People`s Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

University of Echahid Hamma Lakhdar - El Oued

Faculty of Technology

Department of Electrical Engineering

Dissertation

ACADEMIC MASTER

Domain: Science and Technology

Division: Telecommunications

Specialty: Telecommunication Systems

Presented by:

- HABITA Sara
- BOUTA Hanine

Entitled:

Enhancing Wireless Sensor Network Efficiency through Multiobjective Optimization

Dissertation Submitted in Partial Fulfillment of the Requirements for the Master Degree in ……...

Publicly defended in: 03/06/2024

Board of Examiners:

Dr. Narimane Hadjadji	**Chairman**
Dr. Abir Betka	**Supervisor**
Dr. Imad Eddine Tinedert	**Examiner**

Academic Year: 2023/2024


بسم الله الرحمن الرحيم

Thanks

First of all, we thank Almighty God for us to have given the necessary strength to carry out this work. At the end of this work‘

We would like to thank our promoter ***DR betka abir*** *for having taken us under wing for the realization of this work thanks to his precious advice and to his immeasurable help.*

We would also like to acknowledge the Chair of the Committee and Professor Discussant

Sara and Hanine

Dedication

We will begin by dedicating the fruit of these years of study to:

Our dear parents who never ceased to illuminate our path with their love, prayers, sacrifices and encouragement who made us who we are and gave us a wonderful model of work and perseverance I hope they find in this work all our gratitude and all our love May God protect them forever

To our dear brothers, may god protect them

We will not forget all our friends.

Hanine and Sara

Abstract

ABSTRACT:

This master's dissertation explores the application of a multi-objective metaheuristic algorithm called Multi-Objectives Jellyfish Search (MOJS) to enhance the performance and reliability of Wireless Sensor Networks (WSNs). WSNs represent a recent technology that enables the strategic deployment of a multitude of miniature, battery-powered sensors to monitor and gather data from diverse environmental settings. However, the implementation of WSNs poses significant challenges due to limited energy resources. The proposed approach, WSN-MOJS, aims to find the optimal implementation of WSN that maximizes coverage and minimizes energy consumption. The research provides an overview of optimization problems, including their classification and methodologies, as well as the principles of metaheuristics and the Jellyfish Search method and its multi-objective variant. The application of MOJS in WSN is then described, with various simulation results presented for different scenarios. The simulation results demonstrate that the proposed method is capable of maximizing area coverage with an average number of sensors while minimizing energy consumption in a reduced computation time. Overall, this research contributes to the understanding and enhancement of Wireless Sensor Networks through the innovative application of the MOJS metaheuristic algorithm.

KEYWORD: Wireless Sensor Networks; Optimization; Multi-objective; Metaheuristics; jellyfish search; Multi-objective jellyfish.

ملخص:

مذكرة الماستر هذه تستكشف تطبيق خوارزمية فوق متفوقة متعددة الأهداف تُعرف باسم البحث بقناديل البحر متعدد الأهداف (MOJS) لتعزيز أداء وموثوقية شبكات المستشعرات اللاسلكية (WSNs) تُمثل شبكات WSNs تكنولوجيا حديثة تُمكّن النشر الاستراتيجي لعدد كبير من أجهزة الاستشعار الصغيرة والتي تعمل بالبطارية لرصد وجمع البيانات من مجموعة متنوعة من البيئات. ومع ذلك، فإن تنفيذ WSNs يواجه تحديات كبيرة بسبب محدودية الموارد الطاقية. تهدف النهج المقترح، WSN-MOJS، إلى إيجاد التنفيذ الأمثل لـ WSN الذي يُعظّم التغطية ويُقلل من استهلاك الطاقة. توفر الأبحاث نظرة عامة على مشاكل الأمثلة، بما في ذلك تصنيفها ومنهجياتها، فضلاً عن مبادئ خوارزميات فوق متفوقة وطريقة البحث عن قناديل البحر وتنويعها متعدد الأهداف. ثم يُوصف تطبيق MOJS في WSN، مع تقديم نتائج المحاكاة المختلفة لسيناريوهات مختلفة. وتُظهر النتائج التجريبية أن الطريقة المقترحة قادرة على تعظيم تغطية المنطقة مع عدد متوسط من أجهزة الاستشعار مع تقليل استهلاك الطاقة في وقت حساب أقل. وبشكل عام، تساهم هذه الأبحاث في فهم وتحسين شبكات المستشعرات اللاسلكية من خلال التطبيق المبتكر لخوارزمية MOJS.

كلمات مفتاحية: شبكات الاستشعار اللاسلكية؛ تحسين؛ متعدد الأهداف؛ فوق متفوقة ؛ البحث بقناديل البحر؛ البحث بقناديل البحر متعدد الأهداف.

RÉSUMÉ :

Cette mémoire de master explore l'application d'un algorithme métaheuristique multi objectif appelé Multi-Objectives Jellyfish Search (MOJS) pour améliorer les performances et la fiabilité des réseaux de capteurs sans fil (WSN). Les WSN représentent une technologie récente qui permet le déploiement stratégique d'une multitude de capteurs miniatures alimentés par batterie pour surveiller et collecter des données dans divers environnements. Cependant, la mise en œuvre des WSN pose des défis importants en raison des ressources énergétiques limitées. L'approche proposée, WSN-MOJS, vise à trouver la mise en œuvre optimale des WSN qui maximise la couverture et minimise la consommation d'énergie. La recherche fournit un aperçu des problèmes d'optimisation, y compris leur classification et leurs méthodologies, ainsi que les principes des métaheuristiques et de la méthode Jellyfish Search et de sa variante multi-objectif. L'application de MOJS dans les WSN est ensuite décrite, avec divers résultats de simulation présentés pour différents scénarios. Les résultats expérimentaux démontrent que la méthode proposée est capable de maximiser la couverture de la zone avec un nombre moyen de capteurs tout en minimisant la consommation d'énergie dans un temps de calcul réduit. Dans l'ensemble, cette recherche contribue à la compréhension et à l'amélioration des réseaux de capteurs sans fil grâce à l'application innovante de l'algorithme métaheuristique MOJS.

MOTS-CLÉS : Réseaux de capteurs sans fil; Optimisation; Multi-objectifs; Métaheuristiques; JS ; multi-objectifs JS.

Table of contents

Table of contents

Figures list

Tables list

List of algorithms

Abbreviations list

WSN	Wireless Sensor Network
MEMS	Micro-Electro Mechanical System
SOSUS	Sound Surveillance System
DSN	Distributed Sensor Network
GPS	Global Positioning System
LPA	Local Positioning Algorithms
NLP	Non-Linear Programming
NL	Linear Programming
HAS	Harmony Search Algorithm
GAs	Genetic Algorithms
MOOP	Multi-Objective Optimization Problem
MOO	Multi-Objective Optimization
JS	Jellyfish Search
MOJS	Multi-Objective Jellyfish Search

General Introduction

General Introduction

Wireless Sensor Networks (WSNs) represent a progressive technology in modern communication systems, enabling the deployment of numerous small, battery-operated sensors that monitor and collect data from various environments. These networks have a wide array of applications, including environmental monitoring, health care, military operations, and smart homes. However, the implementation of WSNs poses significant challenges due to constraints such as limited energy resources, bandwidth, and processing power.[1]

To address these challenges, multi-objective optimization (MOO) techniques can be employed. MOO seeks to optimize multiple conflicting objectives simultaneously, which is crucial for enhancing the performance and long life of WSNs. The complexity of MOO problems in WSNs necessitates the use of advanced optimization methods.

Metaheuristics, a class of high-level problem-independent algorithmic frameworks, are particularly effective for tackling MOO problems. These techniques, which include algorithms such as Genetic Algorithms (GAs), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO), are designed to explore large solution spaces efficiently and effectively. [2]

Among the emerging metaheuristic algorithms, the jellyfish search (JS) algorithm has gained attention due to its simplicity and robustness. Inspired by the movement and behavior of jellyfish in nature, this algorithm mimics their foraging and adaptive mechanisms to find optimal solutions. The multi-objective JS (MOJS) technique extends the basic jellyfish algorithm to handle multiple objectives.[3]

This memory's dissertation focuses on the application of the multi-objective (MOJS) metaheuristic algorithm to enhance the overall performance and reliability of Wireless Sensor Networks (WSNs) by addressing multiple objectives simultaneously. The proposed method, called WSN-MOJS, aims to increase node coverage and minimize energy consumption. The research explores the capacity of metaheuristics to optimize WSN design, making WSN deployments more efficient, reliable, and adaptable for real-world applications.

This manuscript is divided into three chapters organized as follows:

- ✓ **The first chapter:** It introduces the concept of sensor networks and the architecture of sensors, which prompted us to delve into a detailed study of wireless sensor networks, with a focus on their features and applications

- ✓ **The second chapter:** This chapter offers a comprehensive review of optimization problems, encompassing their classification and diverse methodologies. It delves into metaheuristics, examining the underlying principles of multiple approaches and their practical applications. Furthermore, it introduces the Jellyfish Search metaheuristic method, along with its multi-objective variant
- ✓ **The third chapter:** In this chapter, the use of the MOJS technique to optimize WSN is explained, and the results obtained are discussed.

Finally, we present our conclusions and perspectives.

Chapter I:

Wireless Sensor Network (WSN)

I.1. Introduction

A Wireless Sensor Network (WSN) comprises sensors interconnected wirelessly, typically consisting of at least two nodes. WSNs serve to monitor physical or environmental conditions, such as temperature, pressure, motion, sound, vibration, or pollutants. These networks are composed of sensor nodes with sensing and radio transceivers, computational devices, and power units, communicating via radio signals. WSNs are constrained by limited processing speed, storage capacity, and communication bandwidth, often employing multi-hop communication. They find application in various fields, including industrial process monitoring, machine health monitoring, agriculture, and environmental monitoring. [1, 4, 5]

In this chapter, we will delve into the sensor network, the architecture of a sensor, and the wireless sensor network.

I.2. Sensor node

A wireless sensor is a small-sized object with very limited resources that is autonomous, with three basic functions: collecting, processing information from the surrounding environment (temperature, speed, pressure, etc.), and transmitting it to other devices via radio waves over a limited distance. [6]

Figure I.1 shows an example of a sensor: DHT11 humidity sensor

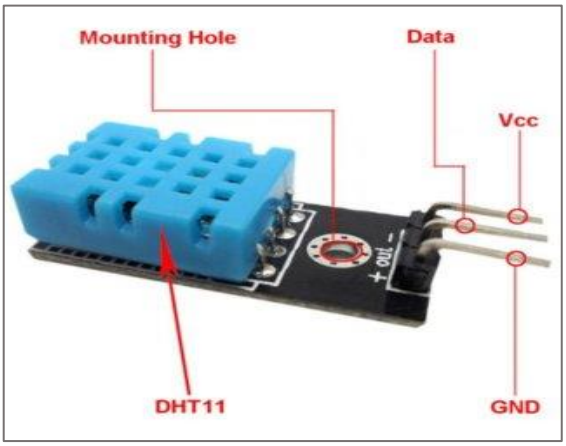


Figure I.1 : *Example of a sensor [7]*

The architecture of a sensor node consists of a collection of elements and components that combine to create a coherent and operational structure. It encompasses both software and hardware architecture.

I.2.1. Software architecture of a sensor node

The energy constraint of sensors requires the use of lightweight operating systems such as TinyOS or Contiki. However, TinyOS remains the most widely used and popular in the field of Wireless Sensor Networks (WSNs). It is free and is used by a large community of simulation scientists to develop and test network algorithms and protocols. [4]

Below we explain some of these systems.

a) Contiki

Contiki is an open-source operating system designed to serve as a resource manager for cheap sensor nodes. It utilizes Micro-Electro Mechanical System (MEMS) based sensor technology, comprising a microcontroller, transceiver, timer, memory, and analog-to-digital converter. Contiki, written in C language, is lightweight and portable, centered around an event-driven kernel. It offers preemptive multitasking at the individual process level. Configurations typically require 2 kilobytes of RAM and 40 kilobytes of ROM.[8]

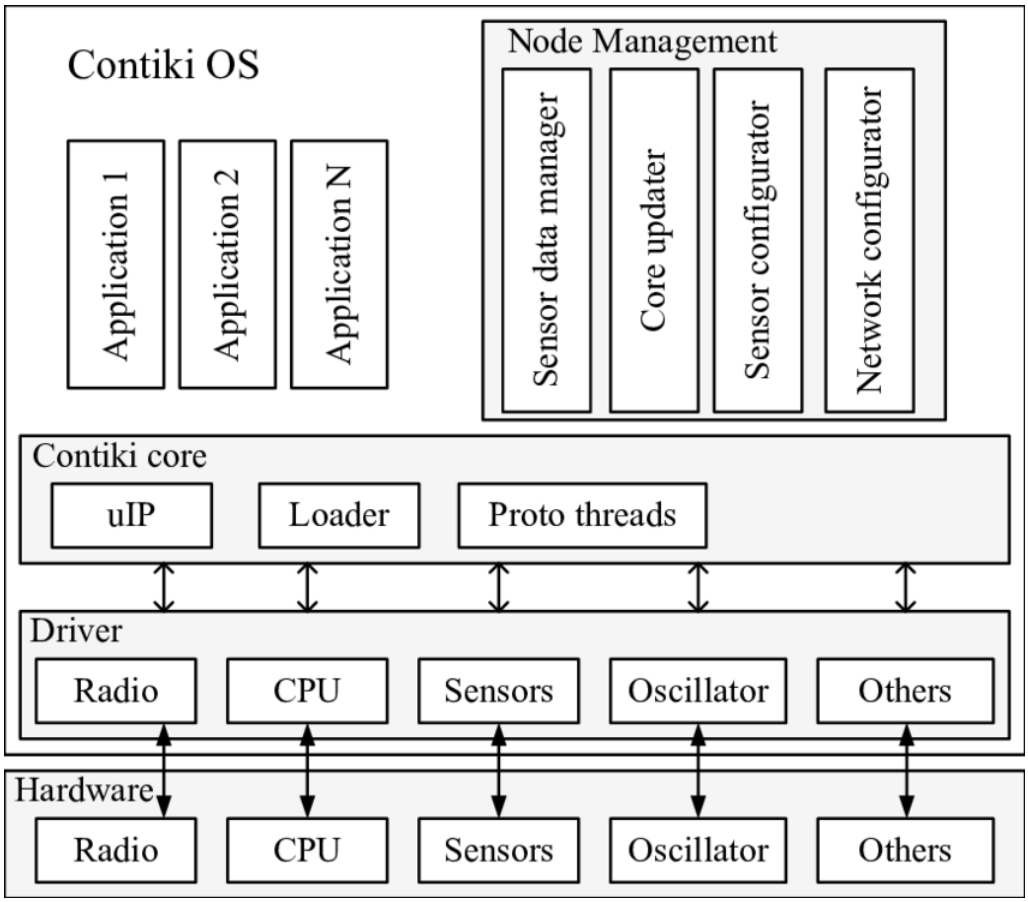


Figure I.2 : *Structure of Contiki [9]*

b) TinyOS

TinyOS employs a component-based programming model, formalized by the NesC language, which is a variant of C. Unlike conventional operating systems, TinyOS functions as a programming framework tailored for embedded systems. It provides a set of components facilitating the creation of application-specific operating systems within each application. On average, a typical TinyOS application spans approximately 15K in size, with the core OS constituting around 400 bytes. The most extensive application, resembling a database query system, extends to about 64K bytes. [10]

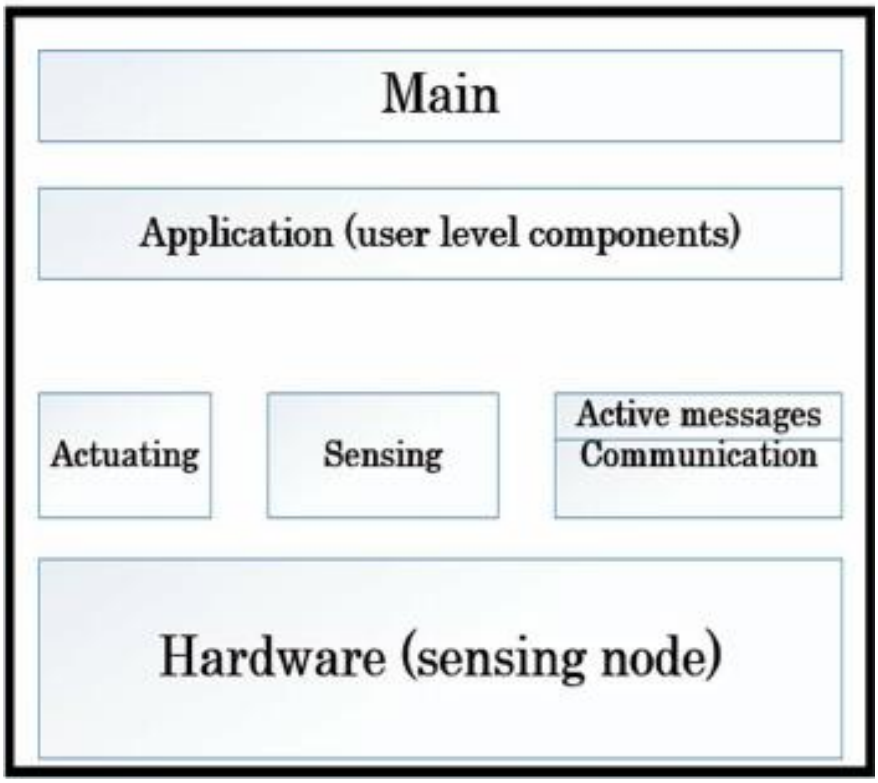


Figure I.3 : *Structure of tinyOS[11]*

I.2.2. Hardware Architecture of a sensor node

A sensor node comprises four fundamental components:

a) Sensor Unit: The sensor typically consists of two subunits, the receptor, and the transducer. It generates analog signals based on observed phenomena, which are then converted to digital signals using an A/D converter.

