated, usually by calculating the PDR (the ratio of actual links to possible links);
  - If the new solution is better than the previous best, it is stored as the current best.

3.6 Performance Evaluation Metrics

Evaluated via:

- Packet Delivery Ratio (PDR);
- Maximum distance between nodes;
- Number of node-disjoint paths;
- Number of active backup nodes after failure.

Metrics are calculated using functions such as: `calculate_pdr()`, `count_node_disjoint_paths()`, `max_distance_in_connected_network()`.

3.7 Conclusion

This chapter outlined the practical implementation of GRASP for improving the fault tolerance of WSNs. It detailed how the algorithm was adapted and deployed to maintain network connectivity and performance under partial failure conditions. By focusing on reconnection strategies and link optimization, this chapter laid the groundwork for a resilient network design. In the next chapter, we proceed to evaluate the effectiveness of our proposed solution through simulation-based performance assessments under various fault scenarios.

Chapter 4 : Simulation and Experimental Evaluation

4.1 Introduction

In this chapter, we evaluate the effectiveness of the proposed GRASP-based approach for designing fault-tolerant Wireless Sensor Networks (WSNs). Building on the theoretical foundations and mathematical modeling presented in previous chapters, this chapter focuses on a comprehensive simulation process aimed at testing the algorithm under realistic network conditions.Two main fault scenarios were examined: **random failures** and **targeted (intentional) failures**. Each scenario was simulated to assess how well the network maintains connectivity and performance under different types of disruptions. The evaluation was carried out using a set of performance metrics tailored to assess resilience and efficiency. Furthermore, a comparative analysis was conducted between the results of the two scenarios, highlighting the algorithm's adaptability and robustness. To further validate our approach, simulation results were compared with those from related research studies, providing insights into the advantages and limitations of our method relative to existing fault-tolerant strategies.

4.2 Simulation Environment

To ensure a reliable and repeatable evaluation,all experiments were performed under a controlled simulation environment.The environment includes the software tools used, the hardware setup, and a detailed explanation of the key libraries involved in the implementation.

4.2.1 Simulation Tools

All simulations were conducted using Python, a high-level and versatile programming language widely used in academic and industrial contexts[23].The implementation and experimentation were carried out using Anaconda distribution, specifically within Jupyter Notebook, which provided an interactive and flexible coding environment ideal for iterative testing[24], visualization, and documentation[25].

Figure 4.1: Python and Jupyter notebooks logos .

Python was selected for multiple reasons:

- It had already been used extensively in previous practical coursework, which made it a familiar and accessible choice for implementing the proposed method;
- It was the primary tool used in the simulation module (Agent based modeling and simulation) in the first semester, which provided additional motivation and confidence in using it for this project;
- Its object-oriented programming model enabled a modular and extensible architecture;
- It supports a broad ecosystem of scientific and optimization libraries, such as NumPy, SciPy, and NetworkX;
- Integration with Jupyter Notebook allowed for clear visualization, interactive scenario navigation, and real-time analysis.

4.2.2 System Specifications

4.2.2.1 Hardware

The simulations were executed on a personal computer with the following specifications:

- Operating System: Windows 10 Professional (64-bit);
- Processor: Intel(R) Core(TM) i5-6300U CPU @ 2.40GHz (up to 2.50GHz);
- RAM: 8.00 GB;
- System Type: 64-bit Operating System, x64-based processor.

These specifications were sufficient to handle the simulation workload for small- to medium-sized network instances, although more complex simulations might benefit from higher computational resources.

4.2.2.2 Software

Libraries Used and Their Roles in the Project: Several Python libraries were utilized to develop and implement the GRASP-based simulation for Fault-Tolerant Wireless Sensor Networks (WSNs). Each library served a specific purpose during different stages of the simulation process. Below is a detailed overview of the key libraries and their functionalities:

1. **Random:** The `random` library was used to introduce stochastic elements into the simulation. Specifically(this randomness is essential to reflect real-world uncertainty and enhance solution diversity.), it was applied during:
    - The random placement of sensor nodes across the network area;
    - The randomized selection of candidate nodes during the construction phase of the GRASP algorithm.
2. **Math:** The `math` module provides standard mathematical functions. In this project, it was primarily used to compute the *Euclidean distance* between sensor nodes, a critical step in determining connectivity and proximity in the network graph.
3. **Networkx:** `networkx` is a comprehensive library for the creation, manipulation, and study of complex networks and graphs. Its role in the simulation included:
    - Constructing the network graph representing sensor nodes and their connections;
    - Identifying connected components to analyze network robustness;
    - Verifying whether the network remains connected to the sink node after node failures or removals.
4. **Matplotlib.pyplot:** This library was employed for the visualization of simulation outcomes. It was used to graphically represent the topology of the WSN, including:
    - Sensor nodes and the sink node;
    - Edges representing communication links;
    - The status of nodes (alive or dead).
5. **Ipywidgets:** `Ipywidgets` provides tools to build interactive widgets in Jupyter Notebooks. Within the simulation environment, it enabled:

- The creation of interactive controls like ``Next Simulation'' and ``Previous Simulation'' buttons;
- A user-friendly interface for navigating between different simulation scenarios dynamically.

6. **IPython.display:** This module was used to dynamically display outputs in notebook cells. Specifically, it facilitated:
   - The live update of network graphs and performance tables;
   - Enhancing the interactive simulation experience by refreshing visual elements as user input changes.
7. **Tabulate:** `tabulate` was used to organize and format the final simulation results into structured tables. It was responsible for presenting performance metrics such as:
   - The number of remaining functional nodes;
   - The count of active communication links;
   - The number of disconnected components.

4.2.3 Network Configuration and Parameters

The following section outlines the key parameters and configuration settings used in the network simulation. Each parameter is carefully selected and justified to reflect realistic and practical conditions for a Wireless Sensor Network (WSN). Understanding these parameters is essential for interpreting the simulation results and assessing the network's performance and reliability. These parameters are summarized in **Table 4.1: Network Simulation Parameters.**

1. **Number of Nodes (15 to 50 nodes)**
   Justification: Selecting a node range between 15 and 50 reflects a moderately sized Wireless Sensor Network (WSN), providing a balanced environment for studying performance under varying network densities. This range allows examining how network behavior, including communication overhead and routing complexity, scales with node count, which is crucial for designing scalable and efficient sensor deployments;
2. **Deployment Area (100m $\times$ 100m)**
   Justification: A fixed 100m by 100m deployment area represents a realistic and manageable spatial environment often used in practical WSN applications such as environmental monitoring or smart buildings. This size allows for controlled experiments on node distribution impact, communication range effectiveness, and network coverage, enabling reliable performance evaluation;

3. **Maximum Communication Range (20 meters)**
   Justification: A communication range of 20 meters corresponds to typical low-power wireless communication standards like ZigBee or IEEE 802.15.4. This constraint realistically models device limitations and environmental interference, influencing network connectivity, energy consumption, and interference levels, thus providing applicable insights for real-world deployment;

4. **Maximum Neighbors per Node (3 neighbors)**
   Justification: Limiting each node to a maximum of three neighbors reduces network congestion and interference, lowering energy usage by minimizing unnecessary transmissions. This parameter simplifies routing and connection maintenance, optimizing network efficiency, especially in energy-constrained WSNs;

5. **Minimum Number of Redundant Paths (2 backup paths)**
   Justification: Maintaining at least two redundant paths enhances network reliability and fault tolerance by ensuring alternative routes for data transmission if the primary path fails. This is critical for mission-critical applications where uninterrupted data flow is essential, such as in healthcare or security systems;

6. **Number of Simulation Runs (30 runs)**
   Justification: Conducting 30 simulation iterations ensures statistical significance and reduces the effect of random variations, leading to more reliable and reproducible performance metrics. This aligns with best practices in simulation-based research to achieve robust results;

7. **Fault Rate (5% to 15% randomly distributed)**
   Justification: Simulating faults in the range of 5% to 15% mirrors real-world failure rates due to hardware faults, energy depletion, or environmental interference. This range allows assessment of the network's fault tolerance and adaptive mechanisms under realistic operating conditions;

8. **Node Distribution (Random with a fixed central node)**
   Justification: Having one fixed central node (sink) with other nodes randomly distributed models typical WSN topologies where data is collected at a central base station. This layout enables studying routing strategies and load balancing under heterogeneous spatial distributions.

Parameter	Value	Comment
Number of Nodes	15 to 50 nodes	Suitable network size.
Deployment Area	100m x 100m	Appropriate experiment area.
Maximum Communication Range	20 meters	Typical communication range.
Maximum Neighbors per Node	3 neighbors	Reduces congestion and energy use.
Minimum Number of Redundant Paths	2 backup paths	Ensures backup routes.
Number of Simulation Runs	30 runs	Reliable statistical results.
Fault Rate	5% to 15% randomly distributed	Simulates realistic faults.
Node Distribution	Random with a fixed central node	Random layout with central sink.

Table 4.1: Network Simulation Parameters.

4.3 Evaluation Scenarios

This section briefly describes two failure scenarios used in the network simulation: random failures representing unpredictable node malfunctions, and targeted failures simulating strategic attacks on critical nodes:

4.3.1 Random Failure Scenario

- **Description:**In this scenario, a random subset of nodes is selected to fail, simulating unpredictable events like hardware malfunctions or environmental disturbances.
- **Implementation Details**
  - A fraction of nodes is randomly chosen to fail;
  - The simulate random failure with backup function deactivates these nodes and attempts to maintain network connectivity by activating predefined backup nodes;
  - The grasp improve connectivity function is then employed to optimize the network's structure post-failure, aiming to restore or enhance connectivity.

