الجمهورية الجزائرية الديمقراطية الشعبية

People's Democratic Republic of Algeria
وزارة التعليم العالي والبحث العلمي
Ministry of Higher Education and Scientific Research
جامعة عين تموشنت بلحاج بوشعيب
Ain Temouchent – University Belhadj Bouchaib
Faculty of Science and Technology
Department of Mathematics & Computer Science

End of Cycle Project
To obtain Master degree in Networks and Data Engineering
Specialty: Computer Systems

Theme

Optimization of Constrained Relay Node Deployment using a Metaheuristic

Submitted by:

- Walid TOUIL

In front of the jury composed of:

DR. Fatima BEDAD	ATU.B.B (Ain Temouchent)	President
DR. Hakim BENDIABDALLAH	ATU.B.B (Ain Temouchent)	Examiner
DR. Ali BENZERBADJ	ATU.B.B (Ain Temouchent)	Supervisor

Academic Year: 2022/2023

ACKNOWLEDGMENT

All praises to Allah and His blessing for the completion of this thesis. I thank God for all the opportunities, trials and strength that have been showered on me to finish writing the thesis. I experienced so much during this process, not only from the academic aspect but also from the aspect of personality.

First and foremost, I would like to sincerely thank my supervisor Dr. Ali Benzerbadj for his guidance, understanding, patience and most importantly, he has provided positive encouragement and a warm spirit to finish this thesis. It has been a great pleasure and honour to have him as my supervisor.

My deepest gratitude goes to all of my family members. It would not be possible to write this thesis without the support from them. I would like to thank my dearest mother for listening to me with so much patience, for encouraging me and for her abundant moral support, my sister Kawtar, my brother Zakaria, my nephews Kamel and Hichem.

I would sincerely like to thank all my dear friends who were with me and support me through thick and thin. In particular I am tryly grateful to Chems Eddine and Khaled for their unwavering friendship. Your presence in my life has brought countless moments of laughter, inspiration, and motivation.

Lastly, I would like to acknowledge all the individuals, whether mentioned explicitly or not, who have played a role in shaping my thinking and helping me reach this milestone.

Thank you all for being a part of this incredible journey and for your unwavering support. Your belief in me has been the driving force behind my achievements. I am grateful beyond words.

Regards,
Touil Walid

CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

WSN	Wireless Sensor Network
MAC	Media Access Control
PDF	Probability Density Function
THT	Tri-Hexagon Tiling
MOO	Multi-Objective Optimization
$\mathcal{P}$	Polynomial
$\mathcal{NP}$	Non-deterministic Polynomial
COP	Combinatorial Optimization Problems
DP	Dynamic programming
CP	Constraint Programming

LP	Linear Programming
A*	A Star
BB	Branch and Bound
GA	Genetic Algorithm
PSO	Particle Swarm Optimization
ABC	Artificial Bee Colony
CRO	Chemical Reaction Optimization
SA	Simulated Annealing
TS	Tabu Search
VNS	Variable Neighborhood Search
ILS	Iterative Local Search
VND	Variable Neighborhood Descent
GVNS	General Variable Neighborhood Search
SVNS	Skewed Variable Neighborhood Search
BVNS	Basic Variable Neighborhood Search
RN	Relay Node

GENERAL INTRODUCTION

Wireless Sensor Networks (WSNs) have emerged as a versatile and efficient technology for a wide range of applications, including surveillance. The deployment of WSNs for surveillance applications requires careful consideration of various factors such as coverage, connectivity, energy efficiency, and latency which related to the hop count in the network. In order to address these challenges, this thesis focuses on the optimization of constrained relay node deployment using a meta-heuristic approach.

The project consists of three chapters. The first chapter provides an overview of the deployment of wireless sensor networks specifically tailored for surveillance applications. It explores the fundamental concepts and requirements of surveillance systems, emphasizing the need for efficient and reliable WSN deployments. The chapter discusses the key factors influencing the deployment process, including coverage requirements, communication constraints, and energy limitations. It also reviews existing approaches and techniques used for WSN deployment in surveillance applications, highlighting their limitations and the need for further optimization.

Chapter 2 delves into the core of this thesis, introducing the concept of metaheuristic-based multi-objective optimization for WSN deployment. It presents an in-depth exploration of various exact and approximate algorithms and their application to combinatorial optimization problems. It explores different types of combinatorial optimization problems and discusses techniques to resolve these problems. This chapter sets the groundwork for utilizing metaheuristic algorithms to tackle the relay node deployment problem.

The last chapter focuses on the practical implementation of the proposed meta-heuristic-based optimization framework. It describes the design and development of the optimization algorithm, namely the Variable Neighborhood Search (VNS) algorithm, including the formulation of objectives, constraints, and the integration of meta-heuristic techniques. The chapter also presents the experimental setup and methodology used for evaluating the performance of the proposed approach. It discusses the metrics employed to measure the effectiveness of the optimized relay node deployment, such as network coverage, connectivity, energy efficiency, and latency.

At the end of the manuscript, we provide a general conclusion and some future works to explore

CHAPTER 1

DEPLOYMENT OF WIRELESS SENSOR NETWORKS FOR SURVEILLANCE APPLICATIONS

1.1 Introduction

Wireless Sensor Networks (WSNs) have recently emerged as a premier research topic. They have great long-term economic potential, ability to transform lives, and pose many system-building challenges; a WSN can be defined as a network of tiny devices, denoted as sensor or relay nodes. Sensor nodes sense the environment and communicate the information gathered, from a monitored field (e.g., sensitive areas such as airports, oil fields, frontiers, country border, etc.) through wireless links. The

data are then forwarded, eventually via multiple hops and through other sensors or relay nodes, to one or multiple sink nodes (also called base stations). The sink node(s) processes the data locally or sends it to a remote decision centre through a high throughput network. The WSN nodes can be stationary or mobile, aware of their location or not and homogeneous or not [14].
In the following sections we are going to tackle the deployment problematic of such networks.

1.2 WSNs: Constraints and Challenges

Now that we have established the context and importance of WSNs, let's discuss significant constraints that can undermine their reliability in various ways, despite their usefulness. It is crucial to consider these unique issues as they can have a significant impact on the overall performance and efficiency of the network, potentially making it unusable.

1.2.1 Constraints

Energy: Nodes in WSNs are typically powered by batteries, which are not easily replaceable or rechargeable. As a result, the available energy of each network node is limited. For instance, optimal node deployment, energy-efficient routing and Media Access Control (MAC) design, etc., are crucial design criteria for WSNs because the power failure of a node not only affects the node itself but also its capacity to forward packets on behalf of other nodes [8].

Bandwidth: WSNs often operate in a constrained frequency spectrum, resulting in limited bandwidth. This limitation can restrict the amount of data that can be transmitted within a given time, potentially affecting the timeliness and accuracy of data collection [8].

Unreliable wireless connections: It can be a result of various factors, including interference, signal strength issues, and packet loss. These problems can lead to inconsistent connectivity between nodes.

Furthermore, in addition to the discussed challenges, WSNs also face other crucial issues such as fault tolerance, scalability, coverage and connectivity, among others.

1.2.2 Challenges

WSNs deployment

The process of optimally placing sensor nodes in a given area to achieve specific objectives. The node deployment problem in WSNs involves determining the optimal locations for deploying the sensor nodes to optimize network performance and coverage while considering various constraints. Node deployment can be deterministic or stochastic (random) [5].

1.3 WSNs deployment approaches

Random Deployment

WSN Random deployment refers to the process of placing sensor nodes in an arbitrary or stochastic manner within a target area, in other words it involves scattering the nodes randomly or using a probabilistic distribution across the deployment region although their density can be controlled [10], sensor positions are defined by a Probability Density Function (PDF). Depending on the deployment strategy, the coordinates of the sensor positions may follow a particular distribution. We categorise the random placement strategies into simple and compound [27] as illustrated in Figure 1.3.1.

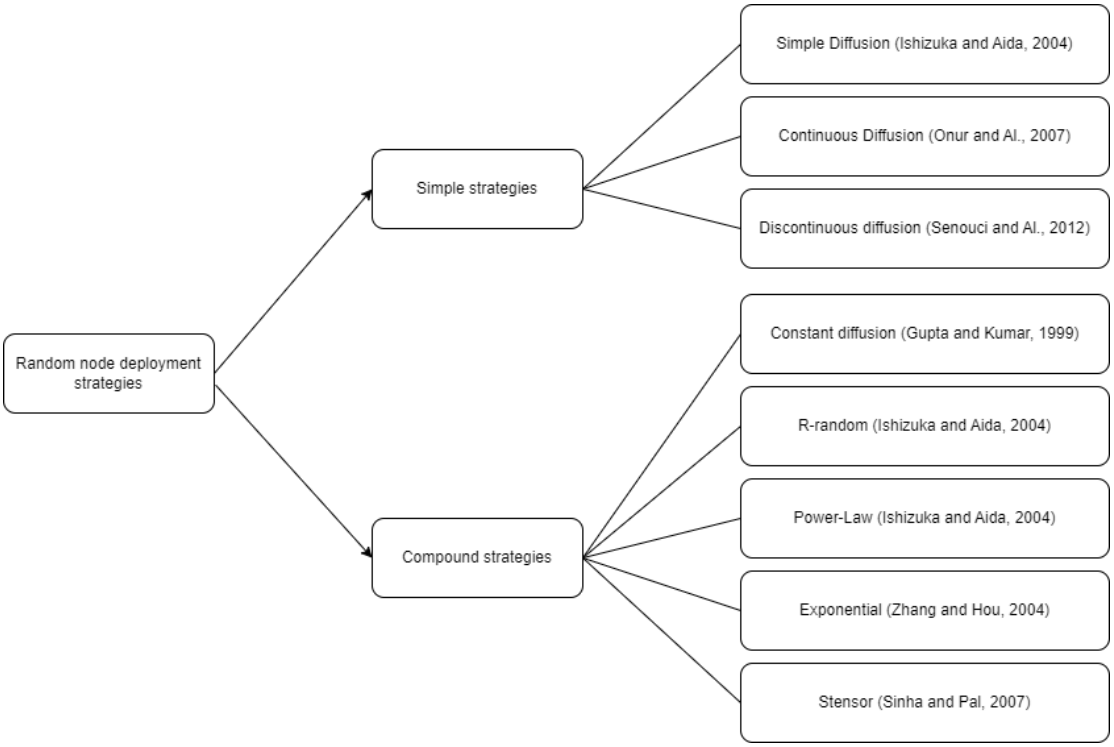

Figure 1.3.1: Random node deployment strategies in WSN

Random deployment is suitable for applications where the details of the regions are not known, inaccessible or hostile. An example of random deployment of sensor nodes would be within a disaster region [6].
Logically, random deployments result in sub-optimal WSN performance in terms of coverage, connectivity, delay and energy efficiency.

Deterministic Deployment

In this approach each sensor node is placed at pre-determined coordinates, this type of deployment is usually pursued when nodes are expensive or when their operation is significantly affected by their position. In contrast to random deployment, deterministic deployment offers an optimal network configuration since the positioning of nodes is determined beforehand to meet the design goals (such as reducing costs and energy usage while enhancing coverage, connectivity, delay and lifetime). From performance and cost point of view, deterministic deployment is

considered a smart deployment approach in comparison to the random one [31]. WSNs deterministic deployment strategies consider the optimization of one or more objectives related to the application needs. Among these objectives, coverage is considered the most important metric in the literature. The research community has extensively investigated methods to maximize the coverage rate while minimizing node usage. In addition to coverage, maintaining connectivity and energy saving are another concerns in WSNs [31].

