République Algérienne Démocratique et Populaire
الجمهورية الجزائرية الديمقراطية الشعبية
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
وزارة التعليم العالي و البحث العلمي

Djilali Bounaama Khemis Miliana University
جامعة الجلالي بونعامه خميس مليانة

Faculty of Science and Technology Department of Mathematics and Computer Science

Thesis Presented
for obtaining a Master's degree
of Computer Science

Option: Software Engineering and Distributed Systems

Automatic Design of Metaheuristics for the Optimization of Operating Room Scheduling Problem

Realized by:
Tawfiq Tabouche
Ibrahim Kridi

In front of the jury members:

name	President
name	Examiner
Dr. Imène AIT ABDRRAHIM.	Supervisor

Academic Year: 2022 / 2023

Abstract

The Operating Room Scheduling Problem (ORSP) is a critical problem in healthcare facilities that aims to optimize the allocation of operating rooms and surgical procedures in a defined order to improve efficiency and resource utilization. In this study, we propose Ant Colony Optimization (ACO) approach and a semi-automated design strategy for ACO to solve the ORSP problem. ACO is a metaheuristic algorithm that is known for its ability to effectively explore large solution spaces and find near-optimal solutions for combinatorial optimization problems, however, this method is also known to be very sensitive to its parameters setting that influences on the performance of the algorithm and quality of the solution, therefore, we suggest to use the semi-automated design strategy to overcome the manual tuning of the ACO algorithm. The proposed approaches were evaluated on three different objective models for the ORSP, where one objective is to maximize the number of operating rooms scheduled, a second objective is to minimize the total tardiness of operations scheduled and the last objective is a bi-objective that combines both previous objectives mentioned previously (Min and Max).In this work, we tackled a static case of ORSP where the data are known in advance, and no changes can happen in the middle of the processing of the data. The obtained results of both approaches ACO and semi-automated ACO were very promising, however, the experiments show that the semi-automated design strategy for ACO outperformed the ACO approach. Finally, the findings of this research contribute to the development of efficient and effective scheduling strategies for operating rooms, ultimately enhancing the overall performance and patient care in healthcare organizations.

Keywords: Operating Room Scheduling Problem (ORSP), Combinatorial Optimization, Scheduling Problems, Ant Colony Optimization (ACO), Metaheristics, Semi-automatic Configuration.

Résumé

Le Problème de Programmation des Salles d'Opération (ORSP) est un problème critique dans les établissements de santé qui vise à optimiser l'allocation des salles d'opération et des procédures chirurgicales selon un ordre défini afin d'améliorer l'efficacité et l'utilisation des ressources. Dans cette étude, nous proposons une approche basée sur l'Optimisation par Colonie de Fourmis (ACO) et une stratégie de conception semi-automatisée pour l'ACO afin de résoudre l'ORSP. L'ACO est un algorithme métaheuristique réputé pour sa capacité à explorer efficacement de vastes espaces de solutions et à trouver des solutions quasi-optimales pour les problèmes d'optimisation combinatoire. Cependant, cette méthode est également connue pour être très sensible à la configuration de ses paramètres, ce qui influence les performances de l'algorithme et la qualité de la solution. Par conséquent, nous suggérons d'utiliser la stratégie de conception semi-automatisée pour pallier l'ajustement manuel de l'algorithme ACO. Les approches proposées ont été évaluées selon trois modèles d'objectifs différents pour l'ORSP. Le premier objectif consiste à maximiser le nombre de salles d'opération programmées, le deuxième objectif vise à minimiser le retard total des opérations programmées, et le dernier objectif est bi-objectif, combinant les deux objectifs précédemment mentionnés. Les résultats obtenus avec les approches ACO et ACO semi-automatisé se sont révélés très prometteurs, cependant, les expériences ont montré que la stratégie de conception semi-automatisée pour l'ACO surpassait l'approche ACO. Enfin, les résultats de cette recherche contribuent au développement de stratégies de planification efficaces pour les salles d'opération, améliorant ainsi les performances globales et les soins aux patients au sein des organisations de santé.

Mots-clés : Problème de planification des salles d'opération (ORSP), Optimisation Combinatoire, Problèmes d'ordonnancement, Optimisation par colonie de fourmis (ACO), Metaheuristiques, configuration semi-automatique.

ملخص

مشكلة جدولة غرف العمليات (ORSP) هي مشكلة حرجة في مرافق الرعاية الصحية تهدف إلى تحسين تخصيص غرف العمليات والإجراءات الجراحية بترتيب محدد لتحسين الكفاءة واستخدام الموارد. في هذه الدراسة، نقترح استخدام نهج الأمثلة النمطية للنملة (ACO) واستراتيجية التصميم شبه الآلي لـ ACO لحل مشكلة ORSP. ACO هو خوارزمية الأدلة العليا المعروفة بقدرتها على استكشاف فعال لمساحات الحل الكبيرة والعثور على حلول قريبة من الأمثلة لمشكلات التحسين التكميلية، ومع ذلك، يُعرف أيضًا أن هذه الطريقة حساسة جدًا لإعداداتها المعلمة التي تؤثر على أداء الخوارزمية وجودة الحل، لذا، نقترح استخدام استراتيجية التصميم شبه الآلي للتغلب على الضبط اليدوي لإعدادات لخوارزمية ACO. تم تقييم النهج المقترح على ثلاثة نماذج مختلفة للأهداف في مشكلة ORSP، حيث يكون الهدف الأول هو تحديد أقصى عدد غرف للعمليات المجدولة، والهدف الثاني هو تقليل التأخير الكلي للعمليات المجدولة والهدف الأخير هو نموذج ثنائي الأهداف يجمع بين الهدفين المذكورين سابقا. أظهرت النتائج المستحصلة من كلا النهجين ACO و ACO شبه الآلي نتائج واعدة جدًا، ومع ذلك، أظهرت التجارب أن استراتيجية التصميم شبه الآلي لـ ACO تفوقت على نهج ACO. أخيرًا، تسهم نتائج هذا البحث في تطوير استراتيجيات جدولة فعالة وفعالة لغرف العمليات، مما يعزز الأداء العام ورعاية المرضى في المؤسسات الصحية.

الكلمات المفتاحية: مشكلة جدولة غرف العمليات (ORSP)، التحسين التكميلي، مشكلات الجدولة، أمثلة النملة للتحسين التكميلي (ACO)، أساليب البحث التكميلي، التكوين شبه الآلي.

Dedication

I dedicate this thesis to my loving parents, brother Ryad, sister Louiza, Merime, and Souhila. Their unwavering support and encouragement have been the driving force behind my academic journey. Their belief in my abilities and their sacrifices have been my constant inspiration.

I also dedicate this thesis to my dedicated supervisor, Dr. Imène Ait Abderrahim. Their guidance, expertise, and invaluable insights have shaped my research and personal growth. Their unwavering commitment to excellence has instilled in me a passion for knowledge and a drive for success.

In loving memory, I dedicate this thesis to my dear grandpa and aunt who constantly encouraged me in my studies. May they rest in peace.

Furthermore, I would like to express my heartfelt gratitude to my friends and colleagues for their friendship and unwavering encouragement throughout this journey

Toufik

Dedication

To my beloved parents, and my brother Amine

Your love and unwavering support have shaped me into who I am today. This thesis is a testament to your sacrifices and belief in me. I am forever grateful for your endless encouragement.

To my dear sister, Hiba,

Wishing you luck and success on your own journey. May you fearlessly chase your dreams and find joy in every step you take. Your presence in my life has been an inspiration.

To my loving wife,

During this tough journey, you have always believed in me and helped me. You have supported and understood me, even when things were hard. Your encouragement has given me the strength to face difficulties. I am grateful for you being there for me, and this project is as much yours as it is mine because it carries your love and support.

To my dedicated teacher Dr. Imène Ait Abderrahim,

Your guidance and mentorship have been invaluable throughout this entire experience. Your passion for knowledge and commitment to nurturing your students' growth have inspired me to reach new heights. Your wisdom, expertise, and constructive criticism have shaped my understanding and helped me refine my ideas. I am grateful for the opportunity to learn from you and for your unwavering belief in my potential.

To my colleague, Aymen,

Thank you for providing me with the space to work and supporting me throughout this process. Your assistance has been instrumental to my success.

To my friends and colleagues,

Your friendship and support have motivated and uplifted me. I cherish the memories we have shared and look forward to celebrating future accomplishments together.

To all those who have shaped my journey,

Ibrahim

Acknowledgement

Our heartfelt appreciation is extended first and foremost to Allah, the Almighty, for endowing us with unwavering strength and unwavering determination to accomplish this endeavor.

We would like to seize this invaluable opportunity to convey our profound gratitude and utmost respect to our supervisor, **Dr. Imène Ait Abderrahim**, whose invaluable guidance and perpetual motivation served as the catalyst that fueled our aspirations to bring this project to fruition.

We also would like to thank **Mr. Henning Witteborg** for his assistance and valuable advice throughout this work, and for providing the necessary information and resources for this project.

Our sincerest gratitude is extended to the esteemed members of the jury, whose gracious acceptance to evaluate this work has bestowed great honor upon us.

We are also deeply indebted to all our esteemed teachers, whose unwavering support and nurturing during our master's studies, and unwavering belief in our capabilities, have been invaluable.

To all those individuals who have contributed, whether directly or indirectly, from near or far, to the development and triumph of this work, we extend our heartfelt appreciation.

Table of content

List of Acronyms

ORSP: Operating Room Scheduling Problem.

ACO: Ant Colony Optimization.

SA-ACO : semi-automated

Tsp : traveling salesman problem

ACO-MO : Ant Colony Optimization with multi-objectives

TS: Tabu Search.

SA: Simulated Annealing.

GA: Genetic Algorithm.

List of Figures

List of Tables

General Introduction

The efficient scheduling of operations in the operating room for a hospital is vital for the smooth functioning of healthcare facilities which makes it a difficult task for humans to find the best schedule of operation without breaking many rules. From here, comes the Operation Room Scheduling Problem (ORSP) which involves the allocation of available resources, such as operating rooms, surgical teams, and equipment, to a set of surgical operations while considering various constraints and objectives. The ORSP is known to be a complex combinatorial optimization problem that requires intelligent and effective scheduling techniques to achieve optimal or near-optimal solutions.

In the field of combinatorial optimization, several algorithms and techniques have been developed to address similar problems. These include metaheuristic algorithms such as Tabu Search (TS), Genetic Algorithms(GA), Simulated Annealing(SA), and Ant Colony Optimization (ACO). Each algorithm has its own strengths and weaknesses, making it important to carefully select the most suitable method for solving a specific problem. metaheuristics have shown their efficiency in solving combinatorial optimization problems and their ability to solve complex computational problems, particularly those demanding the processing of vast datasets.

Among these methods, Ant Colony Optimization (ACO) has gained significant attention due to its ability to effectively tackle combinatorial optimization problems. ACO is inspired by the foraging behavior of ants, where individual ants deposit pheromone trails to communicate and collectively find good paths to food sources. This decentralized, probabilistic search mechanism allows ACO to efficiently explore the search space and find near-optimal solutions.

The choice of ACO for this thesis is based on several considerations. First, ACO has demonstrated impressive performance in various combinatorial optimization problems, including the Traveling Salesman Problem and the Vehicle Routing Problem, Flow Shop Scheduling Problem showcasing its versatility and effectiveness. Second, the nature of the ORSP, with its complex constraints, dynamic scheduling requirements, and need for real-time adaptability, aligns well with the decentralized and adaptive nature of the ACO algorithm. In this work, we focus on our proposed solution for a static ORSP problem where the data for the schedule are known in advance.

Through this thesis, we aim to leverage the strengths of ACO and exploit its potential for solving the ORSP effectively. By customizing and enhancing the ACO algorithm to accommodate the specific requirements and constraints of the ORSP, we anticipate that our proposed approach will offer improved scheduling outcomes compared to other methods.

The development of metaheuristic algorithms has traditionally relied on a manual and iterative process. However, the effectiveness of these algorithms is highly reliant on accurately tuning their configuration and parameters. This process can be time-consuming, prone to errors, challenging to replicate, and often explores only a limited range of design possibilities. In order to address these limitations and overcome the need for manual tuning, automatic algorithm configuration has emerged as a promising technique. Inspired by this idea we investigate a semi-automated ACO approach to improve the performance of the ACO algorithm in terms of solution quality. By automating the process of finding the optimal parameter settings, this approach has proven to be highly efficient in optimizing the performance of metaheuristic algorithms.

In summary, this thesis explores the utilization of Ant Colony Optimization as a powerful optimization technique for solving the Operation Room Scheduling Problem. The research conducted herein seeks to advance the current understanding of ORSP and provide practical insights and recommendations for healthcare professionals and administrators involved in operation room scheduling.

As a reminder, this manuscript is organized in 4 chapters:

In the first chapter, we will give a literature review and a description of the ORSP problem. The second chapter covers Ant Colony Optimization (ACO) method. Chapter three consists of the application of ACO algorithm for solving the ORSP problem. The last Chapter talks about the semi-automated ACO (SA-ACO) algorithm that we applied to the ORSP. Finally, we conclude our work with a general conclusion and an Appendix that contains extra details of the last experiments related to the SA-ACO.

Chapter 1

Literature Review and Problem Description

1.1 Introduction

The Operating Room Scheduling Problem (ORSP) is a combinatorial optimization problem that involves scheduling surgeries in an operating room, taking into account various constraints such as the availability of operating rooms, surgeons, and nurses, as well as the patient's medical condition. The objective is often to minimize the total time required for all surgeries or to maximize the utilization of available resources. However, this problem can be seen as well as a multi-objective optimization problem where the goal is to find the best optimal solution among many objectives [33].

In this chapter, we aim to provide a comprehensive exploration of scheduling problems, with a particular focus on the challenging case of operating room scheduling. We will delve into the intricacies of the operation room scheduling problem (ORSP), elucidating its key characteristics and inherent complexities. Moreover, we will delve into the mathematical formulation employed to model and optimize ORSP. Additionally, we will conduct a short review of various scheduling problems that exist in the literature. By delving into the classification, description, mathematical formulation, and review of scheduling problems, this chapter endeavors to offer a clearer understanding of the intricacies and advancements in this critical field.

1.2 Scheduling Problems

Scheduling problems are computational difficulties that involve allocating resources and ordering actions or tasks over a predetermined period of time. Numerous industries, including manufacturing, project management, transportation, healthcare, and telecommunications are affected by such problems. In order to achieve specified goals, such as lowering completion time, maximizing resource usage, cutting costs, or fulfilling deadline limitations, scheduling problems must be solved with the primary purpose of optimizing

the allocation and sequencing of limited resources [6][29].

In the literature, scheduling problems can be classified based on various criteria, such as characteristics of the problem instances or the specific domain they are applied to. The classification helps in understanding the different types of scheduling problems and their unique characteristics. One common classification is known as Pinedo's classification[27], divides scheduling types into two main categories: Work Scheduling and Service System Scheduling. Figure 1.1 provides a complete overview of the entire structure.

Overall, scheduling serves as a crucial tool for optimizing workforce management and resource allocation, enabling organizations to meet demands efficiently and maintain competitiveness in a dynamic global environment.

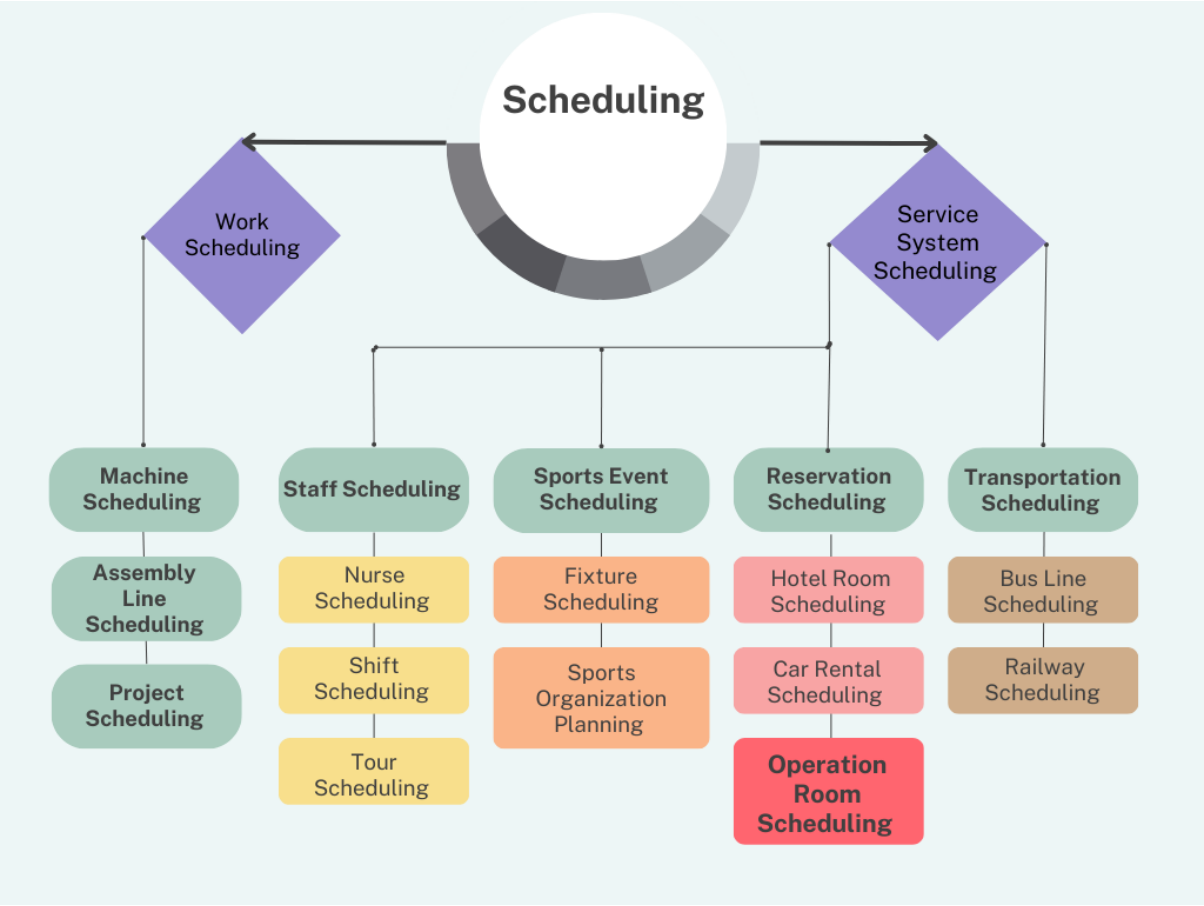


Figure 1.1: Scheduling categories [27]

1.3 Review of Scheduling Problems

The computational complexity of scheduling problems makes their solution difficult. Many scheduling problems fall under the category of NP-hardness, which denotes that, for

large problem instances, computing an optimal solution in a reasonable amount of time is computationally challenging. In order to solve scheduling problems, academics frequently use a variety of methodologies, including mathematical programming techniques, heuristic algorithms, metaheuristic methods, constraint programming, evolutionary algorithms, and simulation-based optimization.

Scheduling problems are a broad class of combinatorial optimization problems that involve allocating resources over time to perform tasks. They are common in many areas, including manufacturing, transportation, project management, and healthcare.

Some common examples of scheduling problems include:

- **Job Shop Scheduling (JSS)**: It represents a problem where a set of jobs must be processed on a set of machines. The objective is often to minimize the total processing time or to minimize the makespan (the time required to complete all jobs)[47].

- **Flow Shop Scheduling**: This problem consists on a set of jobs that must be processed on a set of machines in a specific order. The objective is often to minimize the makespan[22].

- **Resource-Constrained Project Scheduling**: This is a problem where a set of tasks must be scheduled to complete a project within a given set of constraints to be satisfied. The objective is often to minimize the completion time or to minimize the cost of the schedule[35].