4.3.2 Targeted Failure Scenario

- **Description:** This scenario simulates deliberate attacks on the network by targeting and disabling the most connected (high-degree) nodes, reflecting strategic disruptions.

- **Implementation Details**
  - Nodes are ranked based on their connectivity (degree), and the top fraction is selected for failure;
  - The simulate targeted failure function deactivates these critical nodes;
  - Similar to the random failure scenario, grasp improve connectivity is applied to reconfigure the network, striving to maintain or restore overall connectivity.

4.4 Evaluation Metrics

The following **Table 4.2: Evaluation Metrics** summarizes the key evaluation metrics used in this study's network simulation. It includes detailed descriptions of each metric along with comments that highlight their importance in evaluating the network's performance and fault tolerance.

Metric	Description	Comment
Number of Nodes	The total count of nodes in the network, indicating the network size	Reflects the network's scale and complexity, fundamental for performance assessment.
Max Distance (m)	The maximum geographical distance between any two connected nodes, showing how spread out the nodes are	Indicates the geographic spread, impacting latency and communication quality.
Backup Nodes Active	The number of backup nodes activated to replace failed or disabled nodes	Shows the network's ability to recover and maintain service continuity after failures.
Node-Disjoint Paths (0 to N-1)	The count of node-disjoint paths between node 0 and node N-1, indicating the network's fault tolerance	Measures fault tolerance by providing alternative routes in case of node failures.
Network Connection	A boolean indicator showing whether the network remains connected after failures (random or targeted)	Indicates if the network maintains connectivity and functionality post-failure.
Percentage of Operational Nodes	The percentage of nodes still operational and not failed after the failure event	Reflects the operational capacity and resilience of the network after failures.
Failed Nodes	A list of nodes that have failed during the failure scenario	Provides detailed failure information to analyze vulnerabilities and failure causes.

Table 4.2: Evaluation Metrics.

4.5 Simulation Interface

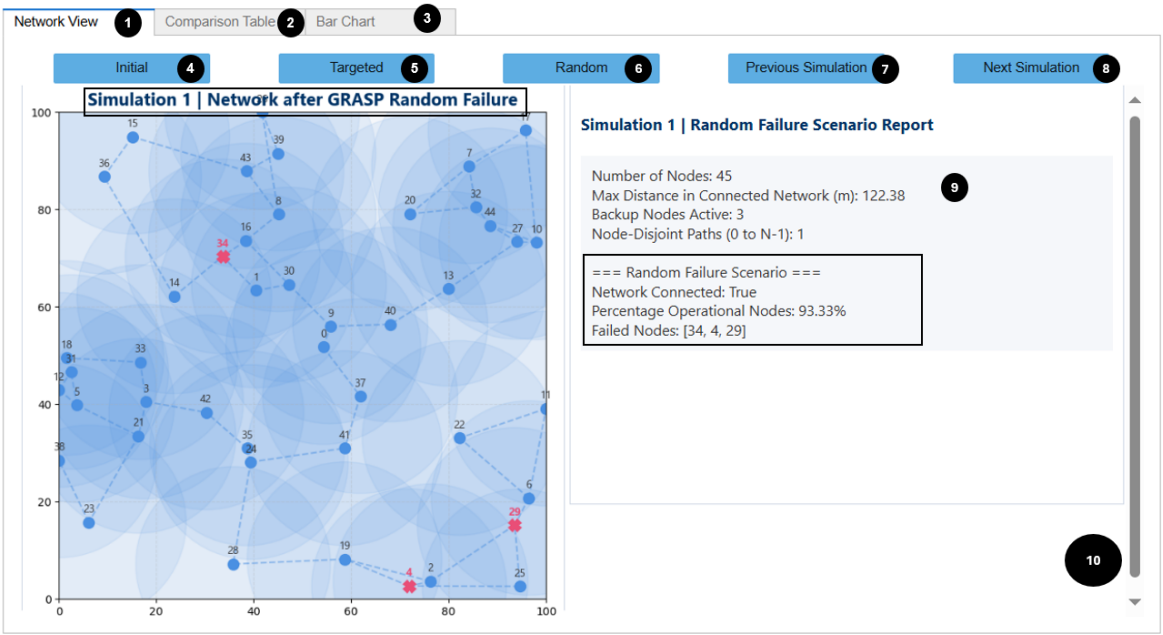

Figure 4.2: The Network Simulation Interface.

The above **Figure 4.2: The Network Simulation Interface** illustrates the graphical interface of the network simulation used in this study. The interface consists of several main sections that allow the user to view the results:

1. **Network View:** This is the main tab in the interface that displays the network graph and its details along with associated performance reports. Selecting this tab allows you to visually observe the distribution of nodes and links;

2. **Comparison Table:** A button or tab that presents a comparison table of various statistics across scenarios;

3. **Bar Chart:** A button to display results in the form of vertical bar charts (Bar Chart) for visually comparing different statistics across scenarios;

4. **Initial:** A button that shows the initial state of the network before any failures or improvements are applied. It displays the original distribution of nodes and links;

5. **Targeted:** A button showing simulation results when specific nodes (usually high-degree nodes) are targeted for failure, representing a directed attack scenario;

6. **Random:** A button showing simulation results under random node failures, representing a random failure scenario;

7. **Previous Simulation:** A button to navigate to the previous simulation if multiple experiments exist;

8. **Next Simulation:** A button to navigate to the next simulation if multiple experiments exist;

9. **Simulation Report:**
   Displays statistical information about the current network, such as:

   - Total number of nodes ;
   - Maximum distance between connected nodes ;
   - Number of active backup nodes ;
   - Number of disjoint paths between the first and last nodes .

   This information is updated only after failures occur:

   - Whether the network remains connected after failure ;
   - Percentage of operational nodes post-failure ;
   - List of failed nodes in this scenario .

10. **Scroll Bar:** Allows scrolling through the report if it is long or contains additional details not visible at once.

4.6 Performance Evaluation and Results

To evaluate the performance of the GRASP algorithm in designing fault-resistant networks, we show Figures from Simulations 01 and 28, comparison table of all simulations and bar chart of WSN performance using GRASP:

4.6.1 Simulation 01

Pre-Failure Network

The image below, **Figure 4.3: Initial State (Pre-Failure Network Simulation 01)**, illustrates the network's initial condition of simulation 01 before any failures occur. The following provides a detailed explanation of the image, including the visualization, key performance metrics, and subsequent observations and analysis:

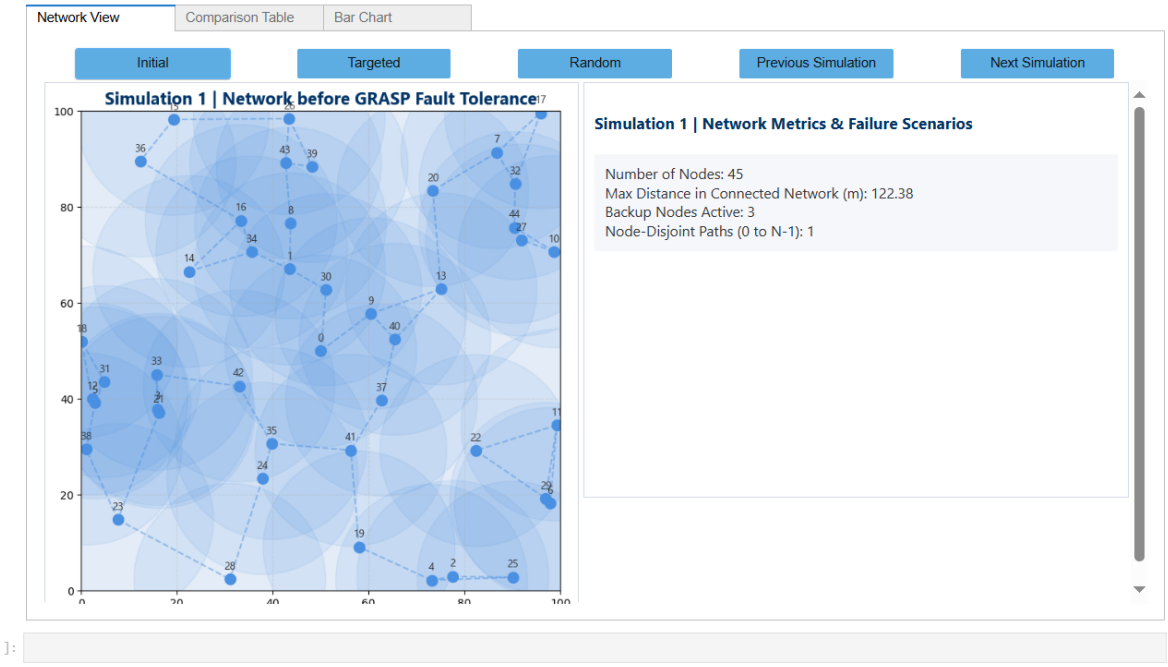

Figure 4.3: Initial State (Pre-Failure Network simulation 01).