b) Processing Unit: This unit includes a processor (microcontroller) and a small storage unit. Its responsibility lies in implementing communication protocols, enabling the node to collaborate with other nodes in the network.

c) Transceiver Unit: Connecting the node to the entire network, this unit facilitates the sending and receiving of messages between nodes.

d) Energy Unit: Serving as the energy source for the sensor node, it can be associated with or powered by a power generation unit such as solar cells. [12]

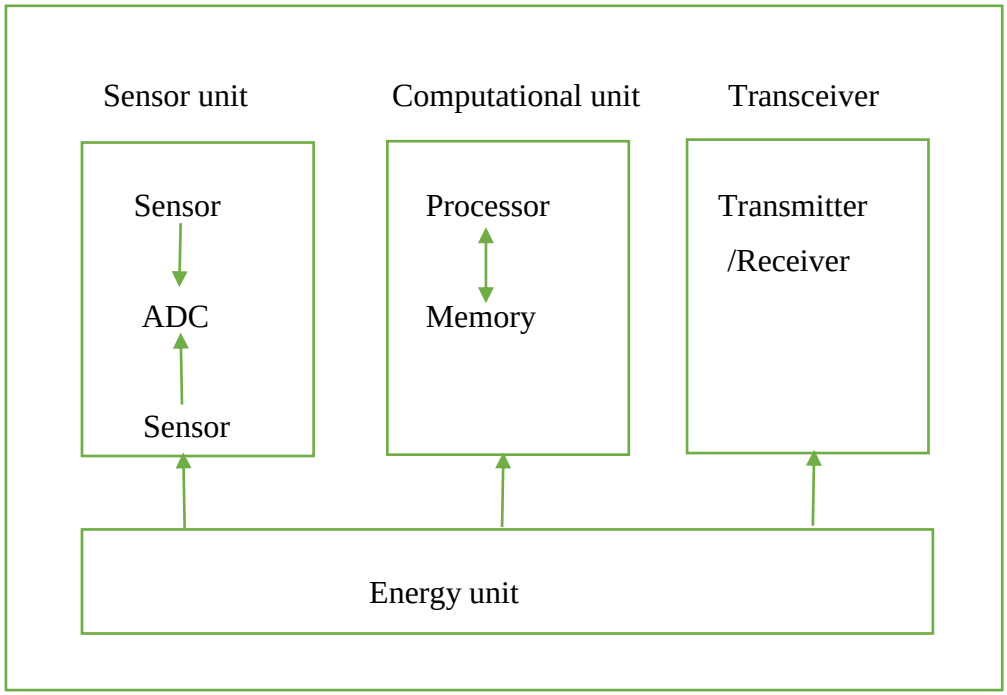


Figure I.4 : *Sensor node architecture*

I.3. Wireless Sensor Networks

Sensor networks reside at the lowest level of network functionality. Typically positioned at the base of the industrial automation hierarchy, sensors are intentionally designed to be cost-effective due to the frequent need for numerous units. These sensors play a crucial role in supplying fundamental data to the control system, including information about object position or physical properties like temperature. [13]

I.3.1. Historical overview of research on Wireless Sensor Networks

Wireless Sensor Networks (WSNs) have evolved significantly, starting from military and heavy industrial contexts to widespread applications in light industrial and consumer domains today. The precursor to modern WSNs, the Sound Surveillance System (SOSUS), emerged in the 1950s for tracking Soviet submarines and now serves peaceful purposes like monitoring marine life. The 1980 Distributed Sensor Network (DSN) program by DARPA, in collaboration with institutions like Carnegie Mellon University and MIT Lincoln Labs, paved the way for WSN integration into academia and civilian research. Governments and

universities have since employed WSNs in air quality monitoring, forest fire detection, and weather stations. Despite robust demand, challenges such as high costs and proprietary protocols hindered widespread adoption beyond niche applications. [14]

I.3.2 Definition

A Wireless Sensor Network (WSN) is a network comprised of numerous sensor nodes capable of autonomously collecting and transmitting environmental data. The position of these nodes is not necessarily predetermined; they can be randomly deployed in a geographical area known as a "capture field," corresponding to the area of interest for the observed phenomenon. [15]

The WSN offer distinct advantages compared to traditional standalone sensors and controllers. Firstly, they provide site specificity by allowing sensors to be strategically positioned in close proximity to production fields. Secondly, WSN allows for target specificity as network nodes can be tailored to monitor specific variables of interest, resulting in a reduction in the deployed sensor count and overall network cost. Additionally, WSN enables high spatial resolution by employing multiple nodes to increase the density of sensors and controllers per unit area. [16]

Sensor nodes operate in either a continuous or event-driven mode. Location and positioning information can be acquired using either the Global Positioning System (GPS) or local positioning algorithms (LPA). [5]

I.3.3 Architecture of wireless sensor networks

A wireless sensor network comprises numerous nodes organized into sensor fields. Each sensor is equipped with an acquisition module for measuring environmental data such as temperature, humidity, pressure, acceleration, sound, images, and videos. The collected data from these sensor nodes is routed to one or more base stations or sink nodes, which serve as data collection points. The sink node can communicate the gathered data to the end user through a communication network, potentially utilizing the Internet or a satellite. The end user can, in turn, use the base station as a gateway to transmit requests into the network. [17]

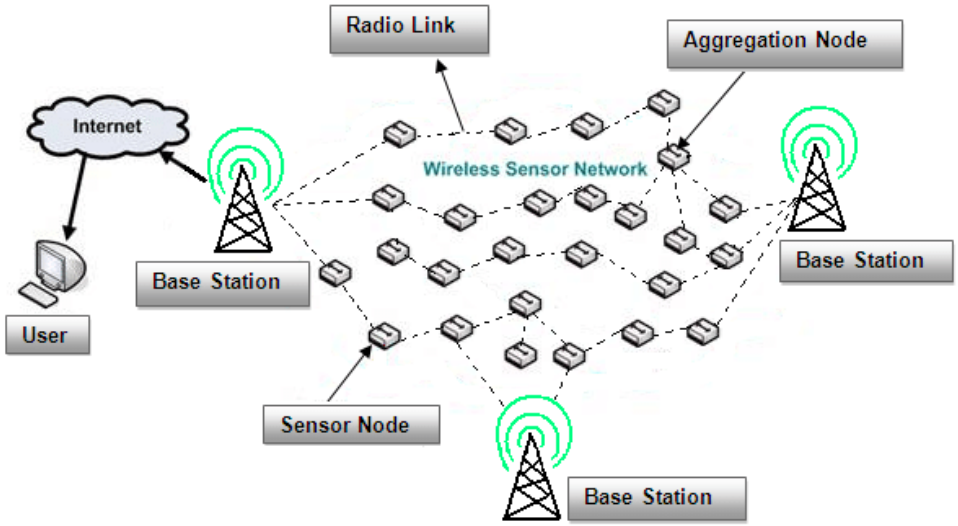


Figure I.5 : *Architecture of wireless sensor networks [18]*

I.3.4 Protocol architecture

The main purpose of this stack is to standardize communication among participants, allowing various manufacturers to develop compatible products in both software and hardware domains. Comprising 5 layers that mirror the functions of the OSI model, it also includes 3 layers specifically focused on power management, mobility management, and task management. The layered system aims to break down the problem into distinct parts (the layers) based on their abstraction level. Each layer in the model communicates with its adjacent layer (either above or below), enabling the utilization of services from lower layers while providing services to the layer above. [19]

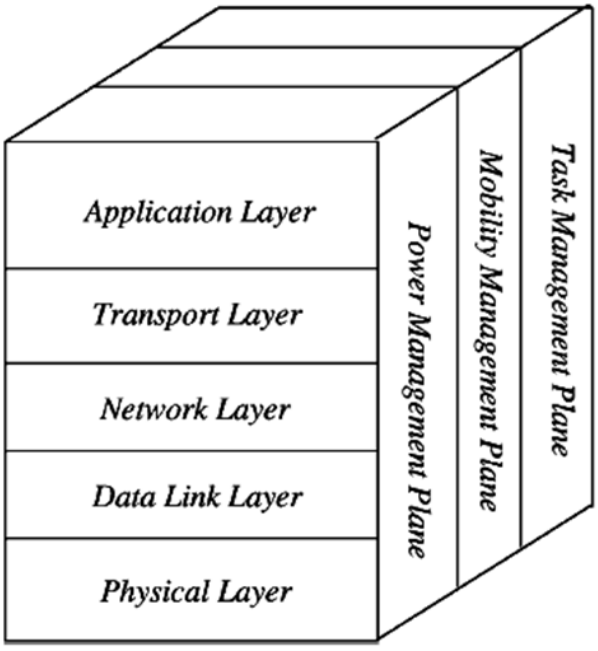


Figure I.6 : *Protocol stack for wireless sensor networks[20]*

a) The protocol layers

It comprises five layers:

- ✓ **Physical Layer:** Enables the conversion and routing of data in a specific physical medium while selecting the appropriate frequencies.
- ✓ **Data Link Layer:** Specifies how data is sent between two nodes/routers within a single hop. It is responsible for data multiplexing, error control, access to communication media, and ensuring point-to-point and multipoint connections in a communication network.
- ✓ **Network Layer:** This layer allows for finding a reliable route for data transmission from sensor nodes to the sink, optimizing the use of sensor energy.
- ✓ **Transport Layer:** This layer is responsible for data transport, packet segmentation, flow control, maintaining packet order, and managing any transmission errors.
- ✓ **Application Layer:** Ensures the interface with applications and is thus the closest layer to users, directly managed by software. [21]

b) Management Plans

It contains three plans

- ✓ **Energy Management Plan:** Integrated functions at this level involve managing energy consumption by sensors. For instance, a sensor can deactivate its receiving interface upon receiving a message from a neighboring node to avoid duplicate message reception. Additionally, when a node has low energy levels, it can broadcast a message to other sensors, opting out of routing tasks and preserving remaining energy for sensing functions.
- ✓ **Mobility Management Plan:** This level detects and records the movement of sensor nodes. Thus, a continuous link to the user is maintained, and the node can keep track of its neighboring nodes.
- ✓ **Task Management Plan:** Balances and schedules various data collection tasks in a specific region. It's not necessary for all nodes in this region to perform data collection tasks simultaneously; it depends on the sensor's nature and energy level. Therefore, the task management level ensures the balancing and distribution of tasks across different network nodes to ensure cooperative and efficient energy consumption, thereby extending the network's lifespan. [22]

I.3.5. Features of wireless sensor network

WSNs possess several features, including resource organization, energy management, and dynamic behavior.

Resource Constraints: Sensors are limited in resources such as memory, processing power, and bandwidth, with a primary focus on conserving energy.

Self-Organization and Scalable Topology: Due to the high node density in critical areas, sensor networks necessitate self-organization and adaptability to topology changes without human intervention to maintain normal operation.

Energy Constraints: Sensors face critical power limitations, especially in remote locations where battery replacement is challenging.

Dense Networks: Deployed sensors are numerous across the monitored area, presenting challenges such as interference and data packet damage due to high deployment density.

Limited Coverage: Each sensor provides a partial view, and the overall coverage results from combining these partial views from multiple nodes.

Dynamic Node Identification: Sensor nodes are identified based on captured attributes rather than fixed IP addresses, unlike traditional wireless networks.

Mobility Challenges: Mobility, particularly in environments with frequent movement, can lead to significant energy consumption for control messages related to topology structure and neighbor detection. [23]

I.3.6. Design factors

Design factors of sensor networks include the following:

a) Network Topology

Deploying a large number of nodes requires topology maintenance, which occurs in three phases:

- ✓ Deployment.
- ✓ Post-deployment (sensors may move, cease functioning, etc.).
- ✓ Redeployment of additional nodes.

b) Transmission Media

To be easily installed in targeted areas without incurring significant wiring costs, sensors utilize radio frequency links to cooperate within the network. [24]

c) Physical Constraints

The primary material constraint is the size of the sensor. Other constraints include energy consumption, which needs to be minimized for the network to last as long as possible. It should also adapt to diverse environments (high temperatures, water, etc.) and be autonomous and durable, given its frequent deployment in harsh conditions.

d) Network Lifetime

It is the time interval between the deployment of the network and the exhaustion of energy from the first node. The required network lifetime varies depending on the application, ranging from a few hours to several years. [25]

I.4. Classification of Wireless Sensor Network

Based on various specific criteria, including the method of data acquisition and delivery to the well, the distance between sensor nodes and the well, the deployment model in the network, and the capabilities of network nodes, WSNs can be categorized into multiple classes.

I.4.1. In accordance with the mode of data acquisition and delivery to the well:

The WSN adopts different models based on the application and its specific requirements. There are four communication modes for sensor networks:

a) Continuous Mode (Time-Driven): Also known as periodic mode, sensors must periodically awaken their transmitters to send captured data to the well. This mode is suitable for applications requiring regular monitoring of the designated area, as seen in agriculture where sensors are deployed in fields to monitor soil conditions and optimize resource inputs.

b) Event-Driven Mode: Sensors transmit their measurements to the well when a specific event occurs. This model is ideal for critical event monitoring applications, prioritizing the rapid acquisition of information about the event. For instance, in waste management and glass container applications.

c) Query-Driven Mode: In this on-demand model, sensors measure phenomena and store the data in their memory, transmitting it only upon receiving requests from the base station. A classic example is the remote reading of water meters.

d) Hybrid Model: This model combines elements from the three previous modes, offering a versatile approach to data acquisition and delivery in WSN.

I.4.2. According to the distance between sensor nodes and the well:

This classification distinguishes three communication modes: single hop, multi-hop, and hybrid.

a) In a single-hop WSN, sensor nodes send their captured data directly to the well without passing through any other intermediate nodes.

b) However, in a multi-hop WSN, the distance between some sensor nodes and the base station exceeds their maximum range. Consequently, each node serves as both the data initiator and router. In multi-hop communication, each sensor node transmits its information to the neighboring node, which then relays the information to its neighboring node until it reaches the base station.

c) In hybrid WSN, sensor nodes transmit their captured data using a combination of the two aforementioned methods.

I.4.3. According to the mobility model in the network:

This classification involves a combination of the mobility of sensor nodes and that of the base station. Through this combination, we can distinguish two major categories of networks: static networks and dynamic or mobile networks.

a) A static sensor network considers no mobility of nodes, neither of the base station. This requires an initial setup of communication infrastructures to create the path between the base station and the sensors.

b) In a mobile sensor network, nodes or the base station are mobile. How can a node be mobile? It can be attached to a robot, a human, a vehicle, etc. The goal of mobile sensor networks is to gather more information about an environment using fewer sensor nodes and allow the network to self-organize its nodes. Additionally, it becomes capable of moving its sensors dynamically based on environmental changes, making it adaptable to its evolving surroundings.