- **Operating Room Scheduling**: It consists of a problem where surgeries/operations must be allocated and scheduled in an operating room, taking into account the availability of resources such as operating rooms, surgeons, and nurses, as well as the patient's medical condition[33].

- **Educational Timetabling Problem:** Educational Timetabling involves the allocation of teacher-student meetings to specific days, timeslots, and classrooms. While this task may seem simple, it becomes complex due to the unique rules, conventions, and specific requirements of each educational institution. As a result, the literature on Educational Timetabling has proposed various problem formulations tailored to different types of institutions (high school, university, etc.), meeting types (lectures, exams, etc.), and specific settings, constraints, and objectives.[11].

1.4 Classification of Scheduling Problems

Scheduling problems can be classified into several categories based on various criteria, such as the number of resources, the type of resources, the objective function, and the

constraints [43]. Some common classifications are:

- **Single vs. Multi-Objective:** Scheduling problems can be either single-objective, where the goal is to optimize one specific criterion (such as minimizing the completion time), or multi-objective, where multiple criteria must be considered and balanced (such as minimizing the completion time and maximizing the utilization of resources).
- **Deterministic vs. Stochastic:** Scheduling problems can also be either deterministic, where the processing times and resource availability are known in advance, or stochastic, where they are subject to uncertainty or variability.
- **Preemptive vs. Non-Preemptive:** Scheduling problems can be either preemptive, where a task can be interrupted and resumed later, or non-preemptive, where a task must be completed once it has started.
- **Offline vs. Online:** Scheduling problems can be either offline, where the entire problem is known in advance, or online, where the problem must be solved incrementally as new information becomes available before the program is completed.
- **Open Shop vs. Closed Shop:** Scheduling problems can be either open shop, where jobs can be processed in any order on any machine, or closed shop, where jobs must be processed in a specific order on specific machines.

These are just a few examples of how scheduling problems can be classified. The specific classification will depend on the problem at hand, and some problems may fit into multiple categories.

1.5 Combinatorial Optimization

Combinatorial optimization is a field of optimization that deals with finding the best solution among a finite set of possible solutions to a problem that involves discrete decision variables. Combinatorial optimization problems arise in many practical applications such as scheduling, routing, packing, and clustering[41]. The goal of combinatorial optimization is to find a solution that optimizes a given objective function subject to a set of constraints. The objective function may be to minimize or maximize a certain performance metric, such as cost, time, or resource utilization. The constraints may include resource limitations, precedence relations, and logical conditions.

Combinatorial optimization problems are often NP-hard, meaning that finding an optimal solution requires an exponential amount of time in the worst case.

Combinatorial optimization has applications in many fields, including computer science, engineering, economics, finance, and logistics. The ability to find optimal or near-optimal solutions to complex problems is essential for businesses, governments, and other organizations to make informed decisions and improve their operations.

1.6 Operating Room Scheduling Problem (ORSP)

1.6.1 Problem Description

Operating Room Scheduling Problem (ORSP) is a combinatorial optimization problem that deals with the allocation of surgery cases to the available operating rooms over a period of time. The main objective of ORSP is to maximize the utilization of resources and minimize the waiting time for patients. It is a critical issue in hospital management, as efficient scheduling can reduce the cost of operation and improve the quality of care [39].

The ORSP involves various constraints such as the availability of operating rooms, surgical staff, and equipment, as well as the priority of surgeries, the duration of each surgery, and the need for postoperative recovery time. The scheduling problem becomes even more complex in the case of multiple surgery cases, as the inter-dependencies between cases need to be taken into account.

ORSP is a challenging problem in hospital management that requires efficient scheduling techniques to improve the quality of care and reduce the cost of operation. Metaheuristic algorithms such as ACO can provide effective solutions to ORSP, and future research focuses on further improving the performance of these algorithms for this problem as we will see in our work[10].

In the figure 1.2 above it becomes evident that surgical planning is intricately linked with optimization, including Ant Colony Optimization (ACO). Within the realm of solution methodologies, genetic algorithms have emerged as prominent contenders. Multi-objective optimization and uncertainty represent two pivotal keywords within the depicted network, underscoring their significance.

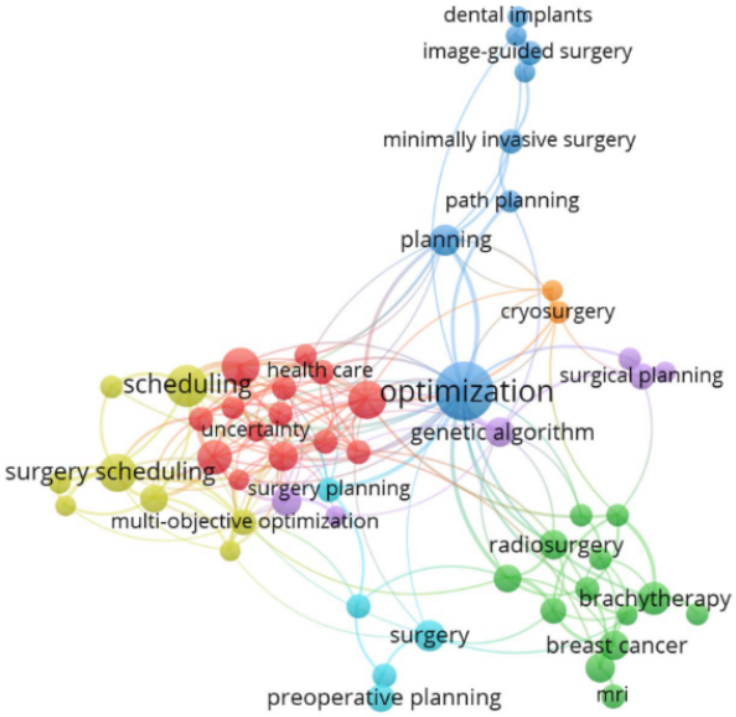


Figure 1.2: Keywords network of research areas in ORSP [32]

1.6.2 Mathematical Formulations of ORSP

We can find many formulations of the ORSP problem in the literature, and this is due to its complexity and different objectives and constraints that can be modeled for such a big problem. Among the most used formulations we can find:

Formulation1

The studied problem was formulated as a mathematical model by Yang et al.[18]

Objective function:

$$\min \sum_{d=1}^{ND} \sum_{k=1}^{Md} C_{d,k} = \sum_{d=1}^{ND} \sum_{k=1}^{Md} \max \left(R_{d,k} - \sum_{i \in \Omega} t_{i,z,d,k}, \beta \left(\sum_{i \in \Omega} t_{i,z,d,k} - R_{d,k} \right) \right) \tag{1.1}$$

subject to:

$$\sum_{i=1}^{Di}\sum_{k=1}^{Md} z_{i,d,k} = 1, \quad i \in \Omega, \quad d \leq ND \tag{1.2}$$

$$\sum_{d=1}^{ND}\sum_{k=1}^{Md} z_{i,d,k} \leq 1, \quad i \in \Omega, \quad d > ND \tag{1.3}$$

$$\sum_{i\in\Omega} t_{i,z,d,k} \leq R_{d,k} + S_{d,k}, \quad k \in \{1,\ldots,Md\}, \quad d \in \{1,\ldots,ND\} \tag{1.4}$$

$$\sum_{i\in\Omega} l_{t,i,z,d,k} \leq A_{d,l}, \quad d \in \{1,\ldots,ND\}, \quad l \in \{1,\ldots,NS\} \tag{1.5}$$

$$z_{i,d,k} \in \{0,1\}, \quad i \in \Omega, \quad k \in \{1,\ldots,Md\}, \quad d \in \{1,\ldots,ND\} \tag{1.6}$$

The objective function aims to minimize the combined cost of unexploited opening hours and overtime in the operating room. The cost per regular opening hour is considered constant and excluded from the objective function formula.

Constraints 1.2 and 1.3 ensure that each surgical case with a deadline before the end of the planning period (one week) is treated exactly once, while other cases can be treated at most once.

Constraint 1.4 limits the total operating time assigned to each operating room per day to its maximum opening hours.

Constraint 1.5 guarantees that the total operating time assigned to each surgeon per day does not exceed their maximum working hours. This constraint is essential as a surgeon cannot work on multiple surgical cases simultaneously. While it may appear that a surgeon can be assigned to multiple cases simultaneously in this planning model, a feasible operating schedule can always be achieved by ensuring proper patient sequencing during the daily scheduling stage, while adhering to this constraint at the planning stage.

Formulation2

The study conducted by Yang et al. [24] represents the continuation of their previous research [18].

the Formulation

Objective:

$$\min \sum_{d=1}^{}\sum_{k=1} C_{d,k} \tag{1.7}$$

Subject to:

$$\sum_{i\in\Omega} D_{i,d} = 1 \quad \sum_{k=1} M_{d,k} = 1 \quad x_{i,d,k} = 1 \quad \text{for } i \in \Omega \text{ and } D_i \leq H \tag{1.8}$$

$$\sum_{d=1} M_{d,k} \sum_{k=1} x_{i,d,k} \leq 1 \quad \text{for } i \in \Omega \text{ and } D_i > H \tag{1.9}$$

$$\sum_{d=1} M_{d,k} \sum_{k=1} x_{i,d,k} \leq 1 \quad \text{for } i \in \Omega \tag{1.10}$$

$$\sum_{i\in\Omega} t_{i,x,d,k} \leq RT_{d,k} + OT_{d,k} \quad \text{for } k = 1,\ldots,M_d; d = 1,\ldots,H \tag{1.11}$$

$$\sum_{k=1} \sum_{i\in\Omega_l} t_{i,x,d,k} \leq A_{d,l} \quad \text{for } d = 1,\ldots,H; l = 1,\ldots,L \tag{1.12}$$

$$C_{d,k} \geq RT_{d,k} - \sum_{i\in\Omega} t_{i,x,d,k} \quad \text{for } k = 1,\ldots,M_d; d = 1,\ldots,H \tag{1.13}$$

$$C_{d,k} \geq \alpha \left(\sum_{i\in\Omega} t_{i,x,d,k} - RT_{d,k} \right) \quad \text{for } k = 1,\ldots,M_d; d = 1,\ldots,H \tag{1.14}$$

$$C_{d,k} \geq 0 \quad \text{for } k = 1,\ldots,M_d; d = 1,\ldots,H \tag{1.15}$$

$$x_{i,d,k} = 0 \text{ or } 1 \quad \text{for } i \in \Omega; k = 1,\ldots,M_d; d = 1,\ldots,H \tag{1.16}$$

The objective function was formulated as follows:

$$\min \sum_{d=1}^{ND} \sum_{k=1}^{Md} C_{dk} = \max \left(\sum_{i\in\Omega} tiz_{dik}, \beta \sum_{i\in\Omega} tiz_{dik} - R_{dk} \right) \tag{1.17}$$

The objective function is nonlinear due to the presence of the maximum term. It captures the minimization of the total cost, considering the tradeoff between the total used time and overtime.

Additionally, it is important to note that the work in [18] failed to consider Constraint 1.10 . Without Constraint, there is a possibility that surgery i with a duration (Di) less than or equal to H might be scheduled from day d = 1 to Di and then again from day d = Di to H. This violates the condition that every surgery can only be scheduled at most once over the planning horizon, rendering it infeasible. they modified the model in [18] so that they can find an optimal solution for the studied problem. First, they revised the objective function in [18] as Constraints 1.8, 1.13, and 1.14 so that the objective function is linear. Second, they added Constraint 1.10 to prevent surgery from being scheduled twice. The modified mixed integer programming (MIP) model for the studied problem is as follows: The decision variable is defined as follows:

$$x_{dik} = \begin{cases} 1, & \text{if surgery } i \text{ is assigned to operating room } k \text{ on day } d \\ 0, & \text{otherwise} \end{cases} \tag{1.18}$$

The objective function aims to minimize the cost of total unused time and overtime.

The constraints are defined as follows:

Constraint (1.8): If a surgery's deadline is smaller or equal to H, it should be scheduled exactly once before its deadline.

Constraint (1.9): If a surgery's deadline is larger than H, it can be scheduled at most once over the planning horizon.

Constraint (1.10): Each surgery can only be scheduled at most once over the planning horizon.

Constraint (1.11): The total operating time of any operating room on any planning day should not exceed its maximum permissible overtime.

Constraint (1.12): The total operating time assigned to each surgeon per day cannot exceed their maximum working hours on that day.

Constraints (1.13) and (1.14) ensure that C_{dk} takes either the value of waste cost for the unused time of operating room k on day d, or the value of α times the overtime operating cost of operating room k on day d, depending on which value is larger.

Constraints (1.15) and (1.16) represent the nonnegativity and integrality restrictions, respectively.

Formulation 3

The studied problem was formulated as a mathematical model by Xiang [44]. In this study they assume that there is a set of surgeries I to be performed in an operating theatre with different types of resources C. There are two kinds of decision variables. $x_{ij} \cdot c_m$ is a 0–1 variable indicating whether resource m of resource type c is assigned to stage j of surgery i. Sti1 is another variable that determines the start time of surgery i. The OR scheduling problem is to minimize three objectives (described in function $f1$, $f2$ and $f3$) involving makespan, overtime, and leveling.

$$\min\ F = (f_1, f_2, f_3) \tag{1.19}$$

The three objective functions are:

1. Objective function 1: Eq. (1.20) is defined to minimize the makespan, i.e., the end time to finish all surgeries. This objective is set to evaluate the patient-related measure mentioned in 2.1. ET_{i3} is the end time of the last stage of surgery i. As shown in Eq. (1.21), it is determined by accumulating the longest operating time of the individual surgery stages by resource m of resource type c. T_{ijcm} is the duration time of stage j of surgery i by resource m of resource type c.

$$f_1 = \min\ \max_{i \in I} ET_{i3} \tag{1.20}$$

$$ET_{ij} = ST_{ij} + \max_{c \in C, m \in M_c} (T_{cmij} \cdot x_{cmij}) \quad \forall i \in I, j = 1, 2, 3 \tag{1.21}$$

2. Objective function 2: Eq. (1.22) is to minimize the variation of resources' working time. CV_c is the variation coefficient of the working time of resources in resource type c and is defined as the ratio of the standard deviation to the mean, as shown in Eq. (1.23-'1.25). It is used to evaluate the leveling of resources. M_c is the set of resources in resource type c.

$$f_2 = \min \max_{c \in C} CV_c \tag{1.22}$$

$$CV_c = \frac{r_c}{l_c} \tag{1.23}$$

$$l_c = \frac{1}{|J| \cdot |M_c|} \sum_{j \in J_i, m \in M_c, i \in I} T_{cmij} \cdot x_{cmij} \tag{1.24}$$

$$r_c = \sqrt{\frac{1}{|J| \cdot |M_c|} \sum_{j \in J_i, m \in M_c, i \in I} (T_{cmij} \cdot x_{cmij} - l_c)^2} \tag{1.25}$$

3. Objective function 3: Eq. (1.26) is to minimize the total overtime of all resources. RT_{cm} is the regular working time for a specific resource m in resource type c. This objective is defined to ensure the staff-related measure.

$$f_3 = \min \sum_{c \in C, m \in M_c} \max_{i \in I, j \in J_i} \left(T_{ij} \cdot x_{cmij} \cdot \left(1 - \frac{T_{ij} \cdot x_{cmij}}{RT_{cm}} \right) \right) \tag{1.26}$$

1. A resource can be allocated to one surgery at a time. H is an arbitrarily large positive number. $Z_{ij;i'j'} = 1$ if stage j of surgery i precedes stage j' of surgery i' on the same resource, $Z_{ij;i'j'} = 0$ otherwise.

$$ET_{i'j'} = ET_{ij} + H \cdot (1 - Z_{ij;i'j'}) \cdot T_{cmi'j'} \quad \forall i', j', i \in I, i' \neq i \tag{1.27}$$

$$ET_{ij} = ET_{i'j'} + H \cdot Z_{ij;i'j'} \cdot T_{cmij} \quad \forall i', j', i \in I, i' \neq i \tag{1.28}$$

2. The operating sequence of three stages must be followed completely and in sequence. Any two stages of a surgery cannot be performed at the same time.

$$ET_{ij} = ST_{i(j+1)} \quad \forall i \in I, j = 1, 2 \tag{1.29}$$

$$ET_{ij'} \geq ET_{ij} + T_{cmij'} \quad \forall j, j' \in J_i, j \neq j', i \in I, c \in C, m \in M_c \tag{1.30}$$

3. The exact number of required resources in resource type c is assigned to perform stage j of surgery i.

$$\sum_{m \in M_c} x_{cmij} = n_{cij} \quad \forall c \in C_{ij} \tag{1.31}$$

4. Some other constraints on state variables, like the start time of all surgeries should be greater than or equal to time 0. Constraints (1.33) are binary constraints.

$$ST_{i1} \geq 0 \quad \forall i \in I \tag{1.32}$$

$$x_{cmij} \in \{0, 1\} \quad \forall i \in I, j = 1, 2, 3, c \in C_{ij}, m \in M_c \tag{1.33}$$

1.7 Conclusion

In conclusion, the Operating Room Scheduling Problem (ORSP) is a complex optimization problem that requires the allocation of operating rooms, surgeons, and equipment to surgical procedures while satisfying various constraints. Overall, the ORSP is a significant challenge in healthcare management that requires advanced optimization techniques to improve patient outcomes and resource utilization. To address this problem, various optimization methods have been proposed in the literature, including deterministic and heuristic methods. However, these methods often suffer from limitations such as scalability and robustness when applied to large-scale instances, and due to this, metaheuristic methods or also known as approximate methods are the most preferred methods for solving such computational problems. Among these methods, we find the Ant Colony Optimization(ACO) algorithm which has shown promise as a potential solution to the ORSP due to its ability to handle complex and dynamic environments. The literature review suggests that the application of ACO algorithms to the ORSP is still limited, and there is a need for further research to investigate their potential. In the next chapter, we will introduce the ACO method and a short overview of its applications in optimization problems.

Chapter 2

Ant Colony Optimization for Combinatorial Optimization

2.1 Introduction

Ant Colony Optimization (ACO) is a metaheuristic algorithm that is inspired by the foraging behavior of real ant colonies. ACO has been widely used in combinatorial optimization problems, such as the traveling salesman problem (TSP), vehicle routing problem (VRP), and scheduling problems.

ACO has been shown to be effective in finding near-optimal solutions to combinatorial optimization problems, especially for problems with a large search space and complex constraints. One of the advantages of ACO is that it does not require any prior knowledge of the problem structure or search space, making it applicable to a wide range of problems. Overall, ACO is a powerful metaheuristic algorithm for solving combinatorial optimization problems, and its effectiveness has been demonstrated in numerous applications.[17]

2.2 Metaheuristics

Metaheuristics are high-level problem-solving strategies used to find good-quality solutions to complex optimization problems. They are general algorithmic frameworks that can be applied to various problem domains without requiring detailed knowledge about the problem structure. The term "meta" means "higher level" or "beyond," indicating that metaheuristics are not focused on finding optimal solutions but rather on finding good solutions within a reasonable amount of time, especially for problems that are difficult or impossible to solve exactly[2].

Metaheuristics are used for a wide range of problems, including combinatorial optimization, continuous optimization, multi-objective optimization, and constrained optimization, among others. They are particularly effective for problems that are NP-hard

or NP-complete, meaning that finding the optimal solution would take an inordinate amount of time. Common examples of problems that can be solved using metaheuristics include the traveling salesman problem, the job-shop scheduling problem, and the vehicle routing problem.

Metaheuristics include many methods such as genetic algorithms, simulated annealing, tabu search, Swarm colony, Bee colony, particle swarm optimization, and many others. Each metaheuristic has its own strengths and weaknesses, and the choice of which one to use depends on the problem being solved and the available computational resources.[7] The metaheuristics methods are subdivided into two categories: Trajectory-based methods and population-based methods.