1.3.1 Deployment stategies to meet coverage and connectivity in WSNs

A sensor's prime function is to sense the environment for any occurrence of the event of interest. Therefore as mentioned before coverage is one of the major concerns in WSN. In fact it is a key for evaluating the quality of service in WSN [32]. Connectivity and coverage problems are caused by the limited communication and sensing range. The solution lies in how the sensors are positioned with respect to each other. To solve both problems, the sensors need to be placed optimally. In order to maximize coverage, the sensors should not be positioned too close to each other, allowing the sensing capability of the network to be fully utilized. At the same time, they should not be located too far apart to avoid gaps or areas without coverage. From a connectivity standpoint, the sensors should be placed close enough to ensure they are within each other's communication range, thus ensuring connectivity [4]. Some researchers believe that by positioning the node in a particular location and keeping it stationary, it will provide equal and consistent sensing or monitoring capabilities at every point throughout the sensor field because of its deterministic architecture. Poe and Schmitt presented three deployments for large-scale WSNs namely, a uniform random, a square grid, and a pattern-based Tri-Hexagon Tiling (THT) node deployment [34]. The figures below show the different grid techniques of node positioning.

In a triangular lattice sensor nodes are positioned at the vertices of equilateral triangles The distance between neighboring nodes in the lattice is kept uniform, ensuring a regular distribution throughout the network.

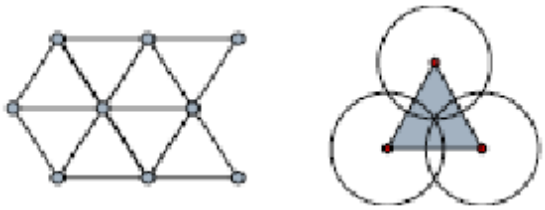

Figure 1.3.2: Triangular Lattice

In a hexagonal grid the arrangement of sensor nodes are positioned at the vertices of regular hexagons, with each node equidistant from its neighboring nodes.

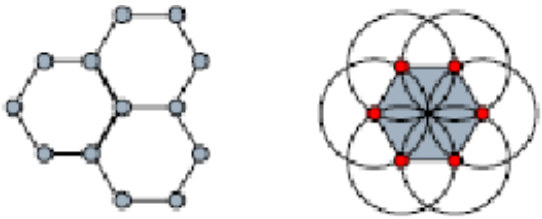

Figure 1.3.3: Hexagonal Grid

In a Square Grid pattern the given area of interest is divided into equal squares and nodes are placed at the corners.
It is assumed that the sensing range is equal to the length of a cell.

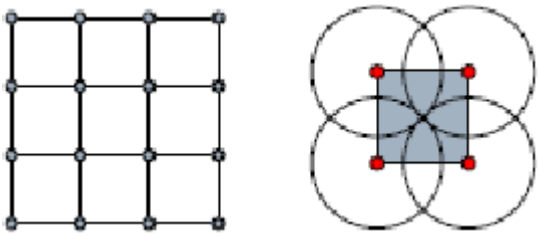

Figure 1.3.4: Square Grid Pattern where nodes are placed in the corners

Same as the square grid in the left figure, but the nodes are placed at the middle of the squares instead.

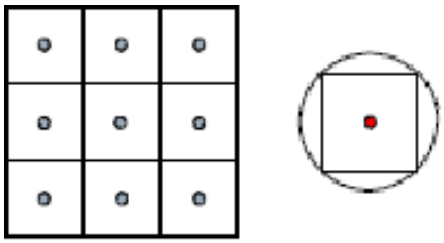

Figure 1.3.5: Square Grid Pattern where nodes are placed in the middle

1.4 Surveillance applications

Security problems are a major concern in several domains, and surveillance applications, especially those that rely on WSNs, are no exception. WSNs are widely employed in surveillance systems due to their ability to monitor and gather data from distributed sensor nodes and transmit it to a central monitoring station through relay nodes. These surveillance applications can be found in various scenarios For example intrusion detection systems, traffic surveillance or sensitive fenced areas (see Figure 1.4.1).

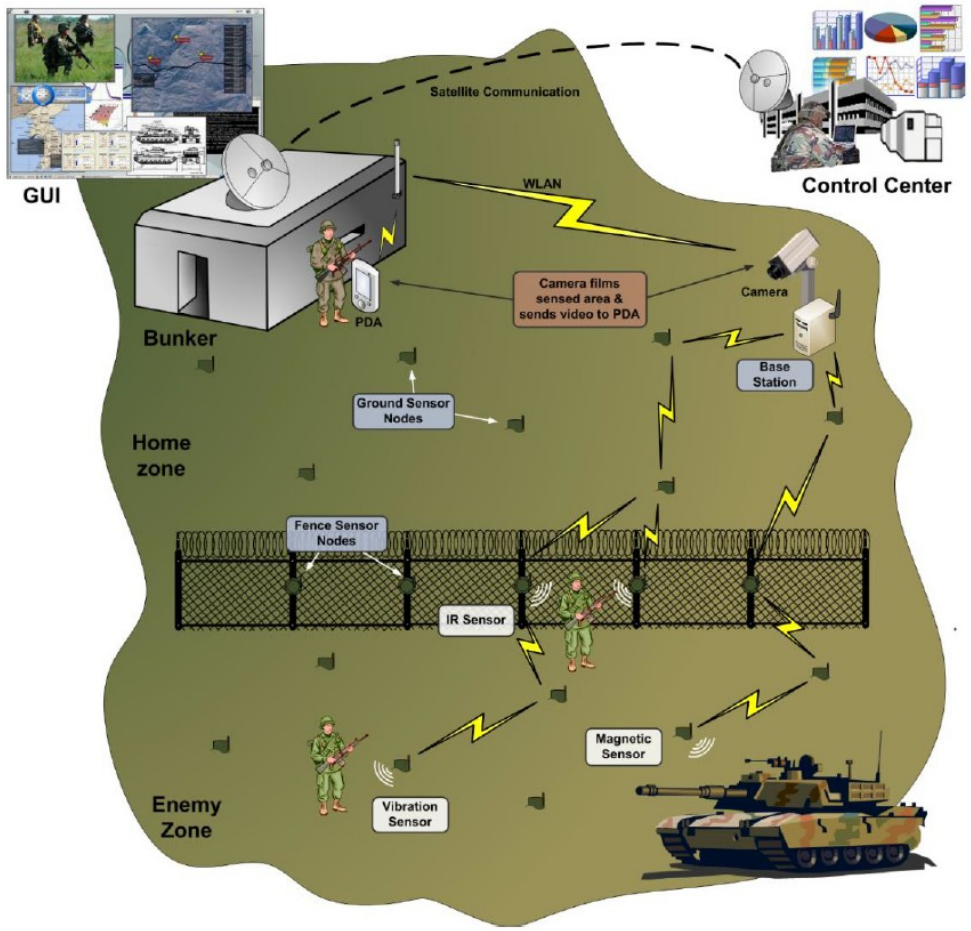

Figure 1.4.1: Fenced military area that use WSNs for surveillance

Surveillance applications have special requirements such an extended network lifetime, a reasonable latency which is generally related to the hop count and a high degree of coverage and connectivity.

In this thesis, we have tackled the problem of the surveillance of fenced sensitive areas using two-tiered topology based WSNs. The latter are composed of sensor and relay nodes. Sensor nodes are used as sentinel to monitor the border of the fenced sensitive area while the relay nodes are used to forward alert messages until the unique sink node as illustrated by Figure 1.4.2

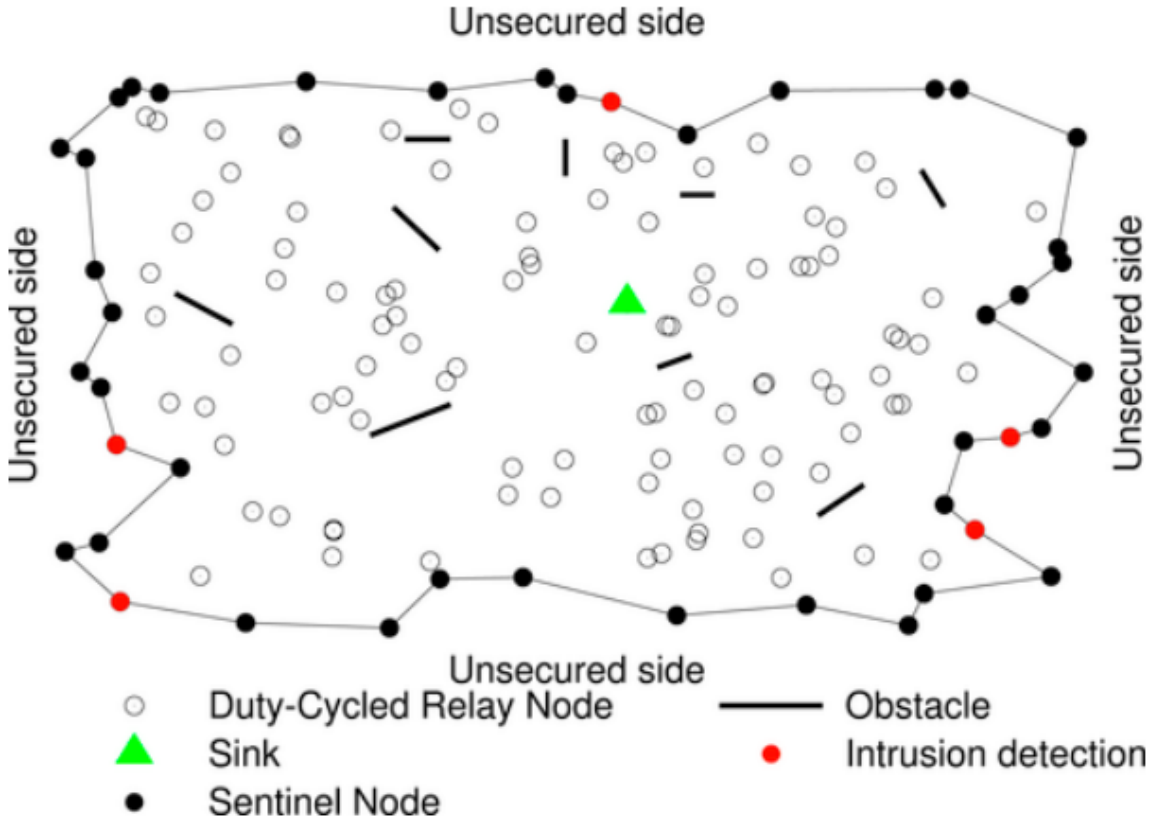

Figure 1.4.2: WSN based surveillance model

1.5 Conclusion

In summary, node deployment is a critical aspect of WSN design, influencing network performance and resource utilization. Random deployment offers simplicity and broad coverage in emergency situations, while deterministic deployment provides better control and optimization. By understanding the characteristics of both strategies and tailoring them to the application's needs, we can enhance the effectiveness and efficiency of Wireless Sensor Networks in various domains (Multi objective, exact approximte, heuristic metaheuristic).
In the upcoming chapter, we delve into the details of Meta-Heuristic based Multi-Objective Optimization for WSN deployment and explore its potential to improve network performance and resource utilization. Specifically, we investigate how metaheuristic algorithms can be employed to tackle the complex trade-offs involved in WSN design, ultimately leading to more effective and efficient deployments.

CHAPTER 2

META-HEURISTIC BASED MULTI-OBJECTIVE OPTIMIZATION FOR WSN DEPLOYMENT

2.1 Introduction

Multi-Objective Optimization (MOO) is a computational approach that aims to optimize multiple conflicting objectives simultaneously. In the context of WSNs, by employing MOO techniques, it becomes possible to balance objectives such as maximizing network coverage while minimizing energy consumption or ensuring reliable communication. MOO algorithms seek to identify a set of Pareto optimal solutions (Pareto optimality is a state in which no improvement can be made in one

objective without degrading another, named after the Italian polymath Vilfredo Pareto). Researchers continue to explore new MOO algorithms, protocols, and approaches to address the challenges faced by WSNs where the aim is to develop smarter and more autonomous WSNs that can optimize multiple objectives simultaneously, leading to improved performance and resource utilization [41] [33]. One emerging area of research that complements the MOO approach in WSNs is combinatorial optimization which will be covered in the next sections.