1. **Displaying and Explaining the Image**

   - **Node Distribution:** In this simulation, we see 45 nodes distributed across a 100x100 unit area. The nodes are fairly evenly spread, which helps to avoid congestion and ensures wide coverage;
   - **Blue Circles:** Each blue circle represents the communication range of a node. The overlaps between circles indicate that most nodes can connect to several neighbors, supporting basic network connectivity;

- **Dashed Lines:**Dashed lines show the active links between nodes. In this initial setup, each node is connected to at least one neighbor, but the overall redundancy is limited;
- **Node Labels:** Each node is labeled with a unique number, which makes it easy to reference specific nodes when discussing scenarios or failures.

2. **Explaining the Key Metrics**

- **Number of Nodes (45):** The network consists of 45 nodes, providing a substantial coverage area and potential for robust connectivity;
- **Max Network Distance (122.38 meters):** The maximum distance between any two connected nodes is 122.38 meters, indicating that all nodes are within communication range and the network is physically cohesive;
- **Backup Nodes (3):** There are 3 backup nodes that have been pre-assigned. These nodes are on standby and ready to be activated if failures occur;
- **Node-Disjoint Paths (1):** Currently, there is only one node-disjoint path between the source and destination nodes. This means that if any single node along the path fails, connectivity could be lost.

3. **Observations & Analysis**

- **Pre-Failure Fragility:** With only one node-disjoint path, the network is fragile at this stage. A single node or link failure could disrupt connectivity between certain pairs of nodes;
- **Backup Nodes Pre-Assigned:** Although backup nodes are **ready** and assigned, they are not yet active in this initial phase. Their effectiveness will be seen once failures are introduced;
- **GRASP Path Redundancy:**The GRASP algorithm has maintained the existing path redundancy but, in this initial configuration, has not yet improved it. This highlights the need for further optimization or backup activation to enhance fault tolerance.

Targeted Failure Scenario

The image below,**Figure 4.4: Targeted Failure Scenario Network of simulation 01**, illustrates the state of the network following a targeted failure scenario of simulation 01. The following provides a detailed explanation of the visualization, key performance metrics, and subsequent observations and analysis:

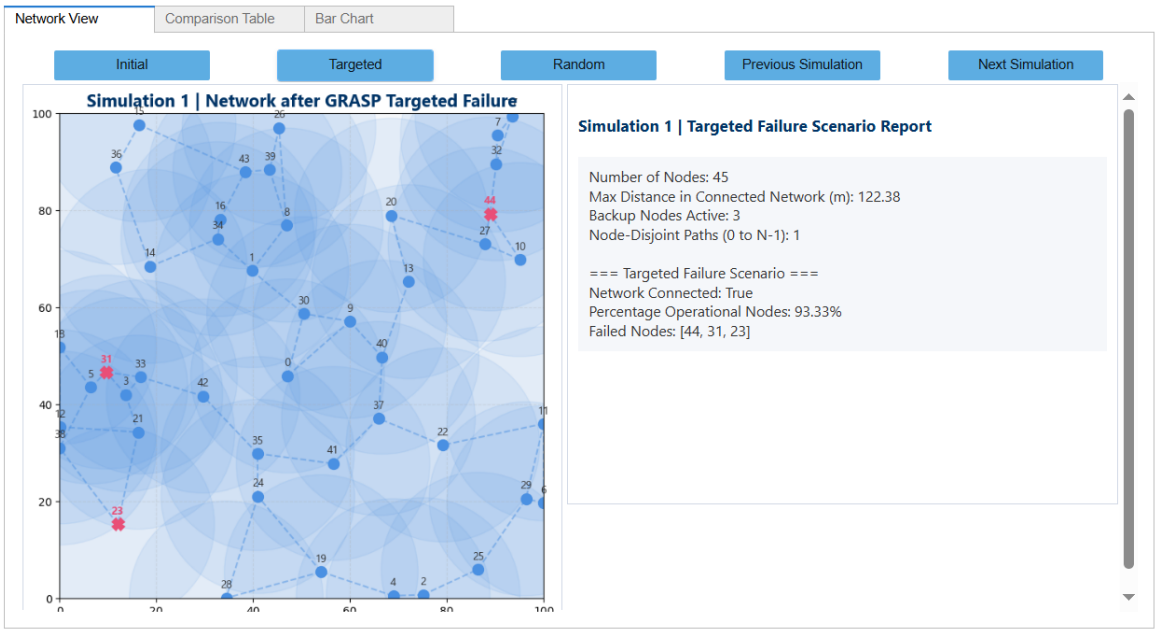

Figure 4.4: Targeted Failure Scenario Network of simulation 01.

1. **Displaying and Explaining the Image**

   - **Node Distribution:** As we can see, the network consists of 45 nodes distributed across a 100x100 unit area. The nodes are spread out to ensure broad coverage and minimize congestion;
   - **Blue Circles:** Each blue circle represents the communication range of a node. The overlapping circles show that most nodes can connect to several neighbors, which is important for maintaining network connectivity;
   - **Dashed Lines:** The dashed lines indicate the active links between nodes. These links form the backbone of the network, allowing data to travel between nodes even if some connections fail;
   - **Node Labels and Failures:** Each node has a unique label for identification. In this scenario, nodes 44, 31, and 23 are marked with red stars, indicating they have failed as part of a targeted failure scenario.

2. **Explaining the Key Metrics**

   - **Total Nodes (45) :** The network starts with 45 nodes, providing a robust base for connectivity;
   - **Max Network Distance (122.38 meters):** The maximum distance between any two connected nodes is 122.38 meters, ensuring all nodes are within communication range;
   - **Active Backup Nodes (3):** There are 3 backup nodes actively assigned, ready to compensate for failures and maintain network performance;
   - **Node-Disjoint Paths (1):** Currently, there is only one node-disjoint path between the source and destination. This means the network is more vulnerable: if a single node on this path fails, connectivity could be lost for some node pairs;
   - **Operational Nodes (93.33%):** After the failure of three nodes, 93.33% of the nodes remain operational and connected;
   - **Network Connectivity Maintained:** Despite the targeted failure of three likely high-degree or central nodes, the network remains fully connected, with no isolated nodes.

3. **Observations & Analysis**

   - **Impact of Targeted Failures:** Three nodes failed—likely **high-degree** or **central nodes**. Normally, such failures could fragment the network, but the GRASP algorithm successfully compensated for these losses;
   - **Effectiveness of GRASP: The operational percentage (93.33%**) is identical to that observed in the random failure scenario, suggesting that GRASP provides equal resilience to both random and targeted failures;
   - **Redundancy Limitation:** However, with only one node-disjoint path, the network is fragile at this stage. Any additional failure along the main path could disrupt connectivity.

Random Failure Scenario

The image below,**Figure 4.5: Random Failure Scenario Network of simulation 01**, illustrates the state of the network of simulation 01 after the occurrence of random node failures. The following provides a detailed explanation of the visualization, key performance metrics, and subsequent observations and analysis.

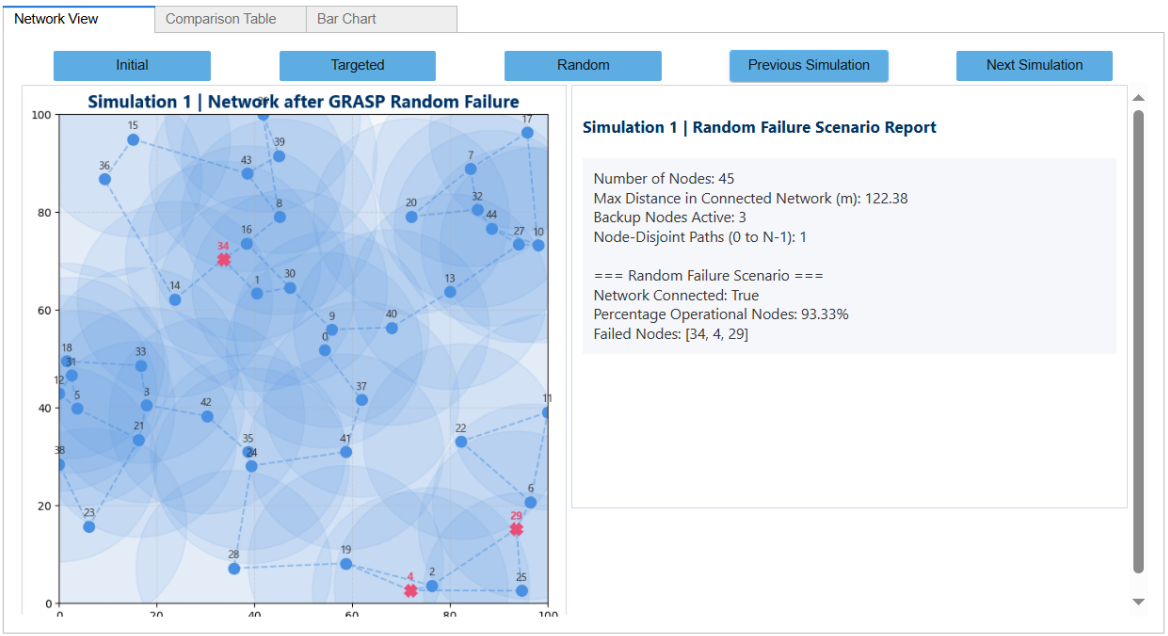

Figure 4.5: Random Failure Scenario Network of simulation 01.