I.4.4. According to the capabilities of network nodes:

In this category, two types of networks are distinguished: homogeneous and heterogeneous networks.

a) In a homogeneous sensor network, all nodes have the same capabilities in terms of energy, computation, and storage.

b) While in a heterogeneous sensor network, there are some sophisticated nodes that have higher processing, energy, and communication capabilities than normal nodes. [26]

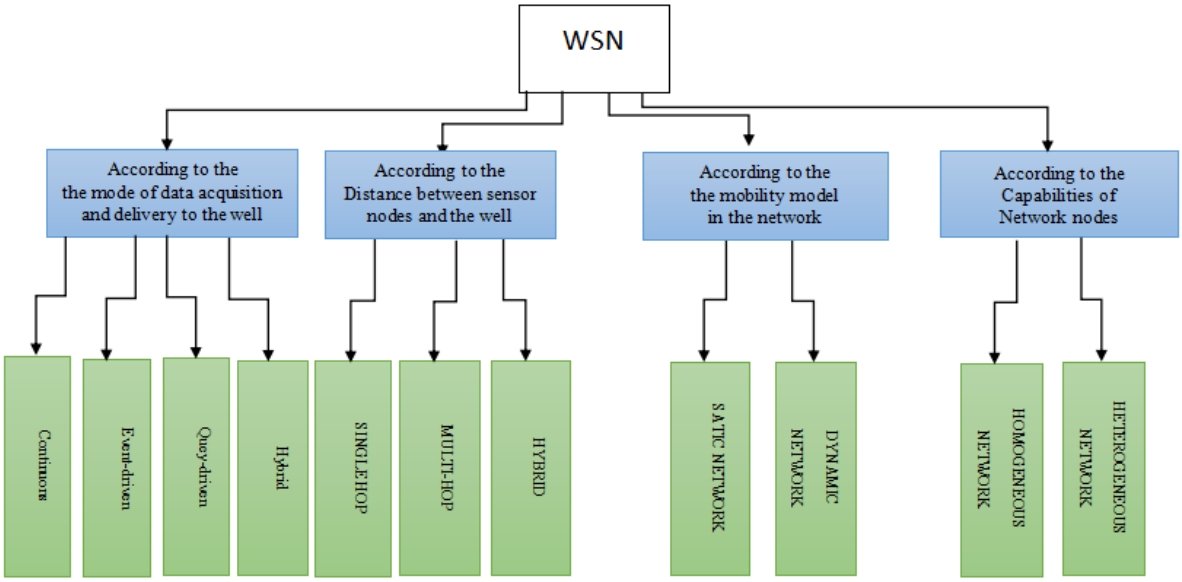


Figure I.7 : *Classification of Wireless Sensor Network.*

I.5. Topology of a Wireless Sensor Network

The data routing in Wireless Sensor Networks (WSNs) follows various routing schemes that are inherent to different topologies. Multiple topologies exist for sensor networks:

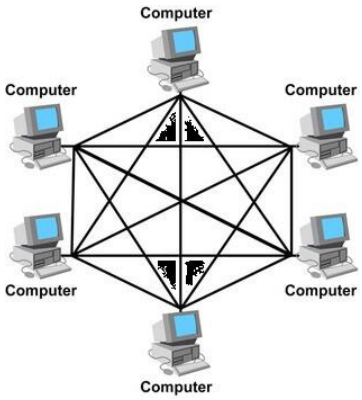


Figure I.8 : *Mesh Network Topology [27]*

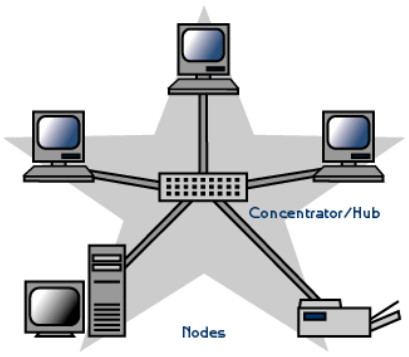


Figure I.9 : *Star Network Topology [28]*

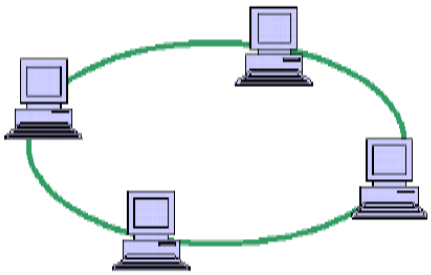


Figure I.10 : *Ring topology.*

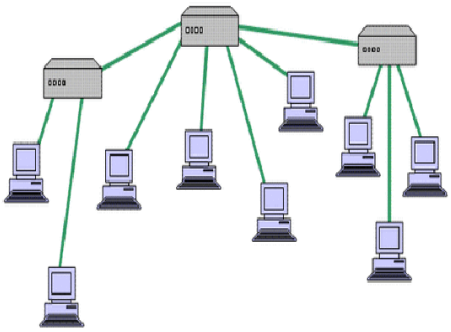


Figure I.11 : *Tree topology .[29]*

I.5.1. Mesh Network Topology: A multi-hop system facilitating communication between sensor nodes and the gateway, either directly or through a multi-hop routing scheme. This prevalent scenario involves nodes not only transmitting their own sensed data but also acting as relays for other sensors. In this topology, routing emerges as one of the most significant challenges in WSNs. [30]

I.5.2. Star Network Topology: It's a single-hop system where sensor nodes communicate directly with the collection point. This topology is the simplest and requires the distance between nodes and the gateway to be limited. [30]

I.5.3. Ring topology: In this topology, each node in the network connects to precisely two other nodes, with the first and last nodes linked together. Data travels through each node in the ring until it reaches its destination. In this setup, data and signals move in a single direction, ensuring orderly transmission. The dual ring topology expands on this by establishing two connections between each node, enabling data flow in opposite directions. Unlike centralized setups, ring topologies don't rely on a central server to manage connections, promoting efficient network operation.

I.5.4. Tree topology: Also referred to as hierarchical topology, the tree topology features a central root node connected to one or more nodes of a lower hierarchy. Among various network topologies, the tree topology combines aspects of both the bus and star topologies. Its tree-like structure enables the integration of multiple servers and facilitates network expansion in diverse directions. This setup proves particularly beneficial for institutions like colleges, universities, and schools, allowing each branch to identify its relevant systems while remaining connected to the larger network. Tree structures are most suitable for extensively spread networks with numerous branches. However, like other topologies, tree topology has its advantages and disadvantages. It may not be ideal for small networks due to potential cable waste, and its configuration should be tailored to address specific limitations.

I.6. Application Domains

Ideally, sensor networks are supposed to be built using low-cost nodes deployed in large quantities. Currently, they are not widely used in our daily lives as the manufacturing cost of nodes remains very high, and some techniques are far from being mature and reliable. However, there are already numerous applications, some of which are still experimental, in various fields.

Military Applications: The anticipated characteristics of sensor networks, such as low cost, self-organization, security, and fault tolerance, have led to their utilization in military domains. This is also the field where one often encounters the most advanced technologies. [31]

Environmental Applications: Thermo-sensors dispersed from an aircraft over a forest can detect potential fires, improving firefighting efficiency. In agriculture, sensors integrated with seeds can identify dry areas, enhancing irrigation. In industrial settings, sensors can detect leaks of toxic substances, alerting for prompt intervention. Sensor deployment in forests or wildlife areas can gather data on surroundings and animal behaviors, aiding conservation efforts. For instance, the University of Pisa developed sensor networks to monitor natural parks, facilitating wildlife observation without disturbance and proposing effective conservation solutions. [32]

Medical Applications: Implanting miniature video sensors beneath the skin enables the capture of real-time images from a targeted body area, all without the need for surgery. This facilitates the continuous monitoring of disease progression or muscle reconstruction.

Agricultural Applications: Embedding nodes in the soil enables the network to assess the field's condition, pinpointing the driest areas for prioritized irrigation. Furthermore, fitting livestock herds with sensors ensures constant location tracking, eliminating the necessity for shepherd dogs. [33]

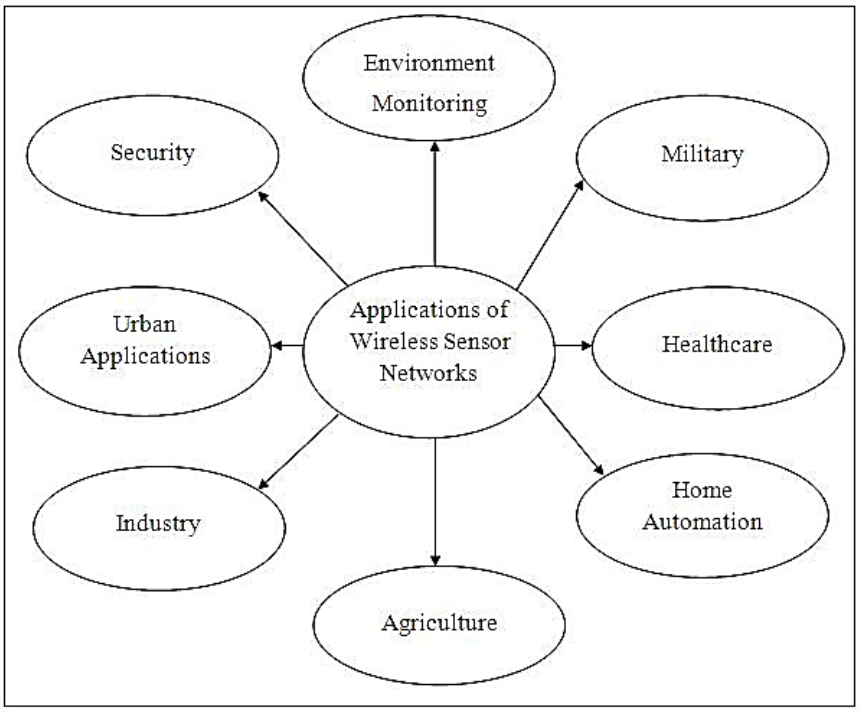


Figure I.12 : *Applications of WSN[34]*

I.7. Conclusion

In this chapter, we studied the wireless sensor device, delving into its definition, characteristics, applications, and more. Through this, we inferred that the sensor device allows for measurement and monitoring in a manner much closer to the phenomenon than ever before, leading to continuous and high-precision data collection.

In the next chapters, we will explore how to enhance wireless sensor network performance in terms of coverage and energy consumption using multi-objective optimization techniques.

Chapter II:

Metaheuristics and Multi-Objective Jellyfish Search method

II.1. Introduction

Metaheuristics are approximate optimization methods designed to solve difficult optimization problems when exact methods fail or require an unacceptable amount of computational time. Generally, all metaheuristic techniques are based on random mechanisms to explore the search space in order to determine or approximate a global optimum. Metaheuristics have been used in various domains such as image processing, electronics, mechanics, encryption, etc. [35]

This chapter provides an overview of optimization problems, including their classification and various techniques. It delves into metaheuristics, exploring the principles behind several methods and their applications. Additionally, it introduces the Jellyfish Search metaheuristic method, along with its multi-objective version.

II.2. Optimization

Optimization is the process of achieving optimal operation within a system, whether it be a machine, a company, or any other entity. This involves creating the most favorable conditions or making the best possible use of resources to maximize performance. Optimization is a fundamental concept within Operations Research, which is the science of decision-making.

In mathematics, an optimization problem involves determining the best solution for a given problem, aiming to optimize specific criteria or objectives.

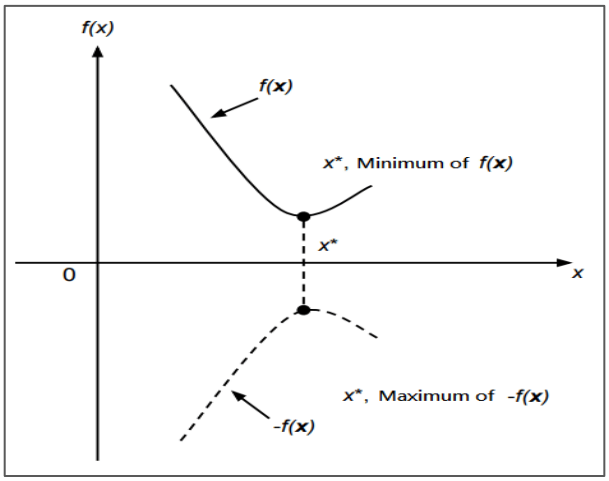
Definition 1. Optimization problem

Let:

- ✓ $n \in N^*$: be a strictly positive integer,
- ✓ $X \subset R_n$: a non-empty subset
- ✓ $f: X \rightarrow R$: a real-valued function.

An optimization problem consists of finding the best solution $x \in X$ called the global solution, that maximizes or minimizes the function f over the set X. [36] We denote:

$$x = \min_{x \in X} f(x) \text{ or } x = \max_{x \in X} f(x) \qquad \text{(II.1)}$$



***FigureII.1**: Minimum of f(x) is same as maximum of - f(x) [37]*

II.2.1. Classification of Optimization Problems

Optimization problems can be categorized in various manners, as outlined below.

a) Classification Based on the Deterministic Nature of the Variables: Optimization problems can be categorized into deterministic and stochastic programming problems based on the deterministic nature of the variables involved.

b) Classification Based on the Permissible Values of the Design Variables: Optimization problems are classified as integer and real-valued programming problems depending on the values permitted for the design variables

c) Classification Based on the Number of Objective Functions: Optimization problems can be categorized as single- and multi-objective programming problems based on the number of objective functions to be minimized.

d) Classification Based on the Existence of Constraints: Optimization problems are classified as constrained or unconstrained depending on whether constraints exist in the problem.

e) Classification Based on the Nature of the Equations Involved: This classification groups optimization problems into four categories: non-linear linear, geometric, and quadratic.

- *Non-linear*: If any function among the objective functions or constraints is non-linear, the problem is called a Non-linear Programming (NLP) problem.
- *Linear*: If the objective function and all constraints are linear functions, the problem is called a Linear Programming (LP) problem.

- *Geometric*: In the Geometric Programming (GMP) problem, the objective function and constraints are expressed as posynomial in X. [38]
- *Quadratic*: If the objective function is quadratic and the constraints are linear, the problem is referred to as quadratic. Typically, it is formulated as follows:

$$f(x) = c + \sum_{i=1}^{n} q_i x_i + \sum_{i=1}^{n} \sum_{j=1}^{n} Q_{i,j}, x_i x_j \quad \text{(II.2)}$$

Under the constraints:

$$\sum_{i=1}^{n} a_{i,j}.x_i = b_j$$

Where c, q_i, $Q_{i,j}$, b_j and $a_{i,j}$are constants. [35]

II.2.2. The optimization techniques

Combinatorial optimization methods can be classified into two main families: exact methods and approximate methods. Figure II.2 illustrates the taxonomy of optimization problem-solving methods. [36]

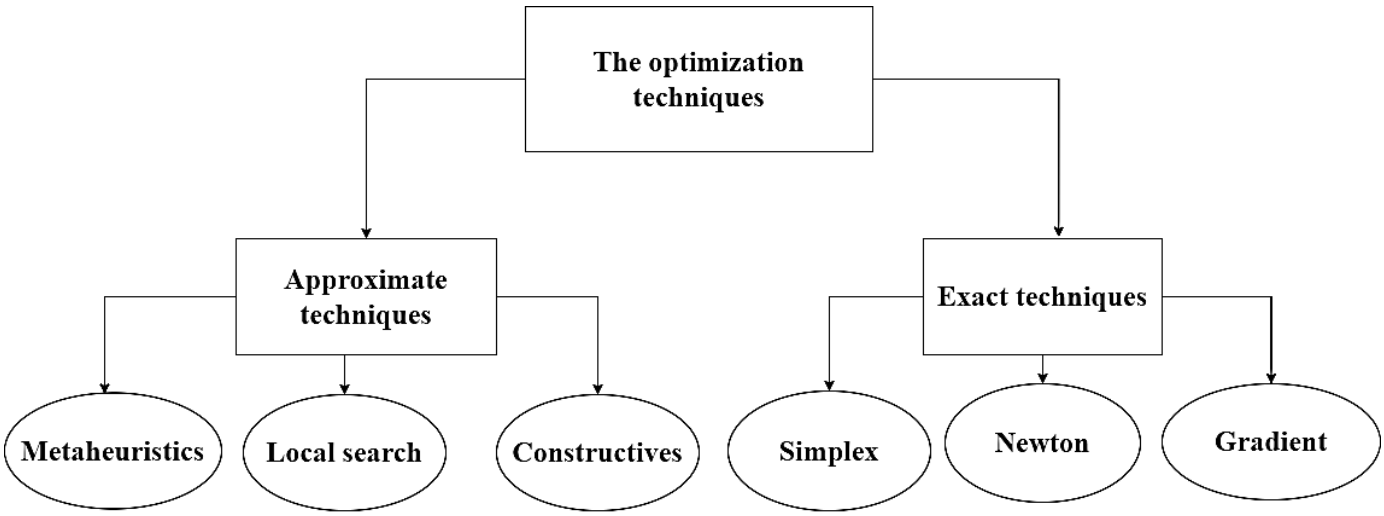


Figure II.2: *Classification of optimization techniques*

a) Exact Methods: Exact methods produce an optimal solution for a given instance of an optimization problem. They typically rely on tree search and partial enumeration of the solution space. These methods are guaranteed to find an optimal solution.

The most well-known exact algorithms include the branch and bound method, dynamic programming, and linear programming. The major drawback of these methods is combinatorial explosion: the number of combinations increases with the dimension of the

problem. The efficiency of these algorithms is promising only for small-sized problem instances.

b) **Approximate Methods:** In contrast to exact methods, approximate methods do not guarantee an optimal solution but rather aim to provide a good solution of reasonable quality within a reasonable amount of time. [36] They often use heuristics or simplifications to guide the search process. While not guaranteeing optimal solutions, they provide valuable alternatives in complex or computationally intensive problems.

II-3 Metaheuristics

The term "metaheuristic" was introduced by Glover in 1986 to describe a set of methodologies positioned conceptually above heuristics, guiding their design. A metaheuristic is a higher-level procedure aimed at discovering, generating, or selecting a lower-level procedure (a partial search algorithm) that could yield a sufficiently good solution to an optimization problem. By exploring a large set of feasible solutions, metaheuristics often uncover good solutions with less computational effort compared to calculus-based methods or simple heuristics.[2]

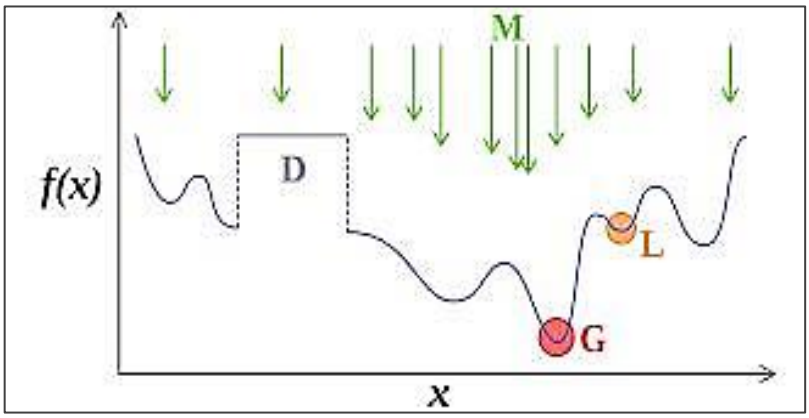


Figure II.3: *The general principle of metaheuristics[39]*

Metaheuristics (M) are often algorithms that utilize probabilistic sampling. They aim to find the global optimum (G) of a challenging optimization problem (with discontinuities — D —, for example), without being trapped by local optima (L).