2.2 Combinatorial Optimisation

Combinatorial optimization is a branch of mathematical optimization that has applications in artificial intelligence, applied mathematics, software engineering, and many other domains. A main motivation is that thousands of real-life problems can be formulated as abstract combinatorial optimization problems and it involves identifying the best solution from a finite set of possible solutions for an objective function with a discrete domain and a vast configuration space. Although Combinatorial optimization is about solving optimization problems it doesn't give specific instructions on how to turn real-world problems into mathematical questions [16] [25]. Today, combinatorial optimization finds widespread application in the study of algorithms, and it holds particular relevance in our case for optimizing WSN nodes deployment.

2.3 Problems of Combinatorial Optimisation

2.3.1 Combinatorial problems

Combinatorial problems are encountered in many areas of computer science and various other disciplines that utilize computational methods. Well known combinatorial problems are cases such as planning, scheduling and resource allocation. The essence of these problems lies in identifying optimal groupings,

orderings, or assignments for a discrete, finite set of objects, all while abiding to specific conditions or constraints. Combinations of these solution components form the potential solutions of a combinatorial problem [13].

Decision problems

Many combinatorial problems can naturally be characterized as decision problems, where the solutions of a given instance are determined by a set of logical conditions [13]

- The search variant is a problem-solving approach that involves finding a solution or determining the non-existence of a solution given a specific problem instance.
- The decision variant refers to a scenario where one needs to determine whether a solution exists for a given problem instance.

These variants are closely related because algorithms solving the search variant can always be used to solve the decision variant. Interestingly, this holds true for many combinatorial decision problems [13].

Combinatorial Optimization problems

Optimization problems can be viewed as generalizations of decision problems, where solutions are evaluated based on an objective function, and the goal is to find solutions with optimal objective function values. The objective function is often defined for both candidate solutions and solutions, and the value of the objective function for a given candidate solution (or solution) is also referred to as its solution quality [13]. For each combinatorial optimisation problem, we distinguish two variants:

- The search variant: given a problem instance, find a solution with minimal (or maximal, respectively) objective function value.

- The evaluation variant: given a problem instance, find the optimal objective function value (i.e., the solution quality of an optimal solution).

Many combinatorial optimisation problems are defined based on an objective function as well as on logical conditions (Constraint) [13].

Complexity issues

In theoretical computer science, the classification and complexity of common problem definitions have two major sets, $\mathcal{P}$ which is "Polynomial" time and $\mathcal{NP}$ which "Non-deterministic Polynomial" time. There are also $\mathcal{NP}$-Hard and $\mathcal{NP}$-Complete sets, which we use to express more sophisticated problems as depicted in Figure 2.3.1.

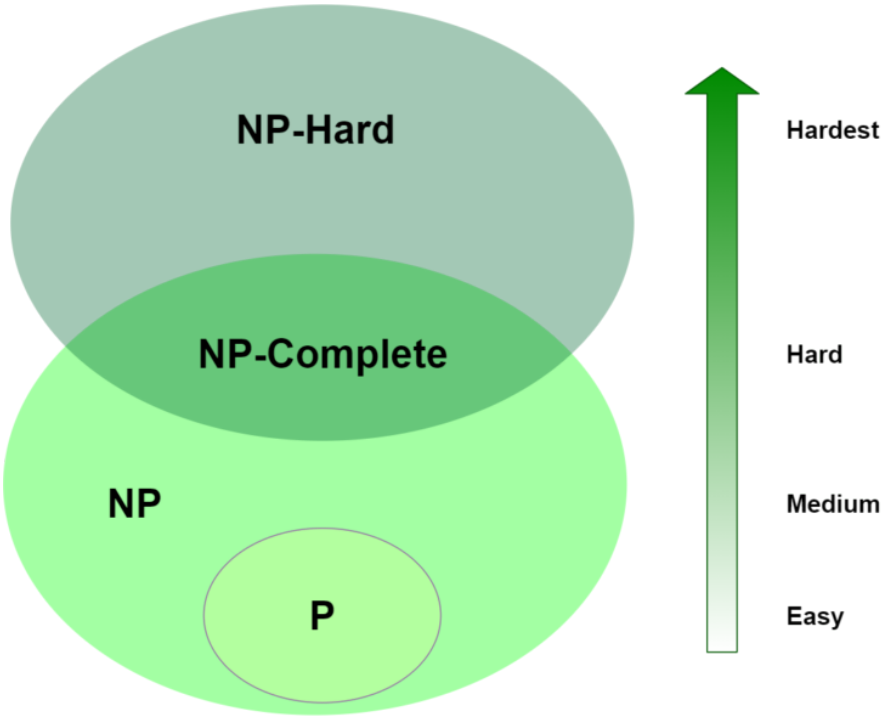

Figure 2.3.1: Complexity classes labeled from Easy to Hard scale

➢ $\mathcal{P}$**:** The class of problems that have polynomial-time deterministic algorithms. That is, they are solvable in O($\mathcal{P}$(n)), where $\mathcal{P}$(n) is a polynomial on n. A deterministic algorithm is essentially one that always computes the correct answer [28].

➢ $\mathcal{NP}$**:** The second set of problems cannot be solved in polynomial time. However, they can be verified (or certified) in polynomial time [2] For example, we'll see complexities like O(n^n), O(2^n), O($2^{0.000001\times n}$) [28].

➢ $\mathcal{NP}$**-Complete:** This set is very similar to the previous set. Taking a look at the diagram, all of these all belong to $\mathcal{NP}$, but are among the hardest in the set. What makes them different from other $\mathcal{NP}$ problems is a useful distinction called completeness. For any $\mathcal{NP}$ problem that's complete, there exists a polynomial-time algorithm that can transform the problem into any other $\mathcal{NP}$-complete problem (In other words, an $\mathcal{NP}$-complete problem is an $\mathcal{NP}$-hard problem that is also in $\mathcal{NP}$). This transformation requirement is also called reduction [2].

➢ $\mathcal{NP}$**-Hard:** The last set of problems, it contains the hardest, most complex problems in computer science. They are not only hard to solve but are hard to verify as well, These algorithms have a property similar to ones in $\mathcal{NP}$-Complete they can all be reduced to any problem in $\mathcal{NP}$. Because of that, these are in $\mathcal{NP}$-Hard and are at least as hard as any other problem in $\mathcal{NP}$. A problem can be both in $\mathcal{NP}$ and $\mathcal{NP}$-Hard, which is another aspect of being $\mathcal{NP}$-Complete [2].

The $\mathcal{P}$ versus $\mathcal{NP}$ Problem

The $\mathcal{P}$ versus $\mathcal{NP}$ problem was introduced independently in 1971 by Stephen Cook and Leonid Levin. Since that time, extensive efforts have been made to find a proof for this problem, but no definitive solution has been discovered thus far it asks whether $\mathcal{P}$ is equal to $\mathcal{NP}$ or not. In other words, it asks whether every problem for which a solution can be verified in polynomial time can also be solved in polynomial time [1] [39].

> If $\mathcal{P}$=$\mathcal{NP}$, we could find solutions to search problems as easily as checking whether those solutions are good. This would essentially solve all the

algorithmic challenges that we face today and computers could solve almost any task. However because there are problems for which no efficient algorithm exists, and finding a solution requires an exponential amount of time [1]. In this case, there would always be a gap between verifying a solution and finding it thus $\mathcal{P} \neq \mathcal{NP}$.

2.4 Techniques to resolve Combinatorial Optimisation Problems

To tackle the complexity of combinatorial optimization problems, various techniques have been developed to efficiently search through the vast solution space and find the desired high-quality solutions. These techniques leverage different approaches, ranging from exact methods that guarantee optimality to heuristic methods that quickly find good solutions or metaheuristic methods that provide higher-level strategies that guide the search process, while approximation algorithms offer solutions with known bounds on their quality [36]. By applying these diverse techniques, researchers can effectively address combinatorial optimization problems and make significant strides in solving real-world challenges across multiple domains. Now, let's explore some of the most commonly employed techniques in more detail.

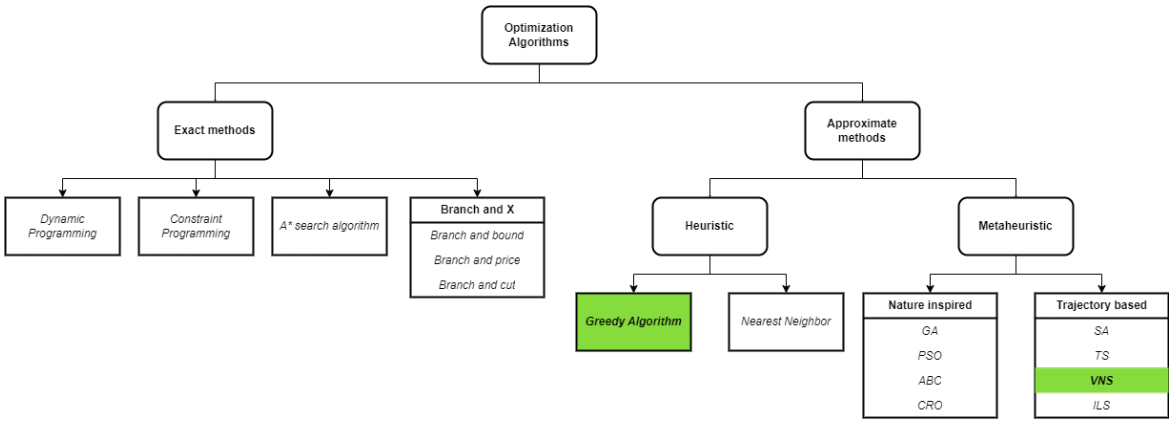

Figure 2.4.1: Optimization methods classification

2.4.1 Exact Methods

Exact method algorithms provide a precise approach to solving combinatorial optimization problems. This method guarantee finding the optimal solution by exhaustively search through all possible solutions or utilize mathematical techniques for every finite size instance of a COP within an instance dependent finite run time, or show that no possible solution exists [36].
Let's explore a few examples of exact method algorithms to better understand their applications.

Dynamic programming

Dynamic programming is an algorithmic approach for investigating an optimization problem by breaking it down into smaller, simpler sub-problems. A key aspect of dynamic programming is the proper structuring of optimization problems into multiple levels and solving them in a sequential manner, one level at a time. Each level is solved using typical optimization techniques, and the solution obtained helps to define the characteristics of the next level problem in the sequence. Typically, these levels correspond to distinct time periods within the overall problem [29].
0-1 Knapsack Problem example: a thief robbing a store finds n items, the ith item is worth v_i algerian dinars and weights w_i kilograms, where v_i and w_i are positive integers. He wants to take as valuable a load as possible, but he can carry at most W kilograms in his knapsack for some positive integer W. What items should be taken? [3]
Formally, the 0-1 knapsack problem can be stated as follws:

Given: n items of values v_1, v_2, . . . , v_n (positive integers) and of the weight w_1, w_2, . . . , w_n (positive integers), and a total weight W (positive integer).
Find: a subsect $\mathcal{S} \subseteq 1, 2, \ldots, n$ of the items such that:

$$\sum_{i \in \mathcal{S}} w_i \leq W \quad \text{and} \quad \sum_{i \in \mathcal{S}} v_i \quad \text{is maximized.}$$

In the Figure 2.4.2 below, we can see a visual example of the 0-1 Knapsack problem. The problem involves a thief who needs to choose the best combination of items from a selection of four different candies, each with different values and weights.