1. **Displaying and Explaining the Image**

   - **Node Distribution:** As shown in the image, the network consists of 45 nodes distributed across a 100x100 unit area. The nodes are spread out to ensure broad coverage and minimize congestion, which is essential for maintaining stable connectivity;
   - **Blue Circles:** Each blue circle represents the communication range of a node. The overlapping circles indicate that most nodes have multiple neighboring connections, which is crucial for redundancy and network robustness;
   - **Dashed Lines:** Each node is labeled with a unique identifier. In this scenario, nodes 34, 4, and 29 are marked with red stars, indicating they have failed randomly during the simulation;
   - **Node Labels and Failures):** Each node is labeled with a unique identifier. In this scenario, nodes 34, 4, and 29 are marked with red stars, indicating they have failed randomly during the simulation.

2. **Explaining the Key Metrics**

   - **Total Nodes (45):** The network starts with 45 nodes, providing a strong base for connectivity;
   - **Max Network Distance (122.38 meters):** TThe maximum distance between any two connected nodes is 122.38 meters, confirming that all nodes are within communication range and the network remains physically cohesive;
   - **Active Backup Nodes (3):** There are 3 backup nodes actively assigned. These nodes are ready to compensate for failures and help maintain network performance;
   - **Node-Disjoint Paths (1):** : Currently, there is only one node-disjoint path between the source and destination. This means the network is more vulnerable: if a single node on this path fails, connectivity could be lost for some node pairs;
   - **Operational Nodes (93.33%):** After the random failure of three nodes, **93.33% of the nodes remain operational and connected**;
   - **Network Connectivity Maintained:** Despite the failure of three nodes (about 6.67% of the network), the network remains fully connected, with no isolated nodes.

3. **Observations & Analysis**

   - **Random Failures:** Three nodes failed randomly. The network's ability to remain connected demonstrates resilience against unpredictable failures;
   - **Backup Node Activation:** The activation of three backup nodes helped ensure continuity and prevented network partitioning;
   - **Strength: :** A key strength is the robustness in maintaining connectivity even after multiple random node failures;
   - **Weakness:** However, with only one node-disjoint path, the network is fragile at this stage. Any further failure along the main path could disrupt connectivity.

Comparative Summary

The **Table 4.3: Comparative summary of network metrics** below ,summarizes and compares the key network metrics across the three scenarios previously discussed. It is followed by key conclusions highlighting the effectiveness and limitations of the GRASP algorithm:

Metric	Pre-Failure	Random Failure	Targeted Failure
Total Nodes	45	45	45
Max Distance	122.38m	122.38m	122.38m
Backup Nodes	3 (inactive)	3 (active)	3 (active)
Node-Disjoint Paths	1	1	1
Operational Nodes	100%	93.33%	93.33%
Failed Nodes	--	[34, 4, 29]	[44, 31, 23]
Connectivity	Yes	Yes	Yes

Table 4.3: Comparative summary of network metrics.

- **Key Conclusions**
  - **GRASP Effectiveness**
    - Maintains connectivity under both failure types;
    - 3 backup nodes sufficient for ~7% node loss.
  - **Limitations**
    - Single node-disjoint path is critical bottleneck;
    - No topology optimization post-failur.

4.6.2 Simulation 28

Pre-Failure Network

The image below, **Figure 4.6: Pre-Failure Network of Simulation 28**, illustrates the network's initial condition of simulation 28 before any failures occur. The following provides a detailed explanation of the image, including the visualization, key performance metrics, and subsequent observations and analysis;

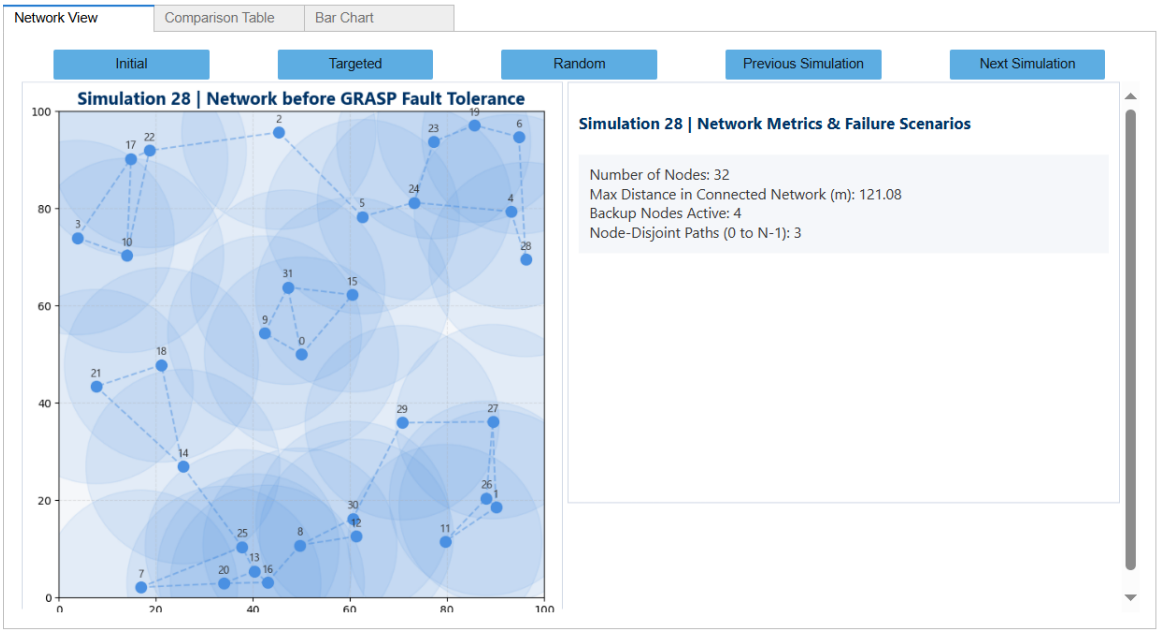

Figure 4.6: Pre-Failure Network of simulation 28.

1. **Displaying and Explaining the Image**

   - **Node Distribution:** As we can see, the nodes are fairly evenly distributed across the area (100x100 units), ensuring there are no congested or empty zones. This helps avoid communication bottlenecks within the network;
   - **Blue Circles:** Each blue circle represents the communication range of a node. Notice the overlaps between these circles, which means each node can connect to multiple neighbors. This overlap is crucial for providing alternative paths in case of failures;
   - **Dashed Lines:** The dashed lines indicate the actual links between nodes. Some nodes have multiple connections, which increases the network's flexibility and fault tolerance;
   - **Node Labels:** Each node is labeled with a unique identifier, which helps us track and discuss specific scenarios, such as node failures, during the presentation.

2. **Explaining the Key Metrics**

   - **Number of Nodes (32):** The network consists of 32 nodes, which is sufficient to cover the area effectively while providing multiple paths between most nodes;
   - **Maximum Network Distance (121.08 meters):** The longest direct distance between any two connected nodes is approximately 121 meters. This indicates that all nodes are within communication range, with no isolated nodes;
   - **Backup Nodes (4):** There are 4 active backup nodes strategically placed to take over when needed, ensuring quick recovery from faults;
   - **Node-Disjoint Paths (3):** Each pair of nodes can communicate through 3 separate node-disjoint paths. This means the network can tolerate failures of up to two nodes or links without losing connectivity between any two points.

3. **Observations & Analysis**

   - **Fault Tolerance:** Having 3 node-disjoint paths guarantees that the network remains connected even if multiple nodes or links fail;
   - **Geographical Distribution:** The balanced geographic distribution minimizes the risk of isolated areas or overloaded nodes, reducing the chance of communication breakdowns due to node failures;
   - **Use of Backup Nodes :**Backup nodes are strategically distributed to quickly replace failed nodes, minimizing downtime and improving network reliability;
   - **Network Readiness :**Overall, these metrics indicate that the network is highly resilient and well-prepared to handle various failure scenarios before any faults occur.