II.3.1. Classification of Metaheuristics

There are multiple approaches to defining a taxonomy for metaheuristics. Depending on the characteristics chosen for differentiation, various classifications are feasible, each representing a distinct viewpoint. Here, we provide a brief overview of the primary methods used to classify metaheuristics. [40]

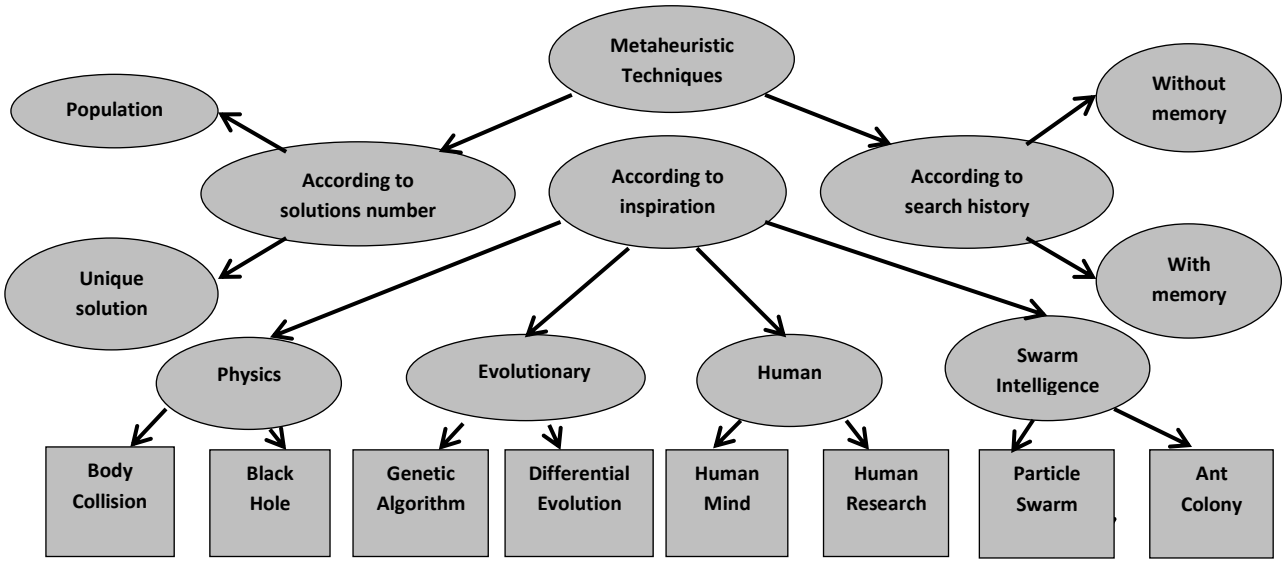


Figure II.4 : *Classification of metaheuristics*

II.3.2. Some Metaheuristic Techniques

There are several methods, among which we mention the following:

a) Simulated annealing

Simulated annealing is an optimization technique (metaheuristic) inspired by a process used in metallurgy. It involves alternating cycles of slow cooling and reheating (annealing) to minimize the material's energy. This method is adapted to optimization to find extrema (minimum or maximum) of a function. It was developed by three researchers from IBM, S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi in 1983, and independently by V. Černy in 1985.

Simulated annealing algorithm relies on the Metropolis acceptance criterion, a key concept within the Metropolis-Hastings algorithm. This criterion describes the probability of accepting a new state in the search for an optimal solution. By analogy with the physical process, the objective function to be optimized is equated to the material's energy (E). A fictitious parameter, the system's temperature (T), is also introduced.

This method can be summarized by Algorithm II.1. [41]

Simulated annealing iteratively applies the Metropolis algorithm to generate a sequence of configurations that tend towards thermodynamic equilibrium:

Algorithm II.1: Simulated annealing algorithm
1- Choose a starting temperature T=T0 and an initial solution $S = S0$
2- Generate a random solution in the neighborhood of the current solution
3- Compare the two solutions according to the Metropolis criterion
4- Repeat steps 2 and 3 until statistical equilibrium is reached
5- Decrease the temperature and repeat until the system is frozen. [42]

b) The Tabu Search

Tabu search is an optimization metaheuristic introduced by Fred W. Glover in 1986. It is an iterative metaheuristic categorized as a form of generalized local search. The core idea is to explore the neighborhood of a given solution and select the best candidate that minimizes the objective function. This exploration may lead to an increase in the objective function value, which occurs when all neighboring solutions are worse. This mechanism helps escape local optima, but there is a risk of falling back into a previously escaped local minimum. To prevent this, the algorithm uses memory by maintaining a tabu list, which stores recently visited solutions and prohibits revisiting them. [41]

This method is generally superior in terms of execution time. It can be summarized by Algorithm II.2.

Algorithm II.2: The Tabu Search algorithm
1- Choose an initial solution i from S (the set of solutions). Apply $i^* = i$ k = 0 and T = 0 .
2- Apply $k = k + 1$ and generate a subset of solutions in N(i,k) such that: - Tabu moves are not chosen. - One of the aspiration criteria a(i, m) is applicable.
3- Choose the best solution i' among the set of neighboring solutions N(i, k). Apply i = best i'.
4- If f(i) <= f(i*) , then we have found a better solution. Apply i* = i
5- Update the list T and the aspiration criteria.
6- If a stopping condition is met, stop. Otherwise, return to Step 2.[43]

c) The Harmony Search Algorithm

The Harmony Search Algorithm (HSA) draws inspiration from the musical process of seeking perfect harmony. Musical performances aim for pleasing harmony, much like optimization processes aim for a global solution defined by an objective function. The aesthetic quality of musical tones is akin to the values of decision variables in optimization. The algorithm models the improvisation process of musicians, who typically follow three rules: playing a tone from memory, playing an adjacent tone, or playing a completely random tone. [44]

d) Genetic algorithm

Genetic Algorithms (GAs) are widely regarded as the most popular and commonly used technique among evolutionary algorithms. Their origins can be traced back to the early 1970s, with the pioneering work of John Holland and his students at the University of Michigan on adaptive systems (Holland, 1975). David E. Goldberg's influential book (Goldberg, 1989) significantly contributed to their widespread adoption. These algorithms are primarily distinguished by the representation of genotype data, initially as a binary vector and more generally as a string of characters. The algorithm utilizes solutions from the initial population to generate a new population using genetic operators such as crossover and mutation. [41]

II.3.3. Extensions of metaheuristics

In this section, we review some of the extensions that have been proposed to address specific optimization challenges. [45-47]

■ ***Parallelization:*** Parallelization in metaheuristics refers to the process of running multiple parts of a metaheuristic algorithm simultaneously, typically on multiple processors or computing units. This approach aims to speed up the optimization process by dividing the work among the available computing resources.[48]

■ ***Multi-objective Optimization***: An increasing number of challenges require the simultaneous consideration of multiple conflicting objectives. In such scenarios, a single optimum does not exist; instead, a range of Pareto-optimal solutions is sought, forming the "compromise surface" of the problem. These solutions may require final arbitration by the user.

■ ***Multimodal Optimization*** Multimodal optimization refers to the optimization of functions that have multiple optima, or peaks, in their search space. Unlike unimodal optimization, which seeks a single global optimum, multimodal optimization aims to find multiple local optima. Variants such as "multi-population" concurrently exploit multiple populations, each aiming to discern distinct optima.

■ ***Hybrid Methods:*** The rapid growth of metaheuristics can be attributed to the challenges classical optimization methods encounter in complex engineering problems. Following the initial enthusiasm among advocates of specific metaheuristics, a realistic evaluation is required, recognizing the synergy among these innovative methods and with alternative approaches. As a result, hybrid methods are currently gaining prominence.

II.3.4. Application of metaheuristics

a) Test functions

Every new metaheuristic technique must be validated using test functions where the global optimum is known in advance and the number of iterations is fixed to allow for comparison of results with other techniques.

b) Real applications

At the application level, metaheuristic techniques have been successfully applied in various research areas such as: telecommunications networks, production cell formation, medical applications, electrical systems, traveling salesman problem, project scheduling, communication system security, knapsack problem, multimedia applications, automated storage and retrieval systems, quadratic assignment problems, resource allocation problems, vehicle routing, reconfigurable production systems, allocation in logistics chains, assembly sequence planning, scheduling in Open Shop environments, and routing in optical networks. [35]

To apply metaheuristics to real-world problems effectively, one should follow these steps:

1. Define the problem and its objective function.
2. Choose a suitable metaheuristic algorithm (e.g., Genetic Algorithms, Simulated Annealing).
3. Fine-tune algorithm parameters using techniques like grid search.
4. Represent solutions appropriately and generate an initial population.

5. Implement the objective function and handle any constraints.
6. Allow the metaheuristic to iteratively explore the search space.
7. Define termination criteria.
8. Analyze results, validate solutions against real-world constraints, and consider trade-offs. [49]

II.4. Multi objective optimization

II.4.1. Main concept

Multi-objective optimization involves the simultaneous optimization of two or more often conflicting objectives. This concept applies not only to computer architecture but also to many real-world problems that require optimizing multiple objectives. Typically, there is a trade-off between these conflicting objectives.

In multi-objective optimizations, there are usually multiple solutions to the problem. No single solution fully minimizes/maximizes each objective. The goal of multi-objective optimization algorithms is to find solutions where each objective is optimized to the fullest extent possible.

Various search methods are used in multi-objective optimization, including simulated annealing, genetic algorithms, and particle swarm optimization. These algorithms are often adapted from single-objective ones and typically address two major issues: assigning an order or fitness. [50]

II.4.2. General form of Multi-Objective Optimization Problem

A set of decision variables forms a vector that adheres to constraints while optimizing a vector function composed of objective functions. These functions mathematically describe performance criteria typically in conflict. Thus, "optimize" refers to finding a solution that yields acceptable values for all objective functions to the decision maker.

The general MOOP can now be formally defined as follows:

***Definition*:** find the vector which satisfies the m inequality constraints:

$$gi(\vec{x}) \geq 0 \ = \ 1,2,\dots,m \tag{II.3}$$

The p equality constraints

$$h_i(\vec{x}) = 0 \quad i = 1,2,\dots,p \tag{II.4}$$

And optimizes the vector function

$$\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), \dots, f_k(\vec{x})]^T \quad \text{(II.5)}$$

In other words, it is aimed to determine from among the set of all solutions which satisfy (II.3) (II.4) and the particular set $x_1^*, x_2^* \dots, x_n^*$ which yields the optimum values of all the objective functions.

- ✓ The constraints given by (II.3) and (II.4) define the feasible region Ω and any point $\vec{x}$ in Ω defines a feasible solution.
- ✓ The vector function $\vec{f}(\vec{x})$ is a function which maps the set Ω into the set A which represents all possible values of the objective functions.
- ✓ The k components of the vector $\vec{f}(\vec{x})$ represent the number of criteria which must be considered.
- ✓ The vector x^*, is reserved to denote the optimal solutions (normally there is more than one). [51]

II.4.3. Dominance

The concept of dominance is a key element in Multi-Objective Optimization (MOO). One solution dominates another if it is superior in every objective and at least equal in others. This notion supports Pareto optimality, where a solution is Pareto optimal if no other solution improves upon it in at least one objective without worsening others.

MOO utilizes various generalized dominance structures, such as ranking-dominance, strength dominance relationships, NLAD, D-Dominance, and L-Dominance, each offering specific advantages based on the application context. Research in MOO spans decades, encompassing evolutionary algorithms, constraint handling, and interactive methods to assist users in making informed decisions among Pareto optimal solutions. [52, 53]

In Multi-Objective Optimization (MOP), the concept of dominance is employed to determine if a given solution is superior to another. Solution w dominates solution v if w is strictly superior to v in at least one objective and equal or superior to v in the remaining objectives (refer to Figure II.5).

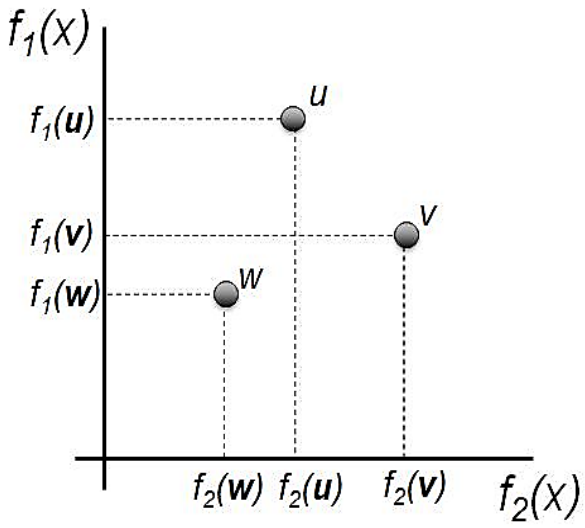


Figure II.5 : *Dominance in multi-objective optimization[54]*

II.4.4. Pareto optimal solution

Definition (Pareto-optimal solution): A feasible solution x of a minimization problem is Pareto-optimal if there exists no other solution y such that $f_k(y) \leq f_k(x)$, k=1,…,r with for at least one objective.[55]

During this stage, a single set of ground motions is chosen in the scenario of single-objective optimal selection or multi-objective optimal selection with predetermined weight values established in the preprocessing phase. Additionally, several different sets of ground motions can be obtained on the Pareto front of the selection objectives in the case of multi-objective optimal selection without pre-established weight values. These Pareto-optimal solutions epitomize the most favorable compromises between the various selection objectives, as depicted in Figure II.6 for the scenario involving two selection objectives.

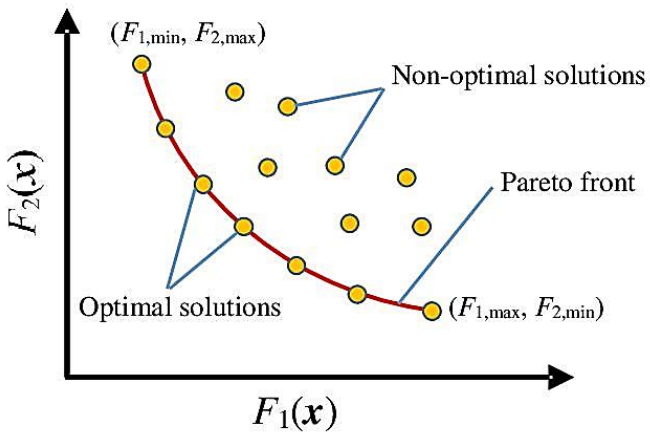


Figure II.6: *Graphical depiction of Pareto optimal solution [56]*

II.4.5. Goals in multi objective optimization

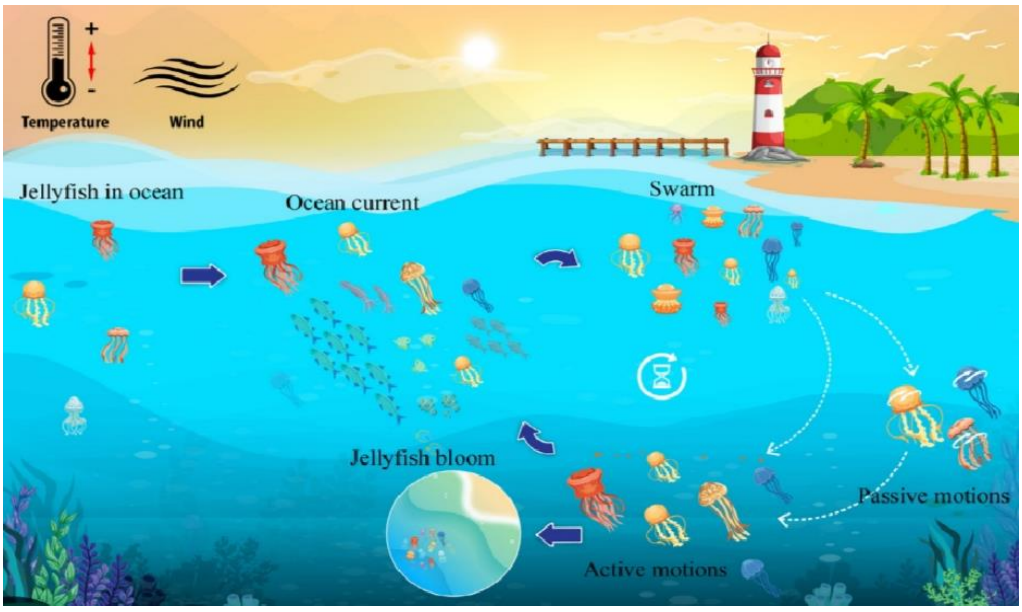
Multi-objective optimization (MOO) is a technique employed to discover optimal solutions when faced with multiple conflicting objectives or criteria that need simultaneous satisfaction. The primary aim of MOO is to identify Pareto-optimal solutions, which remain unmatched by any other feasible solution, offering a spectrum of trade-offs between objectives.[57]

Essential steps in MOO involve defining the problem, formulating objective functions, designing the search space, and selecting an optimization algorithm. MOO finds applications

in various fields such as engineering, finance, healthcare, and transportation, facilitating well-informed decision-making amidst conflicting goals. [57-59]

II.5. Principle of Jellyfish Search method

The Jellyfish Search Algorithm (JSA) is a recent novel metaheuristic optimizer inspired by the behavior of jellyfish in the ocean. It was proposed by Jui-Sheng Chou and Dinh-Nhat Truong in 2020. Its main inspiration is depicted in Figure II.7, illustrating the movements of jellyfish in the ocean.