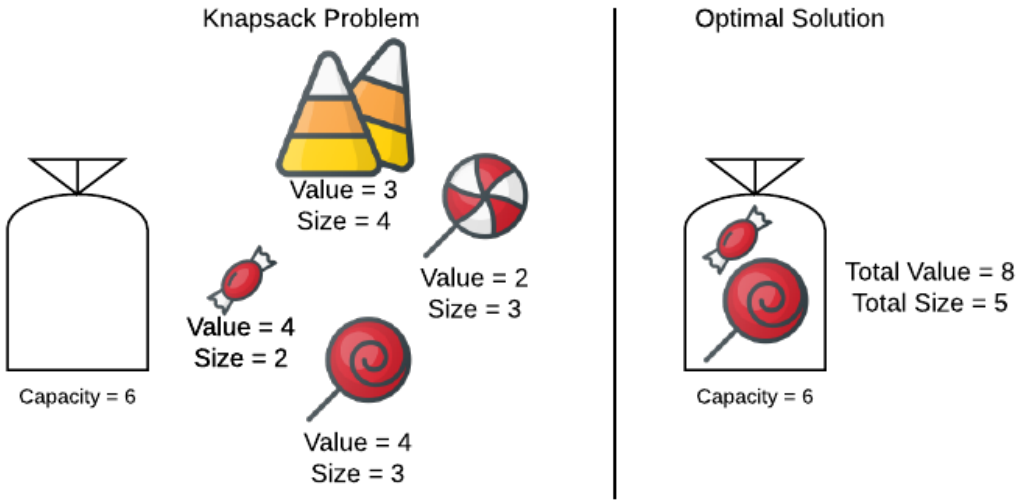

Figure 2.4.2: Example of 0-1 Knapsack Problem

This is called the 0-1 knapsack problem because each item must either be taken or left behind the thief cannot take a fractional amount of an item or take an item more than once [3].

Constraint Programming

Constraint Programming (CP) is a powerful paradigm for solving combinatorial search problems that imposes restrictions on the possible solutions and reduce the search space of a problem [20]. CP utilizes a mathematical/logical modeling language to encode the formulation (translating the problem into a formal representation that a computer can understand), enabling users to employ diverse search strategies. These strategies can be adjusted to locate solutions effectively, making CP highly flexible in terms of formulation power and solution approach but requires skill in

declarative-style logic programming and in developing good search strategies [17].
The Figure 2.4.3 below shows the different types of constraint programming:

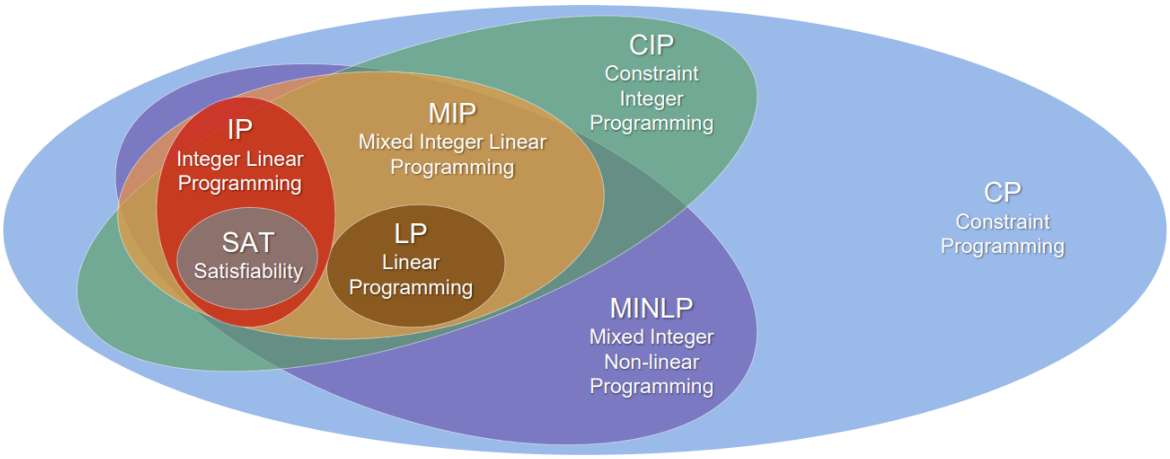

Figure 2.4.3: Constraint programming optimization methods

There are many different types of constraint programming, and the ones we're interested in are those that are used for optimization purposes such as Linear Programming (also called Linear Optimization). In this type of constraint programming, the goal is to to solve problems where we want to make the most or the least of something, while following certain constraints. These constraints may be equalities or inequalities. The optimization problems in linear programming involve calculating profits and losses. Linear programming is a useful tool for finding a feasible region to get the optimal solution within a given set of conditions, aiming for the highest or lowest value of the thing we want to optimize.
In other words, linear programming helps us find the best outcome by maximizing or minimizing the objective function for a given mathematical model, while taking into account certain requirements that are expressed in terms of linear relationships. The main goal of linear programming is to find the best possible solution (it is possible to get either multiple optimal solutions or no solution) [26] [24].

Element	**Characteristics**	**Examlpes**	
		Acceptable	**NOT acceptable**
Objective function	Linear	*minimize (2x+5y)*	*minimize (2xy+5y)*
		maximize (6x+7y+8z)	*maximize ($2x^2$+log(y))*
Constraints	Linear	*2x+5y ≤ 7*	*2xy +5y ≤ 7*
	Inequalities must be of the form (≥ or ≤). No strict inequalities (>or <) are allowed	*2x+5y ≤ 7*	*2x+5y <7*
Decision variables	Continuous	*x ∈ [0,1] where x can take any real value ranging from 0 to 1 (The variable value must alway be ≥0)*	*x ∈ (0,1) where x must be either 0 or 1*

Table 2.1: Characteristics of a linear program

An example of a production problem with telephones: A telephone company produces and sells two kinds of telephones, namely desk phones and cellular phones. Each type of phone is assembled and painted by the company. The objective is to maximize profit, and the company has to produce at least 100 of each type of phone. There are limits in terms of the company's production capacity, and the company has to calculate the optimal number of each type of phone to produce, while not exceeding the capacity of the plant [24].

A possible **descriptive model** of the telephone production problem is as follows:

➢ **Objective:** Maximize profit.

➢ **Desision varibles:**

- Number of desk phones produced (DeskProduction).
- Number of cellular phones produced (CellProduction).

➢ **Constraints:**

1. The DeskProduction should be greater than or equal to 100.
2. The CellProduction should be greater than or equal to 100.
3. The assembly time for DeskProduction plus the assembly time for CellProduction should not exceed 400 hours.
4. The painting time for DeskProduction plus the painting time for CellProduction should not exceed 490 hours.

Now for the **mathematical model** of the telephone production problem, the results are as follows:

Maximize: *12 desk_production+20 cell_production*

Subject to:

$$desk\_production \geq 100$$

$$cell\_production \geq 100$$

$$0.2\ desk\_production\ +\ 0.4{+}cell\_production \leq 400$$

$$0.5\ desk\_production{+}0.4\ +\ cell\_production \leq 490$$

With the modeling phase complete, we can now delve into the results:

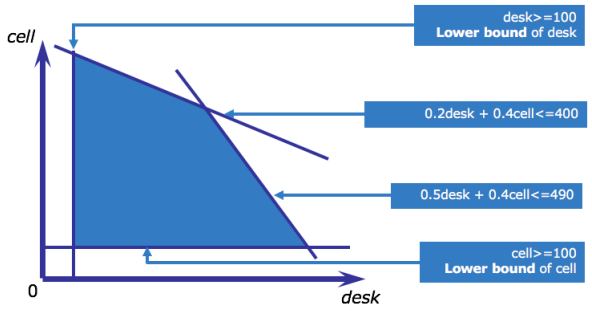

Figure 2.4.4: The feasible set of solutions

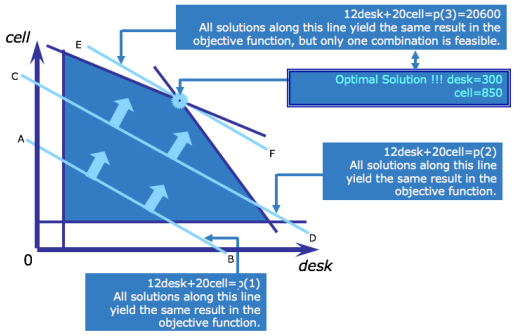

Figure 2.4.5: The optimal solution

A* search algorithm

The A* algorithm, pronounced "A star" is a search algorithm used to find the shortest path between an initial and final point. It is widely regarded as one of the

best and most popular techniques for pathfinding and graph traversal. Unlike other traversal techniques, A* is considered a smart algorithm due to its ability to significantly reduce the search space therefore saving time [30].
The idea behind A* algorithm is really simple [23]:

$f(n) = g(n) + h(n)$ where:
***g(n)*:** The cost of traversing from the start node to node n.
***h(n)*:** The cost to reach from node n to goal node.
***f(n)*:** Estimated cost of the cheapest solution.

8-puzzle problem example using A* algorithm: Find the most cost-effective path to reach the final state from initial state using A* Algorithm where $g(n) =$ Depth of node and $h(n) =$ Number of misplaced tiles [19].
The Figure 2.4.6 below shows the initial problem of 8-puzzle and the solution we should be having:

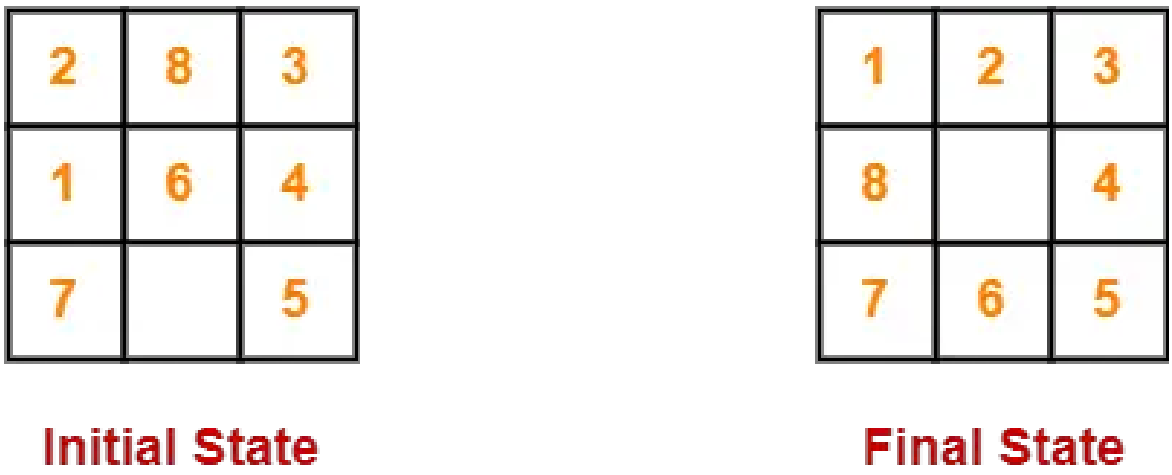

Figure 2.4.6: 8-Puzzle problem

The Figure 2.4.7 below represents the A* algorithm steps to reach an optimal solution:

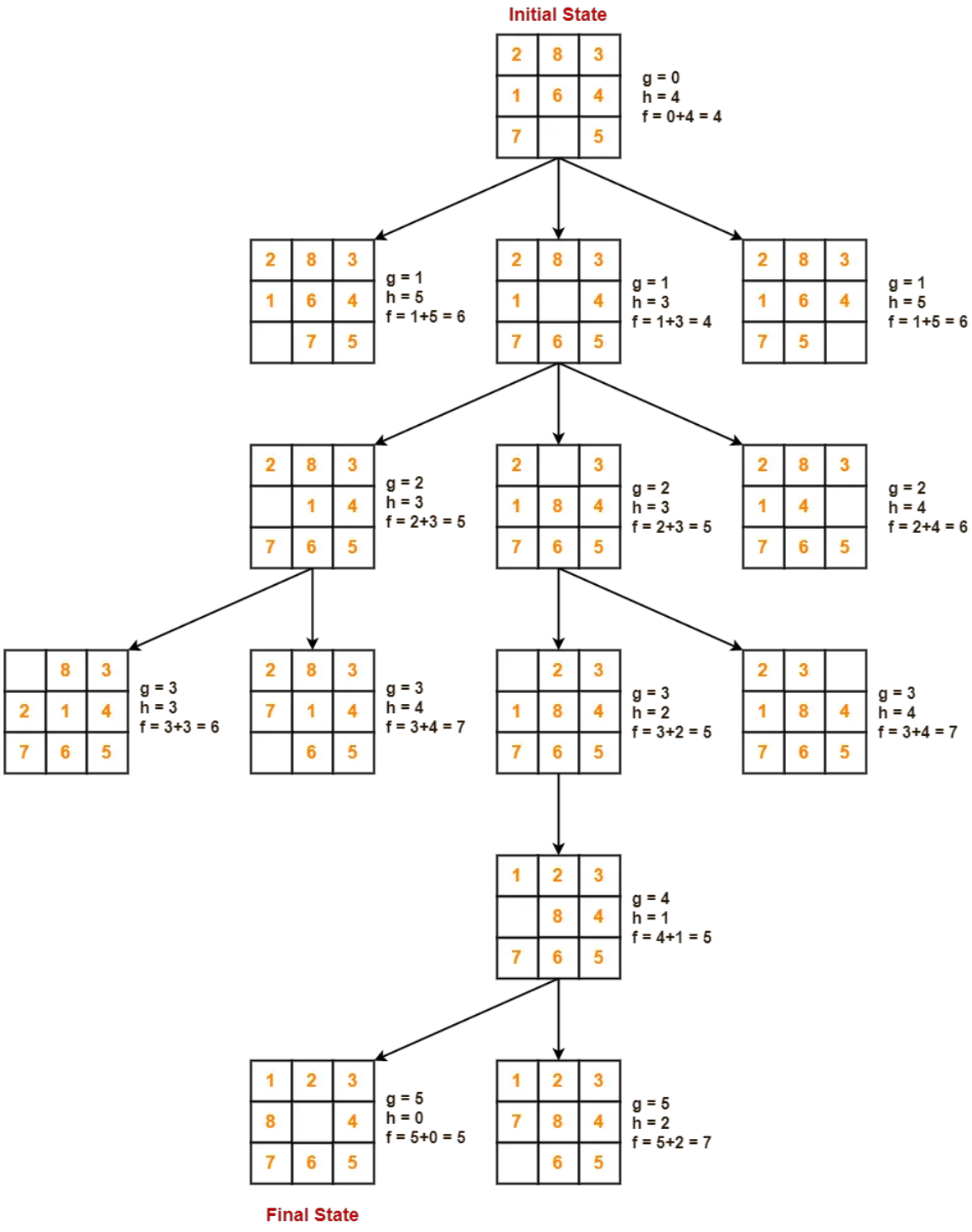

Figure 2.4.7: The solution to 8-Puzzle problem

Branch and bound

Branch and bound (BB) algorithms are utilized to find optimal solutions for combinatorial, discrete, and general mathematical optimization problems. Generally, when faced with an NP-Hard problem, a branch and bound algorithm systematically explores the entire search space of potential solutions and provides an optimal solution. There are also other similar algorithms, such as Branch and Price (a hybrid of branch and bound and column generation algorithms), which is used for solving problems with large solution spaces, and Branch and Cut (a hybrid of branch and bound and cutting planes algorithms), which is employed to solve mixed-integer programming problems [9].

A branch and bound algorithm consist of stepwise enumeration of possible candidate solutions by exploring the entire search space.

Job assignment problem using branch and bound example: Let there be N workers and N jobs. Any worker can be assigned to perform any job, incurring some cost that may vary depending on the work-job assignment. It is required to perform all jobs by assigning exactly one worker to each job and exactly one job to each agent in such a way that the total cost of the assignment is minimized [35].

The Figure 2.4.8 below shows an example of job assignment problem with a brief explanation:

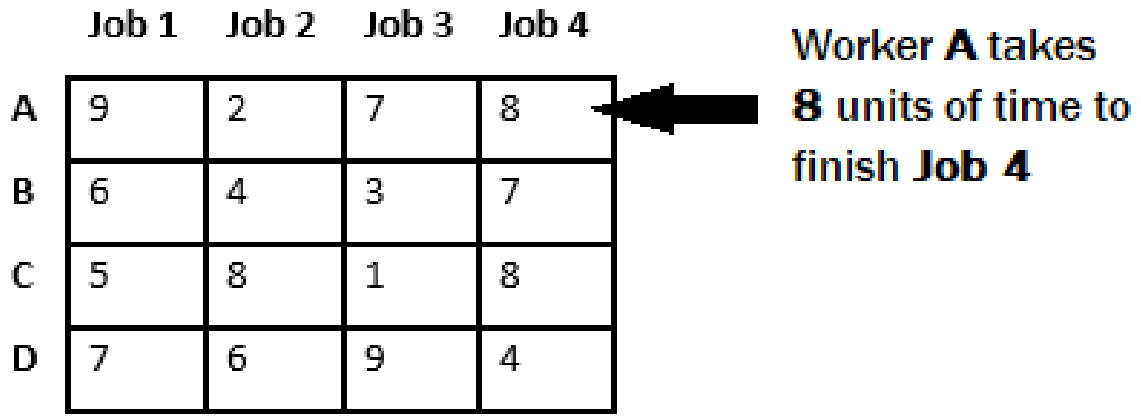

Figure 2.4.8: Job assignment problem

For each worker, we'll choose the job with minimum cost from the list of unassigned jobs, Let's take the Figure 2.4.9 below and try to calculate promising cost when Job 2 is assigned to worker A.

	Job 1	Job 2	Job 3	Job 4
A	9	2	7	8
B	6	4	3	7
C	5	8	1	8
D	7	6	9	4

Figure 2.4.9: Job assignment problem worker A

Now we assign job 3 to worker B as it has minimum cost from list of unassigned jobs. Cost becomes $2 + 3 = 5$ and Job 3 and worker B also becomes unavailable as shown in Figure 2.4.10.

	Job 1	Job 2	Job 3	Job 4
A	9	2	7	8
B	6	4	3	7
C	5	8	1	8
D	7	6	9	4

Figure 2.4.10: Job assignment problem worker B

Finally in Figure 2.4.11, job 1 gets assigned to worker C as it has minimum cost among unassigned jobs and job 4 gets assigned to worker C as it is only Job left. Total cost becomes $2 + 3 + 5 + 4 = 14$.

	Job 1	Job 2	Job 3	Job 4
A	9	2	7	8
B	6	4	3	7
C	5	8	1	8
D	7	6	9	4

Figure 2.4.11: Job assignment problem worker C,D

The below diagram (Figure 2.4.12) shows complete search space diagram showing optimal solution path in green.

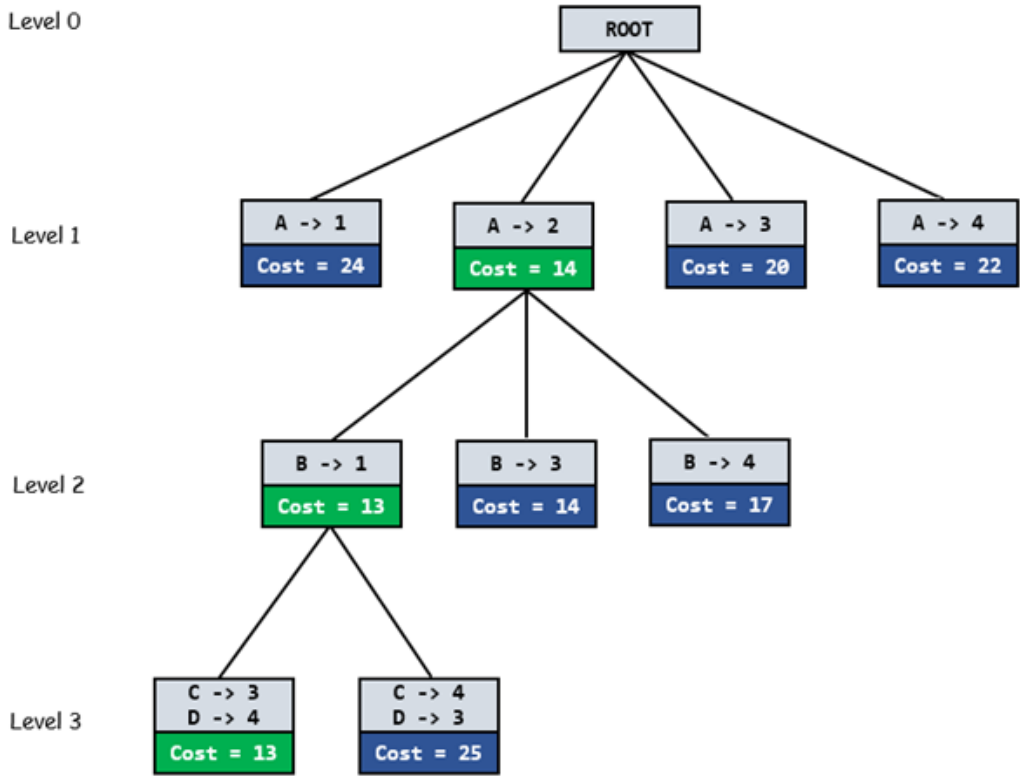

Figure 2.4.12: Branch and bound diagram of Job assignment problem

2.4.2 Approximate Methods

Most optimization problems, including those arising in important application areas, are $\mathcal{NP}$-hard. Therefore, under the widely believed conjecture that $\mathcal{P} \neq \mathcal{NP}$, their exact solution is prohibitively time consuming so unless $\mathcal{P} = \mathcal{NP}$, there are no efficient algorithms to find optimal solutions for such problems, because a wide class of optimization problems cannot be solved exactly in polynomial time where waiting for a solution isn't worth it, expensive or impossible because the search space is too large, Approximate algorithms came into existence that sacrifice results accuracy for time efficiency to get a solution that is close to optimal while reducing the computational resources required [38].
There are various options to choose from for solving NP-hard problems sub-optimally, and the ones we're interested in are heuristics and metaheuristics.

Heuristic algorithms

Heuristic algorithms were first introduced by G. Polya in 1945. They were proposed as a means to solve problems more quickly than traditional methods. Heuristics employ practical methods and shortcuts to generate solutions that may or may not be optimal, but are satisfactory within a limited time frame [11] [22].

Greedy algorithm: Greedy algorithm is a simple approach used to solve optimization problems. It is based on the principle of making locally optimal choices at each step with the hope that the overall solution will be globally optimal. The algorithm makes the best possible choice (greedy choice that looks best at the time) at each step without considering the larger context or looking ahead to future steps [7].