2.2.1 Trajectory-based metaheuristics

The trajectory-based approach involves replacing an initial solution with another, usually the best, solution found in its neighborhood at each iteration of the search. This technique leads to the rapid generation of a local optimal solution. These methods, also known as Exploitation-Oriented methods, are a type of metaheuristic that is utilized to solve optimization problems. They can be viewed as steps across the neighborhood, tracing a search trajectory toward the solution space of the problem. Below are some of the most commonly used metaheuristic families that manipulate a single solution[1].

- Hill climbing
- Simulated annealing
- Tabu search
- Iterated Local Search
- Variable Neighborhood Search
- Guided Local Search
- Scatter Search
- Reactive Search Optimization

2.2.2 Population-based metaheuristics

Population-based metaheuristics utilize a set of solutions, or population, which is typically generated randomly. They iteratively select and combine existing solutions from the population in order to find good solutions. These methods are also known as Exploration-Oriented methods, as they aim to explore the search space. Two main mechanisms are

used in these metaheuristics: (1) selecting high-quality solutions from the population, and (2) recombining those solutions into new ones using specialized operators. After recombination, new solutions are reintroduced into the population, subject to conditions such as feasibility or minimum quality demands, which may require them to replace low-quality solutions [36]. A few examples of these methods are:

- Genetic algorithms (GA)
- Artificial Bee Colony (ABC)
- Ant Colony Optimization (ACO)
- Particle Swarm Optimization (PSO)

2.2.3 Hybrid Methods

Hybrid methods are considered a complementary category to the two previously presented categories of methods, Exact and Metaheuristics. Hybridization refers to the combination of two or more different types of optimization methods, especially for tackling large instances of various problems. Two types of hybrid methods are known: hybrid metaheuristics, which combine two or more metaheuristics, and hybrid methods, which merge an exact method with a metaheuristic. Hybrid methods have gained significant attention in the literature, and there are two primary classification taxonomies for these methods: the one proposed by Talbi in 2002 [40] and the one suggested by Raidl in 2005 [31].

2.3 Ant Colony Optimization (ACO)

More than 30 years ago, the first Ant Colony Optimization (ACO) technique was presented. Numerous other ACO algorithms variants were introduced after the main journal article describing Ant System (AS) was published in [12] with the intention of outperforming AS's performance and demonstrating that ACO algorithms could produce highly competitive results for a variety of well-known combinatorial optimization problems. These updated algorithms include Ant Colony System (ACS) [14], Max-Min Ant System (MMAS) [37], rank-based Ant System (RAS)[9], and several more[16]. The Traveling Salesman Problem (TSP) [9, 15, 37] and the Quadratic Assignment Problem (QAP) [15, 37], among others, were the prominent exam problems of the time.

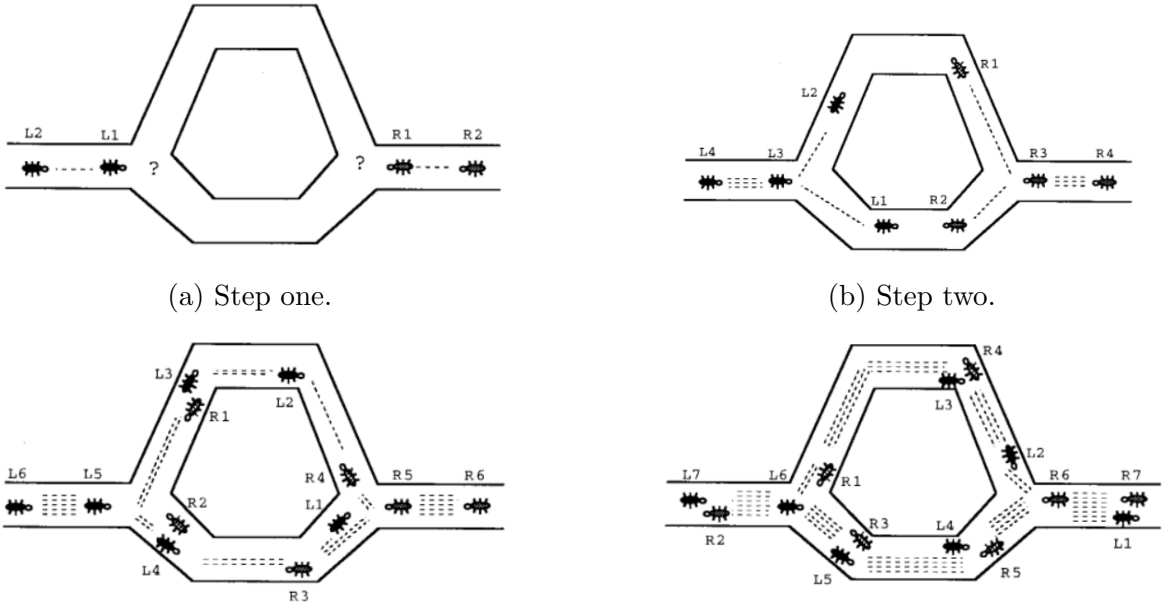
ACO is a metaheuristic optimization algorithm that models the behavior of real ant colonies to solve optimization problems. ACO works by iterative building and updating a pheromone trail, where the ants choose their next move based on the amount of pheromone present on the trail. ACO has been applied to a wide range of optimization

problems and has shown to be a powerful optimization method that can find high-quality solutions in a reasonable amount of time. The original paper introducing ACO is by Dorigo and Gambardella (1997), where they demonstrate the effectiveness of the Ant Colony System algorithm on the Traveling Salesman Problem.[14].

2.4 How real ants finds the path

In Figure 2.1 below we can observe the fascinating behavior of real ants as they navigate and discover a path. Here's a breakdown of their journey:

- (a) Ants arrive at a decision point.
- (b) Some ants choose the upper path and some the lower path. The choice is random.
- (c) Since ants move at approximately a constant speed, the ants that choose the lower, shorter, path reach the opposite decision point faster than those that choose the upper, longer, path.
- (d) Pheromone accumulates at a higher rate on the shorter path. The number of dashed lines is approximately proportional to the amount of pheromone deposited by ants.[20]



(a) Step one.

(b) Step two.

(c) Step three.

(d) Step four.

Figure 2.1: How real ant find the path.[20]

2.5 Ant Colony Optimization Algorithm

The algorithm works by simulating the behavior of ants as they search for food. Ants leave pheromone trails as they move, and other ants follow these trails to find food. The pheromone trails evaporate over time, so shorter paths with more pheromones become more attractive.

In the ACO algorithm, a colony of artificial ants is created to explore the solution space. Each ant builds a solution by moving from one solution component to another. The probability of an ant moving from one component to another depends on the amount of pheromone deposited on that component. After that, each ant completes a solution, and the quality of the solution is evaluated. The best solution found so far is used to update the pheromone trail. The amount of pheromone deposited on each component is proportional to the quality of the solution. Over time, the pheromone trail builds up on the best solutions, leading to the convergence of the algorithm to the optimal solution. However, to prevent the algorithm from getting stuck in local optima, random perturbations are introduced into the solution-building process.

The ACO algorithm has been successfully applied to a wide range of optimization problems and has been shown to produce high-quality solutions in a relatively short amount of time.[16]

Algorithm 1 Ant Colony Optimization

Require: $G = (V, E)$: the graph of the problem;
Require: n: number of ants;
Require: α: pheromone influence factor;
Require: β: heuristic influence factor;
Require: Q: pheromone trail intensity;
Require: ρ: pheromone trail evaporation factor;
Require: t_{ij}: amount of pheromone on edge (i, j);
Require: η_{ij}: heuristic value of edge (i, j);
Require: s: starting node;
Require: e: ending node;

```
Initialize pheromone trails t_ij;
for each ant k ∈ {1, 2, ..., n} do
    Choose a starting node s_k;
    Set the current node to s_k;
    while the ant has not reached the ending node e do
        Choose the next node using the transition rule;
        Move to the chosen node;
    end while
    Calculate the length L_k of the ant's tour;
    Update the pheromone trail on the ant's tour;
end for
Evaporate pheromone trails;
Update the pheromone trails with the new pheromone deposits;
Return the best tour found so far.
```

2.6 Optimization Problems Solved with ACO

Ant Colony Optimization (ACO) has been successfully applied to a wide range of optimization problems, including:

- **The Traveling Salesman Problem (TSP)**: it is a well-known problem in combinatorial optimization where the task is to find the shortest possible route that visits a set of cities and returns to the starting city. The problem is NP-hard, which means that it becomes increasingly difficult as the number of cities increases.

  Various approaches have been proposed to tackle the TSP, including exact methods, approximation algorithms, and heuristics. Exact methods can guarantee the optimal solution but are only feasible for small instances due to their computational

complexity. Approximation algorithms and heuristics provide sub-optimal solutions that can be computed faster and are more suitable for larger instances.

One popular heuristic for the TSP is the nearest neighbor algorithm, which starts at a random city and iteratively selects the nearest unvisited city until all cities have been visited. Another widely used heuristic is the 2-opt algorithm, which improves the solution by swapping pairs of edges and testing for improvements.

Overall, the TSP remains a challenging problem in combinatorial optimization with many applications in areas such as logistics, transportation, and network design [42].

- **The Vehicle Routing Problem (VRP)**: it is a classic combinatorial optimization problem that involves finding the optimal set of routes for a fleet of vehicles to serve a set of customers with known demands. The objective is to minimize the total distance traveled or the number of vehicles used while satisfying all constraints, such as vehicle capacity and customer time windows. VRP has numerous real-world applications in logistics, transportation, and distribution systems. Due to its computational complexity, many metaheuristic algorithms, such as genetic algorithms, simulated annealing, and ant colony optimization, have been developed to solve VRP instances efficiently.[13]

- **The Graph Coloring Problem (GCP)**: it is a classical combinatorial optimization problem that seeks to assign a color to each vertex of a graph such that no adjacent vertices share the same color. The objective is to minimize the number of colors used to color the graph. GCP has important applications in various fields, such as scheduling, register allocation, and frequency assignment.

  Several techniques have been proposed to solve GCP, including exact algorithms, heuristics, and metaheuristics. Exact algorithms, such as the backtracking algorithm and the branch and bound algorithm, can guarantee finding the optimal solution, but they can be computationally expensive for large graphs. On the other hand, heuristics and metaheuristics, such as greedy algorithms, tabu search, and genetic algorithms, can find suboptimal solutions efficiently but with no guarantee of optimality.[8]

- **The Knapsack Problem (KP)**: it is a well-known combinatorial optimization problem that seeks to maximize the value of items that can be placed into a knapsack of limited capacity. The objective is to select a subset of items that fit into the knapsack and has the highest total value. KP has a wide range of applications in various fields, including finance, resource allocation, and production planning.

  Various algorithms have been proposed to solve KP, including exact algorithms, heuristic algorithms, and metaheuristics. Exact algorithms, such as the dynamic

programming algorithm and the branch and bound algorithm, can guarantee finding the optimal solution, but they can be computationally expensive for large instances of KP. On the other hand, heuristic algorithms and metaheuristics, such as greedy algorithms, simulated annealing, and genetic algorithms, can find good quality solutions efficiently but with no guarantee of optimality.[30]

- **The Quadratic Assignment Problem (QAP)**: it is a combinatorial optimization problem that involves the assignment of facilities to locations. It seeks to minimize the sum of the weighted distances between all pairs of facilities, where the weight is a product of the flow between two facilities and the distance between their assigned locations. QAP has various real-world applications in areas such as telecommunications, logistics, and facility location.

  QAP is an NP-hard problem, and exact algorithms can be computationally expensive for large instances. As a result, heuristic and metaheuristic algorithms have been proposed to solve QAP efficiently. These include genetic algorithms, simulated annealing, tabu search, and ant colony optimization. However, finding high-quality solutions remains a challenging task.[28]

ACO has also been applied to other optimization problems such as facility location problem, job shop scheduling problem, and wireless sensor network coverage problem. It has been shown to produce high-quality solutions for these problems in a relatively short amount of time. However, its performance can be sensitive to the choice of parameters such as the pheromone update rate and the exploration-exploitation balance. Therefore, careful tuning of these parameters is important for achieving good performance.

Here's a simple example of the Ant Colony Optimization Algorithm applied to the Traveling Salesman Problem[14]:

- Initialization: The algorithm starts by creating a set of artificial ants that will explore the solution space. Each ant is placed at a random city and selects a random neighboring city to move to.

- Solution Building: The ants move through the solution space, constructing solutions by selecting neighboring cities to visit. The probability of an ant moving from one city to another depends on the amount of pheromone deposited on that edge and the distance to the neighboring city.

- Solution Evaluation: After each ant completes a tour, the quality of the tour is evaluated. In this example, the tour length is used as the objective function to be minimized.

- Pheromone Update: The pheromone trail is updated by depositing the pheromone on the edges of the best tour found so far. The amount of pheromone deposited on each edge is proportional to the quality of the tour.

- Termination: The algorithm terminates after a fixed number of iterations or when a stopping criterion is met. In this example, the algorithm terminates after 10 iterations.

  The final solution is the best tour found so far, which is the tour with the highest amount of pheromone deposited on its edges.

2.7 Overview for Operation Room Scheduling Problem Solved using ACO algorithm

In numerous studies addressing the ORSP problem, researchers have employed heuristic and meta-heuristic approaches to obtain near-optimal solutions. These methods include genetic algorithms [26], simulated annealing[4, 5], tabu search [23], and ant colony optimization[45]. However, when exploring the literature we noticed that only a few studies have explored the application of Ant Colony Optimization (ACO) to address the challenges of the Operation Room Scheduling Problem (ORSP), where we can cite:

One work is from Xiang et al2015 [45]. This study proposes an Ant Colony Optimization (ACO) approach with a hierarchical ant graph to address the surgery scheduling problem in large operating suites. The problem involves considering surgery sequencing, resource allocation, and multiple resource constraints. A two-level ant graph model, comprising an outer surgery graph and an inner resource graph, is introduced. The ACO algorithm aims to minimize the makespan of all surgeries by achieving sub-optimal solutions. Specific mechanisms such as pheromone representation, heuristic visibility, pheromone updating rule, and state transition rule are developed for the hierarchical graph. Experimental results on five test cases show that the proposed ACO outperforms a simulation model. Surgeons' specialties and experiences are also taken into account. The integration of nurse rostering and further improvements to the algorithm are suggested as future research directions. This ACO-based approach demonstrates effective and efficient solving of the surgery scheduling problem, resulting in shortened total makespan and balanced resource allocations. The consideration of surgeons' specialties and experience adds an additional layer of complexity to the problem. Future research aims to integrate nurse rostering and enhance the algorithm's intensification, diversification, and local search capabilities

another work by Xiang et al 2017[44] proposes a meta-heuristic approach integrating Pareto sets and Ant Colony Optimization (ACO) to solve the multi-objective OR

scheduling problem with objectives such as minimizing waiting time, reducing overtime, and increasing resource utilization. The paper compares the performance of the proposed approach with other methods using a test case from MD Anderson Cancer Center.

in The work by Yin et al 2012[46] addresses the complex task of surgery scheduling, considering the entire process from preoperative to postoperative stages, multiple resource constraints, and the integration of upstream and downstream surgical resources. The authors frame the problem as an extended multi-resource constrained flexible job-shop scheduling problem and propose an optimization approach based on an improved ant colony algorithm.

The algorithm incorporates a resource selection rule and a strategy for overtime judging and adjusting to enhance the scheduling process. Experimental results from both test instances and real-world hospital data demonstrate that the proposed algorithm effectively solves the surgery scheduling problem. It successfully reduces the total time required for surgeries and optimally allocates resources.

In summary, the paper presents an improved ant colony algorithm tailored for surgery scheduling. By considering the various stages of surgery and resource constraints, the algorithm achieves efficient resource allocation and minimizes the makespan. The results highlight its effectiveness in real-world scenarios, where it optimizes surgery scheduling by reducing time and optimizing resource allocation.

in this paper by Tan et al 2015 [19] the authors propose a modified ant colony optimization (ACO) algorithm, specifically designed for multi-objective OR scheduling. The algorithm incorporates Pareto sets construction and introduces two types of pheromone setting strategies. The objectives aimed to be optimized include minimizing makespan, reducing nurses' overtime, and balancing resource utilization. They compare the scheduling results obtained by three different approaches: simulation, ACO with a single objective of makespan (ACO-SO), and the proposed ACO with multi-objectives (ACO-MO). The computational results demonstrate that the ACO-MO algorithm achieves promising outcomes, effectively shortening the makespan, reducing nurses' overtime, and improving resource utilization equilibrium.

In conclusion, the authors highlight the significance of their modified ACO algorithm for the combinatorial OR scheduling problem with conflicting objectives. The integration of Pareto sets and the introduction of two pheromone setting strategies adapt traditional ACO to address the multi-objective nature of the problem. The performance evaluation, comparing ACO-MO with simulation and ACO-SO, reveals the superiority of the proposed algorithm.

This summarized work provides valuable insights into the proposed modified ACO algorithm's effectiveness in solving the OR scheduling problem with multiple conflicting objectives. It establishes a foundation for further research and development in optimizing

OR scheduling in real-world hospital settings.

2.8 Conclusion

Ant Colony Optimization (ACO) is a powerful metaheuristic algorithm for solving combinatorial optimization problems. It has been shown to produce high-quality solutions in a reasonable amount of time for many combinatorial optimization problems, and it has been extended and adapted in various ways to enhance its performance.

However, there are still many open research questions and challenges in the field of ACO, such as determining the optimal parameter settings, dealing with dynamic environments, and integrating ACO with other optimization algorithms. Nonetheless, Applications of ACO on optimization problems remain a promising and active research area, with new and innovative variants of the algorithm constantly being developed and tested. Therefore, ACO has the potential to make significant contributions to solving complex combinatorial optimization problems in various real-world applications. In the following chapter, we aim to present the application and adaptation of ACO for solving the ORSP problem.

Chapter 3

Ant Colony Optimization for Operating Room Scheduling Problem

3.1 Introduction

In this chapter, we present our proposed models Operations Research Scheduling Problem (ORSP), then we describe the Ant Colony Optimization (ACO) approach we adapted to solve the proposed models. Our ACO algorithm is tailored to address the challenges of scheduling tasks in real-world scenarios.The ACO algorithm was adapted by incorporating different ant-searching structures and different objective functions specifically designed for the ORSP. We conduct experiments using training and real-world datasets, varying in size and resource configurations, to evaluate the performance of our approach. Additionally, we discuss the scheduling strategy employed by our approach and provide insights into its effectiveness. Through this research, we aim to enhance scheduling optimization techniques for the ORSP.

3.2 Proposed Models

3.2.1 Objective 1: Maximize the Total Number of Scheduled Operations

Model Description

The objective of this optimization model is to maximize the total number of prioritized operations scheduled for the duration of a week, ensuring efficient utilization of resources and meeting the constraints of the scheduling problem. By assigning surgeries to appropriate time slots, the aim is to accommodate as many operations as possible within the given time frame per day.

To achieve this objective, the decision variables x_i represent the assignment of surg-

eries i to time slots, where a value of $x_i = 1$ indicates that a surgery is assigned to a specific time slot, and if not then $x_i = 0$. The start time (s_i) and duration (d_i) of each surgery are also considered. Each surgery has a priority p_i that defines the importance and emergency of the surgery i, where the priority of the surgery is calculated as following:

$$p_i = \sum_{i=1}^{n} (r_i * \frac{t}{r_i}) = \sum_{i=1}^{n} (r_i * w_i) \tag{3.1}$$

where r_i represents the required number of resource i, T is the total number of resources and w_i is the weight of resource i which is calculated by the formulation: t/R_i.

The objective function is defined as the sum of all assigned important/prioritized surgeries x_i, indicating the total number of scheduled operations. By maximizing this objective, the scheduling algorithm aims to optimize the total number of scheduled operations.