FigureII.7 : *Jellyfish in the ocean*

II.5.1. Jellyfish search algorithm

The algorithm operates based on several rules:

- ✓ Jellyfish either ***follow the ocean current*** or ***navigate within the jellyfish swarm***, with a "time control" mechanism regulating the transition between these motion types.
- ✓ Jellyfish ***explore the ocean*** in search of food, exhibiting a stronger attraction to areas with higher food availability.
- ✓ The quantity of food encountered is determined by both the ***location*** and the ***objective function.***

a) Ocean current

The ocean current contains a lot of nutrients, so the jellyfish are attracted to it. The direction of the ocean current is formulated as:

$$\overrightarrow{trend} = X^* - 3 \times rand(0,1) \times \mu \qquad \text{(II.6)}$$

where X^* is the location of current best jellyfish in the swarm; μ is the mean value of all jellyfish locations.

Hence, the updated location of each jellyfish is given by

$$X_i(t+1) = X_i(t) + rand(0,1) \times \overrightarrow{trend} \qquad \text{(II.7)}$$

where $X_i(t)$ is location of $i\ th$ jellyfish at time t.

b) Jellyfish swarm

In a jellyfish swarm, which is a large congregation of jellyfish, the jellyfish exhibit two types of motion:

- ➢ **Passive motion** (type A), where they move around their own position,
- ➢ **Active motion** (type B), where they move around another position.

Upon formation of the swarm, most jellyfish initially demonstrate type A motion. However, as time progresses, they increasingly adopt type B motion.

Type A motion involves jellyfish moving around their own locations, and the updated location of each jellyfish can be determined using the following formula:

$$X_i(t+1) = X_i(t) + 0.1 \times rand(0,1) \times (Ub - Lb) \qquad \text{(II.8)}$$

To simulate ***type B motion*** (II.9), a jellyfish (j) is randomly selected, excluding the one of interest, and a vector from the jellyfish of interest (i) to the selected jellyfish (j) determines the direction of movement (II.10).

If the quantity of food at the location of the selected jellyfish (j) exceeds that at the location of the jellyfish of interest (i), the latter moves towards the former. Conversely, it moves directly away from that location if the quantity of food available to the selected jellyfish (j) is lower than that at the location of the jellyfish of interest (i).

(II.11) and (II.12) provide the updated position of a jellyfish.

$$\overrightarrow{Step}\ is\ simulated\ as\ rand(0,1) \times \overrightarrow{Direction} \quad \text{(II.9)}$$

$$\overrightarrow{Direction} = \begin{cases} X_j(t) - X_i(t) if\ f(X_i) \geq f(X_j) \\ X_i(t) - X_j(t) if\ f(X_i) \leq f(X_j) \end{cases} \quad \text{(II.10)}$$

$$\overrightarrow{Step} = X_i(t\ +\ 1) -\ X_i(t) \quad \text{(II.11)}$$

Hence, $X_i(t\ +\ 1) = X_i(t) + \overrightarrow{Step}$ (II.12)

c) Time control mechanism

The ocean current, rich in nutrients, attracts jellyfish. Over time, more jellyfish gather within the current, forming a jellyfish swarm. Changes in temperature or wind can prompt the jellyfish within the swarm to relocate to another ocean current, thus forming a new jellyfish swarm. Within a jellyfish swarm, jellyfish exhibit both type A and type B motion, transitioning between them. Initially, type A motion is favored, but over time, type B becomes increasingly preferred.

To model this scenario, a time control mechanism is introduced. This mechanism regulates jellyfish movement within the ocean current and the jellyfish swarm using a time control function, denoted as $c(t)$, and a threshold constant, Co. The time control function varies randomly from 0 to 1 over time, as expressed in (II.13)

When $c(t)$ exceeds Co, jellyfish follow the ocean current, while if it is less than Co, they move within the jellyfish swarm. Since the exact value of Co is unknown and the time control function varies randomly from zero to one, Co is set to 0.5, representing the midpoint between zero and one.

$$c(t)\ =\ (1\ -\frac{t}{Max_{iter}})\ \times\ (\ 2\ \times\ rand\ (0,1\ )\ -\ 1) \quad \text{(II.13)}$$

where t is the time specified as a number of iterations and Max_{iter} is the maximum number of iterations, which is an initialized parameter.

Like $c(t)$, the function $1 - c(t)$ is used to simulate the motion of a jellyfish in a swarm (type A or B).

- ✓ When rand (0, 1) exceeds $(1 - c(t))$, a jellyfish exhibits type A motion.
- ✓ When rand (0, 1) is lower than $(1 - c(t))$, it exhibits type B motion.

Since $1 - c(t)$ increases from zero to one over time, the probability that $(rand(0,1) > 1 - c(t))$ initially exceeds the probability that $(1 - c(t) > rand(0,1))$. Therefore, type A motion is favored over type B.

As time passes, $1 - c(t)$ approaches one, and the probability that $(1 - c(t) > rand(0,1))$ ultimately exceeds $(rand(0,1) > 1 - c(t))$. Therefore, type B motion becomes more likely.[3]

Algorithm II.3 Pseudocode of JS optimizer algorithm. [60]

Begin
Define the objective function $f(X)$, where $X = (x_1, \ldots, x_d)^T$
Set the search space, population size (n_{Pop}), and maximum iteration (Max_{int})
Initialize the population of jellyfish $X_i (i = 1,2,..,n_{Pop})$ using a logistic chaotic map
Calculate the quantity of food at each, $X_i, f(X_i)$
Find the jellyfish at the location currently with the most food (X^*)
Initialize time: $t = 1$
Repeat
  For i=1: n_{Pop} do
    Calculate the time control function c(t) using(Eq.II.13)
    If *c(t) ≥0.5 : the jellyfish follows the ocean current*
      Determine the ocean current using (Eq.II.6)
      Calculate the new location of the jellyfish using (Eq.II.7)
    Else: *the jellyfish moves inside a swarm*
      If *rand(0,1) > (1 - c(t)): the jellyfish exhibits type A motion (Passive motions)*
        Calculate the new location of the jellyfish using (Eq.II.8).
      Else: *the jellyfish exhibits type B motion (Active motions)*
        Determine the direction of the jellyfish using (Eq.II.10)
        Calculate the new location of the jellyfish using(Eq.II.12)
      End if
    End if
Check the boundary conditions and calculate the quantity of food at the new location Update the location of the jellyfish (X_i) and the location of the jellyfish currently with the most food (X).*
End For
Update the time: t=t+1
Until *stop criterion is met (e.g., t>(Max_{int}))*
Output the best results and visualization (Jellyfish bloom)
End

II.5.2. Multi-Objective Jellyfish Search

The three main features of Multi-Objective Jellyfish Search (MOJS) are as follows:

1. Integration of an archive into JS to save and retrieve Pareto-optimal solutions.

2. Utilization of crowding distance and roulette-wheel selection to efficiently manage the archive population, containing the optimal non-dominated solutions during space exploration.

3. Incorporation of Lévy flight, elite population, and opposition-based jumping methods into MOJS to prevent premature convergence in the presence of numerous local optima.

a) Elite Population

An elite population is maintained to preserve the best solutions obtained by individual jellyfish in each iteration. It involves evaluating the trial solution U_i(t + 1) against the target solution X_i(t) associated with the current elite population. If the trial solution dominates the target, it replaces it in the elite population. After each iteration, the elite population is used to update the archive population.

b) Elitist Selection

An archive is introduced to store and retrieve accurate approximations of genuine Pareto-optimal solutions. To ensure a well-distributed Pareto-optimal front, a member of the elite population is identified within the least populated region of the acquired Pareto-optimal front. This involves segmenting the search space, determining optimal and suboptimal objectives, and defining hyper-spheres and grid cells covering all solutions. The selection process employs a roulette-wheel mechanism, with the probability (P_i) for each segment calculated using the equation :

$$Pi = \frac{c}{Ni} \qquad \text{(II.14)}$$

where c is a positive constant that is greater than one (i.e., c is set to 10 herein), and Ni is the number of obtained Pareto-optimal solutions in the i^{th} segment. (II.14) gives MOJS a higher probability of choosing the best member (the elitist member) of the population from less populated segments. Therefore, the jellyfish are encouraged to move around such regions, improving their distribution over the whole Pareto-optimal front, as shown in Figure II.8.

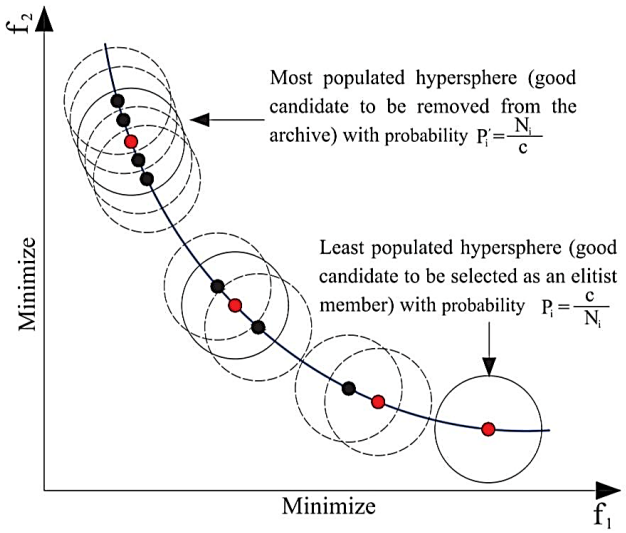


Figure II.8: *Conceptual model of best hyper-spheres for selecting an elitist member or removing a solution from the archive*

c) Lévy Flight

In Lévy flights, each jump, regardless of its magnitude, consumes one unit of time. Numerous studies have illustrated that the movement patterns of various flying animals and insects can be accurately described as Lévy flights . The direction of a step in a Lévy flight follows a uniform distribution, and Mantegna's algorithm is employed to establish a stable and symmetric Lévy distribution. The step length (s) in Mantegna's algorithm is determined by (II.15).

$$\text{Lévy(s)} \sim s = \frac{u}{|v|^{\frac{1}{\tau}}},\ 0 < \tau \leq 2 \qquad \text{(II.15)}$$

where u and v are normally distributed:

$$u \sim N(0,\ \sigma_u^2),\quad v \sim N(0,\ \sigma_v^2)$$

$$\sigma_u = \left\{ \frac{\Gamma(1+\tau)\ sin(\pi\tau/2)}{\Gamma\left[\frac{1+\tau}{2}\right]\tau 2^{(\tau-1)/2}} \right\}^{\frac{1}{\tau}} \quad ,\sigma_v = 1,\ \tau \text{ is set to 1.5}$$

Here, $\Gamma(z)$ is the Gamma function

$$\Gamma(z) = \int_0^{\infty} t^{z-1}\, e^{-t} dt$$

d) Archive Population Update

Updating the archive is crucial in each iteration of the optimization process and may reach its maximum capacity. Therefore, it requires a management mechanism. If a solution is dominated by any solution in the archive, it's excluded. If a solution dominates any Pareto-

optimal solutions in the archive, they are all replaced by the new solution. If a solution isn't dominated by any solution in the archive, it's added. When the archive is full, the least promising solutions are removed from the most crowded segments. To efficiently select solutions for removal, the worst (most populated) hyper-sphere is chosen, discouraging exploration in crowded areas. This selection process is detailed in Figure II.8 and implemented using a roulette-wheel mechanism with specific probabilities (Pi') assigned to each segment.

$$p_i{}' = \frac{N_i}{C} \quad \text{(II.16)}$$

e) Mathematical multi-objective jellyfish search optimizer

Fig.II.9 illustrates the movement of jellyfish in the ocean: they align with the ocean current and navigate within a swarm. A Lévy walk is employed to generate new solutions around the best solution obtained so far, enhancing local search along the ocean current. Hence, (II.7) is updated to (II.17).

$$\mathrm{X_i}(t+1) = \mathrm{EL\_X_i}(t) + \overrightarrow{\mathrm{trend}} \otimes \mathrm{Lévy(s)} \quad \text{(II.17)}$$

$$\overrightarrow{\mathrm{trend}} = X^*(t) - 3 \times rand(0,1) \times \frac{\sum EL\_X}{\mathrm{n}_{pop}} \quad \text{(II.18)}$$

where $\mathrm{EL\_X_i(t)}$ is the elite population of $\mathrm{X_i(t)}$; $\sum \mathrm{EL\_X}$ is the entire elite population; $\mathrm{n_{pop}}$ is the population size; the product $\otimes$ is the entry-wise multiplications of random Lévy movements. $\mathrm{X^*(t)}$ is the elitist solution at time t, which is selected from archive. Next, jellyfish move inside swarm, exhibiting both passive and active motions (types A and B, respectively). In these motions, the elitist solution replaces the current best solution, so (II.8) and (II.12) are revised as (II.19) and (II.20), and represented below.

$$\text{Type A motion : } \mathrm{X_i}(t+1) = \mathrm{X^*(t)} + (\mathrm{EL\_X_i}(t) - \mathrm{X^*(t)}) \otimes \mathrm{Lévy(s)} \quad \text{(II.19)}$$

$$\text{Type B motion : } \mathrm{X_i}(t+1) = \mathrm{X^*(t)} + \overrightarrow{\mathrm{Step}} \quad \text{(II.20)}$$

where

$$\overrightarrow{\mathrm{Step}} = \mathrm{rand(0,1)} \times \overrightarrow{\mathrm{Direction}} \quad \text{(II.21)}$$

$$\overrightarrow{\mathrm{Direction}} = \begin{cases} \mathrm{EL\_X_j(t)} - \mathrm{EL\_X_i(t)} \text{if } \mathrm{EL\_X_i(t)} > \mathrm{EL\_X_j(t)} \\ \mathrm{EL\_X_i(t)} - \mathrm{EL\_X_j(t)} \text{if } \mathrm{EL\_X_i(t)} < \mathrm{EL\_X_j(t)} \end{cases} \quad \text{(II.22)}$$

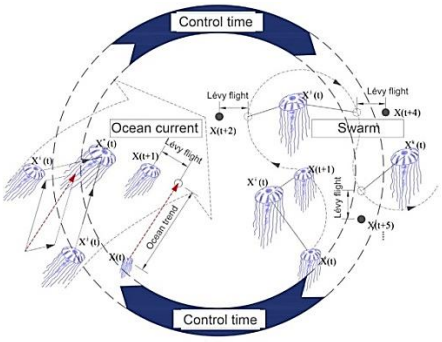


Figure II.9: *Simulated ocean current, swarm and control time mechanism[3]*

Figure II.10 represents a flowchart for the MOJS method.

II.6. Conclusion

This chapter provides an overview of optimization problems, their classification, and solution techniques. It explores metaheuristic principles, categorizing them according to different criteria, with presenting their applications in diverse fields. Multi-objective optimization is discussed, emphasizing its importance in balancing conflicting objectives. The chapter also introduces the Jellyfish Search method (JS), a recent metaheuristic inspired by jellyfish behavior, detailing its principles and adaptation for multi-objective optimization, showcasing its potential for efficiently solving complex real-world problems.

In the next chapter, we will utilize the multi-objective Jellyfish Search Algorithm to optimize wireless sensor networks for both objectives coverage and energy consumption.