The Figure 2.4.13 below depicts a comparison of the results obtained from the Greedy algorithm heuristic and an optimal algorithm in finding the shortest path from node A to node J in a weighted graph:

Greedy Path	Greedy Weight	Optimal Path	Optimal Weight
A-C-G-I-J	3+3+12+9 = 27	A-B-E-H	4+5+2+1 = 12

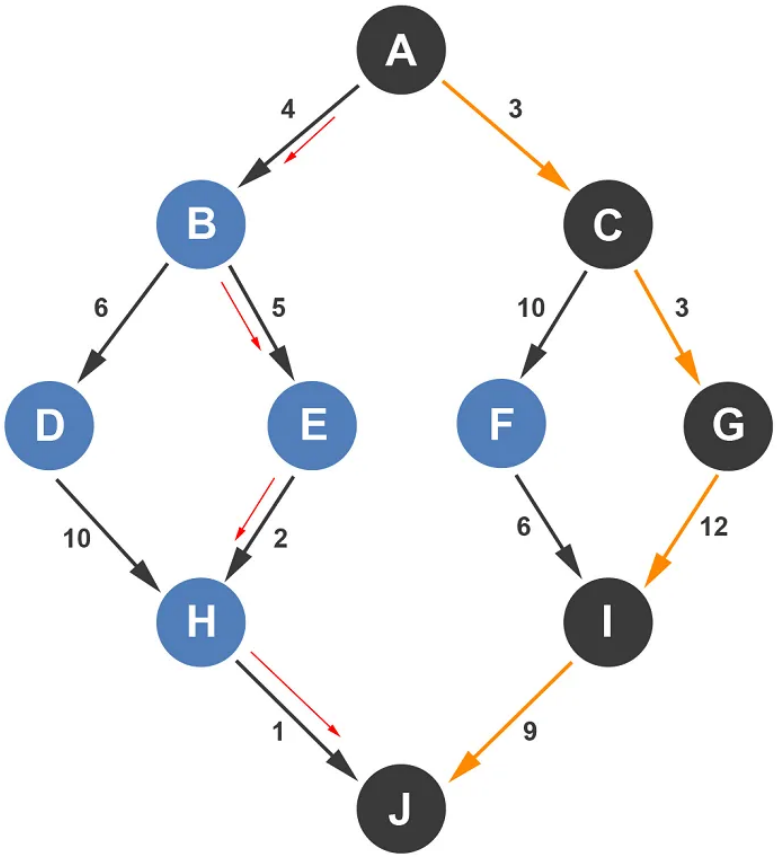

Figure 2.4.13: Comparison between Greedy and Exact algorithm

Metaheuristic

Metaheuristic algorithms are a class of optimization algorithms that are designed to solve complex optimization problems for which traditional heuristic algorithms may not be effective or efficient enough [18]. While heuristic algorithms typically use specific problem-solving techniques to find good solutions, metaheuristic algorithms

take a higher-level approach and provide a framework for guiding the search process [37].

Metaheuristic algorithms often mimic natural or social phenomena to guide the search process. By leveraging these concepts, metaheuristic algorithms explore the solution space in an intelligent and adaptive manner, aiming to find near-optimal solutions within a reasonable amount of time [37].

One of the key advantages of metaheuristic algorithms is their ability to handle complex problems with large search spaces, non-linear constraints, and multiple conflicting objectives [37].

Some popular examples of metaheuristic algorithms include Simulated Annealing (SA) and Variable Neighborhood Search (VNS), which are single-solution metaheuristics that aim to improve a single candidate solution. On the other hand, Genetic Algorithms (GA) and Particle Swarm Optimization (PSO) are population-based metaheuristics that strive to improve multiple candidate solutions. These algorithms have been successfully applied to a wide range of optimization problems in various fields such as engineering, logistics, finance, and scheduling.

Search Space: Before we proceed any further, it is important to have a clear understanding of some terms. In optimization, the goal is often to find the global minimum or maximum of a function, which represents the best or worst possible solution to the problem being optimized, depending on the context of the solution. The most favorable value of a solution is referred to as the optimum. However, optimization algorithms may occasionally return a local optimum, which is a point in the search space where the objective function has the highest or lowest value among neighboring points, but it may not be the optimum value in the entire search space [21].

The Figure 2.4.14 below is an example of solutions made from a metaheuristic algorithm depicted in a graph:

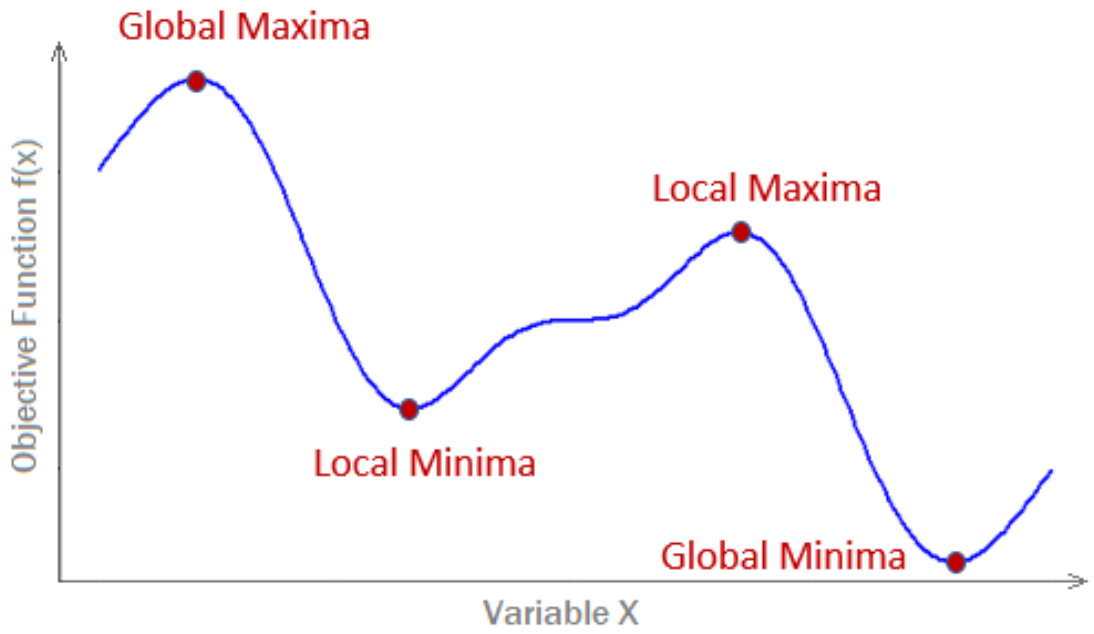

Figure 2.4.14: A visualisation of Global and Local optima

Variable Neighborhood Search (VNS) algorithm: VNS is a metaheuristic algorithm used for solving optimization problems. It was proposed by Mladenović and Hansen in 1997 as an extension of the Local Search method [15].

The Variable Neighborhood Search algorithm is designed to explore different neighborhoods around a given solution to improve its quality. It operates by iteratively searching the solution space using a combination of local search and perturbation techniques. It explores a set of neighborhoods either at random or systematically to escape from local optima [37] [15].

There are various types of VNS algorithms proposed by Mladenović and Hansen, including Variable Neighborhood Descent (VND), General VNS (GVNS), and Skewed VNS (SVNS). Each of these algorithms has its unique characteristics. There is also Basic VNS variant, which is the type I have chosen for the practical implementation [12].

The following Figure 2.4.15 shows 2 different solutions one initial and the other VNS based:

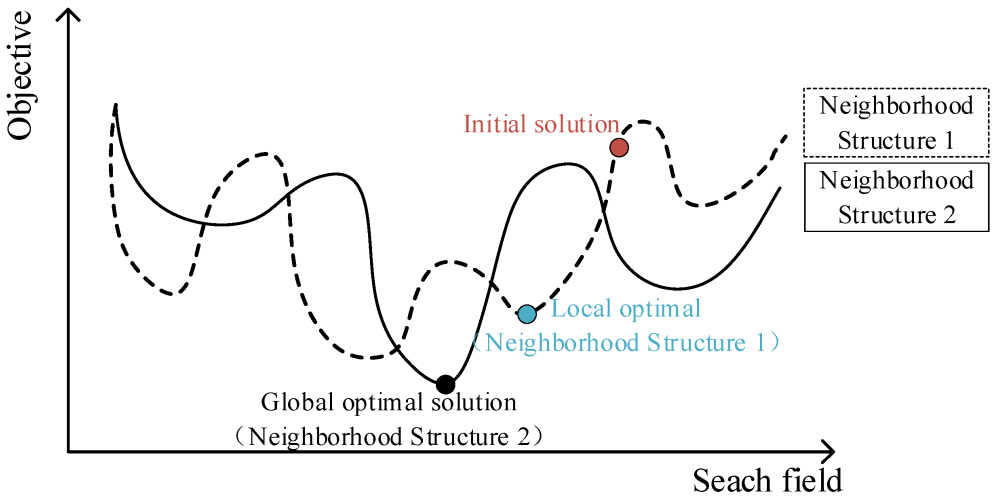

Figure 2.4.15: Two neighborhood structures of VNS comparaison

MOO methods

The goal of MOO is to find a set of optimal solutions when there are multiple conflicting objectives. In traditional single-objective optimization, there is only one objective to be optimized. However, many real-world problems involve multiple objectives that need to be considered simultaneously. In such cases, a user may need only one solution, regardless of whether the associated optimization problem is single-objective or multi-objective. When dealing with multi-objective optimization, the user faces a dilemma in choosing among the optimal solutions. Ideally, in multi-objective optimization, efforts should be made to find a set of trade-off optimal solutions by considering all objectives as important. Once a set of such trade-off

solutions is found, the user can then use higher-level qualitative considerations to make a choice [33].

Weighted sum method : The weighted sum method is a technique that combines multiple objectives into a single objective by assigning weights to each objective. It is a straightforward approach and commonly used in classical optimization methods. When dealing with multiple objectives, the weighted sum method is often the first choice due to its simplicity and widespread application [40].

The weighted sum method combines all the multi-objective functions into one scalar, composite objective function using the weighted sum:
$F(x) = w_1 f_1(x) + w_2 f_2(x) + ... + w_M f_M(x)$ where $\sum_{i=1}^{M} w_i = 1, w_i \in (0, 1)$ [40]
For example, if we have two objective sets: one for minimizing the RN deployment and the other for restricting the hop count, let's say we prioritize minimizing the RN deployment over the hops constraint. The multiobjective function would be written as follows:
$F(x) = 0.7 * total\ RN(x) + 0.3 * hop\ count$ now the Multiobjective function will favor minimizing RNs over hops because we gave it a weight of 0.7 which is bigger than 0.3 and $\sum_{i=1}^{2} w_i = 1$

ϵ-Constraint Method: The method selects one main objective to maximize and treats the other objectives as restrictions. The restrictions are defined using a tolerance parameter, ϵ, which represents the acceptable deviation from the optimal value of each objective.The ϵ-Constraint Method produces a set of solutions by solving the optimization problem for the main objective multiple times, each time using a different ϵ value. By adjusting ϵ, different levels of compromise between the objectives can be attained. A smaller ϵ indicates a stronger preference for the corresponding objective, while a larger ϵ value allows for more flexibility and compromises [40].

2.5 Conclusion

In conclusion, this chapter delved into the fascinating field of Multi-objective optimization, Combinatory optimization, and the various techniques used to solve these complex problems. Through an exploration of both exact and approximate methods, as well as discussed the role of metaheuristics and the VNS algorithm we have gained valuable insights into the challenges and advancements in this area of research. By understanding the characteristics, strengths, and limitations of these methods, researchers and practitioners can make informed decisions when selecting appropriate approaches for their specific problem domains. The next chapter will go deep into the practical application of these concepts, focusing on their relevance in real-world scenarios.

CHAPTER 3

IMPLEMENTATION AND PERFORMANCE RESULTS

3.1 Introduction

In this chapter, we explore the practical implementation of VNS algorithm and present the performance results obtained. The previous chapters have laid the foundation by discussing WSNs deployment, Multi-Objective optimization and metaheuristics. Now, it is time to showcase how these concepts were translated into a tangible solution and evaluate its effectiveness.

3.2 Implementation details

3.2.1 Hardware

The details below show the hardware components used to run VNS algorithm:

Processor: : Intel® Core™ i5-6300U 2.40Ghz
Memory (RAM): 8GB DDR4 2400 MHz
Storage: : SSD M.2 256Gb SSD
Operating System: : Windows 10 PRO version 22H2

3.2.2 Software

The details below show the software components used to run VNS algorithm:

Programming language: : Python version 3.11
Integrated Development Environment (IDE): Pycharm 2023

3.2.3 Algorithm employed

For the optimization of constrained relay node Deployment I have chosen BVNS algorithm (procedure depicted below) due to it's simplicity. The primary objective is to optimize the deployment of RNs while simultaneously minimizing the number of hops in the network.