Objective function: Maximize: $\sum_{i=1}^{n} x_i \cdot p_i$

constraints:

1. Each surgery must be assigned to exactly one-time slot: $x_{ij} = 1$ for all $i = 1, 2, \ldots, n$
2. Time slot constraints: $s_i \geq 0$ for all i (start time of each surgery)
   $s_i + d_i \leq s_j$ or $s_j + d_j \leq s_i$ for all $i \neq j$ (no overlap between surgeries scheduled in the same room)
3. Surgeon availability: Constraints ensuring that each surgery i is scheduled only when the required surgeons Sr_{ki} are available.
4. Equipment and staff availability: Constraints ensuring that the necessary equipment and staff ES_i such as anesthesia providers and nurses, are available for each surgery i.

Mathematical Formulation

$$\min \sum_{i=1}^{n} x_i \cdot p_i \tag{3.2}$$

Subject to:

$$x_{ij} = \begin{cases} 1 & \text{if an operation is assigned to a time slot} \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

$$s_i \geq 0, \quad \forall i = 1, 2, \ldots, n \tag{3.4}$$

$$s_i + d_i \leq s_j \text{ or } s_j + d_j \leq s_i, \forall i \neq j \tag{3.5}$$

$$Sr_{ki} \leq Sr, \quad \forall i, k = 1, 2, \ldots, n \tag{3.6}$$

$$ES_i \leq ES, \quad \forall i = 1, 2, \ldots, n, \tag{3.7}$$

3.2.2 Objective 2: Minimize the Total Weighted Tardiness of Operations

Model Description

The objective of this second model is to minimize the total weighted tardiness of operations, considering the importance and urgency of each surgery. The weighted tardiness reflects the deviation between the actual start/end time and the desired start/end time of each operation, with higher weights assigned to surgeries with stricter time constraints.

The variables x_{ij} and s_i represent the assignment and start time of each surgery i, respectively, while d_i represents the duration of each surgery i. Additionally, the decision variable Sr_{kj} indicates the availability of surgeon k at time slot j.

The objective function is defined as the sum of the weighted tardiness of all surgeries, where w_i represents the weight associated with surgery i and T_i represents the tardiness of the surgery i. Therefore, by minimizing this objective, the scheduling algorithm aims to prioritize surgeries with stricter time constraints, reduce delays, and ensure the timely completion of operations.

ovjective function: Minimize: $\sum_{i=1}^{n} w_i \cdot T_i$

Constraints:

1. Each surgery must be assigned to exactly one time-slot: $x_{ij} = 1$ for all $i = 1, 2, \ldots, n$
2. Time slot constraints:

$$s_i \geq 0 \text{ for all } i \text{ (start time of each surgery)}$$
$$s_i + d_i \leq s_j \text{ or } s_j + d_j \leq s_i \text{ for all } i \neq j \text{ (no overlap between surgeries)}$$

3. Surgeons availability constraints:$\sum Sr_{kj} \geq \sum x_{ij}$ for all i surgeries (ensures that there are enough available surgeons for each assigned surgery).
4. Equipment and staff availability: Constraints ensuring that the necessary equipment and staff ES_i such as anesthesia providers and nurses are available for each surgery i.

Mathematical Formulation

$$\min \sum_{i=1}^{n} w_{ij} \cdot T_i \tag{3.8}$$

Subject to:

$$x_{ij} = \begin{cases} 1 & \text{if an operation is assigned to a time slot} \\ 0 & \text{otherwise} \end{cases} \tag{3.9}$$

$$s_i \geq 0, \quad \forall i = 1, 2, \ldots, n \tag{3.10}$$

$$s_i + d_i \leq s_j \text{ or } s_j + d_j \leq s_i, \forall i \neq j \tag{3.11}$$

$$Sr_{ki} \leq Sr, \quad \text{for } i, k = 1, 2, \ldots, n \tag{3.12}$$

$$\sum Sr_{kj} \geq \sum x_{ij}, \text{for } i = 1, 2, \ldots, n \tag{3.13}$$

$$ES_i \leq ES, \quad \text{for } i = 1, 2, \ldots, n, \tag{3.14}$$

Calculation of the weight w_i:

The specific method for calculating the weights can vary depending on the specific context and requirements of the operating room scheduling problem being addressed.

1. *Resource utilization*: The weights can be determined based on the impact of each surgery on resource utilization. For instance, surgeries that require specialized equipment or scarce resources can be assigned higher weights to ensure their efficient utilization.

2. *Expert judgment*: Experts in the field, such as surgeons, administrators, or healthcare professionals, can assign weights based on their knowledge and experience. They can evaluate the relative importance of each surgery considering factors like medical urgency, patient conditions, and clinical requirements.

3. *Patient priority*: The weights can be based on the urgency or priority level assigned to each patient. For example, surgeries for emergency cases or patients with critical conditions can be assigned higher weights to reflect their priority in the scheduling process.

4. *Cost considerations*: The weights can be derived from the cost associated with each surgery. For example, surgeries with higher financial implications or revenue potential can be assigned higher weights to reflect their economic importance.

For the case of our model in this work, we are using the first weight calculation type based on ***Resource utilization***.

3.3 ACO approach for solving the ORSP

We propose an adaptation of the Ant Colony Optimization (ACO) algorithm to tackle the Operations Room Scheduling Problem (ORSP) in line with our models. Our approach builds upon the fundamental principles of the ACO algorithm but with specific modifications to address the unique challenges of the scheduling problem. We reframe the scheduling problem on the basis of the traveling salesman problem (TSP), by considering each operation as a city, and the connections between operations as arcs that represent the schedulability of subsequent operations. Through the use of pheromones and a weighted graph, our modified ACO algorithm guides the ants (scheduling agents) in making informed decisions about the next operation to schedule, taking into account the availability of resources and with respect to the problem constraints.

To evaluate the quality of schedules generated by our approach, we employ a fitness/objective function.This function considers constraints such as resource utilization, completion time, and other relevant criteria, aiding in the selection of the optimal schedules. Additionally, our adaptation of the ACO algorithm incorporates a scheduling strategy that optimizes the allocation of operations within the specified time frame. This strategy considers resource availability and precedence constraints to ensure efficient scheduling.

3.3.1 Description of the ACO algorithm process

In the proposed ACO algorithm for ORSP, we start by loading schedule data (Instances) from an Excel file and defining overtime and *DayOfWeek* objects for each day of the week. Then, it generates a pheromone matrix and weight matrix using *ACO_Calculation* to define the ACO parameters and objective of the algorithm. Next, an *AntColonyAlgorithm* object is created using the defined parameters and objective, where the operations list is pre-processed. The pre-processing step aims to optimize the search process by prioritizing high-priority operations. This helps ants navigate through the schedule more efficiently, reducing the overall search time. Then, the algorithm gets the weekly schedule by initializing variables for the best schedule, best fitness, and start time, and iterating through a loop where it creates a list of *WeeklyAntAgent* objects. For each ant, a new schedule is created by initializing variables for the current day, current period, and current operation index, and scheduling operations while there are still operations to be scheduled. The fitness of the schedule is calculated using the objective function, and the best schedule and best fitness are updated every time the ants find a better solution than the current best. The pheromone matrix is updated using the global pheromone update rule $\tau_{xy} = (1 - \rho) \cdot \tau_{xy} + \Delta\tau_{xy}$, and the loop is exited if the running time has been exceeded, in our case, it represents the max number of iterations or max time limit.

The following pseudo-code (see algorithm 2) represents the steps of ACO algorithm for solving the ORSP problem (minimization case).

Algorithm 2 Ant Colony Algorithm for ORSP

```
Inputs : Operations, AvailableRessources          //Load the data from an Excel file
AllocateScheduleTimeFrame()          //Define overtime and create days time-frame
GenerateInitialPheromoneMatrix()
Pre-processing data: GenerateWeightMatrix()
Define ACO parameters: alpha ,beta , Q , initialPheromoneValue ,MaxIterations
, numberOfAnts
PreProcessOperations(Operations)    //Starting from the most important operations
Initialize: BestSchedule, BestFitness, and StartTime
Iteration ← 1
repeat
    Create a list of WeeklyAntAgent objects
    for each ant i ∈ 1, 2, . . . , numberOfAnts do
        Generate a new schedule
        Initialize: CurrentDay, CurrentPeriod, and CurrentOperationIndex
        while (NbrScheduledOperation ≤ ScheduleTotalCapacity) do
            TempOperationIndex ← NewOperation
            if (Duration(NewOperation) ≤ CurrentPeriod) then
                Schedule ← Schedule + NewOperation
                currentOperationIndex ← TempOperation.index
                UpdatePeriod(CurrentPeriod)
            else
                Move to next period or next day if necessary
            end if
        end while
        CurrentFitness ← CalculateFitness(CurrentSchedule)
        if (CurrentFitness ≤ BestFitness) then
            BestFitness ← CurrentFitness
            BestSchedule ← CurrentSchedule
        end if
    end for
    UpdatePheromonesMatrix()
    Iteration + +
until (Iteration ≤ MaxIterations )
Print BestSchedule + BestFitness
```

3.4 Experimental setup and Results

3.4.1 Dataset/Benchmarks

Due to the wide range of constraints and goals that the ORSP has, researchers have worked and generated many different sets of instances that would match their model,and because our models have slightly different objectives than what exists in the literature, therefore, what exists does not match with our problem model, thus, we generated and collected our proper dataset in order to launch our experiments. Our proposed approach has been thoroughly evaluated using two distinct datasets to validate its performance and applicability:

Training dataset

The training dataset is designed to simulate some operations and contains the following information:

- **Index**: A unique identifier for each operation.
- **Name**: Name or description of the operation.
- **Duration**: Duration of the operation in minutes.
- **Surgeons**: Number of surgeons required for the operation.
- **Nurses**: Number of nurses required for the operation.
- **Beds**: Number of beds required for the operation.
- **Rooms**: Number of rooms or blocks required for the operation.

This dataset represents a simplified scenario for small-scale operations, where the resources needed for each operation are relatively limited. It serves as a training dataset to train the algorithm and optimize the scheduling of these small operations based on the available resources.

Real-world dataset

The real-world scenario dataset represents a more complex and realistic scenario with a wider range of resources and varying operation characteristics. This dataset was manually collected from the "Boukhroufa Abdelkader" hospital in Ben Aknoun, Algiers, Algeria. The dataset includes the following information:

- **Index**: A unique identifier for each operation.

- **Name**: Name or description of the operation.
- **Duration**: Duration of the operation in minutes.
- **Professeur**: Number of professors required for the operation.
- **Maitre assistant**: Number of assistant professors required for the operation.
- **Résidents**: Number of resident doctors required for the operation.
- **Bloc**: The block or the room where the operation is scheduled to take place.
- **Anesth**: Number of anesthesiologists required for the operation.
- **Paramedicaux**: Number of paramedical staff required for the operation.

This dataset represents a real-world scenario in a healthcare setting, where operations have specific resource requirements based on the expertise of the medical staff, the facilities available, and the duration of the operations.

Our developed algorithm exhibits high flexibility, allowing it to adapt to different scheduling scenarios. It can efficiently handle varying resource requirements and operation characteristics, making it a versatile solution. This flexibility enables the algorithm to accommodate small-scale operations with limited resources as well as more complex real-world scenarios with diverse resource demands. It dynamically adjusts and optimizes the scheduling process based on the specific dataset, resulting in efficient and effective scheduling solutions.

3.4.2 Experiment environment

The experiments conducted in this thesis were performed in a controlled software environment. The following details outline the software tools, programming language, and hardware used for the experiments:

1. Software Tools:
    - Code Editor: Visual Studio Code
    - Compiler/Interpreter: Python 3.11.4
2. Programming Language:
    - The implementation of the algorithms and experiments was done using Python. Python was chosen for its flexibility, extensive library support, and ease of use in scientific computing and data analysis.

3. Hardware:
    - Laptop 1:
        - Processor: Intel Core i5-10300H CPU @ 2.50 GHz (8 CPU)
        - RAM: 8 GB
        - Storage: 250 GB SSD + 1 TB HDD
        - GPU: NVIDIA GeForce GTX 1650Ti
    - Laptop 2:
        - Processor: Intel Core i7-11th Gen
        - RAM: 16 GB
        - Storage: 500 TB SSD
4. Operating System:
    - The experiments were conducted on two different operating systems:
        - Windows 11 Home 64-Bit
        - Ubuntu 20.04

3.4.3 Experiment setups

In the obtained results, we executed our code using the following set of parameters:

- Number of ants: 25
- Number of iterations: 50
- Alpha value: 1
- Beta value: 1
- Initial pheromone level: 0.1
- Q value: 100
- number of operations: 50

3.4.4 Experiments with objective 1: Maximization

3.4.4.1 Results of the training dataset

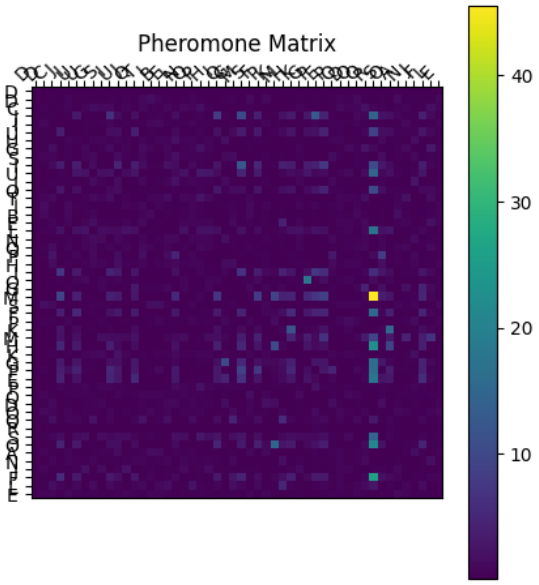


Figure 3.1: Pheromone matrix (Maximization objective)

The figure above (3.1) provides a visual representation of the Pheromone matrix, which serves as a measure of the probability of scheduling each operation in comparison to the others. The intensity of the yellow color within the matrix conveys the likelihood of finding an optimal schedule, with a brighter shade indicating a higher probability. Conversely, areas without color represent operations with a lower likelihood of being scheduled. By examining this figure, one can easily identify the relative probabilities assigned to each operation, thereby gaining insights into the scheduling preferences determined by the Pheromone matrix.

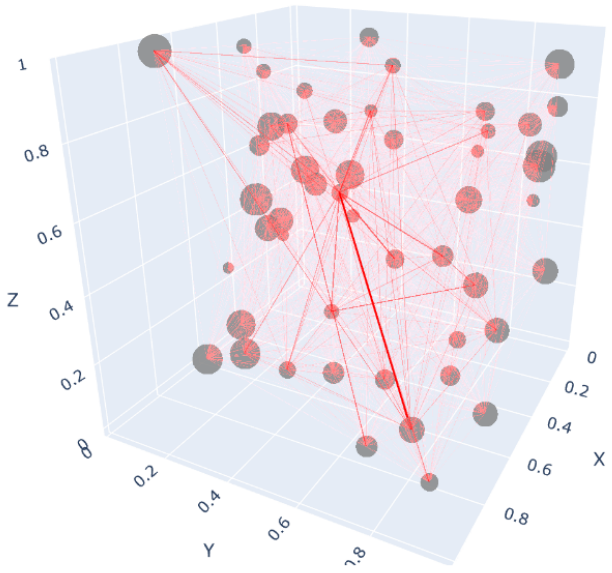


Figure 3.2: Operations' exploration Graphic based on Pheromones on the training set

The figure above (3.2) depicts the Pheromone matrix, presenting a graphical representation of the relationships between operations. Each node within the matrix represents a specific operation, and the size of the node corresponds to the importance/priority of the operation based on the resources needed in terms of scheduling. Larger nodes indicate operations that carry greater significance.

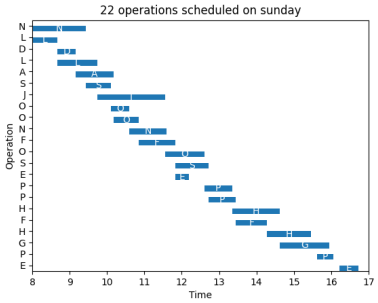
The arcs connecting the nodes illustrate the number of times the ants have visited each operation during the scheduling process. The intensity (thickness of the arc) of the red color on the arcs reflects the frequency of visits by the ants. Darker shades of red signify more frequent visits, indicating that certain operations have been explored and considered more often by the ants.

Number of Operation	Instances	Metrics	Best	Avg
10 Operation	instance 1	Fitness	3003.3330	3003.3330
		CPU time Best	7.7875	8.6072
15 Operation	instance 2	Fitness	5678.0000	5648.5335
		CPU time Best	15.8821	17.5879
20 Operation	instance 3	Fitness	7040.8330	7006.8334
		CPU time Best	30.4884	29.2164
30 Operation	instance 4	Fitness	11211.5000	11072.5250
		CPU time Best	52.2527	53.8608
50 Operation	instance 5	Fitness	19476.3330	18646.8743
		CPU time Best	142.5442	140.4870
100 Operation	instance 6	Fitness	32919.0800	31999.2430
		CPU time Best	441.7997	428.7896

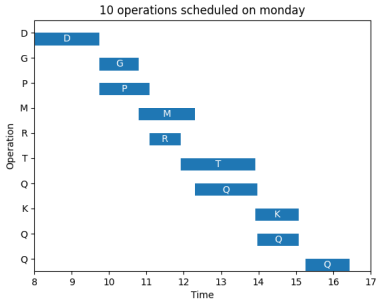
Table 3.1: Experiment results for the Maximization problem (10 run/instance) for the training set

The algorithm was tested first on this training dataset, and the Above (table 3.1) showcases the outcomes obtained by executing 10 runs on six distinct instances, each characterized by a varying number of operations. The table comprises five columns: the first column denotes the number of operations in each instance, the second column represents the instance name, the third column indicates the metric evaluated: fitness and CUP-time, the fourth column displays the best value attained in the 10 runs, and the fifth column showcases the average value across the 10 runs.

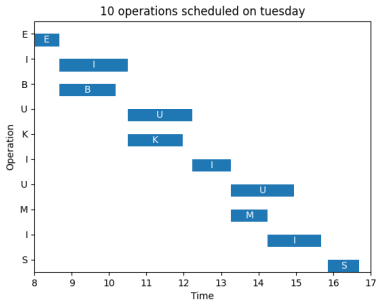
Upon analyzing the table, several observations can be made. Firstly, there is a noticeable enhancement in fitness values as the number of operations increases. This implies that with a larger number of operations, the problem becomes more complex and requieres more time to find a solution.



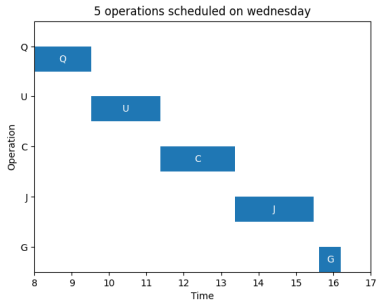
(a) Day one



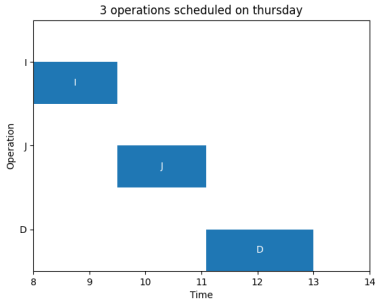
(b) Day two



(c) Day three



(d) Day four



(e) Day five

Figure 3.3: Schedule of a week for 50 operations on training set

The figure above (3.3) provides a schedule example of 50 operations on 5 working days (one week). As we can see, all fifty (50) operations were scheduled. However, there is a noticeable variation in the number of operations scheduled on each day. On the first day, a relatively high number of operations (22) was scheduled, while the remaining days had fewer operations scheduled. The last day of the week only had 3 operations scheduled. This disparity in the distribution of scheduled operations is attributed to the high availability of resources.