Targeted Failure Scenario

The image below, **Figure 4.7: Targeted Failure Scenario Network of simulation 28**, illustrates the network's state following a targeted failure scenario of simulation 28. The following provides a detailed explanation of the visualization, key performance metrics, and subsequent observations and analysis:

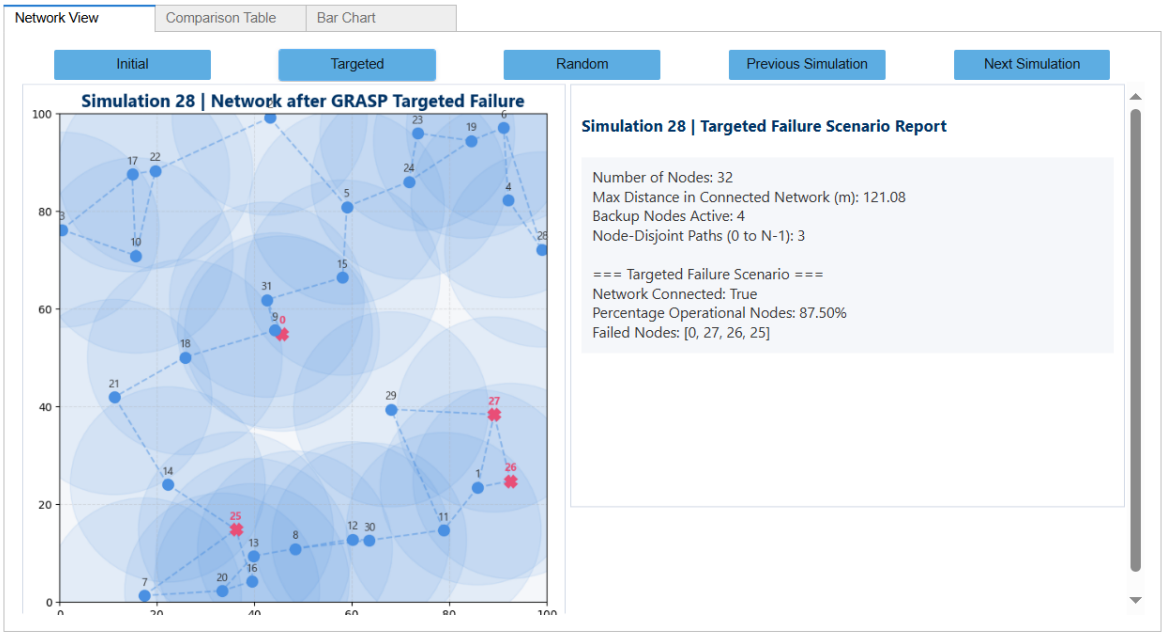

Figure 4.7: Targeted Failure Scenario Network of simulation 28.

1. **Displaying and Explaining the Image**

   - **Node Distribution:** Here, we see the network after a targeted failure scenario. The nodes (blue circles) are still well distributed across the 100x100 unit area, ensuring broad coverage ;
   - **Blue Circles:** Each blue circle represents a node's communication range. The overlapping areas indicate that nodes can still connect to multiple neighbors, which is crucial for maintaining redundancy;
   - **Dashed Lines:** Dashed lines show the active links between operational nodes. Despite the failures, the network remains interconnected, demonstrating the effectiveness of the design;
   - **Failed Nodes (Red X):** Nodes marked with a red X (nodes 0, 25, 26, and 27) represent the failed nodes in this targeted attack scenario. These account for 12.5% of the network.

2. **Explaining the Key Metrics**

   - **Failed Nodes (12.5% failure rate):** Four nodes have failed: 0, 25, 26, and 27, which is 12.5% of the total nodes;
   - **Operational Nodes (87.5%):** Despite these failures, 87.5% of the nodes remain operational and connected;
   - **Node-Disjoint Paths (3):** Importantly, there are still 3 node-disjoint paths between node pairs, meaning the network maintains high fault tolerance even after targeted attacks;
   - **Connectivity Maintained**: The network remains fully connected, with no isolated nodes or partitions, thanks to the redundancy built into the design.

3. **Observations & Analysis**

   - **Critical Node Failure:** Node 0, likely a central hub or root in routing paths, was one of the failed nodes. Normally, losing such a node would disrupt the network, but the GRASP algorithm preserved overall connectivity;
   - **Resilience to Targeted Attacks:** The fact that the number of node-disjoint paths did not decrease confirms the network's resilience, even when high-degree or critical nodes are targeted;
   - **Effective Backup Activation:** Backup nodes and redundant paths successfully compensated for the 12.5% node loss, with no noticeable degradation in network performance.

Random Failure Scenario

The image below, **Figure 4.8: Random Failure Scenario Network of Simulation 28**, illustrates the network's state after random node failures have occurred of simulation 28. The following provides a detailed explanation of the visualization, key performance metrics, and subsequent observations and analysis:

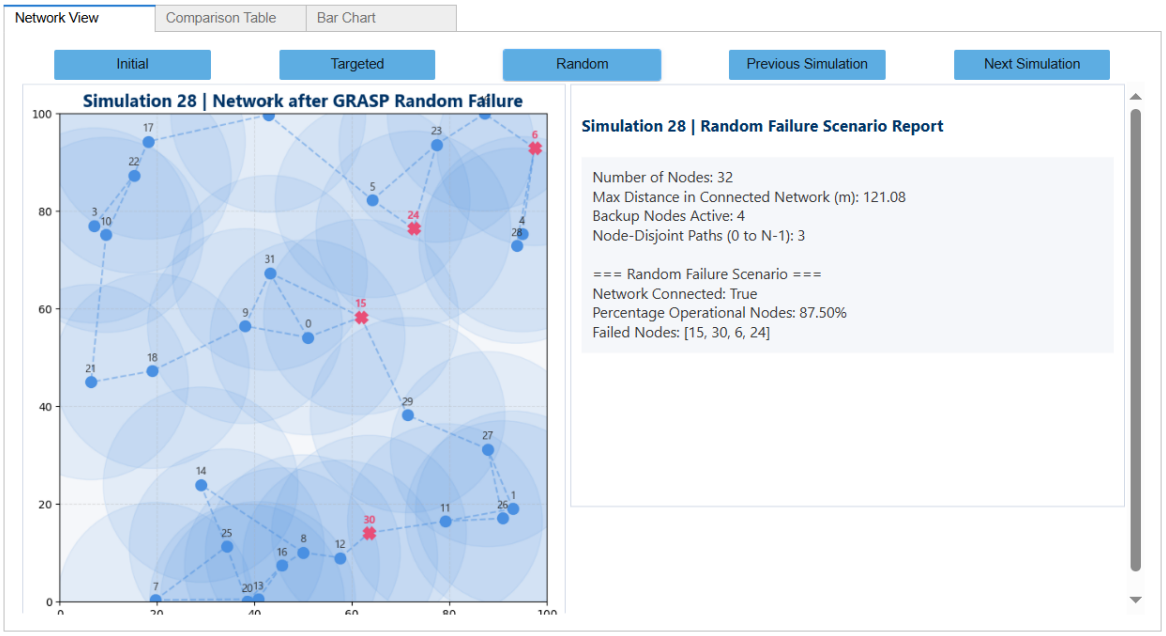

Figure 4.8: Random Failure Scenario Network of simulation 28.

1. **Displaying and Explaining the Image**

   - **Node Distribution:** In this simulation, we see 32 nodes distributed across a 100x100 unit area. The placement is balanced, ensuring there are no isolated or overly dense regions;
   - **Blue Circles:**Each blue circle represents the communication range of a node.The overlapping circles indicate that nodes have multiple neighboring connections, which is essential for redundancy and robust communication;
   - **Dashed Lines:** Dashed lines show the active links between nodes. These connections allow for multiple paths for data to travel, enhancing the network's ability to reroute traffic if a node fails;
   - **Failed Nodes (Red X):**Nodes 15, 30, 6, and 24 are marked with a red X. These represent the failed nodes in this random failure scenario, accounting for 12.5% of the network.

2. **Explaining the Key Metrics**

   - **Number of Nodes & Failures:** The network starts with 32 nodes. In this scenario, 4 nodes have failed, which is a 12.5% failure rate. Despite these failures, 87.5% of the nodes remain operational and connected;
   - **Maximum Network Distance:** The maximum distance between any two connected nodes is about 121 meters, showing that the network remains cohesive without any disconnected parts;
   - **Backup Nodes:** There are 4 backup nodes active, ready to take over if additional failures occur;
   - **Node-Disjoint Paths:** : There are still 3 node-disjoint paths between node pairs, meaning the network can tolerate multiple failures without losing overall connectivity.

3. **Observations & Analysis**

   - **Performance Parity :** The network maintains the same operational rate (87.5%) as in the targeted failure scenario, demonstrating the balanced and robust design;
   - **Non-Adjacent Failures:** The failed nodes are not adjacent, which helps prevent cascading failures and further disruption;
   - **Path Redundancy:** Having 3 redundant paths between nodes prevents bottlenecks and ensures data can still be routed efficiently, even with random node failures;
   - **Network Resilience:** Overall, the network remains fully connected and operational, highlighting the effectiveness of the redundancy and backup strategies implemented by the GRASP algorithm.

Comparative Summary

The table below, **Table 4.4: Comparison of Metrics under Targeted and Random Failures**, presents a concise comparison of key performance indicators between the targeted and random failure scenarios previously discussed. This is followed by Notable Findings and key conclusions highlighting the strengths and limitations of the GRASP algorithm in maintaining network resilience under different failure conditions:

Metric	Targeted Failure	Random Failure
Operational Nodes	87.5%	87.5%
Node-Disjoint Paths	3 (no change)	3 (no change)
Failed Nodes	[0, 27, 26, 25]	[15, 30, 6, 24]
Connectivity	Maintained	Maintained

Table 4.4: Comparison of Metrics under Targeted and Random Failures.

- **Notable Findings**
  - Identical outcomes for both failure types highlight GRASP's agnosticism to failure mode;
  - Path redundancy (3 paths) is the primary driver of resilience no degradation post-failure.
- **Key Conclusions**
  - Superior fault tolerance due to increased node-disjoint paths (from 1 to 3);
  - Effective backup nodes (4 activated) ensured seamless recovery;
  - Consistent performance under both targeted and random failures.