Algorithm 1 Basic Variable Neighborhood Search

```
Generate initial solution S (in our case we performed Greedy algorithm)
while Stopping condition not met do
    k ← 1;
    while k ≤ k_max do
        S' ← Shake(S, k);                          ▷ Shaking or perturbation
        S" ← Local_Search(S', k);
        if S" is better than S then
            S ← S";                                ▷ Neighborhood change
            k ← 1;
        else
            k ← k + 1;
        end if
    end while
end while
```

3.3 Experimental parameters

3.3.1 The surveyed fenced area scenario

We are going to apply the Basic Variable Neighborhood Search algorithm to a two-dimensional grid composed of sensor (sentinel) nodes that monitor the fenced area along the entire border of the grid. The grid is initially generated by a greedy algorithm including RNs that forward alert messages to a sink located in the middle of the area. The Figure 3.3.1 below illustrates an example of the scenario being discussed where sink is in red color, sentinels in orange and relays in black.

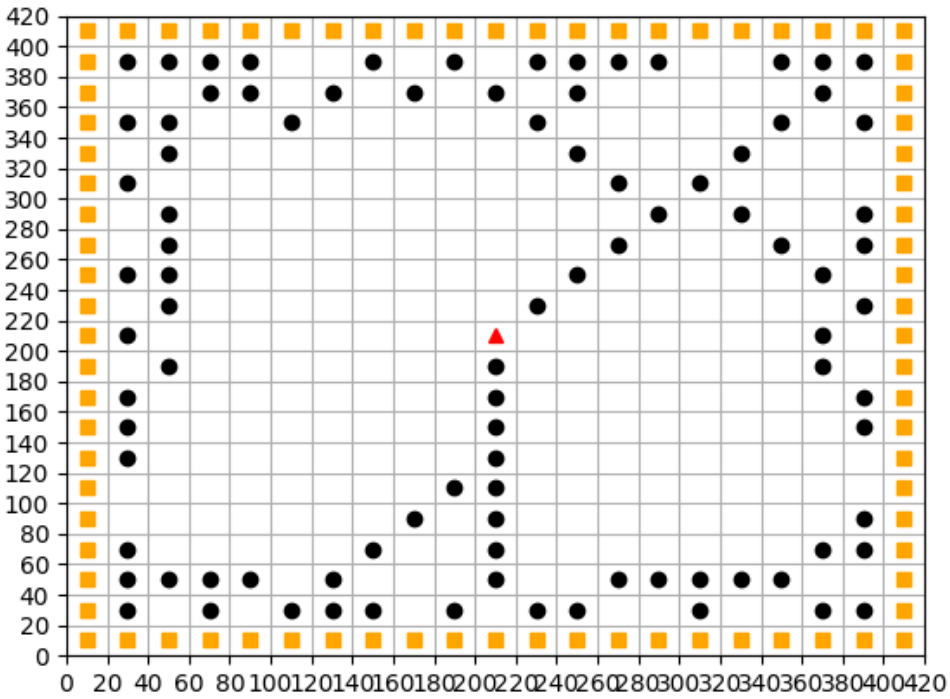

Figure 3.3.1: Example of a surveilled fenced area

The BVNS algorithm focuses on enhancing the performance of a solution space by incorporating various operations such as adding, modifying, or removing elements. Two important functions, perturbation and local search, leverage these operations to navigate away from local optima and explore the adjacent neighborhood structures in search of improved solutions.

3.3.2 Experimental setup parameters

Parameter	**Values**				
Grids	*15x15*	*20x20*	*30x30*	*40x40*	*50x50*
Single mesh size	*20m*				
Surface area	*90 KM²*	*160 KM²*	*360 KM²*	*640 KM²*	*1000 KM²*
RN sensing range	*30m*				
Sink, Sentinels and RNs communication range					
Scenarios	*5*				
Number of iterations per scenario	*10*				
Algorithm used for the initial solution	*Greedy algorithm*				

Table 3.1: Parameter values used before practical application of BVNS

3.3.3 Performance measures

Fitness: The overall fitness (or objective function) of the results will be calculated using weighted sum MOO method: *0.7*total relays + 0.3*average hops count*

Average total relays: total relays of each scenario/total of scenarios (which is 5 in our case)

Average fitness: fitness value of each scenario/5

Average of average hops : average hops of each scenario/5

Average time : time elapsed for each scenario/5

3.4 Performance Results

After running the algorithm for an extended period, I have obtained the following statistics:

Grids	Average time spent	Average initial fitness	Average BVNS fitness	Avg initial total relays	Avg BVNS total relays	Avg of initial average hops	Avg of BVNS average hops
15x15	*33 seconds*	*185.02*	*183.52*	*58*	*58*	*8.6*	*8.51*
20x20	*2 minutes 33 seconds*	*358.56*	*357.78*	*92*	*91*	*12.89*	*12.91*
30x30	*28 minutes 04 seconds*	*791.06*	*789.5*	*159*	*158*	*19.53*	*19.51*
40x40	*02 hours 29 minutes 52 seconds*	*1445.3*	*1444.5*	*226*	*224*	*27.51*	*27.52*
50x50	*10 hours 16 minutes 25 seconds*	*2167.14*	*2165.86*	*314*	*310*	*33.12*	*33.14*

Table 3.2: Statistics table containing after executing 5 scenarios for each grid

By using a maximum of 10 iterations as stopping condition, we observed that BVNS still delivered encouraging outcomes for extremely expansive search scopes within reasonable time frames. The Figure 3.4.1 below is a bar chart comparing the average total relays and average of average hops of five different grids between the initial greedy solution and the final optimal BVNS solution:

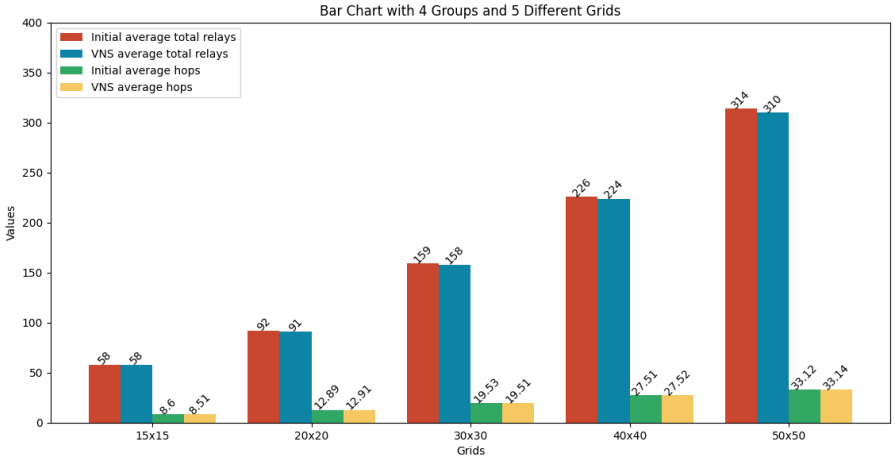

Figure 3.4.1: Relays and Hops Comparaison between Greedy algorithm and BVNS

In this other Figure 3.4.2 we are comparing the initial fitness value with BVNS for each grid:

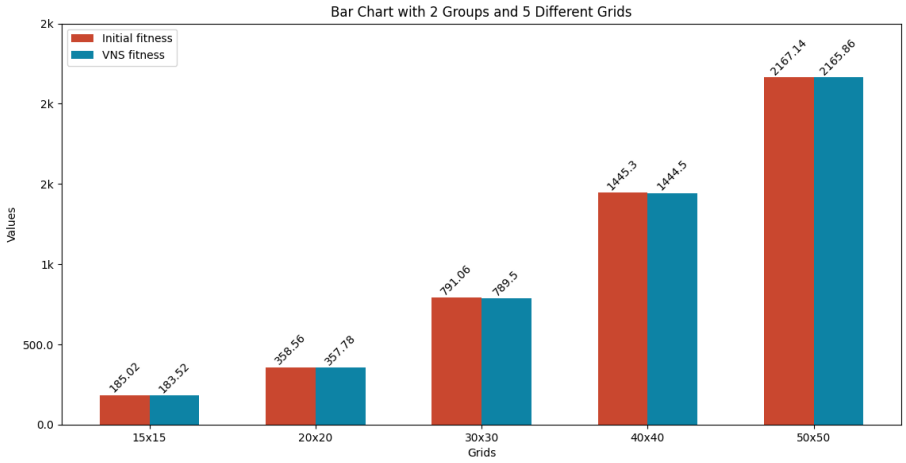

Figure 3.4.2: Initial and BVNS fitness comparaison

Finally, we're going to determine the execution time each grid took to obtain the final solutions in the Figure

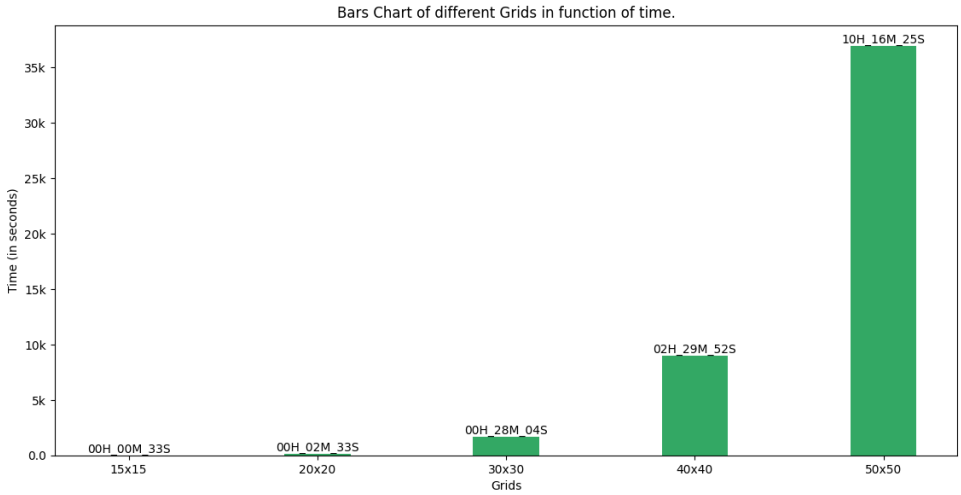

Figure 3.4.3: Total execution time of each grid

3.5 Conclusion

In this final chapter, we implemented the BVNS algorithm with the Greedy algorithm as the initial solution to optimize the deployment of RNs while minimizing the hop count in a surveyed fenced area. The objective was to generate and optimize relay nodes efficiently, ensuring optimal connectivity within the specified constraints. Based on the results obtained, it is evident that metaheuristics play a crucial role in the field of optimization. They provide indispensable approaches for tackling complex optimization problems and achieving close to optimal solutions.

GENERAL CONCLUSION

In this thesis, we have addressed the challenges associated with the deployment of WSNs for surveillance applications. The goal has been to optimize the number of relay nodes deployed and the hop count in the network, under coverage and connectivity constraints. We have tackled this problem using a meta-heuristic approach to be able to solve a big instances in reasonable time.

We have implemented the VNS meta-heuristic and the greedy algorithm used to find the initial solution of the problem by using python language. we also note that we have used the weighted sum method as a technique to solve the multi-objective problem.

Finally, we have conducted extensive experiments to evaluate the performance of our solution. The obtained results in terms of number of relay nodes deployed, hop count and latency are very encouraging compared to exact method.

As an immediate future work, we have planned to explore the Pareto front to try to enhance the solution. The comparison with a genetic based solution is also another possible future work.