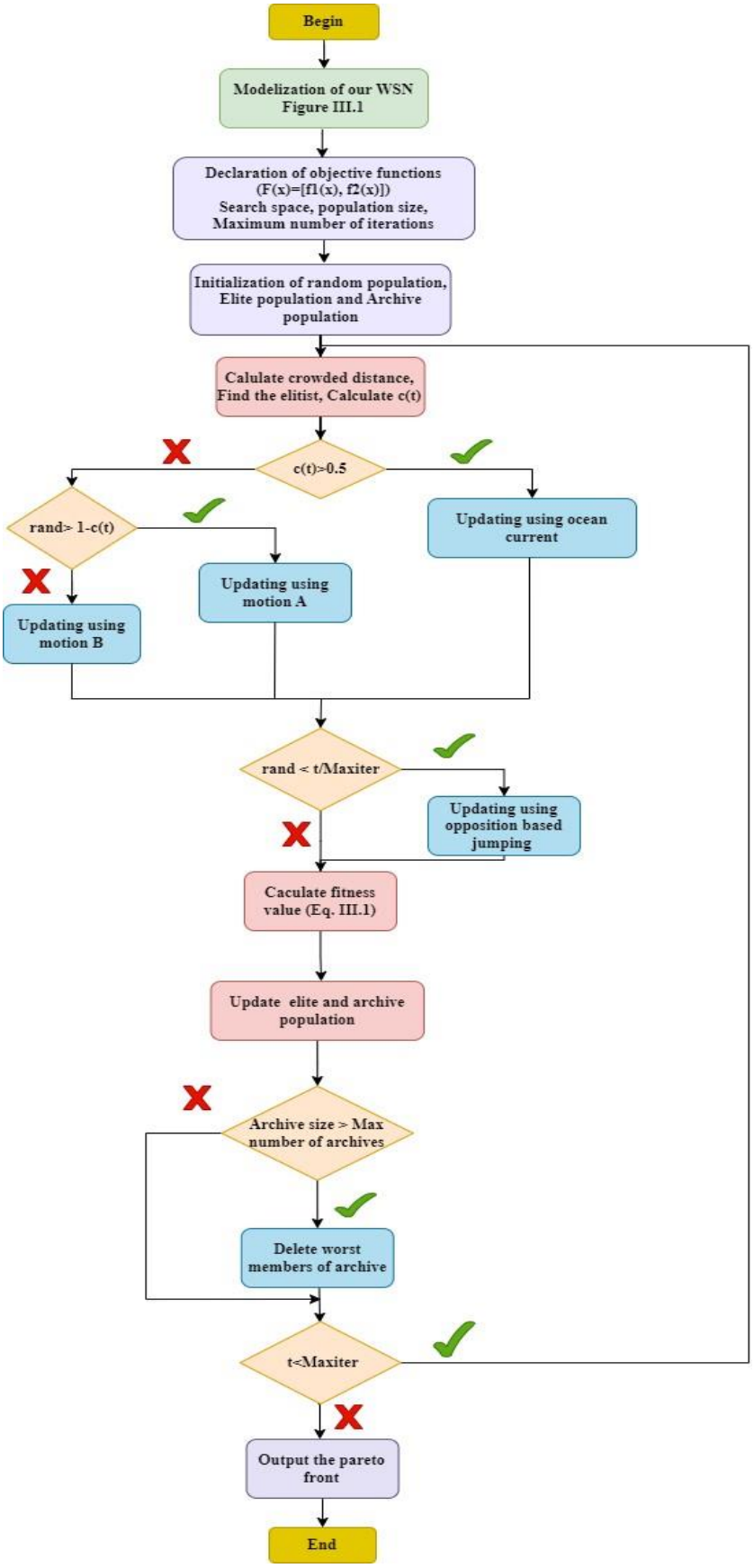


Figure II.10: *MOJS flowchart*

Chapter III:

Optimization of Wireless Sensor Networks Using the Multi-Objective Jellyfish (WSN-MOJS)

III.1. Introduction

Wireless Sensor Networks (WSNs) are crucial components in the infrastructure of modern communication systems, such as environmental monitoring, military surveillance, healthcare and industrial automation. These networks comprise a number of autonomous sensors that monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, or pollutants, and cooperatively pass their data through the network to a main location.

The deployment of WSN faces critical challenges, particularly in terms of coverage and energy efficiency. Maximizing network coverage ensures that the monitored area is sufficiently observed, enhancing the reliability and accuracy of the data collected. Concurrently, minimizing energy consumption is essential as sensors are typically battery-powered and often deployed in inaccessible locations, making frequent replacements or recharges impractical.

Traditional optimization techniques often struggle with the complexity and the multi-objective nature of the problem in WSNs. These challenges necessitate the development of robust, efficient, and adaptive strategies capable of optimizing multiple conflicting objectives, such as maximizing coverage while minimizing energy consumption. This is where metaheuristic algorithms, known for their capability to find near-optimal solutions in complex search spaces, come into play.

This chapter proposes the application of Multi-Objective Jellyfish (MOJS) Metaheuristic Algorithm to optimize the coverage and energy consumption in sensor networks. We conducted various simulations using MATLAB to assess the performance of our Multi-Objective Jellyfish Search (MOJS) approach in optimizing Wireless Sensor Networks (WSNs).

III.2. Presentation of our WSN

For our simulations, we designed a network in MATLAB consisting of N sensor nodes to monitor a zone with dimensions $L \times H$. The designed zone is versatile and can represent a variety of environments, including urban areas, forests, or industrial zones, allowing for the simulation of diverse scenarios and applications. The figure below presents a sample of the network model used in our simulations. It contains 5 nodes positioned at the red circle points, the green and blue circles represent the sensing and communication ranges, respectively.

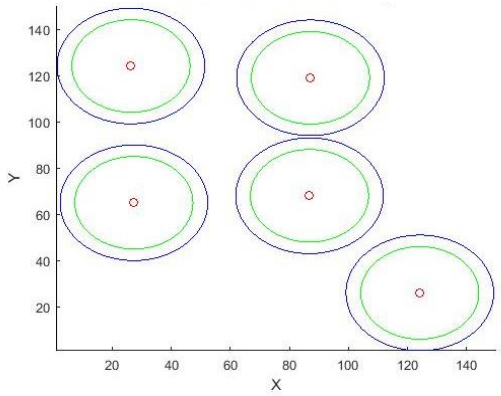


Figure III.1: *Example of the network model used in our simulations*

Each sensor node is assigned a position P in the two axes (x, y) within the zone $\left(P_{i_x},\ P_{i_y}\right)$

Where $i = 1,2, \dots N$

Our objective is to determine the optimal positions for sensor nodes that maximize coverage while minimizing energy consumption.

Table III.1. Parameters used for modelling the WSN

Parameter	**Value**
N: Sensor node number	10
L: Length of the zone	150
H:height of the zone	150
Rc: Communication range	20
Rs: Sensing range	25

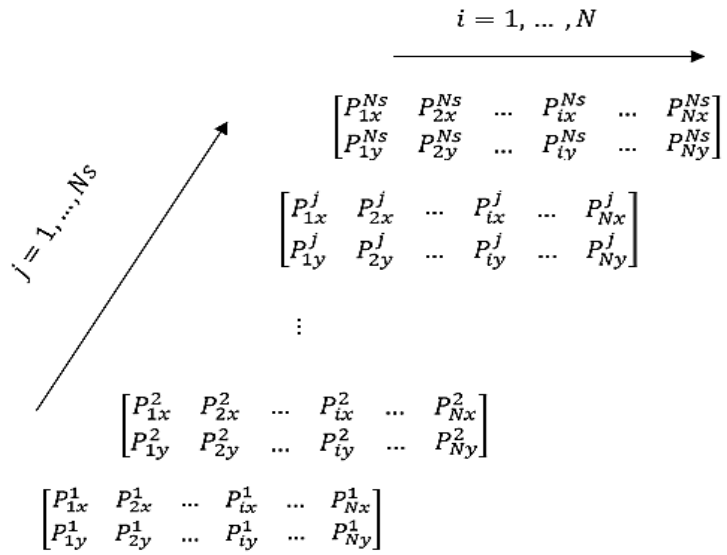
III.3. WSN-MOJS: Optimizing WSN Performance using MOJS

Our proposed method, named WSN-MOJS, aims to enhance WSN performance using the MOJS metaheuristic. It starts by randomly initializing candidate solutions, which are then assessed using the relevant objective functions. The MOJS updating process is iteratively applied to improve these solutions until a predefined stopping criterion is met.

III.3.1. Initialization

Our objective is to find the positions $\left(P_{i_x},\ P_{i_y}\right)$ of each node. Mathematically, we aim to find a matrix of size; 2 rows and N columns. To achieve this, we initialize randomly several

candidate solutions (numerous $2 \times N$ matrices). The initial population can be represented by a 3D matrix of size $2 \times N \times Ns$, as shown below.



The matrix enables the visualization of the sensor's position solution. This solution exhibits the stochastic nature of the algorithm, which converges iteratively towards the optimum solution.

Where: N: number of sensor, Ns: number of solutions

III.3.2. Fitness function

In our optimization problem, the Multi-Objective Jellyfish Search (MOJS) algorithm is employed to optimize Wireless Sensor Networks (WSNs) with two primary objectives: maximizing coverage and minimizing energy consumption. By simultaneously optimizing these two objectives, MOJS helps achieve a balanced and efficient deployment of sensor nodes in WSNs.

$$F = [F_1, F_2] \qquad \text{(III.1)}$$

a) Maximum coverage: The first function aims to maximize the covered area of the zone or decrease its inverse.

Assumed that, the considered zone is divided into grid of points (X_k and Y_k) covering the entire area of interest. For each sensor, we calculate the Euclidean distance between the current sensor position and the grid points.

$$D_{ik} = \sqrt{(P_{ix} - X_k)^2 \quad + \quad (P_{iy} \quad - \quad Y_k)^2}$$

$$i = 1,2, \dots, N \qquad k = 1,2, \dots, Ng$$

$$C_{ik} = \begin{cases} 1 & if\ \ D_{ik} \leq Rs \\ 0 & else \end{cases}$$

$$C = \sum_{i=1}^{N} \sum_{k=1}^{Ng} c_{ik}$$

$$F_1 = \left(1 \quad - \quad \left(\frac{C}{Total\ area}\right)\right) \times 100 \qquad \text{(III.2)}$$

Where $Total\ area$ is the surface of the considered zone: $Total\ area = L \times H$.

When the C value (coverage) increases, the F1 value decreases.

b) Minimum consumption energy: Second function seeks to minimize the energy expended by the sensor nodes, prolonging network lifetime and reducing the need for frequent battery replacements or recharges.

The total energy is the sum of the energy of the transmitter and receiver; equation (III.3)

$$E_{Total} = E_{Trasmitter} + E_{receiver} \qquad \text{(III.3)}$$

Energy is the product of power multiplied by time.

$$E = P \times T \qquad \text{(III.4)}$$

In the transmission, the transmitter energy formula is given as follows:

$$E_{transmitter} = P_{transmitter} \times T \qquad \text{(III.5)}$$

Where:

- Transmission Power ($P_{transmitter}$) is the power level at which the signal is transmitted by the transmitter (in watts)
- Transmission Duration (T) is the duration of transmission (in seconds), representing the time interval over which data is transmitted.

Generally, the transmission power should be set to the minimum value that ensures reliable communication between neighboring nodes. It's important to note that using higher transmission power levels can lead to increased energy consumption.

In the reception: receiver energy is given as follows

$$E_{receiver} = P_{receiver} \times T \qquad \text{(III.6)}$$

The power of receiver depends on the physical distance separating the nodes and the transmitted power, a relationship described by the Friis formula as follows. [61]

$$\frac{P_{receiver}}{P_{transmitter}} = \frac{A_r A_t}{d^2 \lambda^2} \quad \text{(III.7)}$$

where A_r and A_t denote the effective areas of the receiving and transmitting antennas, respectively, d is the distance between the antennas, and λ is the wavelength, calculated as the speed of light c divided by the signal frequency f.

This work does not specifically focus on a particular hardware platform for the sensor nodes; thus, the simplest antenna model, considering a single transmission and reception isotropic antenna at each sensor node, is selected to model the effective antenna areas. Consequently, the effective areas are given by equation (III.7).

$$A_{isotropic} = \frac{\lambda^2}{4\pi} \quad \text{(III.8)}$$

and, thus, making $A_r = A_t = A_{isotropic}$ the power ratio (III.5) simplifies to

$$\frac{P_{receiver}}{P_{transmitter}} = \left(\frac{\lambda}{4\pi d}\right)^2 \quad \text{(III.9)}$$

So, we deduce the expression of the receiver power:

$$P_{receiver} = P_r = P_t \left(\frac{\lambda}{4\pi d}\right)^2 \quad \text{(III.10)}$$

To conclude, the second function to optimize is the total energy

$$F_2 = E_{Total} \quad \text{(III.11)}$$

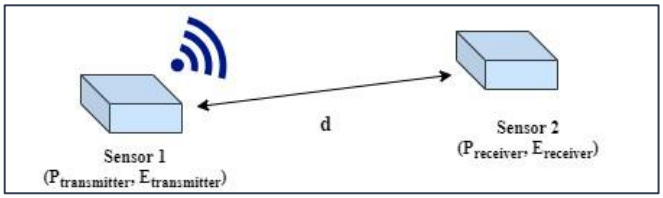


Figure III.2: *Example of transmission between nodes*

III.3.3. Updating process

Population is updating using the MOJS processes; occean current, motion A, motion B and opposition based jumping.

III.3.4. Parameters used for simulation

Table III. 2. Parameters values used for simulation

Parameter	Value
Ns : Population size	100
Max_Iter: Maximun number of iterations	200
Nr: Archive size	100
Ng: grid size	100
$\boldsymbol{P_{transmitter}}$	1 watt
$\boldsymbol{T}$	1 s

The main flowchart of the proposed approach is shown in Figure III.3

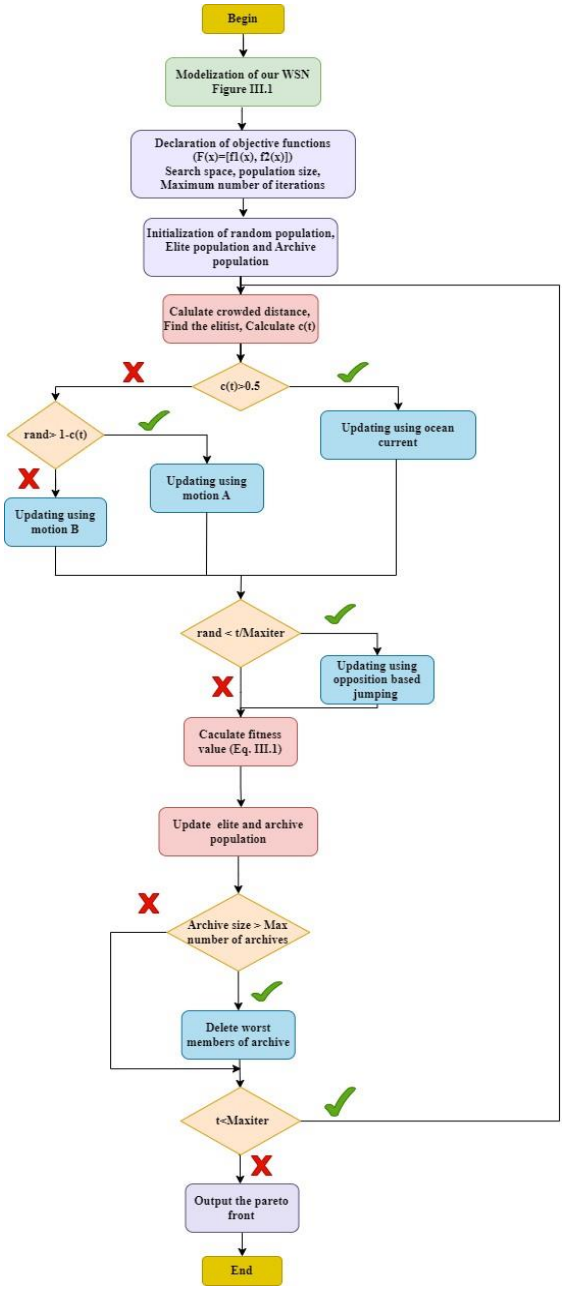


Figure III.3: *Flowchart of our WSN-MOJS approach*

III.4. Simulation results

To test the performance of our WSN-MOJS approach, we will conduct several Simulations by varying the number of nodes, the number of candidate solutions and the number of iterations. We will display the following figures:

- ***Network***: This figure represents the optimal positions of the nodes and illustrates the area covered by the nodes. It is plotted using the leader solution.
- ***The Pareto Front***: This figure represents the non-dominant solutions, it illustrates the trade-off between coverage and energy consumption in the WSN.
- ***Convergence Curve of F1***: This figure plots the values of the objective function F1 (inverse of the coverage) over the iterations, with the goal of minimizing it.
- ***Convergence Curve of F2***: This figure plots the values of the objective function F2 (energy consumed by the network) over the iterations, with the goal of minimizing it.

III.4.1. Simulation 1

In simulation 1, we evaluated the algorithm with different numbers of nodes: 5, 10, and 15. The objective is to verify the method's ability to ensure maximum coverage and minimum energy consumption with a small, medium, and large number of nodes relative to the considered area.