4.6.3 Comparison table

Network View | Comparison Table | Bar Chart

	Simulation	Nodes	Max Distance (m)	Backup Nodes Active	Node-Disjoint Paths	Random Failure %	Targeted Failure %
0	1	45	122.380000	3	1	93.330000	93.330000
1	2	41	114.870000	3	1	92.680000	92.680000
2	3	32	107.860000	3	2	90.620000	90.620000
3	4	22	102.600000	2	0	90.910000	90.910000
4	5	48	125.580000	4	1	91.670000	91.670000
5	6	30	120.080000	3	1	90.000000	90.000000
6	7	36	111.890000	4	0	88.890000	88.890000
7	8	17	120.060000	1	2	94.120000	94.120000
8	9	49	122.000000	3	1	93.880000	93.880000
9	10	40	115.030000	3	0	92.500000	92.500000
10	11	19	108.950000	2	1	89.470000	89.470000
11	12	46	116.480000	5	1	89.130000	89.130000
12	13	30	103.210000	2	0	93.330000	93.330000
13	14	26	107.740000	1	0	96.150000	96.150000
14	15	20	124.580000	1	0	95.000000	95.000000
15	16	28	113.640000	2	2	89.290000	89.290000
16	17	36	101.060000	3	2	88.890000	88.890000

Figure 4.9: comparison table simulation part01.

Network View | Comparison Table | Bar Chart

11	12	46	116.480000	5	1	89.130000	89.130000
12	13	30	103.210000	2	0	93.330000	93.330000
13	14	26	107.740000	1	0	96.150000	96.150000
14	15	20	124.580000	1	0	95.000000	95.000000
15	16	28	113.640000	2	2	89.290000	89.290000
16	17	36	101.060000	3	2	88.890000	88.890000
17	18	28	124.910000	3	1	89.290000	89.290000
18	19	22	113.490000	1	1	95.450000	95.450000
19	20	47	113.250000	3	0	93.620000	93.620000
20	21	29	121.850000	3	2	89.660000	89.660000
21	22	18	107.970000	1	1	94.440000	94.440000
22	23	31	119.490000	3	1	87.100000	87.100000
23	24	23	114.240000	2	2	91.300000	91.300000
24	25	24	113.950000	1	1	95.830000	95.830000
25	26	18	101.570000	1	1	94.440000	94.440000
26	27	17	114.650000	1	0	94.120000	94.120000
27	28	32	121.080000	4	3	87.500000	87.500000
28	29	48	115.280000	6	0	87.500000	87.500000
29	30	23	116.600000	2	0	86.960000	86.960000

Figure 4.10: comparison table simulation part02.

1. **Analytical introduction to the tables presented:** These two images :**Figure 4.9: comparison table simulation part01** and **Figure 4.10: comparison table simulation part02** represent one integrated table summarizing the results of simulating the performance of wireless sensor networks using the advanced Metaheuristic GRASP algorithm:

2. **Network Performance Overview:** The data reveals varying network behavior based on size and characteristics:

    - Network Size
        - Node count ranges: 17-49 nodes;
        - Average: 32 nodes;
        - Larger networks (>40 nodes) show slightly lower performance (87-90%) compared to smaller ones (90-96%).
    - Inter-node Distances
        - Maximum distances: 101.06m to 125.58m;
        - Average: 114.23m (±7.5m standard deviation);
        - No clear correlation between network size and maximum distance.

3. **Fault Tolerance Performance**

    - Backup Node Effectiveness
        - Networks with ≥3 backup nodes maintain 89-93% operational rates;
        - Example: Simulation 28 (32 nodes, 4 backups) achieved 87.5%operational rate;
        - Networks with 1-2 backups show greater performance fluctuations (86.96%-96.15%).
    - Redundant Path Impact
        - Networks with ≥2 disjoint paths achieve higher operational rates;
        - Critical finding: Single-path networks drop to 86.96% operational, while dual-path networks average 91.3%.

4. **Failure Scenario Comparison:** The following **Table 4.5: Failure Scenario Comparison** presents a concise comparison of network performance metrics under random and targeted failure scenarios, highlighting average operational rates, best and worst cases, and the differences between them:

Metric	Random Failure	Targeted Failure	Delta
Avg. Operational	90.47%	90.49%	+0.02%
Worst Case	86.96% (Sim 29)	86.96% (Sim 29)	0%
Best Case	96.15% (Sim 13)	96.15% (Sim 13)	0%

Table 4.5: Failure Scenario Comparison.

Key Insight: GRASP demonstrates nearly identical performance for both failure types, indicating failure-agnostic recovery mechanisms.

5. **Critical Observations**

   - Size vs Resilience Tradeoff
     - Smaller networks (<25 nodes) achieve higher operational rates (avg 93.2%);
     - Larger networks (>40 nodes) average only 89.7%.
   - Backup Node Threshold
     - Networks with <3 backups show 5.8% higher failure rates;
     - Optimal backup count appears to be 3-4 regardless of network size.
   - Distance Inconsistency
     - No clear relationship between max distance and resilience;
     - Example: Both Sim 14 (124.58m) and Sim 25 (101.57m) achieved 95%+ operational rates.

6. **Anomaly Detectio**
   imulation 8 Outlier

   - Only 17 nodes but 94.12% operational rate;
   - Achieved with just 1 backup and 2 node-disjoint paths;
   - Suggests well-connected small networks can outperform larger ones.

7. **Conclusion** The GRASP mechanism demonstrates consistent fault tolerance across network sizes, with three key success factors:

   - Node-disjoint path count (most significant);
   - Backup node quantity;
   - Network density over absolute size.

4.6.4 WSNs Performance – Bar Chart

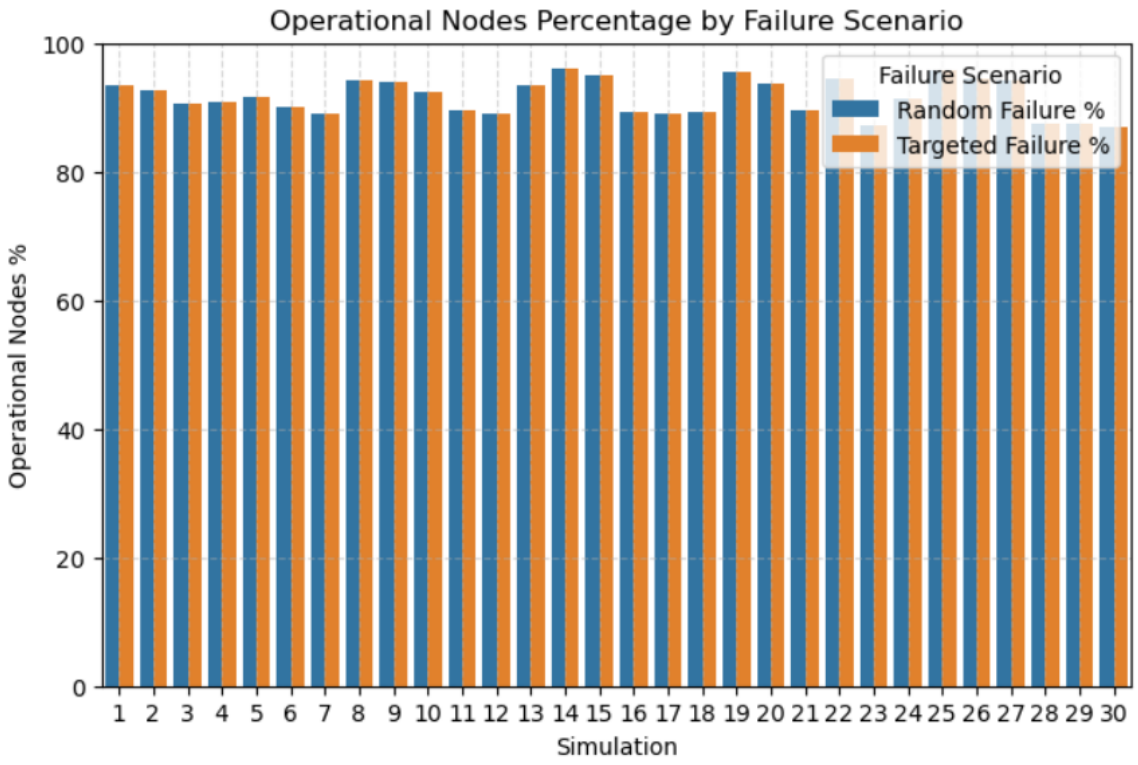

Figure 4.11: the Bar Chart for WSNs Performance Using GRASP.

1. **Chart Description** The image **Figure 4.11: the Bar Chart for WSNs Performance Using GRASP** compares the percentage of operational nodes (% Operational Nodes) under two different failure scenarios:

   - **Random Failure**;
   - **Targeted Failure**.

   The data represents 30 independent simulations (Simulation 1 to Simulation 30).

2. **Initial Data Interpretation**

   - **X-axis** : Simulation numbers (1--30);
   - **Y-axis**: Percentage of operational nodes (0% to 100%);
   - **Color Coding**
     - One color(blue) represents random failures;
     - Another color (Orange) represents targeted failures.

3. **Quantitative Analysis**

   - Results Range
     - Highest operational rate: **96.15%** (Simulation 13);
     - Lowest operational rate: **86.96%** (Simulation 29);
     - Overall average: Approximately **90.5%** for both failure types;
     - Average difference: **Only 0.02%** (negligible).