Abstract:

Wireless Sensor Networks have emerged as a fascinating and rapidly growing area of research, capturing the attention of numerous researchers across a wide range of scientific disciplines. These networks, which consist of a collection of small, autonomous sensor and relay nodes capable of wirelessly communicating with each other, hold immense value and offer an array of limitless possibilities. In this thesis, our primary focus will revolve around the domain of surveillance.
Our objective is to build a two-tierd topology where sensor nodes are deployed on the sensitive site border to cover each point of we aim to minimize the number of relay nodes deployed while ensuring communications between the all sensor nodes and the sink node, in addition to that we strive to minimize the number of hops in the network.
The multi-objective combinatorial optimization problem described above has been resolved using Variable Neighborhood Search metaheuristic algorithm and the wighted sum approach. The obtained results are very encouraging since we can resolve a relative big instances of the problem compared to the exact method. on the other hand, the results in terms of optimal number of relay node to deploy and the average hop count are near optimal.

Keywords: *WSN*, *Meta-heuristic*, *Constrained Relay Node Deployment*, *Node*, *Combinatorial Optimization*, *VNS*, *Multi-Objective Optimization*, *Hop count*

Résumé:

Les réseaux de capteurs sans fil ont émergé en tant que domaine de recherche fascinant et en croissance rapide, attirant l'attention de nombreux chercheurs issus d'une grande variété de disciplines scientifiques. Ces réseaux, composés d'une collection de petits capteurs autonomes et de nœuds relais capables de communiquer sans fil entre eux, ont une valeur immense et offrent une multitude de possibilités illimitées. Dans cette thèse, notre principal objectif portera sur le domaine de la surveillance.
Notre objectif est de construire une topologie à deux niveaux où les capteurs sont déployés sur la frontière du site sensible afin de couvrir chaque point que nous visons, tout en minimisant le nombre de nœuds relais déployés et en assurant les communications entre tous les capteurs et le nœud central, en plus de cela, nous nous efforçons de minimiser le nombre de sauts dans le réseau.
Le problème d'optimisation combinatoire multi-objectif décrit ci-dessus a été résolu à l'aide de l'algorithme métaheuristique de recherche de voisinage variable et de l'approche de somme pondérée. Les résultats obtenus sont très encourageants, car nous pouvons résoudre des instances relativement importantes du problème par rapport à la méthode exacte. D'autre part, les résultats en termes du nombre optimal de nœuds relais à déployer et du nombre moyen de sauts sont proches de l'optimal.

Mots clés: *RCSF*, *Méta-heuristique*, *Déploiement contraint de nœuds relais*, *nœud*, *Optimisation Combinatoire*, *RVV*, *Optimisation Multi-Objectif*, *Nombre de sauts*

ملخص:

تكنولوجيا الشبكات اللاسلكية للمستشعرات قد ظهرت كمجال مثير للإعجاب ونمو سريع في البحث، وقد لفتت انتباه العديد من الباحثين في مجموعة واسعة من التخصصات العلمية. تتكون هذه الشبكات من مجموعة من الأجهزة الصغيرة والمستقلة للمستشعرات والمؤرضات قادرة على التواصل اللاسلكي بينها. وتحمل هذه الشبكات قيمة هائلة وتقدم مجموعة من الإمكانيات اللامحدودة. في هذه الرسالة، سيتم التركيز بشكل أساسي على مجال المراقبة.

هدفنا هو بناء توبولوجيا مكونة من طبقتين حيث يتم نشر أجهزة المستشعر على حدود الموقع الحساس لتغطية كل نقطة، نهدف إلى تقليل عدد الأجهزة المؤرضة المنتشرة مع ضمان التواصل بين جميع أجهزة المستشعر وجهاز التصريف، بالإضافة إلى أننا نسعى لتقليل عدد القفزات في الشبكة.

تم حل المشكلة المتعددة الأهداف للتحسين المركب باستخدام خوارزمية البحث المتغيرة المحلية ونهج الجمع الزائد. النتائج المحصل عليها مشجعة جدًا حيث يمكننا حل مشكلة ذات حجم كبير نسبيًا مقارنة بالطرق الدقيقة. من ناحية أخرى، فإن النتائج من حيث العدد المثلى للأجهزة المؤرضة المنبثقة وعدد القفزات المتوسطة تكاد تكون مثلى.

الكلمات الدالة: *الشبكات اللاسلكية للشبكات، الأدلة العليا، نشر عقدة الترحيل المقيدة، العقدة، الحلول المثلى للمسائل المعدودة، بحث متغير الحي، الامثلة متعددة الاهداف، عدد القفزات*

BIBLIOGRAPHY

[1] Shadia Ajam. What is the p vs. np problem? why is it important? https://science.nd.edu/news-and-media/news/what-is-the-p-vs-np-problem-and-why-is-it-important/, 2013. Accessed: 2023–06-11.

[2] Baeldung. P, np, np-complete and np-hard problems in computer science. https://www.baeldung.com/cs/p-np-np-complete-np-hard, 2022. Accessed: 2023–06-11.

[3] Zhaojun Bai. Ecs122a lecture notes on algorithm design and analysis. https://www.cs.ucdavis.edu/~bai/ECS122A/knapsack01.pdf, 2019. Accessed: 2023–06-12.

[4] A Balamurugan and T Purusothaman. Ipsd: New coverage preserving and connectivity maintenance scheme for improving lifetime of wireless sensor networks. *WSEAS Transactions on Communications*, 11(1):26–36, 2012.

[5] Abdulaziz Barnawi and Ahmed Bawazir. Multi-objective deployment of wireless sensor networks in 3-d environments using metaheuristics. 2023.

[6] Ali Benzerbadj, Bouabdellah Kechar, Ahcène Bounceur, and Mohammad Hammoudeh. Surveillance of sensitive fenced areas using duty-cycled wireless sensor networks with asymmetrical links. *Journal of Network and Computer Applications*, 112:41–52, 2018.

[7] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2022.

[8] Huseyin Cotuk, Bulent Tavli, Kemal Bicakci, and Mehmet Burak Akgun. The impact of bandwidth constraints on the energy consumption of wireless sensor networks. In *2014 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 2787–2792. IEEE, 2014.

[9] Subham Datta. Branch and bound algorithm. `https://www.baeldung.com/cs/branch-and-bound`, 2022. Accessed: 2023–06-12.

[10] Dina S Deif and Yasser Gadallah. Classification of wireless sensor networks deployment techniques. *IEEE Communications Surveys & Tutorials*, 16(2):834–855, 2013.

[11] Mihai Gavrilas. Heuristic and metaheuristic optimization techniques with application to power systems. In *Proceedings of the 12th WSEAS international conference on Mathematical methods and computational techniques in electrical engineering*, page 9, 2010.

[12] Vincenzo Guidi. Less is more approach (lima) in optimization. `https://slideplayer.com/slide/17584318/`, 2020. Accessed: 2023–06-11.

[13] Holger H Hoos and Thomas Stützle. *Stochastic local search: Foundations and applications*. Elsevier, 2004.

[14] Fei Hu and Xiaojun Cao. *Wireless sensor networks: principles and practice*. CRC press, 2010.

[15] Fotoglou Ioakeim. A reduced variable neighborhood search approach for the traveling thief problem. `https://dspace.lib.uom.gr/bitstream/2159/24610/3/FotoglouIoakeimMsc2019.pdf`, 2019. Accessed: 2023–06-11.

[16] Bernhard H Korte, Jens Vygen, B Korte, and J Vygen. *Combinatorial optimization*, volume 1. Springer, 2011.

[17] Joseph YT Leung. *Handbook of scheduling: algorithms, models, and performance analysis*. CRC press, 2004.

[18] Sean Luke. *Essentials of metaheuristics*. 2009.

[19] Not mentioned. A* algorithm | a* algorithm example in ai. `https://www.gatevidyalay.com/a-algorithm-a-algorithm-example-in-ai/`. Accessed: 2023–06-12.

[20] Not mentioned. Constraint programming. `https://www.aiforanyone.org/glossary/constraint-programming`. Accessed: 2023–06-12.

[21] Not mentioned. Global vs. local optimization using ga. `https://www.mathworks.com/help/gads/example-global-vs-local-minima-with-ga.html`. Accessed: 2023–06-11.

[22] Not mentioned. Heuristic techniques. `https://www.javatpoint.com/heuristic-techniques`. Accessed: 2023–06-12.

[23] Not mentioned. Informed search algorithms. https://www.javatpoint.com/ai-informed-search-algorithms. Accessed: 2023–06-12.

[24] Not mentioned. Tutorial: Linear programming, (cplex part 1). https://ibmdecisionoptimization.github.io/tutorials/html/Linear_Programming.html. Accessed: 2023–06-12.

[25] Not mentioned. What is combinatorial optimization? https://www.engati.com/glossary/combinatorial-optimization. Accessed: 2023–06-11.

[26] Not mentioned. Linear programming. https://byjus.com/maths/linear-programming/#:~:text=Linear%20programming%20(LP)%20or%20Linear,calculation%20of%20profit%20and%20loss., 2021. Accessed: 2023–06-12.

[27] S Mini, Siba K Udgata, and Samrat L Sabat. Sensor deployment and scheduling for target coverage problem in wireless sensor networks. *IEEE sensors journal*, 14(3):636–644, 2013.

[28] Shaik Naseera. P, np, np-hard and np-complete problems. https://jntua.ac.in/gate-online-classes/registration/downloads/material/a159262902029.pdf. Accessed: 2023–06-11.

[29] Sukanta Nayak. *Fundamentals of optimization techniques with algorithms*. Academic Press, 2020.

[30] Ravikiran A S. A* algorithm concepts and implementation. https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/a-star-algorithm#what_is_an_a_algorithm, 2023. Accessed: 2023–06-12.

[31] Mustapha Reda Senouci, Mohamed El Yazid Boudaren, Mohamed Abdelkrim Senouci, and Abdelhamid Mellouk. A smart methodology for deterministic deployment of wireless sensor networks. In *2014 International Conference on Smart Communications in Network Technologies (SaCoNeT)*, pages 1–6. IEEE, 2014.

[32] Chin-Shiuh Shieh, Trong-The Nguyen, Hung-Yu Wang, and Thi-Kien Dao. Enhanced diversity herds grey wolf optimizer for optimal area coverage in wireless sensor networks. In *Genetic and Evolutionary Computing: Proceedings of the Tenth International Conference on Genetic and Evolutionary Computing, November 7-9, 2016 Fuzhou City, Fujian Province, China 10*, pages 174–182. Springer, 2017.

[33] Omkar Singh, Vinay Rishiwal, Rashmi Chaudhry, and Mano Yadav. Multi-objective optimization in wsn: Opportunities and challenges. *Wireless Personal Communications*, 121:127–152, 2021.

[34] Rajeev Singh and Matendra Singh Manu. An energy efficient grid based static node deployment strategy for wireless sensor networks. *International Journal of Electronics and Information Engineering*, 7(1):32–40, 2017.

[35] factworx4i2 sumitgumber28, ruhelaa48. Job assignment problem using branch and bound. `https://www.geeksforgeeks.org/job-assignment-problem-using-branch-and-bound/`, 2023. Accessed: 2023–06-12.

[36] Hesamoddin Tahami and Hengameh Fakhravar. A literature review on combining heuristics and exact algorithms in combinatorial optimization. *European Journal of Information Technologies and Computer Science*, 2(2):6–12, 2022.

[37] El-Ghazali Talbi. *Metaheuristics: from design to implementation.* John Wiley & Sons, 2009.

[38] Vijay V Vazirani. *Approximation algorithms*, volume 1. Springer, 2001.

[39] Frank Vega. P versus np. `https://hal.science/hal-01343812/document`, 2018. Accessed: 2023–06-11.

[40] Xin-She Yang. Nature-inspired optimization algorithms: Challenges and open problems. *Journal of Computational Science*, 46:101104, 2020.

[41] Selim Yilmaz and Sevil Sen. Metaheuristic approaches for solving multiobjective optimization problems. In *Comprehensive Metaheuristics*, pages 21–48. Elsevier, 2023.