As the week progressed, operations with shorter periods were scheduled on the first days, and those that demanded more time and resources were scheduled by the end of the week, resulting in a lower number of operations being accommodated on each subsequent day. The scheduling decisions were likely influenced by factors such as the availability of

equipment, staff, or any other limited resources.

3.4.4.2 Results of the real-world dataset for testing

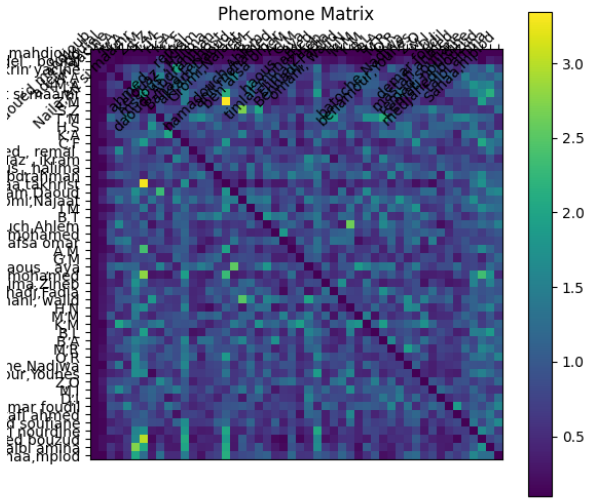


Figure 3.4: Pheromone matrix

Figure 3.4 visualizes the Pheromone matrix on the test data from a real-world scenario, depicting the probability of scheduling each operation in relation to the others.

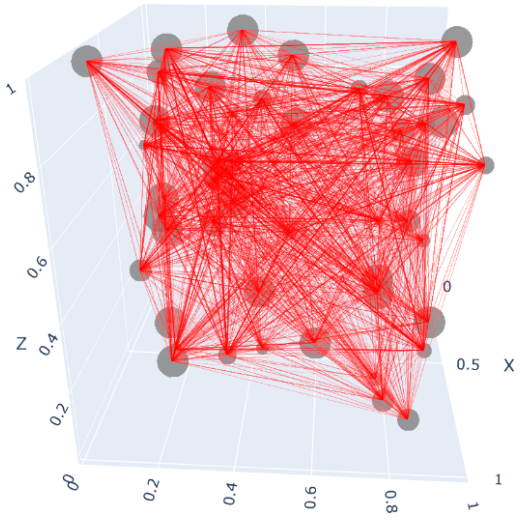


Figure 3.5: Operations' exploration Graphic based on Pheromones on real-world data

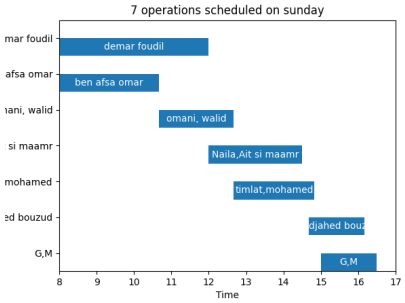
This figure 3.5 above represents the Pheromone matrix exploration space, where each node corresponds to an operation. The size of the node reflects the importance of the operation in terms of scheduling. Larger nodes indicate operations that are considered more significant. The arcs connecting the nodes represent the number of times the ants have visited each operation. The intensity of the red color on the arcs indicates the frequency of visits, with darker red indicating more frequent visits by the ants.

Number of Operation	**Instances**	**Metrics**	**Best**	**Avg**
10 Operation	instance 1	Fitness	2602.8750	2602.8750
		CPU time Best	9.8081	10.3470
15 Operation	instance 2	Fitness	3530.5420	3528.3420
		CPU time Best	18.6576	20.3267
20 Operation	instance 3	Fitness	7390.6250	7264.5375
		CPU time Best	77.3165	69.8369
30 Operation	instance 4	Fitness	7356.2500	7222.4167
		CPU time Best	65.0930	67.5291
50 Operation	instance 5	Fitness	10659.4580	10286.3758
		CPU time Best	163.4976	147.1615
100 Operation	instance 6	Fitness	17995.5400	17257.8560
		CPU time Best	327.8681	332.5044

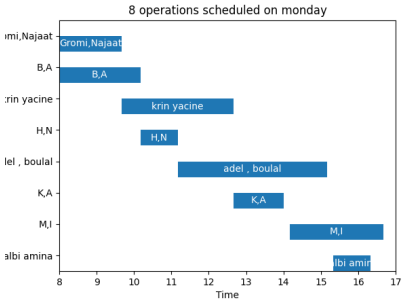
Table 3.2: Experiment results for the Maximization problem (10 run/instance) for the real-world dataset

This table presents the results obtained from running our algorithm 10 times on six different instances from the real-world dataset, each with varying numbers of operations.

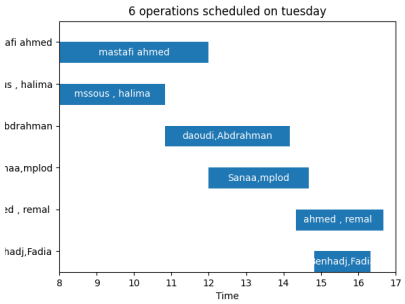
By examining the results in the table, we can observe that as the number of operations increases, the complexity of the algorithm increases and the CPU time takes longer time to achieve better solutions.



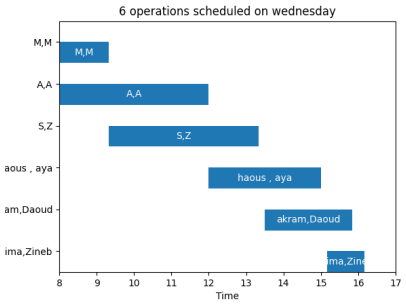
(a) Day one



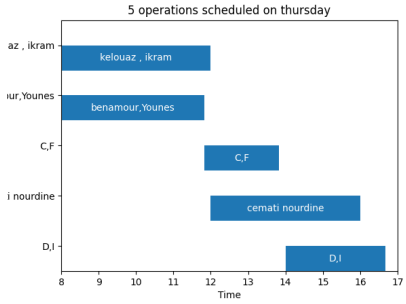
(b) Day two



(c) Day three



(d) Day four



(e) Day five

Figure 3.6: Schedule of a week for 50 operations on real-world dataset

Figure 3.6 represents a schedule for a week on real-world data. We can see that the schedule distribution of operations is relatively balanced across all days. Each day of the week has a consistent number of operations scheduled, with the range in the intervale [5:8] operations per day. It is worth noting that out of the total 50 operations, only 32 operations have been scheduled. This results is due to the influence of several factors that could be driven by the availability of resources. This equitable distribution of operations throughout the week helps in maintaining a steady and manageable workflow, contributing to effective operational planning and execution in hospitals.

3.4.4.3 Comparison study: Training vs. real-world dataset

The provided schedules for a week showcase two distinct scenarios: one based on a training dataset that we generated, and the other derived from a real-world-world dataset representing a hospital case.

In the first schedule, which is based on the training dataset (see figure 3.3), we noticed that all fifty (50) operations were scheduled during the 5 days, however, there is a noticeable variation in the number of operations scheduled each day. On the first day, a relatively high number of operations (22) were scheduled, while the remaining days had fewer operations scheduled. The last day of the week only had 3 operations scheduled. This disparity in the distribution of scheduled operations is attributed to the availability of resources. On the other hand, the second schedule, based on the real-world-world dataset(see figure 3.6), demonstrates a relatively balanced distribution of operations across all days of the week. Each day consistently accommodates a similar number of operations, ranging from 5 to 8 operations per day. However, it is important to note that out of the total 50 operations, only 32 operations have been scheduled in this scenario due to the influence of the factor of resource availability such as surgeons, staff, and nurses.

From here, we can say that the data from real-world are more complex to manage in scheduling surgery rooms due to the influence of many factors and the constraints to be handled. These factors play a crucial role in evaluating and understanding the performance and effectiveness of the scheduling solutions.

3.4.5 Experiments with objective 2: Minimization

3.4.5.1 Results of the training dataset

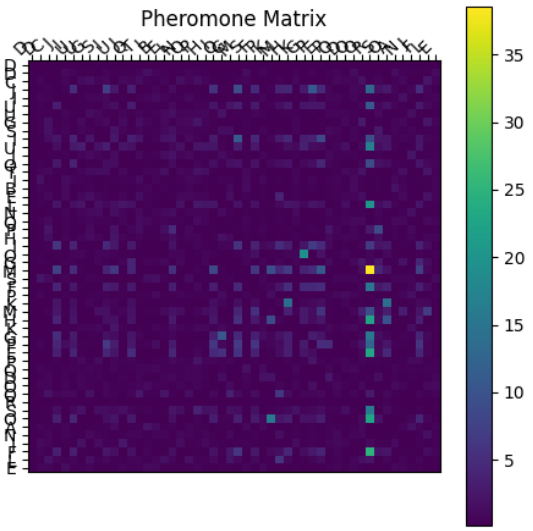


Figure 3.7: Pheromone matrix

This figure 3.7 illustrates the Pheromone matrix, a valuable tool for determining the likelihood of scheduling each operation in relation to others. A brighter shade signifies a higher likelihood to find a better solution, while areas without color indicate lower probabilities.

By analyzing this figure, one can readily identify the relative probabilities assigned to each operation. This provides valuable insights into the scheduling preferences determined by the Pheromone matrix.

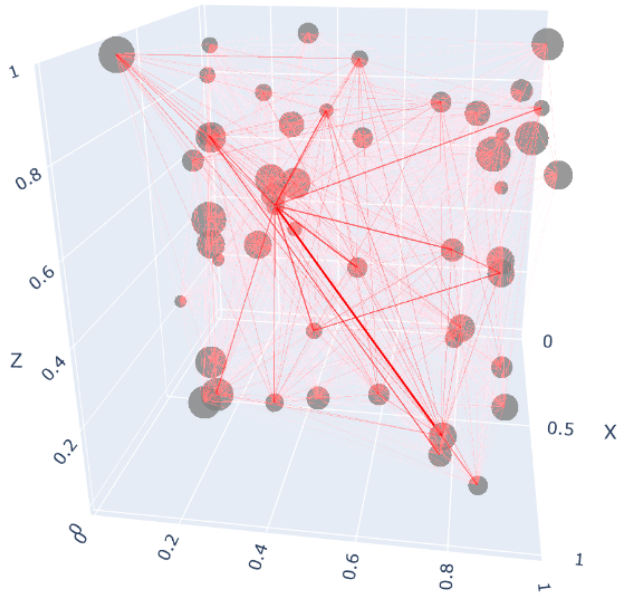


Figure 3.8: Operations' exploration Graphic based on Pheromones on the training set

The following figure 3.8represents the Pheromone matrix, which offers valuable insights into the relationships between operations in the scheduling process. Each node in the matrix corresponds to a specific operation, and the size of the node indicates its importance in terms of scheduling. Larger nodes represent operations of greater significance. The arcs connecting the nodes depict the number of times the ants have visited each operation during the scheduling process. The intensity of the red color on the arcs reflects the frequency of visits by the ants. Darker shades of red indicate more frequent visits, suggesting that certain operations have been explored and considered more often by the ants than other operations.

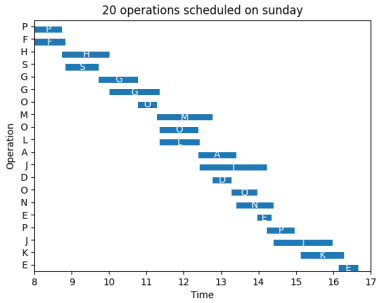
By examining this figure, one can gain insights into the relative importance of operations and their levels of exploration during the scheduling process. The size of the nodes provides a visual indication of the operations' significance and importance, while the color intensity on the arcs offers insights into the frequency of visits by the ants.

Number of Operation	Instances	Metrics	Best	Avg
10 Operation	instance 1	Fitness	3.641	5.0835
		CPU time Best	9.0617	9.7859
15 Operation	instance 2	Fitness	3.6877	4.0928
		CPU time Best	17.6886	20.2078
20 Operation	instance 3	Fitness	27.7184	28.4749
		CPU time Best	30.9078	34.2230
30 Operation	instance 4	Fitness	12.4197	13.6459
		CPU time Best	54.1700	61.9109
50 Operation	instance 5	Fitness	1.7741	3.1747
		CPU time Best	136.5684	149.6612
100 Operation	instance 6	Fitness	18.3583	19.9509
		CPU time Best	397.8395	425.0047

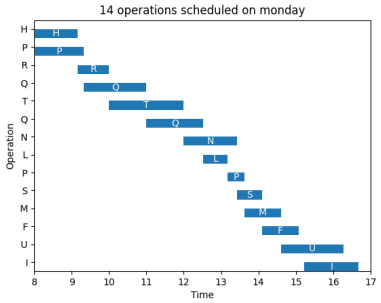
Table 3.3: Experiment results for the Minimization problem (10 run/instance) for the training set

Table 3.3 presents the results obtained from running our algorithm 10 times on six different instances, each with a varying number of operations. The table consists of five columns: the first column indicates the number of operations in each instance, the second column represents the instance name, the third column specifies whether the value is related to fitness or time, the fourth column shows the best value achieved by the algorithm, and the fifth column displays the average value across the 10 runs.

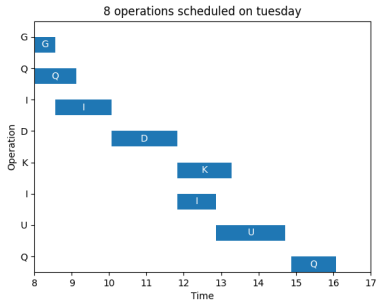
From the results above we can say that as the size of the instance gets larger the search space gets larger too and finding the best solution becomes difficult and takes more time so the algorithm converges. This is due to the complexity of the problem and the constraints to be satisfied.



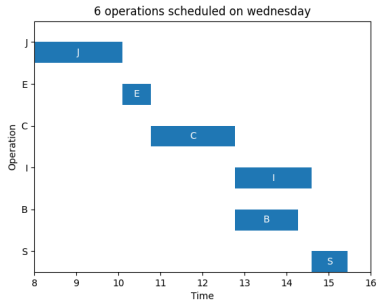
(a) Day one



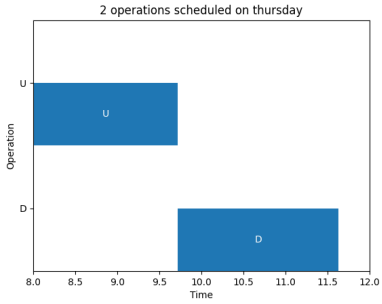
(b) Day two



(c) Day three



(d) Day four



(e) Day five

Figure 3.9: Schedule of a week for 50 operations on training set

In the figure above (3.9), we illustrate the schedule of 50 operations. As we can see, all 50 operations were successfully scheduled; however, there is an evident variation in the number of operations assigned to each day. The first day shows a higher volume of operations (20), while the subsequent days have a lower number of scheduled operations. On the last day, only two operations were scheduled.

This distribution discrepancy can be attributed to resource availability considerations. The scheduling algorithm appears to prioritize operations with lower resource requirements on the first day. As the week progresses, operations that require more time and resources are accommodated, resulting in a reduced capacity for scheduling additional operations each day.

3.4.5.2 Results of the real-world dataset for testing

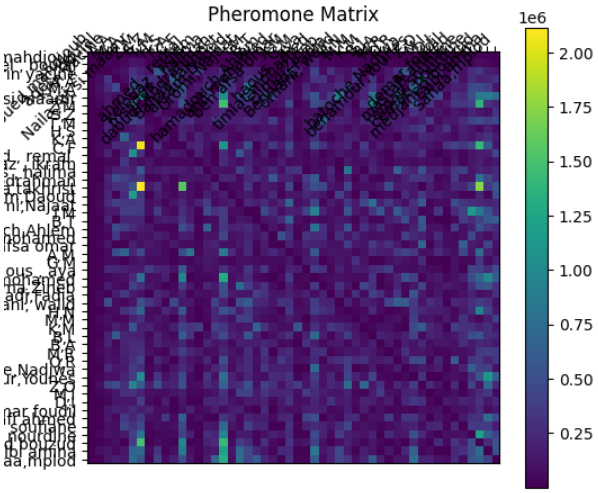


Figure 3.10: Pheromone matrix

In the figure above (3.10), we present the Pheromone matrix, which serves as a powerful tool for assessing the likelihood of scheduling each operation compared to others. Visualized with a color gradient, the intensity of the yellow shade within the matrix corresponds to the probability of scheduling. A brighter shade indicates a higher likelihood, while areas without color represent operations with lower probabilities.

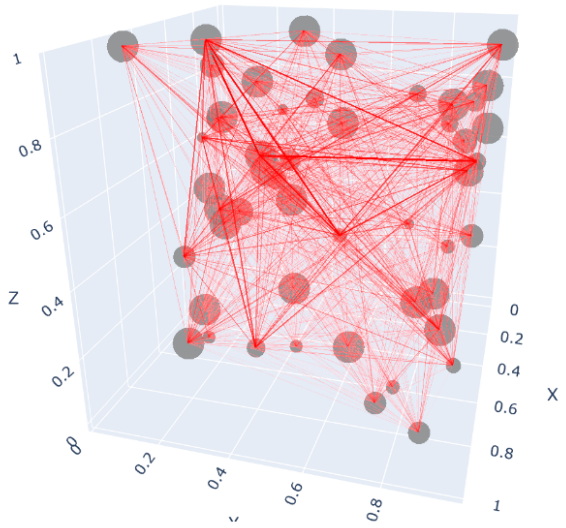


Figure 3.11: Pheromone

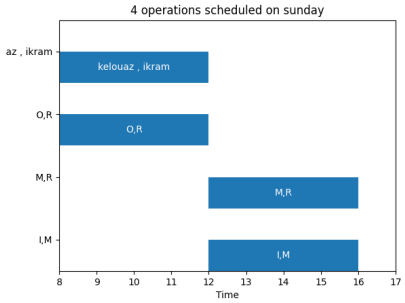
From figure 3.11,one can gain valuable insights into the relative importance of operations and their levels of exploration during the scheduling process. The size of the nodes provides a visual indication of the operations' significance and importance, while the color intensity on the arcs offers insights into the frequency of visits by the ants. This information aids in understanding the dynamics of the scheduling process and can inform decision-making for optimizing the overall scheduling strategy.

Number of Operation	Instances	Metrics	Best	Avg
10 Operation	instance 1	Fitness	6.2972	6.6532
		CPU time Best	11.2344	12.1612
15 Operation	instance 2	Fitness	3.7149	3.8674
		CPU time Best	26.3942	25.4700
20 Operation	instance 3	Fitness	6.3499	7.3246
		CPU time Best	77.1890	74.3650
30 Operation	instance 4	Fitness	6.7113	7.1692
		CPU time Best	75.4859	77.2200
50 Operation	instance 5	Fitness	7.8484	8.7897
		CPU time Best	172.3295	152.2048
100 Operation	instance 6	Fitness	10.4113	11.7402
		CPU time Best	371.7434	369.7136

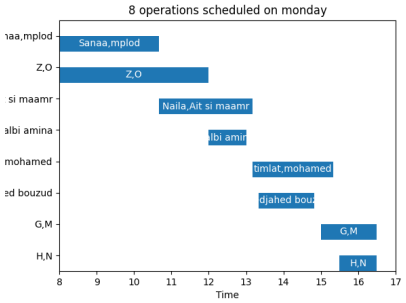
Table 3.4: Experiment results for the Minimization problem (10 run/instance) for the real dataset

Table 3.4 presents the results obtained from running our algorithm 10 times on six different instances of the Minimization problems, each with varying numbers of operations.