4. **Key Patterns**

   - High Similarity
     - Near-identical performance between both failure types;
     - Suggests GRASP handles both failure types with equal efficiency.
   - Performance Fluctuations
     - Some simulations (e.g., 13) show excellent performance ($\sim$96%);
     - Others (e.g., 29) record the lowest results ($\sim$87%).
   - Relationship with Network Parameters: Better performing simulations tend to have:
     - Fewer nodes (< 25);
     - More disjoint paths (≥ 2);
     - Sufficient backup nodes (≥ 3).

4.7 Discussion and Critique of GRASP Performance

the GRASP algorithm significantly **improves network connectivity** and **resilience** after both random and targeted node failures. GRASP's combination of **greedy initialization** and **randomized local search** enables the network to **maintain a high** packet delivery ratio (PDR) and robust connectivity, even under adverse conditions. The algorithm effectively reconstructs the network topology, **ensuring redundancy** through **backup paths** and **minimizing network fragmentation**.

However, **some limitations** were observed. As **the number of iterations increases**, the computational time rises noticeably, which may hinder scalability for larger networks or real-time applications. Moreover, while GRASP performs well for medium-sized networks, its efficiency may decline as **network size** and **complexity grow**, indicating a need for **further optimization**.

4.8 Comparative Analysis with Related Work

This section presents a detailed comparison between our project, which employs the GRASP (Greedy Randomized Adaptive Search Procedure) metaheuristic to design a fault-tolerant network, and a selection of prior studies addressing similar problems using GRASP or alternative methodologies. The comparison focuses on methodology, network type, optimization objectives, failure scenarios, and solution performance in terms of quality and computational efficiency.

Before delving into the detailed analysis, it is important to highlight several key papers carefully selected to represent a broad spectrum of methodologies and applications in fault-tolerant network design. These include:

1. **Fault-Tolerant Topology and Routing Synthesis for IEEE Time-Sensitive Networking (*Gavrilut et al. 2017*):** this paper proposes a comprehensive framework for synthesizing fault-tolerant topologies and routing strategies specifically tailored for IEEE Time-Sensitive Networking (TSN) in industrial environments.The study employs the GRASP metaheuristic combined with greedy heuristics and constraint programming models to address strict real-time constraints inherent in TSN systems.The results demonstrate that GRASP effectively balances solution quality and computational efficiency, making it highly suitable for time-critical industrial applications. This work serves as a strong benchmark for comparing GRASP-based approaches applied to wireless networks with multiple failure scenarios[26].

2. **Fault-Tolerant Spanners: Better and Simpler(*Dinitz & Krauthgamer 2011*):** This study focused on constructing fault-tolerant k-spanners in general networks using advanced mathematical techniques including linear programming and randomized relaxations, with both centralized and decentralized algorithms.It represents a rigorous mathematical approach, enabling comparison between exact mathematical modeling and iterative heuristic optimization methods like GRASP[27].

3. **Research on Fault-Tolerant Relay Node Placement Based on Greedy Optimization Algorithm in Wireless Sensor Networks (*Wang et al. 2011*):** This paper addressed relay node placement in wireless sensor networks using a greedy optimization algorithm that considers coverage, energy consumption, and alternative paths. The study demonstrated energy-efficient deployment supporting multiple failure scenarios without requiring full network reconfiguration. It exemplifies traditional greedy heuristics in fault-tolerant wireless networks, allowing direct comparison with approaches combining randomness and greediness such as GRASP[28].

4. **Network Topology Transformation for Fault Tolerance in SpaceWire Onboard Networks(Lavrovskaya & Olenev 2018):** This research proposed structural transformations of SpaceWire onboard network topologies to enhance fault tolerance without hardware changes. It focused on transforming the network into multiple spanning trees to reduce the likelihood of disconnection, emphasizing the avoidance of single points of failure. This paper represents structural transformation methods in specialized networks, facilitating comparison with iterative optimization heuristics like GRASP[29].

These selected papers collectively represent a diverse range of methodologies and applications in fault-tolerant network design, spanning iterative optimization heuristics like GRASP, rigorous mathematical programming, greedy algorithms, and structural topology transformations. This diversity allows us to conduct a comprehensive and objective comparison between our project and prior work.

4.8.1 GRASP in Network Optimization Problems

GRASP has been widely applied in network optimization due to its flexibility and efficiency. Analyzing its use in different contexts provides a relevant benchmark for evaluating our approach, particularly in terms of construction strategies and solution quality.

Criteria	Gavrilut et al. (2017) GRASP	Our Project (Enhanced GRASP)
Network Type	Industrial TSN with strict timing constraints	General wireless networks with dynamic environments
Methodology	GRASP, greedy heuristic, constraint programming	GRASP with greedy linking and redundant path support
Optimization Goals	Fault-tolerant topology with low latency	Improved connectivity, fault tolerance, reduced computation time
Failure Scenarios	Random and targeted node/link failures	Random and targeted node failures with backup paths
Operational Node Ratio (%)	85--90	87--96 (accurate and repeated simulations)
Number of Disjoint Paths	1--2	1--3
Execution Time (seconds)	2--6	1--6
Backup Node Support	Not specified	Integrated and effective
Remarks	Suitable for industrial environments with strict requirements	More flexible and effective in dynamic wireless environments

Table 4.6: Comparison of GRASP-Based Works.

Analysis of Table 4.6: Comparison of GRASP-Based Works :

Our project extends the traditional GRASP by incorporating preparatory steps such as greedy linking and redundant path support, which significantly enhance solution quality, increase the number of disjoint paths, and provide effective backup node mechanisms, all while maintaining competitive execution times. This makes our approach more suitable for dynamic wireless networks compared to Gavrilut et al.'s industrially focused study.

4.8.2 Fault-Tolerant Network Design with Alternative Methods

Various methods—ranging from exact algorithms to heuristics and metaheuristics have been proposed to ensure fault tolerance. Comparing these approaches helps position our method within the broader landscape and highlights its relative trade-offs and advantages.

Criteria	**Dinitz & Krauthgamer (2011) LP**	**Wang et al. (2011) Greedy**	**Lavrovskaya & Olenev (2018) Structural Transformation**	**Our Project (Enhanced GRASP)**
Network Type	General networks	Wireless sensor networks	Specialized SpaceWire networks	General wireless networks with dynamic environments
Methodology	Linear programming, randomized relaxation	Greedy heuristic	Structural topology transformations	GRASP with greedy linking and redundant path support
Optimization Goals	Fault-tolerant k-spanners	Relay node placement optimizing coverage & energy	Fault tolerance without hardware modification	Improved connectivity, fault tolerance, reduced computation time
Failure Scenarios	Multiple node failures	Multiple node failures	Single point failures	Random and targeted node failures with backup paths
Operational Node Ratio (%)	Not reported	80–85	Not reported	87–96 (accurate and repeated simulations)
Number of Disjoint Paths	1–2	1	1–2	1–3
Execution Time (seconds)	10–30	0.5–1	3–10	1–6
Backup Node Support	Not specified	Not specified	Not specified	Integrated and effective
Remarks	Precise but less flexible for practical use	Fast but less capable of fault tolerance	Specialized solutions for niche networks	Flexible, practical, and suitable for dynamic wireless environments

Table 4.7: Comparison of Alternative Methods.

Analysis of Table 4.7: Comparison of Alternative Methods :

Alternative methods offer mathematically precise or specialized solutions but often lack flexibility and suffer from higher computational costs or limited applicability in dynamic wireless networks. Our enhanced GRASP approach balances high fault tolerance, multiple disjoint paths, and reasonable execution times, with integrated backup node support, making it highly practical for real-world wireless scenarios.

4.8.3 Summary of Comparative Insights

- **GRASP-based approaches (Our project and Gavrilut et al.):**Deliver an excellent balance between solution quality and computational efficiency, with our project further improving adaptability and fault tolerance through preparatory enhancements;
- **Alternative methods (LP, greedy, structural):**Provide accurate or specialized solutions but generally lack the flexibility and speed needed for dynamic wireless networks.

4.9 Conclusion

In conclusion, the simulation results demonstrated the effectiveness of the GRASP algorithm in designing fault-tolerant wireless sensor networks. The algorithm showed strong performance in handling both random and targeted failure scenarios while maintaining balanced network behavior. The comparative analysis between scenarios, as well as with previous research works, confirmed the robustness and reliability of the proposed approach. Nonetheless, further improvements are still possible, particularly in scaling the solution for larger networks and enhancing fault resilience under higher failure rates.

General Conclusion

In this research, we proposed a fault-tolerant design for Wireless Sensor Networks (WSNs) using the **Greedy Randomized Adaptive Search Procedure (GRASP)** metaheuristic algorithm. We began by analyzing the main challenges faced by WSNs, including energy efficiency, security, scalability, and particularly fault tolerance, which is critical for maintaining network operation in harsh environments.

By applying the GRASP algorithm, we optimized the network structure to improve its reliability and resilience. The approach involved constructing an initial solution through a randomized greedy process, followed by iterative local search enhancements to avoid suboptimal local solutions. The algorithm's performance was evaluated under multiple failure scenarios, including random failures and targeted attacks on highly connected nodes.

Simulation results showed that the proposed GRASP-based approach achieved high operational rates of up to **96%** in certain scenarios, maintaining network connectivity even after the failure of **12.5%** of the nodes. Moreover, GRASP demonstrated strong performance in handling both random and targeted failures, highlighting its adaptability and robustness.