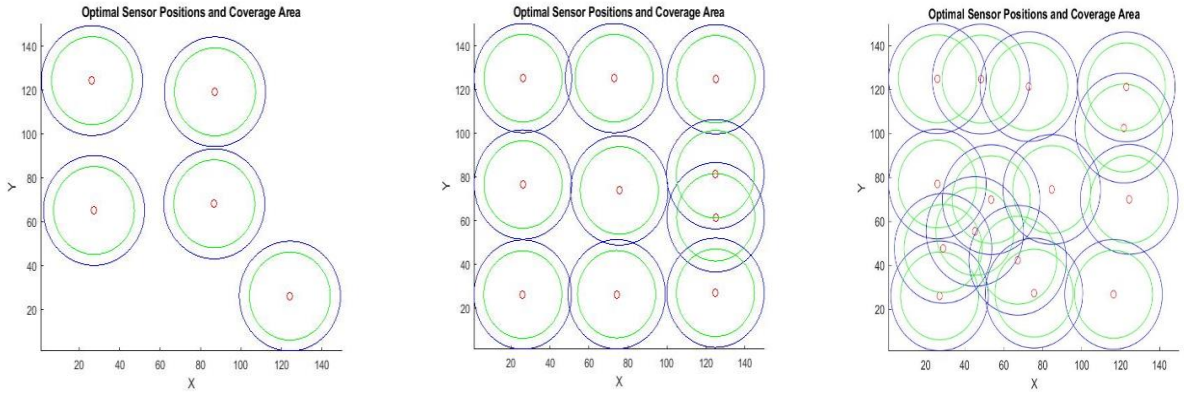


Figure III.4: Network with different number of nodes (5, 10 and 15)

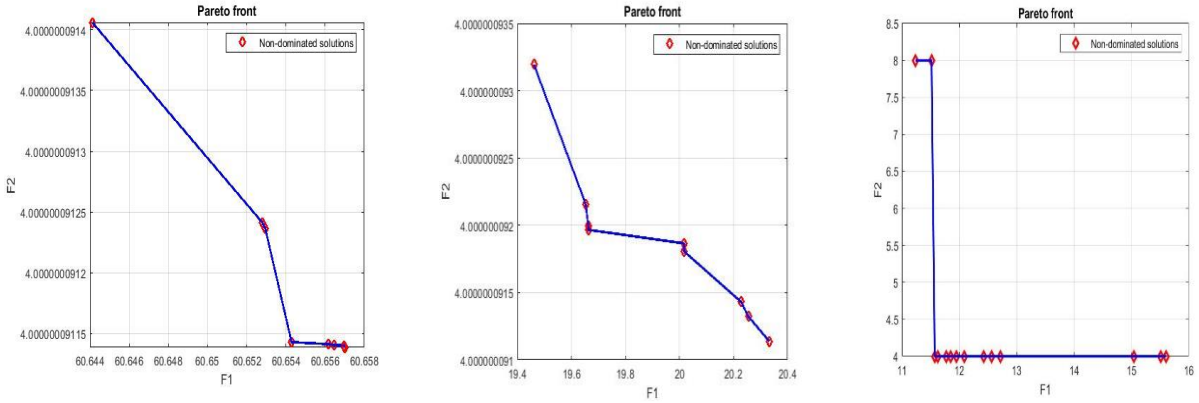


Figure III.5 : Pareto front with different number of nodes (5, 10 and 15)

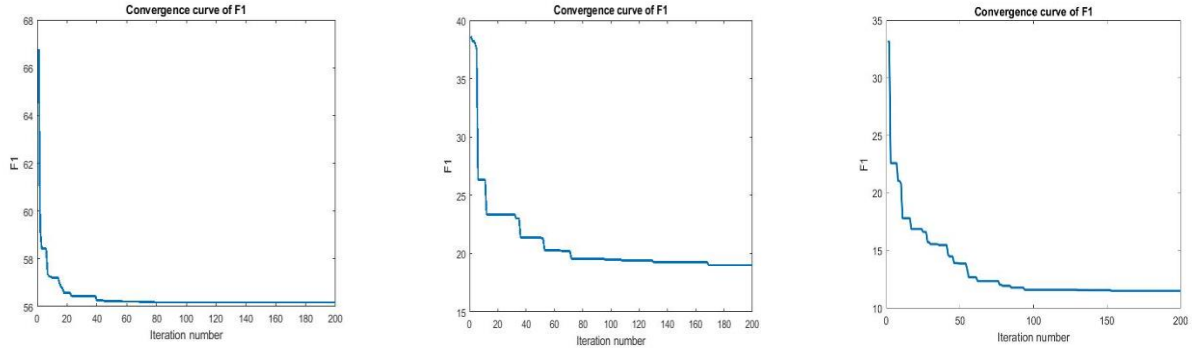


Figure III.6 : Convergence curve of F1 with different number of nodes (5, 10 and 15)

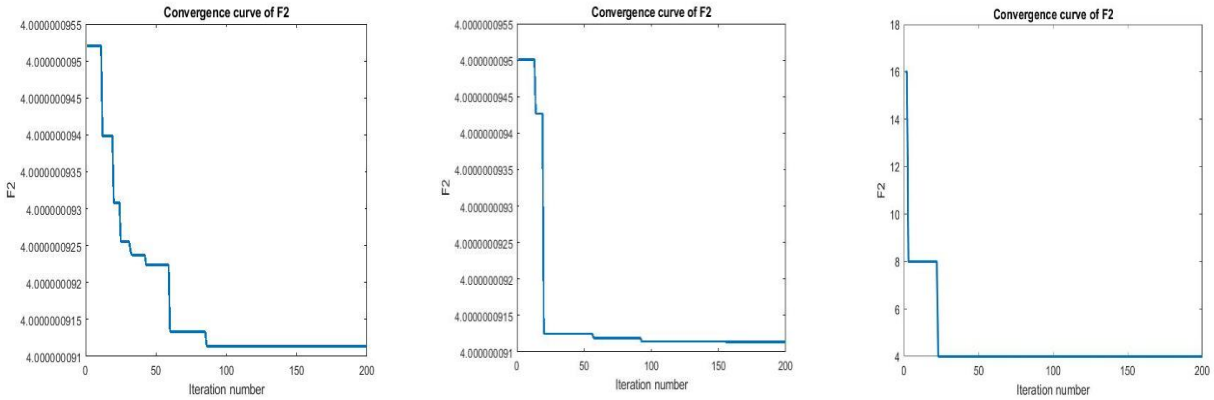


Figure III.7 : Convergence curve of F2 with different number of nodes (5, 10 and 15)

Discussion 1

- According to Figure III.4:
  - With 5 nodes, although the number of nodes is small compared to the area, the algorithm found the optimal positions to cover the maximum surface area.
  - With 10 nodes, almost the entire area is covered with minimal overlap.
  - With 15 nodes, the entire area is covered, but with relatively high overlap.
- According to Figure III.5.

  - With 5 nodes, the Pareto front contains 8 non-dominant solutions; with 10 nodes, it contains 9; and with 15 nodes, it contains 14. The number of non-dominant solutions increases with the number of nodes, because there are more possibilities for positioning the nodes.
  - In the three cases, the solutions are well-distributed along the front, indicating a diverse set of optimal trade-offs.
- According to Figure III.6:
  - The convergence curve of f1 demonstrates that the technique was able to minimize the function f1 (which represents the inverse of coverage).
  - With 5 nodes, f1 stabilizes near the value 56.1; with 10 nodes, it stabilizes at 18; and with 15 nodes, it stabilizes at 12. When the number of nodes increases, the coverage also increases and the inverse decreases, which explains the decrease in f1 with the increase of nodes number.
- According to Figure III.7: It should be noted that with a constant transmission power, the reception power decreases with increasing the distance between nodes. However, with the increase in the number of nodes, the energy consumption increases. We observe that the convergence curves in the three cases stabilize at almost the same results:
  - With 5 nodes: The nodes are positioned a little apart to guarantee maximum coverage.
  - With 10 and 15 nodes: The number of nodes increases, but they are positioned closer together.

Conclusion 1

Adding more nodes is unnecessary, as the algorithm can ensure satisfactory coverage with an average number of nodes.

The Pareto front obtained through our multi-objective method illustrates the trade-off between coverage and energy consumption in the wireless sensor network.

The diversity in the non-dominant solutions allows for flexibility in decision-making, enabling to select a solution that best fits specific needs, whether prioritizing maximum coverage or minimal energy consumption.

The convergence curve of f1 demonstrated that the algorithm was able to minimize the function f1 and ensure maximum coverage. The higher the number of nodes, the more the area is covered.

The convergence curve of f2 demonstrated that the algorithm was able to minimize the function f2 and ensure maximum of energy consumed. The algorithm can ensure almost stable energy consumption even with the increase in the number of nodes

III.4.2. Simulation 2

In simulation 2, we evaluated the algorithm with different numbers of candidate solutions: 20, 50, and 100. The objective of this simulation is to test if the performance of the method improves with the increase in the number of candidate solutions.

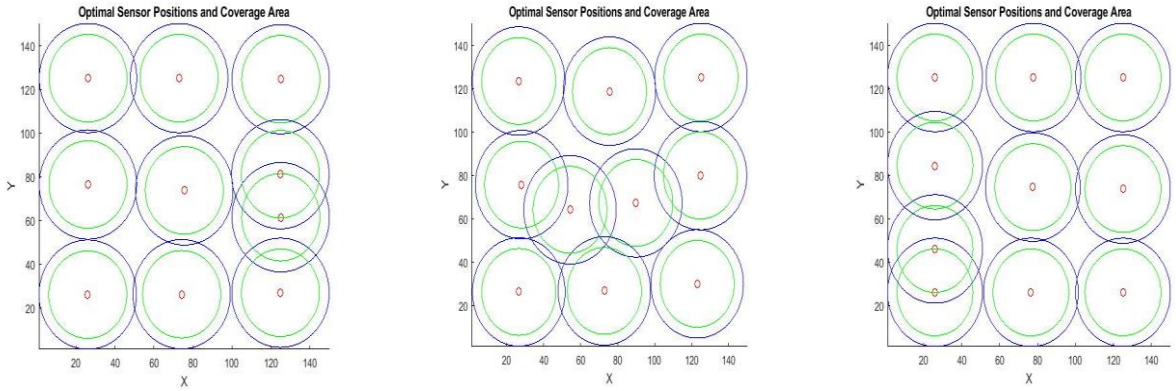


Figure III.8 : Network with different number of candidate solutions (20, 50 and 100)

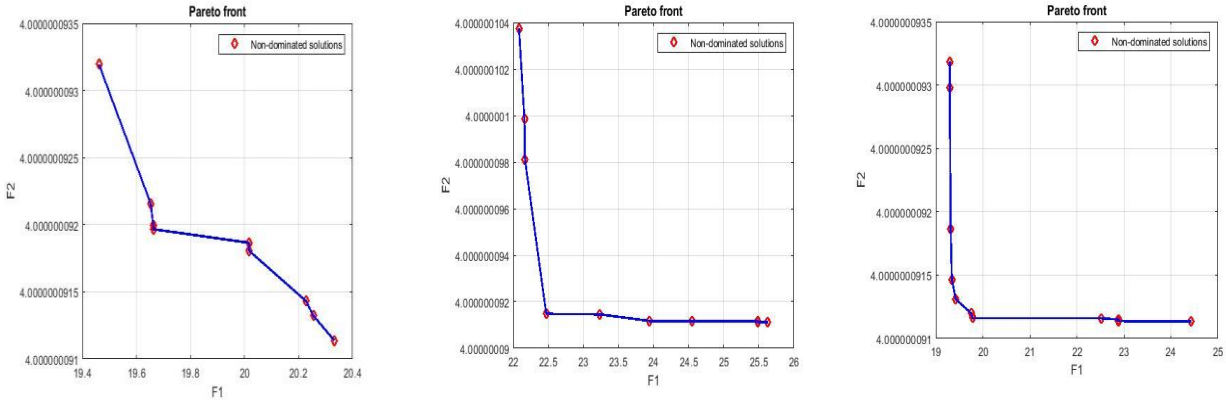


Figure III.9 :Pareto front with different number of candidate solutions (20, 50 and 100)

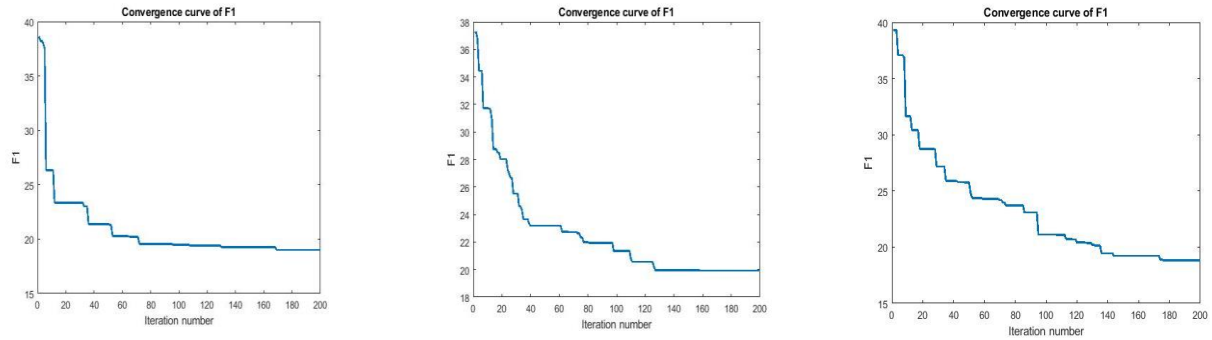


Figure III.10 : Convergence curve of F1 with different number of candidate solutions (20, 50 and 100)

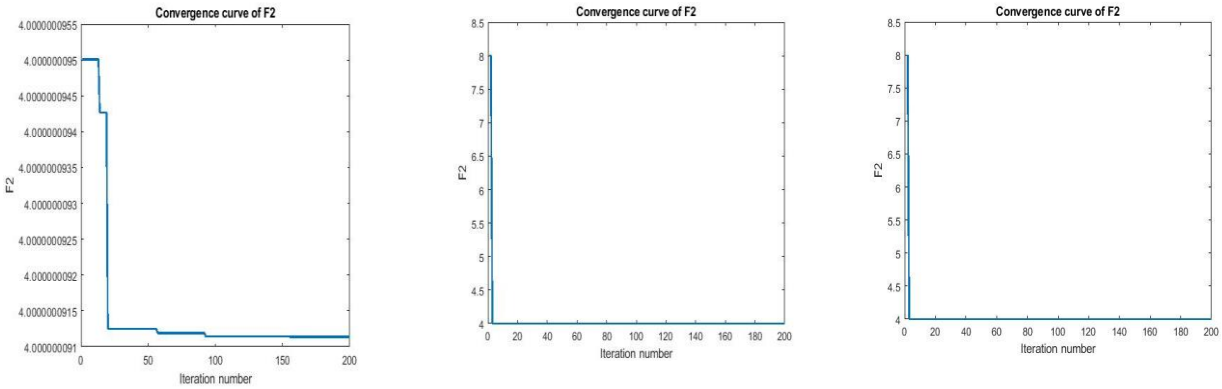


Figure III.11: Convergence curve of F2 with different number of candidate solutions (20, 50 and 100)

Discussion 2

- According to Figure III.8: almost complete coverage is ensured in all three cases. A small number of solutions is sufficient to ensure maximum coverage of the area.
- According to Figure III.9: the non-dominant solutions are well distributed with 50 and 100 candidate solutions. An increase in candidate solutions improves the appearance of the Pareto front.
- According to Figure III.10: the convergence curve with 50 and 100 candidate solutions gradually decreases as it converges towards an optimal solution. The algorithm explores the search space more effectively before stabilizing at a fixed value, which reduces the risk of falling into a local optimum.
- According to Figure III.11: the energy consumed, which is linked with the distance between nodes, stabilizes earlier in the first iterations with 50 and 100 candidate solutions. Since the considered area is not very large, there is no great possibility of positioning the nodes very far apart.

Conclusion 2

The algorithm does not need a large number of solutions to ensure maximum coverage, which provides an advantage in terms of computational complexity.

With the increase in the number of candidate solutions, our approach demonstrates a superior balance between the two objectives.

The increase in the number of solutions leads to a diversification of the solutions, increases the capacity for exploration of the search space, and ensures the obtaining of a global optimal solution.

The increase in the number of candidate solutions makes it possible to minimize the energy earlier, and increases the exploitation capacity of the approach (i.e., its ability to find the best overall solution in unimodal problems).

III.4.3. Simulation 3

In this simulation, we evaluated the algorithm with different numbers of maximum iterations: 200, 300, and 500. The objective of this simulation is to test if the performance of the method improves with the increase in the number of maximum iterations.

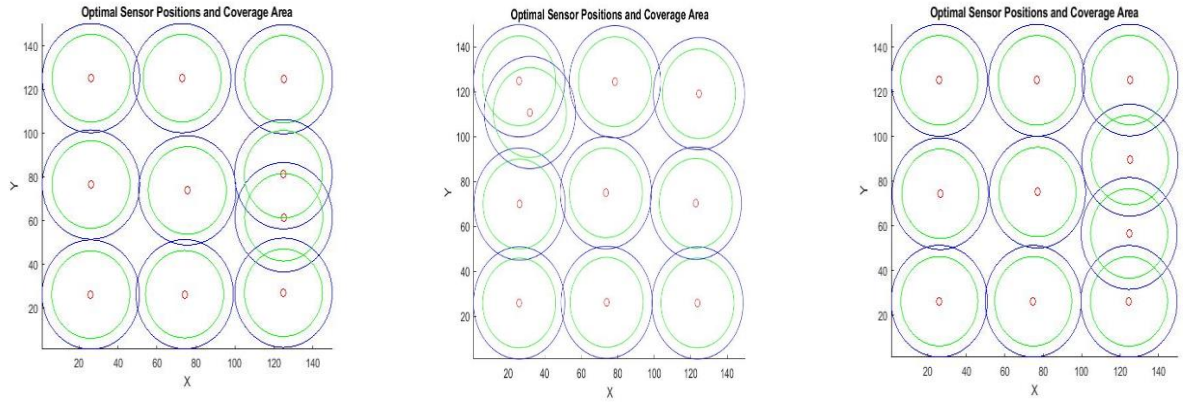


Figure III.12: Network with different number of maximun iterations (200, 300 and 500)

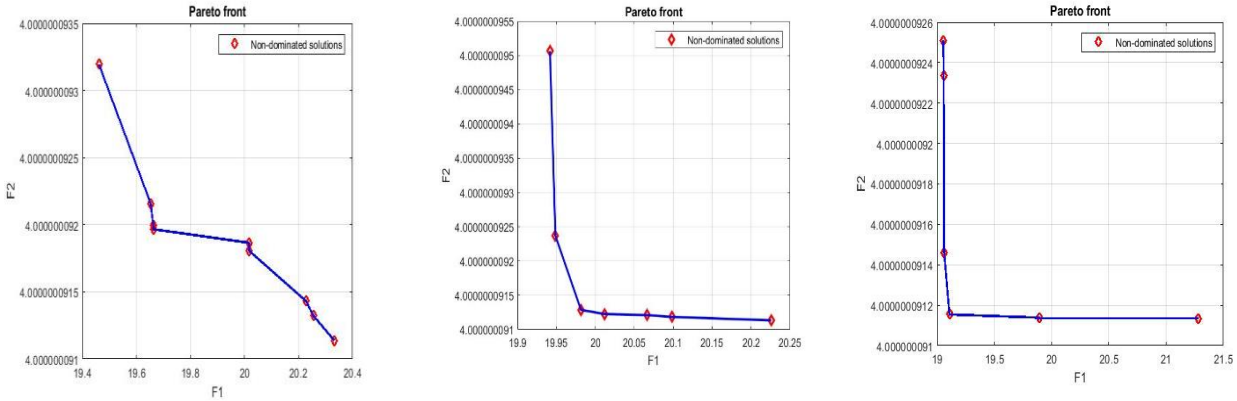


Figure III.13: Pareto front with different number of maximun iterations (200, 300 and 500)