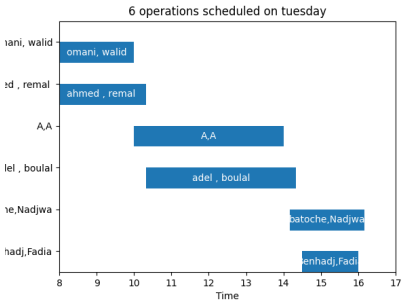
From the results of the table, we can say that for large instances, the algorithm takes a long time to find the best solutions CPU = $\sim$370s where it converges quickly when instances are smaller such as in instance 1 CPU time= 11 $\sim$ 12s.



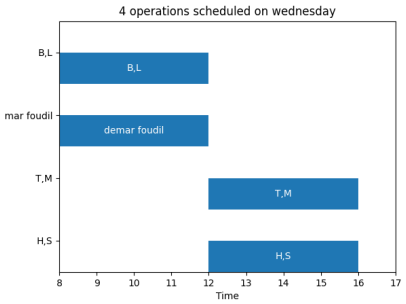
(a) Day one



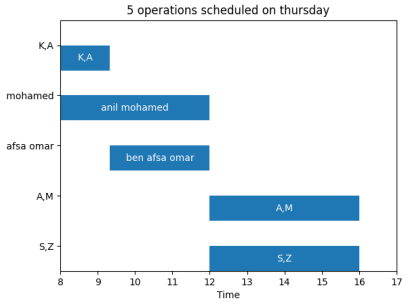
(b) Day two



(c) Day three



(d) Day four



(e) Day five

Figure 3.12: Schedule of a week for 50 operations on the real-world dataset.

The schedule for a week in figure 3.12 demonstrates a well-balanced distribution again of operations across all days for the real-world data. Each day consistently accommodates a similar number of operations, ranging from 4 to 8 operations per day. It should be noted that out of the total 50 operations, only 27 have been scheduled, and that is due to the difficulty of satisfying the constraint of time. By maintaining an equitable distribution of operations, the schedule facilitates effective operational planning and execution. It helps to prevent any overwhelming workload on specific days and allows for efficient utilization of resources throughout the week.

3.4.5.3 comparison study: Training vs. real-world dataset

Two different schedules for a week are presented, representing distinct scenarios: one derived from a training dataset generated by us, and the other based on a real-world dataset from a hospital case study.

In the schedule generated from the training dataset, all 50 operations were successfully scheduled. However, there is a noticeable variation in the daily distribution of operations. The first day accommodated a relatively higher number of operations (20), while the subsequent days had fewer operations scheduled. On the last day of the week, only 2 operations were scheduled. This variation is attributed to resource availability, as operations with fewer resource requirements were prioritized initially, leading to a gradual decrease in the number of operations scheduled each day.

In contrast, the schedule based on the real-world dataset demonstrates a more balanced distribution of operations throughout the week. Each day consistently accommodates a similar number of operations, ranging from 4 to 8 operations. It is important to note that out of the total 50 operations, only 27 have been scheduled in this scenario which is due to the difficulty of satisfaying all constraints in real-world.

Comparing these two schedules reveals differences in distribution patterns, dataset origin, fitness values, and computation time. These factors play a crucial role in evaluating and understanding the performance and effectiveness of the scheduling solutions. By analyzing these schedules, we gain insights into the impact of dataset characteristics and resource constraints on the scheduling process. This knowledge contributes to the advancement of efficient scheduling strategies and aids in improving operational planning in various domains.

3.4.5.4 Comparison study: Max objective schedules vs. Min objective schedule

In this sub-section, we provide a comparative study of the schedules obtained when having two different objectives and constraints. We take the example of an experiment of the instance with 50 operations on the real-world data collected from the hospital. The results are illustrated in the following table 3.5:

Days	Aco Objective 1: Max		Aco Objective 2: Min	
	Training Data	Real Data	Training Data	real-world Data
Day 1	22	7	20	4
Day 2	10	8	14	8
Day 3	10	6	8	6
Day 4	5	6	6	4
Day 5	3	5	2	5
sum	50	32	50	27

Table 3.5: Comparison of schedules in Max and Min Objectives

From the table 3.5 above, we can conclude that working on different objectives for finding the best schedule can vary from one objective to another, where in the first case (maximizing the number of important operations to be scheduled), the algorithm was able to schedule 32 operations out of 50 because it focuses only on the importance of the operation by taking into consideration the factors as resources available, however, on the other case (minimizing the weighted tardiness of operations) the algorithm was able to schedule less number of operations 27 out of 50 compared to the first case, and that is as a consequence to the difficulty of controlling such constraint of tardiness, as we know few operations could exceed the real planned time, next to that we have the availability of resources and staff which makes this second objective more complex to achieve.

3.5 Conclusion

In this chapter, we proposed two different models to represent the Operations Research Scheduling Problem (ORSP) and presented our adapted Ant Colony Optimization (ACO) algorithm to solve these models. Our ACO algorithm was tailored to address the challenges of scheduling tasks in real-world scenarios, incorporating different ant-searching structures and objective functions specifically designed for the ORSP.

Through experiments conducted using training and real-world datasets of varying sizes and resource configurations, we evaluated the performance of our approach. The experiments focused on two proposed models: Objective 1, which aimed to maximize the total number of scheduled operations, and Objective 2, which aimed to minimize the total weighted tardiness of operations.

The experimental results demonstrated the effectiveness of our approach in optimizing scheduling for the ORSP. By systematically exploring the solution space using our ACO algorithm, we were able to achieve high-quality schedules that met the objectives of maximizing the number of scheduled operations and minimizing the weighted tardiness of operations.

Furthermore, we conducted a comparison study between the training and real-world datasets for both Objective 1 and Objective 2. This comparison provided insights into the performance of our approach in different data scenarios, allowing us to assess its robustness and adaptability.

Based on the experimental results and to conclude this chapter, we can say that solving the operating room scheduling problem could lead to many different solutions and all of them are valid based on the objective taken into consideration, however, managing real-world data from the real-world is a complex task, therefore, finding an exact optimum solution is not possible and that is one of the reasons why we use only an approximate method in this work.

Chapter 4

Multi-objective ACO and Semi-Automatic Design for solving the ORSP

4.1 Introduction

4.2 Definition of Multi-Objective problem

The multi-objective problem involves the simultaneous optimization of multiple objectives while considering the constraints of all objectives. It aims to find a set of solutions that offer a good trade-off between different objectives. In this type of problem, there is an objective function vector $f(x)$ that represents the goals to be minimized or maximized and a set of decision variables X. Each solution in the decision variable space corresponds to a point in the objective function space. The effectiveness of a solution is evaluated using a function $f : X \rightarrow Y$, which assigns an objective function vector $(y_1, y_2, ..., y_k)$ in the objective space Y. Additionally, multi-objective problems may include inequality constraints $g_i(x)$ and equality constraints $h_j(x)$, which further shape the optimization process by considering limitations and requirements.[34]

4.2.1 Multi-Objective ORSP

In this part of the work, we have combined both previous objectives mentioned in chapter 3, to turn the problem into a Multi-Objective Optimization Problem (MOOP).

In ORSP and real-world problems, we usually have different objectives to satisfy at the same time, therefor we consider a Multi-Objective ORSP where we need to find an optimal solution for two objectives in the ORSP, in our case we would like to maximize the number of operation scheduled and in the same time to minimize the weighted tardiness for operations in the schedule.

MO-ORSP is a challenging problem that aims to optimize the performance of operating rooms in a hospital by balancing the needs and preferences of patients, medical

staff, and hospital management. It involves assigning operating rooms to surgical specialties, determining the order and duration of surgeries, and allocating other resources such as nurses, equipment, and facilities. The problem is multi-objective because there are several criteria to evaluate the quality of a solution, such as patient satisfaction, staff satisfaction, resource utilization, cost efficiency, etc. These criteria are often conflicting and trade-offs have to be made. MO-ORSP is also combinatorial because there are many possible ways to arrange the operating rooms and surgeries, and finding the optimal or near-optimal solution is computationally difficult.[44]

4.2.1.1 Model Description

To integrate the goals of maximizing the number of operations scheduled and minimizing the weighted tardiness, we can employ a multi-objective optimization approach utilizing the weighted sum method. By assigning appropriate weights to each objective, we can combine them into a unified mathematical model that considers both objectives simultaneously. This allows us to find a set of solutions that achieve a balance between maximizing the number of operations scheduled and minimizing the weighted tardiness, taking into account the relative importance of each objective as determined by their respective weights.

Objectives:

1. Maximize the total number of scheduled operations:
   Maximize $\sum_{i=1}^{n} x_i \cdot p_i$.

2. Minimize the total weighted tardiness of surgeries:
   Minimize $\sum_i w_i \cdot T_i$.

Multi-objective Mathematical Formulation: After combining both objectives using the weighted sum formulation, the multi-objective mathematical formulation become like following:

$$Max \quad \alpha \cdot \left(\sum_i x_i \cdot p_i \right) + (1 - \alpha) \cdot \left(\sum_i w_i \cdot T_i \right)$$

In this formulation, α is a weight parameter that determines the trade-off between the two objectives. When $\alpha = 1$, the objective is to maximize the number of operations scheduled. When $\alpha = 0$, the objective is to minimize the total weighted tardiness. By adjusting the value of α, you can control the emphasis given to each objective.

In the weighted sum method of multi-objective optimization, the value of α determines the trade-off between the objectives. The choice of α depends on the relative importance assigned to each objective by the decision-maker. The literature does not provide specific universally applicable values for α, as it heavily depends on the specific problem context

and the decision-maker's preferences. However, here are some general examples of α values that have been used in the literature:

1. Equal Weights: Setting $\alpha = 0.5$ gives equal weights to both objectives. This implies equal importance placed on maximizing the number of operations scheduled and minimizing the total weighted tardiness.

2. Preference-Based: The choice of α can be guided by the preferences of the decision-maker. For example, if the decision-maker considers maximizing the number of operations scheduled to be twice as important as minimizing the total weighted tardiness, they may set $\alpha = 0.67$ $(2/3)$ to reflect this preference.

3. Sensitivity Analysis: A sensitivity analysis can be conducted by varying α within a specific range, such as $\alpha = 0.1, 0.2, 0.3, ..., 0.9$. This allows for examining the trade-offs and observing how different values of α influence the solution space and the Pareto frontier.

It is important to note that the choice of α is subjective and context-dependent. It should be based on the decision-maker's preferences and the specific requirements of the surgery room scheduling problem. Sensitivity analysis and interactive decision-making techniques can help in finding a suitable value for α that aligns with the decision-makers preferences.

4.2.2 Experiments with Multi-Objective model

The following results represent the experiments of MO-ACO with equal weight for both objectives where $\alpha = 0.5$. the rest of the experimental setups is the same as in chapter 3 (see subsection 4.3).

4.2.2.1 Results of the training dataset

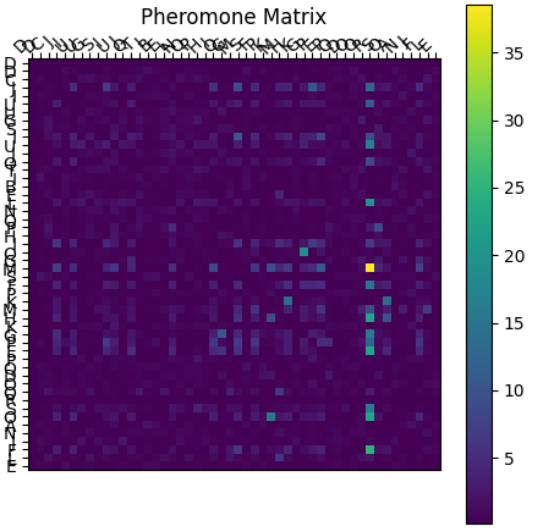


Figure 4.1: Pheromone matrix

Figure 4.1 shows the Pheromone matrix. This matrix helps to evaluate how likely it is to schedule each operation among others. The matrix uses a color gradient to represent the scheduling probability. A more yellow shade means a higher chance, while areas with no color mean lower chances. This figure makes it easy to see the differences in probabilities for each operation. It also reveals the scheduling preferences that the Pheromone matrix sets, which can help to make better choices in using resources and improving the scheduling process.

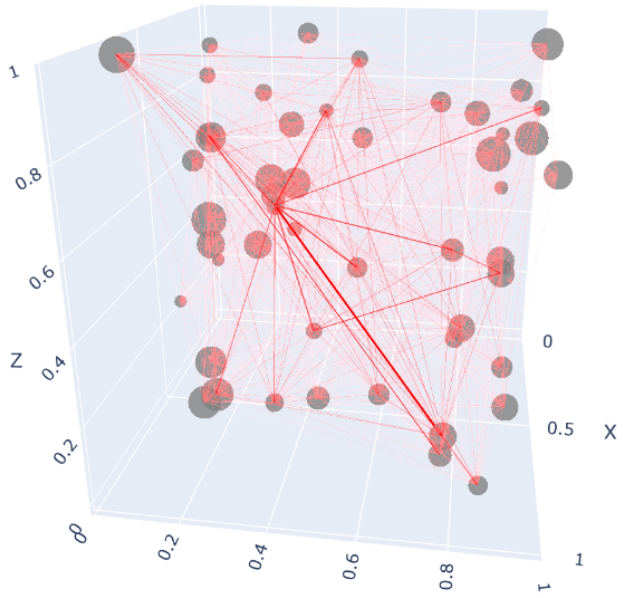


Figure 4.2: Operations' exploration Graphic based on Pheromones on training data

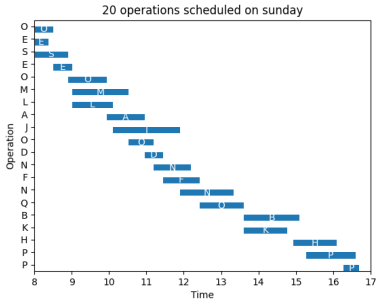
The Pheromone matrix is illustrated differently in the figure 4.2, which helps to understand how operations are related in the scheduling process. Each operation is a node in the matrix, and the bigger the node is, the more important the operation is for being in the schedule. The nodes are connected by arcs that show how many times the ants have visited each operation during the scheduling. The red color on the arcs shows how often the ants have visited them. The darker the red, the more visits, which means that some operations have been more explored and chosen by the ants. This figure can help to see how important and explored each operation is during the scheduling process. The node size shows the importance of the operations, and the color of the arcs shows how often the ants have visited them. This can help to know how the scheduling process works and to make better decisions for making it more efficient.

Number of Operation	Instances	Metrics	Best	Avg
10 Operation	Instance 1	Fitness	1469.5280	1467.6714
		CPU time Best	9.5643	10.8647
15 Operation	Instance 2	Fitness	2745.7540	2729.4698
		CPU time Best	16.7429	18.7256
20 Operation	Instance 3	Fitness	3572.8300	3417.6176
		CPU time Best	27.5075	32.1998
30 Operation	Instance 4	Fitness	5468.4500	5378.1456
		CPU time Best	59.4500	64.8843
50 Operation	Instance 5	Fitness	9084.6080	8915.5552
		CPU time Best	157.1919	164.3137
100 Operation	Instance 6	Fitness	15491.4300	15186.325
		CPU time Best	500.0602	529.8536

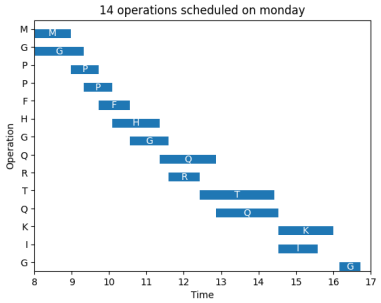
Table 4.1: Experiment results for the Multi-objective problem (10 run/instance) for the training set

Upon analyzing the data presented in Table 4.1, a clear pattern emerges: as the number of operations increases, there is a consistent improvement in the best fitness values obtained.

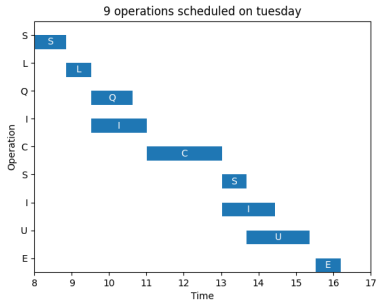
In summary, the experimental findings indicate that the algorithm exhibits enhanced performance when confronted with instances featuring a higher number of operations. Nevertheless, it is imperative to consider the trade-off between achieving better fitness values and the associated increase in computational time. These insights contribute significantly to our understanding of the algorithm's performance characteristics in addressing the Multi-objective problem.



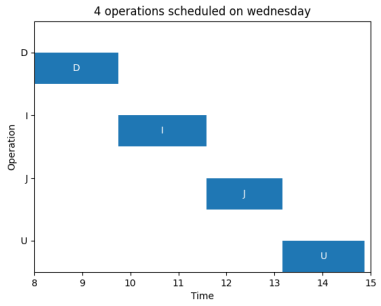
(a) Day one



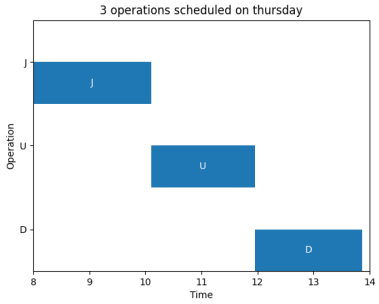
(b) Day two



(c) Day three



(d) Day four



(e) Day five

Figure 4.3: Schedule of a week for 50 operations on training set

The figure 4.3 above illustrates a weekly schedule for instance 5 on the training dataset, and the obtained schedule shows that all 50 operations were scheduled successfully, but, the number of operations for each day is different. The first day has more operations (20), while the next days have fewer operations. The last day has only 3 operations. This difference in distribution is because of resource availability. The scheduling algorithm seems to choose operations that need fewer resources on the first day. Then, as the week goes on, it schedules operations that need more time and resources, which means that there is less space for more operations each day.

4.2.2.2 Results of the real-world dataset for testing

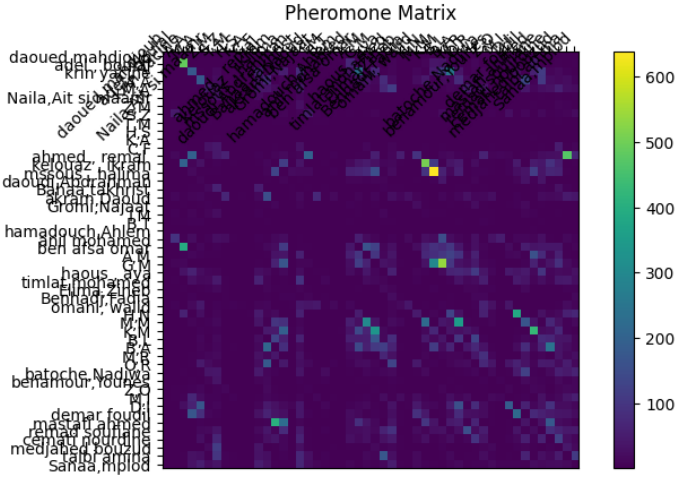


Figure 4.4: Pheromone matrix

This figure 4.4 shows the Pheromone matrix, which helps to compare how likely each operation is to be scheduled. The matrix uses a color gradient to show the scheduling probability. A more yellow color means a higher chance, while no color means a lower chance. This figure makes it easy to see how probable each operation is. It also shows the scheduling preferences that the Pheromone matrix has, which can help to make smarter choices in using resources and making the scheduling process better.