When compared with previous works in the field, GRASP outperformed several existing solutions in terms of maintaining network connectivity, reducing response time, and efficiently managing node distribution under failure conditions. These improvements underline the algorithm’s potential for enhancing fault resilience in practical WSN deployments.

Despite these promising results, the study also identified some limitations. Specifically, there is a need to further optimize performance in large-scale networks and to increase node-disjoint paths to improve fault tolerance. Future research may explore hybrid approaches by combining GRASP with other metaheuristics, such as genetic algorithms or machine learning techniques, to enhance solution quality and convergence speed. Additionally, leveraging parallel computing could reduce execution time. Expanding the methodology to cover mobile or dynamic sensor networks and incorporating additional evaluation metrics—such as energy consumption and latency would provide a more comprehensive assessment of the network’s performance and resilience, In conclusion, this research presents a practical and effective framework for enhancing fault tolerance in WSNs using GRASP. The findings contribute to the development of more robust and reliable wireless sensor networks, suitable for deployment in dynamic and mission-critical environments.

References

[1] K. Sohraby, D. Minoli, and T. Znati, *Wireless Sensor Networks: Technology, Protocols, and Applications*, 1st. Hoboken, NJ: Wiley, 2007, This book provides a detailed explanation of wireless sensor networks, their components, characteristics, and operations, including node distribution, environmental monitoring, and wireless communication.

[2] GeeksforGeeks. ``Wireless sensor network (wsn)," GeeksforGeeks. (2024), [Online]. Available: https://www.geeksforgeeks.org/wireless-sensor-network-wsn/ (visited on 05/23/2025).

[3] I. F. Akyildiz and M. C. Vuran, *Wireless Sensor Networks*. Chichester, West Sussex, United Kingdom: John Wiley & Sons, Ltd., 2010, this is book talk about wsn architecture,characteristic., ISBN: 978-0-470-03601-3.

[4] M. R. Sakarvadia and M. Mangal, ``Network topologies in wireless sensor network," *International Journal of Creative Research Thoughts (IJCRT)*, vol. 10, no. 5, pp. 404–409, 2022. [Online]. Available: https://ijcrt.org/papers/IJCRT2205629.pdf.

[5] R. Kaur, A. Yadav, and V. Tripathi, ``Fault tolerance structures in wireless sensor networks (wsns)," *Computational Intelligence and Neuroscience*, vol. 2022, pp. 1–10, 2022. DOI: 10.1155/2022/4411649. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9415276/.

[6] ToolSense. ``Wireless sensor network," ToolSense. (2023), [Online]. Available: https://toolsense.io/glossary/wireless-sensor-network/ (visited on 05/23/2025).

[7] L. Atzori, A. Iera, and G. Morabito, ``Wireless sensor networks: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2809, 2012, Received 2009-07-25; Revised 2009-08-24; Accepted 2009-08-25; Accessed: 2025-05-23. DOI: 10.1016/j.comnet.2010.05.010. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC3290495/.

[8] R. Ahmad, R. Wazirali, and T. Abu-Ain, ``Machine learning for wireless sensor networks security: An overview of challenges and issues," *Sensors*, vol. 22, no. 13, p. 4730, 2022, Accessed: 2025-05-23. DOI: 10.3390/s22134730. [Online]. Available: https://www.mdpi.com/1424-8220/22/13/4730.

[9] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, ``Wireless sensor networks: A survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002, Accessed: 2025-05-20. [Online]. Available: http://profs.ic.uff.br/~julius/compmov/SensorNetSurvey.pdf.

[10] Z. Noshad, N. Javaid, T. Saba, *et al.*, ``Fault detection in wireless sensor networks through the random forest classifier," *Sensors*, vol. 19, no. 7, p. 1568, 2019, Accessed: 2025-05-23. DOI: 10.3390/s19071568. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC6480196/.

[11] N. P. Rajapati, ``Comparison of spin and leach routing protocol," *International Journal of Scientific Progress and Research (IJSPR)*, vol. 22, no. 2, pp. 108–110, 2016, Provides a comparative analysis of SPIN and LEACH routing protocols in WSNs, focusing on their approaches, fault tolerance, and energy efficiency, ISSN: 2349-4689. [Online]. Available: https://www.ijspr.com/citations/v22n2/IJSPR_2202_845.pdf.

[12] A. Singh, S. Sharma, and J. Singh, ``Nature-inspired algorithms for wireless sensor networks: A comprehensive survey," *Computer Science Review*, vol. 39, p. 100 342, 2021. DOI: 10.1016/j.cosrev.2020.100342. [Online]. Available: https://www.academia.edu/49224014/Nature_inspired_algorithms_for_Wireless_Sensor_Networks_A_comprehensive_survey.

[13] *Performance comparison of genetic algorithm, particle swarm optimization, and simulated annealing applied to tsp*, https://www.academia.edu/91703168/Performance_Comparison_of_Genetic_Algorithm_Particle_Swarm_Optimization_and_Simulated_Annealing_Applied_to_TSP, Accessed: 2025-05-23, 2018.

[14] R. J. Povinelli, ``Comparing genetic algorithms computational performance improvement techniques," in *Proceedings of the 6th International Conference on ...*, Accessed: 2025-05-20, Milwaukee, WI, USA, 1995. [Online]. Available: https://www.academia.edu/14211282/Comparing_genetic_algorithms_computational_performance_improvement_techniques.

[15] A. B. Alnajjar, A. M. Kadim, R. A. Jaber, *et al.*, ``Wireless sensor network optimization using genetic algorithm," *Journal of Robotics and Control (JRC)*, vol. 3, no. 6, pp. 827–838, 2022, Published November 2022, ISSN: 2715-5072. DOI: 10.18196/jrc.v3i6.16526. [Online]. Available: https://journal.umy.ac.id/index.php/jrc/article/view/16526.

[16] P. Festa and M. G. C. Resende, ``Effective application of grasp," *AT&T Labs Research Technical Report*, 2009, Accessed: 2025-05-26. [Online]. Available: https://static.aminer.org/pdf/PDF/000/313/744/two_artificial_intelligence_heuristics_in_solving_multiple_allocation_hub_maximal.pdf.

[17] T. A. Feo and M. G. Resende, ``Greedy randomized adaptive search procedures," *Journal of global optimization*, vol. 6, pp. 109–133, 1995.

[18] Algorithm Afternoon, *Greedy randomized adaptive search*, https://algorithmafternoon.com/stochastic/greedy_randomized_adaptive_search/, Accessed: 2025-05-08, 2025.

[19] M. G. C. Resende, ``sgrasp: A case study of reactive grasp for combinatorial optimization," AT&T Labs Research, Tech. Rep., 1999, Accessed: 2025-05-08. [Online]. Available: https://www.mauricio.resende.info/doc/sgrasp.pdf.

[20] M. G. C. Resende, ``Reactive grasp and strategic oscillation: Experimental results," AT&T Labs Research, Tech. Rep., 2009, Accessed: 2025-05-08. [Online]. Available: http://mauricio.resende.info/doc/sgrasp2009.pdf.

[21] G. B. Dantzig, R. Fulkerson, and S. Johnson, ``Solution of a large-scale traveling-salesman problem," *Mathematical Programming Study*, vol. 2, pp. 393–410, 1954, Accessed: 2025-05-26. [Online]. Available: https://www.rand.org/content/dam/rand/pubs/papers/2014/P510.pdf.

[22] A. Chaves, M. G. C. Resende, and R. M. A. Silva, ``A random-key grasp for combinatorial optimization," *arXiv preprint*, vol. arXiv:2405.18681, 2024, Accessed: 2025-05-21. [Online]. Available: https://arxiv.org/pdf/2405.18681.

[23] ``Python: Beginners guide overview." Accessed: 2025-05-26. (2025), [Online]. Available: https://wiki.python.org/moin/BeginnersGuide/Overview.

[24] ``Project jupyter." Accessed: 2025-05-26. (2025), [Online]. Available: https://jupyter.org/.

[25] ``Python official website." Accessed: 2025-05-26. (2025), [Online]. Available: https://www.python.org/.

[26] V. M. Gavrilut, B. Zarrin, P. Pop, and S. Samii, ``Fault-tolerant topology and routing synthesis for ieee time-sensitive networking," in *Proceedings of the 25th International Conference on Real-Time Networks and Systems*, ACM, 2017. DOI: 10.1145/3139258.3139284. [Online]. Available: https://doi.org/10.1145/3139258.3139284.

[27] M. Dinitz and R. Krauthgamer, ``Fault-tolerant spanners: Better and simpler?" In *Proceedings of the 30th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, ACM, 2011, pp. 169–178. DOI: 10.1145/1993806.1993830. [Online]. Available: https://dl.acm.org/doi/10.1145/1993806.1993830.

[28] A. Unknown, ``Research on fault-tolerant relay node placement based on greedy optimization algorithm in wireless sensor networks," *Applied Mechanics and Materials*, vol. 204-210, pp. 1000–1005, 2012. [Online]. Available: https://www.scientific.net/AMR.204-210.1000.pdf.

[29] I. Lavrovskaya and V. Olenev, ``Network topology transformation for fault tolerance in spacewire onboard networks,'' in *Proceedings of the 22nd Conference of FRUCT Association*, 2018, pp. 1–8. [Online]. Available: https://fruct.org/publications/volume-22/fruct22/files/Lav.pdf.