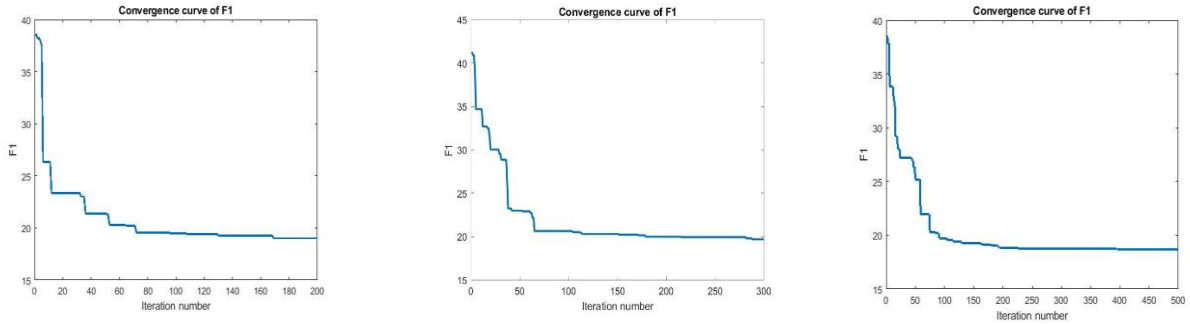


Figure III.14 : Convergence curve of F1 with different number of maximun iterations (200, 300 and 500)

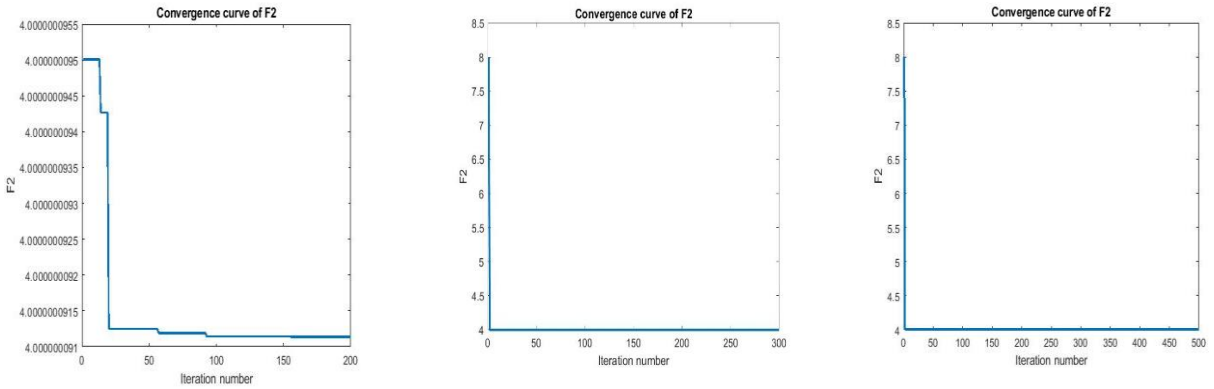


Figure III.15 : Convergence curve of F2 with different number of maximun iterations (200, 300 and 500)

Discussion 3

- According to Figure III.12: in all three cases, nearly complete coverage is guaranteed, so achieving maximum coverage of the area requires only a small number of maximum iterations.
- According to Figure III.13: the non-dominant solutions obtained with 500 iterations are superior, particularly with function f1.

  The minimum value of f1 with 500 iterations is around 19.1, whereas with 200 iterations, it is almost 19.45.
- According to Figure III.14: an improvement in the fitness value is observed as the number of iterations increases.
- According to Figure III.15: since the considered energy problem is unimodal, a small number of iterations is sufficient.

Conclusion 3

The algorithm can achieve maximum coverage without needing a large number of iterations, which is an advantage of the method.

As the number of iterations increases, our approach shows more effective results in both objective functions.

The increase in the number of iterations leads to an improvement in the fitness value obtained for f1.

The increase in the number of iterations does not lead to a significant improvement in the results of f2, given the considered problem.

III.4.4. General discussion

According to the obtained results, the proposed multi-objective approach can ensure maximum coverage with an average number of nodes.

With the increase in the number of candidate solutions, the pareto front is improvesd significantly, and the non-dominated solutions become well-distributed, which provides variety in the results.

The exploration of the search space also improves, allowing the method to avoid local optima. Increasing the number of iterations improved the fitness results, especially with the f1 function.

III.5. Conclusion

This chapter presents an improvement in wireless sensor networks using the multi-objective MOJS (WSN-MOJS) technique. Two objective functions are taken into account: maximizing coverage and minimizing energy consumption. Several simulations are carried out to evaluate the performance of the method.

According to the results obtained, our method is capable of maximizing the coverage of the area with an average number of sensors while minimizing the energy consumed in a reduced computation time. Increasing the number of initialized candidate solutions and the number of iterations can improve the appearance of the Pareto front, producing well distributed non-dominant solutions. Fitness values can also improve.

General Conclusion

General Conclusion

In this memory's dissertation, we studied the optimization of wireless sensor networks using the multi-objective metaheuristic Jellyfish Search (MOJS) technique. To address this subject, we examined wireless sensor networks, their features, and challenges. Subsequently, we explored the principles of the JS optimization technique and its multi-objective variant, MOJS. Finally, we presented our application.

Our objective through this application was to use the MOJS technique to find the optimal arrangement of sensor nodes that ensures maximum coverage of the area and minimum energy consumption. The iterative process began with random positions and finally converged to a globally optimal solution. To test our approach, several simulations were carried out on MATLAB, varying the number of nodes, candidate solutions, and iterations.

As a result, we can observe that the proposed algorithm (WSN-MOJS) ensures maximum coverage with an average number of nodes. It also minimizes energy consumption in minimal time, due to its exploration and exploitation capabilities. Increasing the number of candidate solutions and iterations can significantly improve the Pareto front. The non-dominated solutions become well-distributed, and the fitness values are enhanced.

From the results obtained, it can be concluded that MOJS has been successfully applied in the field of telecommunications, particularly in the optimization of wireless sensor networks. The two objectives considered; maximum coverage and minimum energy consumption, have been effectively optimized in the proposed scenarios. which proves the efficiency and robustness of our WSN-MOJS.

The perspectives of our work are:

- We propose the use of other multi-objective techniques.
- Taking into account other objective functions and constraints.
- Additionally, the use of MOJS in other applications in the telecommunications field.

References

References

[1] Matin, M. A. (Ed.). (2012). Wireless sensor networks: Technology and protocols. BoD–Books on Demand.

[2] Du, K. L., & Swamy, M. N. S. (2016). Search and optimization by metaheuristics. Techniques and Algorithms Inspired by Nature, 1-10.

[3] J.-S. Chou and D.-N. Truong / Chaos, Solitons and Fractals 135 (2020) 109738.

[4] Fdoul, A., Djerifili, S., & Kaddi, K. (2022). Amélioration de l'efficacité énergétique pour les RCSF via une méthode métaheuristique (Doctoral dissertation, Universite ahmed draia-adrar).

[5] https://www.tescaglobal.com/blog/what-is-wireless-sensor/-network-and-types-of-wsn Visited: 31/01/2024 13:51

[6] I.F. Akyildiz, W. Su, Y. Sankara Subramaniam, and E. Cayirci , "A Survey on Sensor Networks". IEEE Communications Magazine, August 2002.

[7] Sipani, J. P., Patel, R. H., Upadhyaya, T., & Desai, A. (2018). Wireless Sensor Network for Monitoring & Control of Environmental Factors using Arduino. International Journal of Interactive Mobile Technologies, 12(2).

[8] Vaishalid, Khairnar terna, Wireless Sensor Networks ENGINEERING COLLEGE (2020).

[9] Miyandoab, F. D. (2015). Design of a body sensor network embedded in textiles for biomedical applications (Doctoral dissertation, Universidade do Porto (Portugal)).

[10] Farooq, M. O., & Kunz, T. (2011). Operating systems for wireless sensor networks: A survey. Sensors, 11(6), 5900-5930.

[11] Ranjan, A., Sahu, H. B., & Misra, P. (2015). A survey report on operating systems for tiny networked sensors. arXiv preprint arXiv:1505.05269.

[12] https://www.redcad.org/members/benhalima/azem/toknow1.html Visited:31/01/2024 14:51

[13] Sensor Networks v4 Uploaded by Marcus Littlewood

[14] Benefit, A. (2013). The evolution of wireless sensor networks. *Silicon Lab. Inc., Austin, TX, USA, White Paper Rev*, *1*, 1-5.

[15] Yasmine, T. C., & Mélissa, R. F. (2015). Déploiement d'un réseau de capteurs sans fil en technologie zigbee (Doctoral dissertation, Université Mouloud Mammeri).

[16] https://edis.ifas.ufl.edu/publication/AE521 Visited: 31/01/2024 13:40

[17] Smaala, A., & Derdouri, L. (2018). Analyse analytique des protocoles multicast géographique dans les réseaux de capteurs.

[18] Raja, B. S., Prabakaran, N., & Dhulipala, V. S. (2011, February). Modified GPSR based optimal routing Algorithm for reliable communication in WSNs. In 2011 International Conference on Devices and Communications (ICDeCom) (pp. 1-5). IEEE.

[19] Lamine, M. M. (2008). Securite dans les Reseaux de Capteurs Sans-Fil. Memoire de Magistere en Informatique Ecole Doctorale d'Informatique de bejaia.

[20] Matin, M. A., & Islam, M. M. (2012). Overview of wireless sensor network. Wireless sensor networks-technology and protocols, 1(3).

[21] Louiza, M. I. (2012). Système de détection d'intrusion hybride et hiérarchique pour les réseaux de capteur sans fil. Mémoire de fin d'études Pour l''obstination du diplôme Master en Informatique Université Mouloud Mammeri de Tizi-Ouzou.

[22] Rachida, A.(2011). Sécurisation d'u protocole de routage Directed Diffusion dans les réseaux de capteurs sans fil. Mémoire de fin d'étude Pour l'obstination du diplôme master en informatique Université Mouloud Mammeri de Tizi-Ouzou.

[23] Mourad, D & Mounia, B. (2021). Routage, Optimisation de la consommation de l'énergie dans un réseau de capteurs sans fil.Mémoire de Master,Présenté au Département : Génie Électrique Domaine : Sciences et Technologies Filière : Télécommunications Spécialité : Systèmes des Télécommunication,Universite Akli Mohand Oulhadji-Bouira.

[24] Lasla, N. (2007). La gestion de clés dans les réseaux de capteurs sans-fil. Mémoire de Magister, Institut National de formation en Informatique (INI), Oued-Smar, Alger.

[25] Zaanane, H., Dihaj, B., & Kaddi, M. (2022). Nouveau protocole de communication et optimisation de l'énergie dans les réseaux de capteurs sans fil (Doctoral dissertation, Universite ahmed draia-adrar).

[26] Lynda, T. (2011). Modele de confiance pour securiser le routage dans les reseaux de capteurs sans-fil, Mémoire de master, Faculté de génie électrique et informatique departement informatique, Université Mouloud Mammeri de Tizi-ouzou.

[27] https://www.javatpoint.com/what-is-mesh-topology Visited: 19/04/2024 18:36

[28] https://fcit.usf.edu/network/chap5/chap5.htm Visited: 19/04/2024 19:06

[29] Hena Tabassum, 'Local Area Network', Course chapter. Link: Visited: 25/04/2024,11:40.

[30] Guadane, M. (2017). Prototypage Rapide et Évaluation en Conditions D'opération Réelles D'algorithmes de Localisation pour les Réseaux de Capteurs Sans Fil (Doctoral dissertation, Institut National de la Recherche Scientifique (Canada)).

[31] Yanbo, Sh. (2014). Cryptographie sur les courbes elliptiques et tolérance aux pannes dans les réseaux de capteurs.Grade de Docteur de l'Université de Franche-Comté Spécialité : Informatique.

[32] Samir, A. (2010). Protocole de sécurité Pour les Réseaux de capteurs Sans Fil. Mémoire de Magister, Faculté des Sciences de l'Ingénieur Département d'Informatique, Université Hadj Lakhder-Batna.

[33] Samir, F & Mohamed, B. (2021). Analyse les performances d'un routage aléatoire sur les réseaux de capteurs sans fil. Mémoire de Master, Faculté mathematiques et informatique département d'informatique, Universite ibn khaldoun-tiaret

[34] Kaur, N., Bedi, R. K., & Gangwar, R. C. (2016, November). A new sink placement strategy for WSNs. In 2016 international conference on ICT in business industry & government (ICTBIG) (pp. 1-5). IEEE.

[35] Betka, A. (2019). Estimation de mouvement par les techniques métaheuristiques (Doctoral dissertation, Université Mohamed Khider-Biskra).

[36] Kherici. N. Optimisation Combinatoire, M1 GADM, UBMA, 2020/2021.

[37] Elhewy, A. M., Hassan, A. M., & Ibrahim, M. A. (2016). Weight optimization of offshore supply vessel based on structural analysis using finite element method. Alexandria Engineering Journal, 55(2), 1005-1015.

[38] Engineering Optimization: Theory and Practice, Fourth Edition Singiresu S. Rao Copyright © 2009 by John Wiley & Sons, Inc.

[39] Dabba . A, Cours d'optimisation des reseaux. (2023-2024).

[40] El Alamin, J. T., & Bouvry, P. (2010). Metaheuristics for Optimal Transfer of P2P Information in VANETs. University of Luxembourg, Master in Information and Computer Sciences.

[41] ILLOUL, Y. (2020). Optimisation des coûts de collecte du lait pour l'entreprise Danone (Doctoral dissertation, M. BRAHAMI Mustapha Anwar).

[42] Bendahmane. A, Le recuit simulé, Cours Master2 RFIA, Module: Optimisation avancée, 25 octobre 2011.

[43] Bouchikhi, N. (2013). La recherche tabou . 2er année Master (RFIA), Université des Sciences Et de la Technologie d'Oran Mohamed Boudiaf USTO.

[44] Baaziz, H., & Houcine, M. Application de la méthode Harmony Search Algorithm pour l'optimisation économique et environnementale du réseau électrique en présence d'une production photovoltaïque (Doctoral dissertation, Université KASDI-MERBAH Ouargla).

[45] SECHEN (C.). ñ VLSI placement and global routing using simulated annealing. Kluwer Acad. Publi., 1988.

[46] Wong (D.F.), Leong (H.W.) et Liu (C.L.). Simulated annealing for VLSI design. Kluwer Acad. Publi., 1988.

[47] Renders (J.M.) et Flasse (S.P.). Hybrid methodsusing genetic algorithms for global optimization. IEEETrans. on Systems, Man, and Cybernetics ñ Part B :Cybernetics, 26, n 2, 1996.

[48] Alba, E. (2005). Parallel metaheuristics: a new class of algorithms. John Wiley & Sons.

[49] https://www.linkedin.com/advice/0/how-can-you-apply-metaheuristics-real-world-6g97f Visited : 08/05/2024 13:00

[50] Calborean, H. (2011). Multi-objective optimization of advanced computer architectures using domain-knowledge. Lucian Blaga" University of Sibiu, Faculty of Engineering, Sibiu, O0.

[51] Coello, C. A. C. (2007). Evolutionary algorithms for solving multi-objective problems. springer. com.

[52] Deb, K., & Ehrgott, M. (2023). On Generalized Dominance Structures for Multi-Objective Optimization. Mathematical and Computational Applications, 28(5), 100.

[53] Bandyopadhyay, S., Chakraborty, R., & Maulik, U. (2015). Priority based∈ dominance: A new measure in multiobjective optimization. Information Sciences, 305, 97-109.

[54] Toutouh, J., & Alba, E. (2016). Natural Computing for Vehicular Networks (Doctoral dissertation, PhD thesis, ETSI Informática, University of Malaga).

[55] Keller, A. A. (2019). Multi-objective optimization in theory and practice II: metaheuristic algorithms. Bentham Science Publishers.

[56] Mergos, P. E., & Sextos, A. (2018). Multi-objective optimum selection of ground motion records with genetic algorithms.

[57] https://www.wallstreetmojo.com/multi-objective-optimization/ Visited: 08/05/2024 11:40

[58] Chang, K. H. (2015). Multiobjective optimization and advanced topics. e-Design, 1105-1173.

[59] Lecture 9: Multi-Objective Optimization Suggested reading: K. Deb, Multi-Objective Optimization using Evolutionary Algorithms, John Wiley & Sons, Inc., 2001.

[60] J.-S. Chou and D.-N. Truong, Applied Mathematics and Computation 389 (2020) 125535.

[61] Friis, H. T. (1946). A Note on a Simple Transmission Formula. Proceedings of IRE, 34(5), 254–256. doi:10.1109/JRPROC.1946.234568.


تم بحمد الله