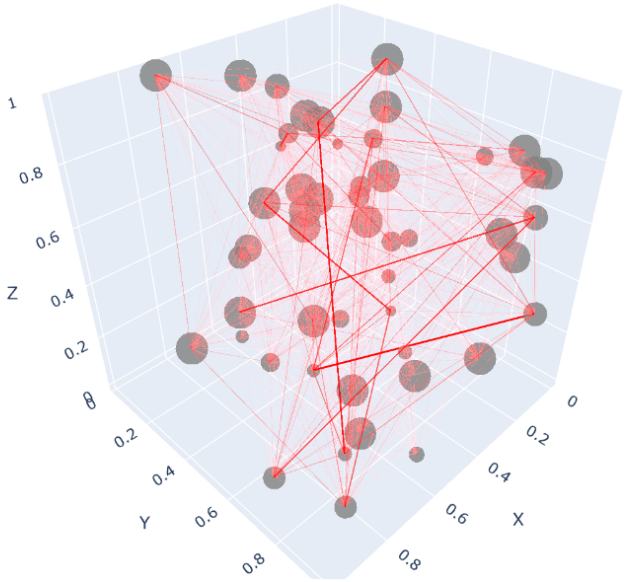


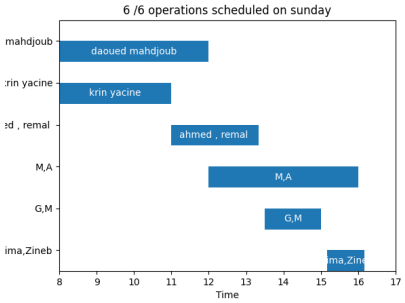
Figure 4.5: Operations' exploration Graphic based on Pheromones on real-world data

The figure 4.5 presented here above showcases the Pheromone matrix exploration graphic, a crucial component for understanding the interrelationships among operations in the scheduling process. In this matrix, each operation is depicted as a node, with its size indicating its relative importance in scheduling. The arcs connecting the nodes represent the frequency of visits by the ants during the scheduling process, with the intensity of red color reflecting the visitation frequency. Darker shades of red signify more frequent visits, suggesting that certain operations have been extensively explored and selected by the ants. This insightful visualization allows us to gauge the significance and exploration level of each operation throughout the scheduling process. By considering the node sizes and the color intensity on the arcs, we gain valuable insights into the dynamics of the scheduling process and can make informed decisions to enhance its efficiency.

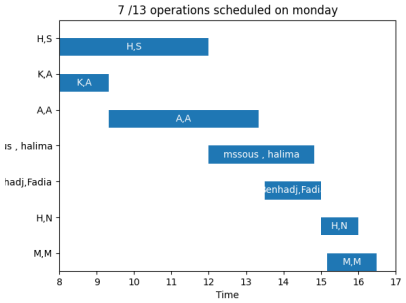
Number of Operation	Instances	Metrics	Best	Avg
10 Operation	instance 1	Fitness	1256.7320	1240.7638
		CPU time Best	10.8951	11.4215
15 Operation	instance 2	Fitness	1700.8410	1687.2634
		CPU time Best	22.22181	21.0346
20 Operation	instance 3	Fitness	3503.6070	3434.5700
		CPU time Best	65.7958	69.9779
30 Operation	instance 4	Fitness	3515.4430	3454.9270
		CPU time Best	65.5806	71.7535
50 Operation	instance 5	Fitness	4980.4853	4833.1167
		CPU time Best	131.2099	132.0776
100 Operation	instance 6	Fitness	8761.054	8406.7210
		CPU time Best	337.9579	367.0906

Table 4.2: Experiment results for the Multi-objective problem (10 run/instance) for the real-world dataset

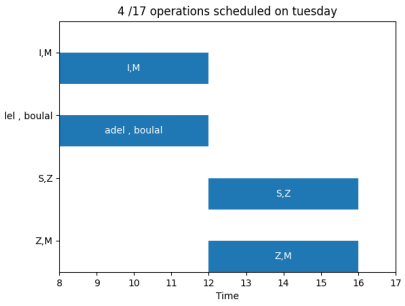
By examining the results in Table 4.2, we observe a consistent trend where the best fitness values tend to increase as the number of operations increases. This indicates that the algorithm's complexity increases the larger the size of the instance, which makes the computational time bigger.



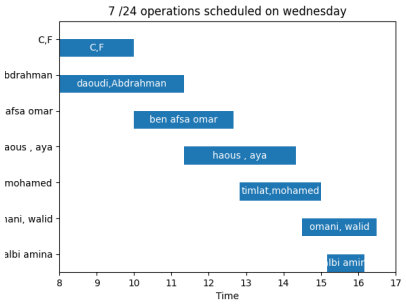
(a) Day one



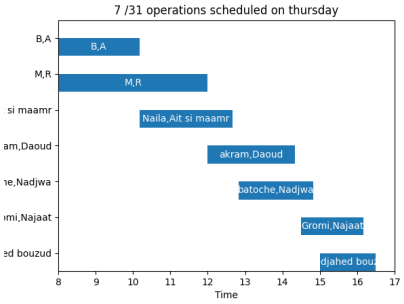
(b) Day two



(c) Day three



(d) Day four



(e) Day five

Figure 4.6: Schedule of a week for 50 operations on real-world dataset

The following weekly schedule represented in figure 4.6 represents the results of instance 5 for scheduling 50 operations using the ACO for the multi-objective model. It exemplifies a commendable equilibrium in distributing operations across all days. Each day consistently handles a similar number of operations, typically ranging from 4 to 7 operations per day. It is important to note that out of the total 50 operations, only 31 have been assigned a schedule.

The decision to evenly distribute the operations throughout the week takes into account various factors, including resource availability, staff availability, and other operational considerations such as tardiness in this case. This approach ensures a fair allocation of workload and promotes a steady and manageable workflow with harder constraints.

4.2.2.3 Comparison study: Training vs. real-world dataset

Two distinct schedules for a week are presented, representing different scenarios: one derived from a training dataset generated by us and the other based on a real-world dataset from a hospital case study on the instance grouping 50 operations to be scheduled.

From the two results, we could see that the distributions on the real-world data are more reasonable for each day and more balanced compared to the training dataset, where the distribution was mostly focused on the first days with a big charge of a number of operations. From here we can say that our approach is more effective on real-world data and that is due to the different factors that contribute to the efficiency of the solution found.

4.3 Schedule results comparison study of the three models based on the number of operations scheduled

The following table 4.3 illustrates a summary of the schedules obtained from the previous experiments of the three different objectives.

In these experiments, we have explored three distinct variants of the multi-objective function, each contingent on the factor, which plays a crucial role in shaping the objectives under consideration. Initially, we set $\alpha = 0.5$ to maintain a balanced weighting between the objectives. Subsequently, we examined $\alpha = 0.33$ and $\alpha = 0.67$, representing scenarios where one objective takes precedence over the other, thereby introducing variability to the objectives' interplay.

Experiment with $\alpha = 0.5$

	Aco Objective 1: Max		Aco Objective 2: Min		Aco Bi-Objective	
	Training Data	Real Data	Training Data	Real Data	Training Data	Real Data
Day 1	22	7	20	4	14	6
Day 2	10	8	14	8	15	6
Day 3	10	6	8	6	14	7
Day 4	5	6	6	4	6	5
Day 5	3	5	2	5	1	7
sum	50	32	50	27	50	31

Table 4.3: Results with alpha = 0.5

From the obtained results we can see that for the real-world dataset, there is a good

balance of the schedule for all the days of the week. In the first objective (Max) 32 operations were scheduled, however, in objective 2 (Min) the number of scheduled operations has decreased, seems like the tardiness constraints are more complex and difficult to be satisfied with the available resources which influence on the performance of the optimal solution. furthermore, in the multi-objective results where both objectives are equaled with an $\alpha = 0.5$, we see that there is a small decrease in the number of operations scheduled compared to objective 1 and that is due to the influence of constraints of tardiness from the objective 2 in the multi-objective function. however, we can note that more than 60% of the operations were scheduled. Thus, we can say that for large-size instances our approach is very promising.

Experiment with $\alpha = 0.33$

In this variant, the fitness factor of the multi-objective function $\alpha = 0.33$, which means that the function's decision considers minimizing the total weighted tardiness to be twice as important as maximizing the number of operations scheduled. The results of this experiment are shown in table 4.4.

	Aco Objective 1: Max		Aco Objective 2: Min		Aco Bi-Objective	
	Training Data	Real Data	Training Data	Real Data	Training Data	Real Data
Day 1	22	7	20	4	14	5
Day 2	10	8	14	8	19	5
Day 3	10	6	8	6	8	6
Day 4	5	6	6	4	7	6
Day 5	3	5	2	5	2	8
Total	50	32	50	27	50	30

Table 4.4: Results with alpha = 0.33

Overall, this table allows for a comprehensive evaluation of the ACO algorithm's performance across different objectives and data types, enabling comparisons and insights into the algorithm's effectiveness under various conditions. The obtained results show that when the algorithm's decision is based on the second objective of minimizing the total weighted tardiness with an $\alpha = 0.33$ are closely similar to the previous experiment where $\alpha = 0.5$, still the number of operations scheduled decreased with one operation due to the influence of the weighted tardiness constraints, though, the algorithm still performs well and 30 out of 50 operations were scheduled (60%). From here we can that our approach is still promising.

Experiment with $\alpha = 0.67$

In this variant, the fitness factor of the multi-objective function $\alpha = 0.67$, which means that the function's decision considers maximizing the number of operations scheduled to be twice as important as minimizing the total weighted tardiness. The results of this experiment are shown in table 4.5.

	Aco Objective 1: Max		Aco Objective 2: Min		Aco Bi-Objective	
	Training Data	Real Data	Training Data	Real Data	Training Data	Real Data
Day 1	22	7	20	4	14	7
Day 2	10	8	14	8	13	7
Day 3	10	6	8	6	16	5
Day 4	5	6	6	4	6	6
Day 5	3	5	2	5	1	6
Total	50	32	50	27	50	31

Table 4.5: Results with alpha = 0.67

The provided table 4.5 shows that in the multi-objective function with an $\alpha = 0.67$ the results change slightly from the previous one but look similar to the results from $\alpha = 0.5$. Thus, from here we can conclude that the total weighted tardiness is a very important objective that influences on finding the optimal solution. If we neglect this objective we can obtain optimal schedules with a higher number of operations but this will not be as realistic as the real-world scenario.

Objective function	data	Total
Aco Objective 1: Max	training	50
	real	32
Aco Objective 1: Min	training	50
	real	27
aco bi-objective (alpha = 0.5)	training	50
	real	31
aco bi-objective (alpha = 0.33)	training	50
	real	30
aco bi-objective (alpha = 0.67)	training	50
	real	31

Table 4.6: Totalmary of Results

For further analysis, more experiments have been done to test the ACO approach, where the algorithm was tested with 30 and 100 operations instances from the real-world dataset, we evaluated the performance of the Ant Colony Optimization (ACO) algo-

rithm using three different values of alpha: 0.5, 0.33, and 0.67. The objective functions considered were ACO Objective 1: Max, ACO Objective 2: Min, and ACO Bi-Objective.

Experiment with the instance of 30 Operations

Days	ACO 1: Max	ACO 2: Min	Aco Bi-Objective	Aco Bi-Objective	Aco Bi-Objective
			$\alpha = 0.5$	$\alpha = 0.33$	$\alpha = 0.67$
Day 1	4	4	4	5	4
Day 2	8	5	4	4	5
Day 3	6	4	7	4	8
Day 4	4	7	5	8	5
Day 5	4	4	6	5	4
Total (30 Operation)	26	24	26	26	26

Table 4.7: Results for 30 Operation using the best parameter settings found

The experiment settings used for the following experiments were the same as those of the previous experiments for the instances of 20 and 50 operations.

Experiment with the instance of 100 Operations

Days	ACO 1: Max	ACO 2: Min	Aco Bi-Objective	Aco Bi-Objective	Aco Bi-Objective
			$\alpha = 0.5$	$\alpha = 0.33$	$\alpha = 0.67$
Day 1	8	4	6	4	6
Day 2	10	8	5	6	7
Day 3	6	5	8	12	8
Day 4	10	8	11	7	9
Day 5	5	5	9	10	12
Total (100 Operation)	39	30	39	39	42

Table 4.8: Results for 100 Operation with alpha = 0.5

Based on the results of these experiments, we can conclude that the choice of the alpha parameter impacts the scheduling decisions to some extent. However, the overall performance in terms of the total number of scheduled operations is similar for all three alpha values. The ACO algorithm demonstrates its ability to handle the multi-objective nature of the problem and find near-optimal solutions under different alpha settings.

These findings provide valuable insights into the behavior of the ACO algorithm and inform the selection of appropriate parameter settings for optimizing the OR scheduling problem.

4.4 Semi-Automatic Design for ACO

Metaheuristics have emerged as powerful techniques for solving large and complex optimization problems, demonstrating their effectiveness in obtaining optimal or near-optimal solutions. However, achieving high-performance optimization algorithms often demands significant computational resources and domain expertise [38]. The performance of these algorithms can be influenced by various factors, including the algorithm's characteristics, the problem being addressed, and the execution environment.

To facilitate the configuration process and enhance algorithm performance, several automatic algorithm configuration tools, known as configurators, have been proposed in the literature review. These configurators aim to guide the selection and tuning of algorithm parameters to improve their efficiency and effectiveness. Notable examples of these configurators include ParamILS [21], GGA [3] , and irace[25].

In this work, we employed a semi-automatic configuration inspired by irace configurator to determine the best parameters' values that increase the performance of the algorithm. A script was developed to automate the process of running the algorithms multiple times with different configurations of parameters' values, allowing for a systematic exploration of the solution space. The objective of the semi-automatic configuration experiment was to find the good/best combination of parameter settings that maximizes the performance of the algorithm. To apply this strategy, we consider the following parameters which represent the most influencing parameters of the ACO algorithm: $Alpha$, $Beta$, $Numberofiterations$, $Numberofants$, Q, $Alphafitness$. We run the algorithm with a limited run time on the semi-automatic configuration technique for 25 configurations. During the execution of the automated design, the algorithm generates a new set/configuration for the parameters being tuned at each execution by keeping track of the best solution found so far and the values of the parameters that led to this solution.

The values of the considered parameters were selected from an interval as described below:

Name	Switch	Type	Valuse
alpha	"–alpha"	r	(0.1 , 1.0)
beta	"–beta"	r	(0.1 , 1.0)
Q	"–Q"	i	(1 , 1000)
iteration_number	"–iteration_number"	i	(25 , 50)
ants_in_colony	"–ants_in_colony"	i	(25 , 50)
alpha_fitness	"–alpha_fitness"	r	(0.1 , 0.99)

Table 4.9: Structure of tuned parameters

where in the type column: "i" signifies "integer" and "r" means "real".

The descriptions used in the "Switch" column represent the way how these parameters are called and pass their values on the command line to be executed on the Linux OS.

4.4.1 Experimental Results of the semi-automated design strategy for ACO

Table 4.10 shows the values of the best and the average result of fitness given by our semi-automated design strategy for each Objective.

Data	**Operation**		**Objective**		
			Maximization	**Minimization**	**Bi-Objective**
Real Data	20	**Best**	7753.6250	3.4482	5313.501
		Avg	7171.9999	4.4841	2998.3328
	50	**Best**	10388.1300	7.1633	4977.6110
		Avg	9916.4641	8.8269	4579.9031
Training Data	20	**Best**	9160.1670	3.2981	4415.489
		Avg	7006.8334	28.3768	3417.6176
	50	**Best**	19074	3.8066	9105.2390
		Avg	18646.8143	2.8352	8915.5552

Table 4.10: Results of the semi-automated design strategy

The following tables 4.11 and 4.12 describe a comparison study between the best value found by ACO and the best value found by the semi-automated ACO algorithms for the instances with 20 and 50 operations.

20 Operation	alpha	beta	number of iterations	number of ants	Q	alpha_fitness	Fitness
Maximization (standard)	1.00	1.00	50	25	1000	-	7212.3333
Maximization(best config)	0.15	0.99	27	38	293	-	**7753.6250**
Minimization (standard)	1.00	1.00	50	25	1000	-	7.5574
Minimization(best config)	0.95	0.69	50	49	864	-	**3.4482**
Bi-Objective (standard)	1.00	1.00	50	25	1000	0.5	3525.4817
Bi-Objective(best config)	0.43	0.67	30	40	856	0.96	**5313.501**

Table 4.11: Comparison results for instance 2: 20 operation

50 Operation	alpha	beta	number of iterations	number of ants	Q	alpha_fitness	Fitness
Maximization (standard)	1	1	50	25	1000	-	10102.5833
Maximization(best config)	0.9	0.78	38	38	107		**10388.1300**
Minimization (standard)	1	1	50	25	1000	-	7.8484
Minimization(best config)	0.16	0.99	40	45	803		**7.1633**
Bi-Objective (standard)	1	1	50	25	1000	0.5	4718.4793
Bi-Objective(best config)	0.61	0.89	30	39	181	0.48	**4977.6110**

Table 4.12: Comparison results for instance 5: 50 operations

From the tables 4.11 and 4.12, we can see that the results obtained by our semi-automated design strategy for ACO outperformed the results obtained by the standard ACO for all 3 different objectives. From here we can say that our proposed design of seùi-automated ACO is promising. For further analysis, we run more experiments for instances 20 and 50 operations using parameter settings of the near-optimal solution found by the semi-automated ACO algorithm. The following table 4.13 illustrates a comparison study of the two approaches in terms of CPU time.

	Aco Objective 1: Max				Aco Objective 2: Min				Aco Bi-Objective			
	20 Op		50 Op		20 Op		50 Op		20 Op		50 Op	
	SA-ACO	ACO	SA-ACO	ACO	SA-ACO	ACO	SA-ACO	ACO	SA-ACO	ACO	SA-ACO	ACO
Day 1	7	7	6	7	5	4	4	4	4	4	6	6
Day 2	4	4	6	8	4	7	7	8	8	8	8	6
Day 3	5	5	6	6	6	4	4	6	4	4	5	7
Day 4	4	4	7	6	4	4	7	4	4	4	6	5
Day 5	0	0	6	5	1	1	4	5	0	0	6	7
Total	20	20	31	32	20	20	26	27	20	20	31	31
Time(s)	60	77	120	163	60	70	120	172	60	70	120	131

Table 4.13: Comparison study for ACO vs. SE-ACO

The experiments were conducted with the parameters settings obtained from the semi-automated ACO (table 4.12), and the obtained results in table 4.13 show that with good parameter settings, we can achieve similar results as before but in less execution time. So far, as we can see, the Semi-automated ACO (SA-ACO) has nearly similar results for the single objective models and the same result in the bi-objective model for a shorter execution time than ACO. We assume that if we execute SA-ACO for a longer time we might achieve better results than the current one.

4.5 Conclusion

In this chapter, we proposed a bi-objective model for the operating room scheduling problem. This model combines two objectives: maximizing the total number of scheduled operations and minimizing the total weighted tardiness of operations. This model aims to strike a balance between operational efficiency and meeting time constraints in a closer way to what exists in real-world problems. Three different variants of the bi-objective model have been considered to test our approach, based on the α factor that influences on the decision taken by the objective function.

Additionally, we introduced a Semi-Automatic Design for ACO, inspired by the irace configurator. This approach enabled us to systematically explore the parameter space and identify near-optimal parameter values for the ACO algorithm that lead to finding

a near-optimal solution. By automating the parameter tuning process, we were able to improve the performance and effectiveness of the algorithm in finding better-quality solutions.

The proposed approach offers a promising avenue for addressing the ORSP problem with a focus on multi-objective optimization and automated parameter configuration. Future research can build upon these foundations by incorporating additional constraints and objectives.

General Conclusion

In this work, we have presented a comprehensive study on the optimization of the operating room scheduling problem (ORSP) using Ant Colony Optimization (ACO) algorithm which is a challenging and complex task. The primary objective of our research work was to find an efficient and effective approach to address the challenges posed by the multi-objective nature of the problem.

To address this challenge, we could summarize our work into the following points.

- We proposed three mathematical models for the ORSP: model 1 aimed to maximize the total number of scheduled operations, while model 2 aimed to minimize the total weighted tardiness of operations These objectives were selected based on their significance in improving the overall performance of operating room scheduling. the third model consists of a bi-objective model that combines models 1 and 2.

- We designed and implemented a Multi-objective ACO model that incorporated different ant-searching structures and objective functions specifically tailored for the operating room scheduling problem.

- We explored the impact of the α parameter of the weighted sum in the Multi-objective ACO approach. This finding emphasizes the significance of considering the total weighted tardiness objective for achieving realistic and optimal schedules.

- Additionally, we incorporated a semi-automatic configuration strategy inspired by the irace configurator to enhance the performance and effectiveness of the ACO algorithm.

Through extensive experimentation using training and real-world datasets gathered from a hospital in Algeria as a case study, we evaluated the performance of our proposed ACO approach. The experimental results demonstrated the effectiveness of the Multi-objective ACO approach in generating high-quality schedules. Then, after incorporating the semi-automated design into ACO, the experiments showed that this strategy improves the quality of the solution for ACO and increases the performance of the algorithm.

Finally, we could say that the proposed SA-ACO outperformed ACO algorithm in terms of quality of solution and CPU-time.

Perspectives

As perspectives to this work, we suggest:

- The implementation of a fully automated design ACO approach using the configurator 'irace'.
- Testing ACO approach on other real-world datasets.
- Implementing other metaheuristic methods for solving the ORSP

Bibliography

[1] Abderrahim, I. A. *Hybridation et auto-configuration des métaheuristiques pour la résolution du problème d'affectation quadratique à trois dimensions*. PhD thesis, Université Oran1 Ahmed Benbella, 2020.

[2] Almufti, S. M. Historical survey on metaheuristics algorithms. *International Journal of Scientific World 7*, 1 (2019), 1.

[3] Ansótegui, C., Sellmann, M., and Tierney, K. A gender-based genetic algorithm for the automatic configuration of algorithms. In *International Conference on Principles and Practice of Constraint Programming* (2009), Springer, pp. 142–157.

[4] Beliën, J., and Demeulemeester, E. Building cyclic master surgery schedules with leveled resulting bed occupancy. *European journal of operational research 176*, 2 (2007), 1185–1204.

[5] Beliën, J., Demeulemeester, E., and Cardoen, B. A decision support system for cyclic master surgery scheduling with multiple objectives. *Journal of scheduling 12*, 2 (2009), 147.

[6] Błażewicz, J., Ecker, K. H., Pesch, E., Schmidt, G., and Weglarz, J. *Handbook on scheduling: from theory to applications*. Springer Science & Business Media, 2007.

[7] Blum, C., and Roli, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM computing surveys (CSUR) 35*, 3 (2003), 268–308.

[8] Brélaz, D. New methods to color the vertices of a graph. *Communications of the ACM 22*, 4 (1979), 251–256.

[9] Bullnheimer, B. A new rank based version of the ant system: A computational study. *Central Eur. J. Oper. Res. Econ. 7* (1999), 25–38.

[10] Cardoen, B., Demeulemeester, E., and Beliën, J. Operating room planning and scheduling: A literature review. *European journal of operational research 201*, 3 (2010), 921–932.

[11] Ceschia, S., Di Gaspero, L., and Schaerf, A. Educational timetabling: Problems, benchmarks, and state-of-the-art results. *European Journal of Operational Research 308*, 1 (2023), 1–18.

[12] Colorni, A., Dorigo, M., Maniezzo, V., et al. Distributed optimization by ant colonies. In *Proceedings of the first European conference on artificial life* (1991), vol. 142, Paris, France, pp. 134–142.

[13] Cordeau, J.-F., Laporte, G., Savelsbergh, M. W., and Vigo, D. Vehicle routing. *Handbooks in operations research and management science 14* (2007), 367–428.

[14] Dorigo, M., and Gambardella, L. M. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on evolutionary computation 1*, 1 (1997), 53–66.

[15] Dorigo, M., Maniezzo, V., and Colorni, A. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 26*, 1 (1996), 29–41.

[16] Dorigo, M., and Stutzle, T. Ant colony optimization. mit press, cambridge, ma.

[17] Dorigo, M., and Stützle, T. *Ant colony optimization: overview and recent advances*. Springer, 2019.

[18] Fei, H., Chu, C., and Meskens, N. Solving a tactical operating room planning problem by a column-generation-based heuristic procedure with four criteria. *Annals of Operations Research 166* (2009), 91–108.

[19] Gu, C., Liu, Q., and Xiang, W. A modified ant colony optimization for the multi-objective operating room scheduling. In *Advances in Swarm and Computational Intelligence: 6th International Conference, ICSI 2015, held in conjunction with the Second BRICS Congress, CCI 2015, Beijing, China, June 25-28, 2015, Proceedings, Part I 6* (2015), Springer, pp. 197–204.

[20] Hingrajiya, K. H., Gupta, R. K., and Chandel, G. S. An approach for solving multiple travelling salesman problem using ant colony optimization. In *Fourth International conference on Computer engineering and Intelligence system* (2015), pp. 1133–1149.

[21] Hutter, F., Stützle, T., Leyton-Brown, K., and Hoos, H. H. Paramils: An automatic algorithm configuration framework. *arXiv e-prints* (2014), arXiv–1401.

[22] Komaki, G., Sheikh, S., and Malakooti, B. Flow shop scheduling problems with assembly operations: a review and new trends. *International Journal of Production Research 57*, 10 (2019), 2926–2955.

[23] Lamiri, M., Grimaud, F., and Xie, X. Optimization methods for a stochastic surgery planning problem. *International Journal of Production Economics 120*, 2 (2009), 400–410.

[24] Lin, Y.-K., and Li, M.-Y. Solving operating room scheduling problem using artificial bee colony algorithm. In *Healthcare* (2021), vol. 9, MDPI, p. 152.

[25] López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M., and Stützle, T. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives 3* (2016), 43–58.

[26] Marques, I., Captivo, M. E., and Pato, M. V. Scheduling elective surgeries in a portuguese hospital using a genetic heuristic. *Operations Research for Health Care 3*, 2 (2014), 59–72.

[27] Özder, E. H., Özcan, E., and Eren, T. A systematic literature review for personnel scheduling problems. *International Journal of Information Technology & Decision Making 19*, 06 (2020), 1695–1735.

[28] Pardalos, P. M., Wolkowicz, H., et al. *Quadratic Assignment and Related Problems: DIMACS Workshop, May 20-21, 1993*, vol. 16. American Mathematical Soc., 1994.

[29] Pinedo, M., and Hadavi, K. Scheduling: theory, algorithms and systems development. In *Operations Research Proceedings 1991: Papers of the 20th Annual Meeting/Vorträge der 20. Jahrestagung* (1992), Springer, pp. 35–42.

[30] Pisinger, D. Where are the hard knapsack problems? *Computers & Operations Research 32*, 9 (2005), 2271–2284.

[31] Puchinger, J., and Raidl, G. R. Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. In *Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach: First International Work-Conference on the Interplay Between Natural and Artificial Computation, IWINAC 2005, Las Palmas, Canary Islands, Spain, June 15-18, 2005, Proceedings, Part II 1* (2005), Springer, pp. 41–53.

[32] Rahimi, I., and Gandomi, A. H. A comprehensive review and analysis of operating room and surgery scheduling. *Archives of Computational Methods in Engineering 28*, 3 (2021), 1667–1688.

[33] Schiele, J., Koperna, T., and Brunner, J. O. Predicting intensive care unit bed occupancy for integrated operating room scheduling via neural networks. *Naval Research Logistics (NRL) 68*, 1 (2021), 65–88.

[34] Sharma, S., and Kumar, V. A comprehensive review on multi-objective optimization techniques: Past, present and future. *Archives of Computational Methods in Engineering 29*, 7 (2022), 5605–5633.

[35] Snauwaert, J., and Vanhoucke, M. A classification and new benchmark instances for the multi-skilled resource-constrained project scheduling problem. *European Journal of Operational Research* (2022).

[36] Sörensen, K., and Glover, F. Metaheuristics. *Encyclopedia of operations research and management science 62* (2013), 960–970.

[37] Stützle, T., and Hoos, H. H. Max–min ant system. *Future generation computer systems 16*, 8 (2000), 889–914.

[38] Stützle, T., and López-Ibáñez, M. Automated design of metaheuristic algorithms. *Handbook of metaheuristics* (2019), 541–579.

[39] Su, M.-C., Lai, S.-C., Wang, P.-C., Hsieh, Y.-Z., and Lin, S.-C. A somo-based approach to the operating room scheduling problem. *Expert Systems with Applications 38*, 12 (2011), 15447–15454.

[40] Talbi, E.-G. A taxonomy of hybrid metaheuristics. *Journal of heuristics 8* (2002), 541–564.

[41] Trevisan, L. Combinatorial optimization: exact and approximate algorithms. *Standford University* (2011).

[42] Voudouris, C., and Tsang, E. Guided local search and its application to the traveling salesman problem. *European journal of operational research 113*, 2 (1999), 469–499.

[43] Wojakowski, P., and Warżołek, D. The classification of scheduling problems under production uncertainty. *Research in Logistics & Production 4*, 3 (2014), 245–256.

[44] Xiang, W. A multi-objective aco for operating room scheduling optimization. *Natural Computing 16* (2017), 607–617.

[45] Xiang, W., Yin, J., and Lim, G. An ant colony optimization approach for solving an operating room surgery scheduling problem. *Computers & Industrial Engineering 85* (2015), 335–345.

[46] Yin, J., and Xiang, W. Ant colony algorithm for surgery scheduling problem. pp. 198–205.

[47] Zhang, C., Song, W., Cao, Z., Zhang, J., Tan, P. S., and Chi, X. Learning to dispatch for job shop scheduling via deep reinforcement learning. *Advances in Neural Information Processing Systems 33* (2020), 1621–1632.

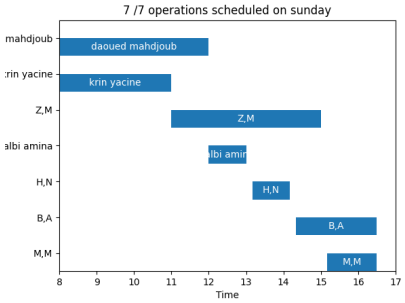
Appendix

4.6 Schedule of the best result of SE-ACO

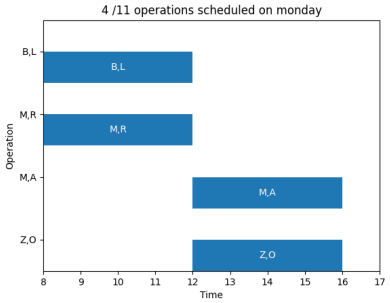
4.6.1 Objective 1: maximization

4.6.1.1 with 20 operations:

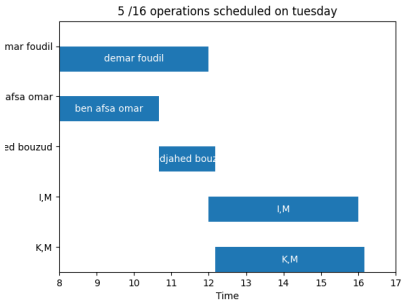
The weekly schedule depicted in Figure 4.7 exemplifies the outcomes obtained for instance 3, focusing on scheduling 20 operations for the maximization model. Notably, the schedule demonstrates the efficient allocation of all 20 operations within the first four days, showcasing the ability to optimize resource utilization and maximize the number of scheduled operations.



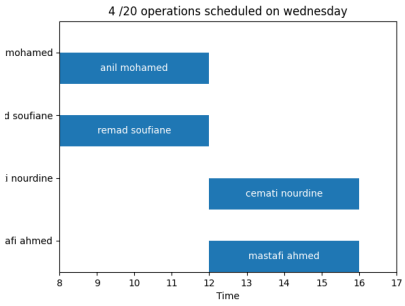
(a) Day one



(b) Day two



(c) Day three

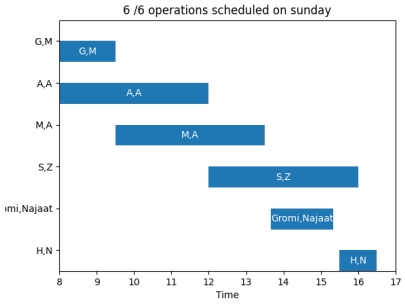


(d) Day four

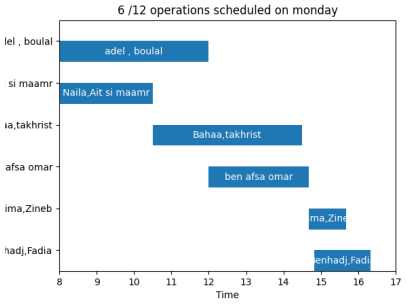
Figure 4.7: Schedule of a week for 20 operations with max on best

The weekly schedule depicted in Figure 4.8 exemplifies the outcomes obtained for instance 3, focusing on scheduling 50 operations for the maximization model. the schedule exhibits an equal distribution of operations across all days, with 31 operations scheduled throughout the week. This balanced allocation emphasizes the effective utilization of resources and ensures a consistent workflow throughout the week.

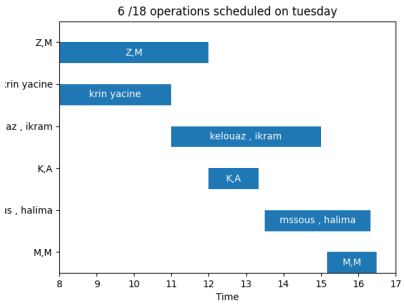
4.6.1.2 with 50 operations:



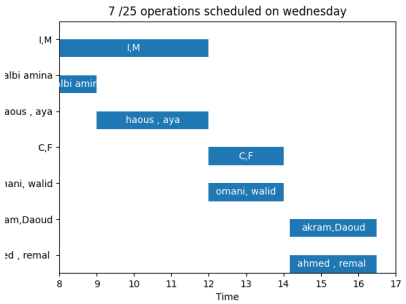
(a) Day one



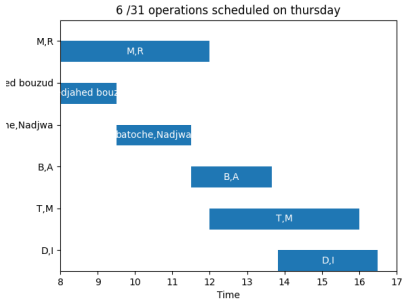
(b) Day two



(c) Day three



(d) Day four



(e) Day five

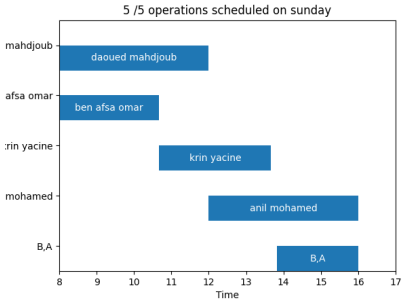
Figure 4.8: Schedule of a week for 50 operations with max on best

4.6.2 Objective 2: Minimization

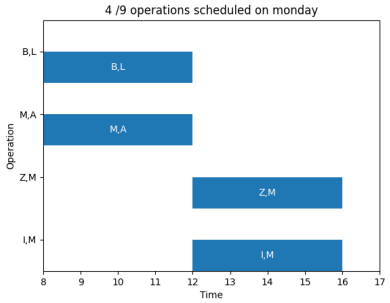
The weekly schedule represented in Figure 4.9 showcases the results achieved for instance 3 in the context of scheduling 20 operations using the minimization model. Notably, the schedule demonstrates the efficient allocation of all 20 operations, with only one operation scheduled on the last day. This indicates the successful minimization of tardiness, as it ensures that operations are completed as close to their target times as possible,

resulting in a more optimal and timely schedule.

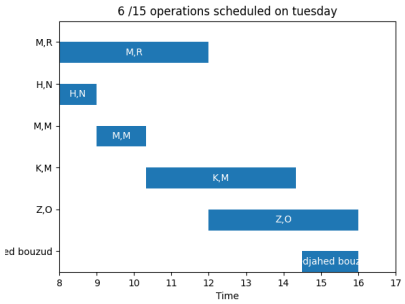
4.6.2.1 with 20 operations:



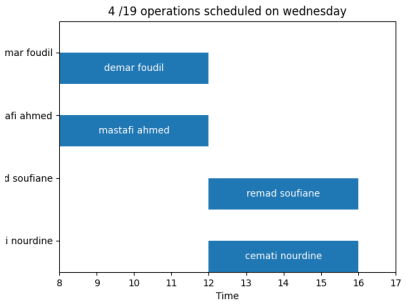
(a) Day one



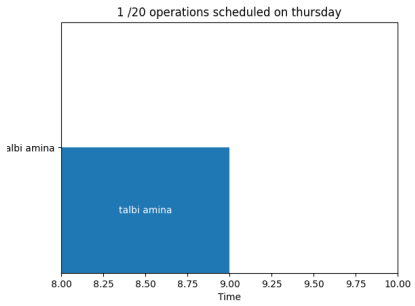
(b) Day two



(c) Day three



(d) Day four



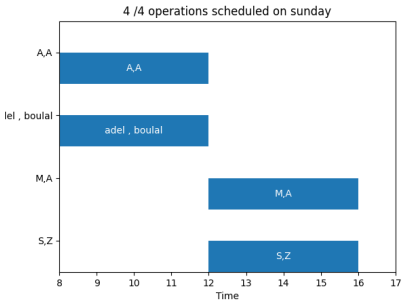
(e) Day five

Figure 4.9: Schedule of a week for 20 operations with min on best

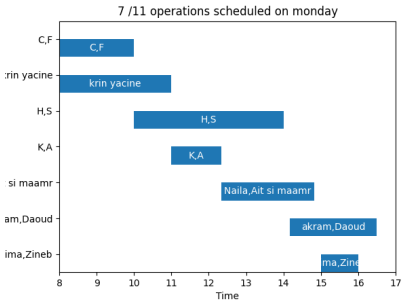
The weekly schedule shown in Figure 4.10 illustrates the results obtained for instance 3, specifically for the minimization model with 50 operations. Notably, the schedule reveals that only 26 operations were scheduled, indicating the focus on minimizing tardiness and ensuring operations are completed as close to their target times as possible. This selective schedul-

ing approach highlights the trade-off between maximizing the number of operations and minimizing tardiness, ultimately aiming to achieve a more optimized and efficient schedule.

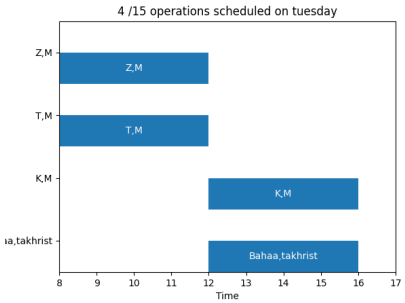
4.6.2.2 with 50 operations:



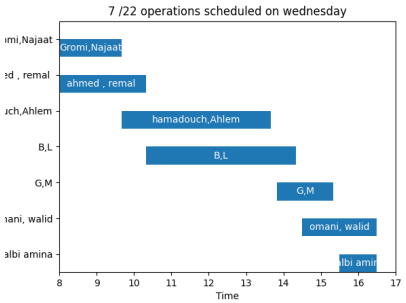
(a) Day one



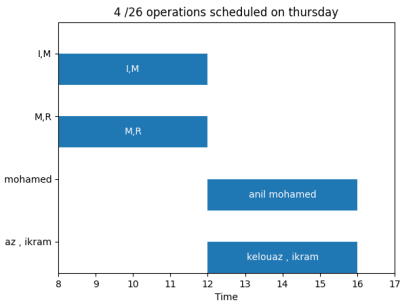
(b) Day two



(c) Day three



(d) Day four



(e) Day five

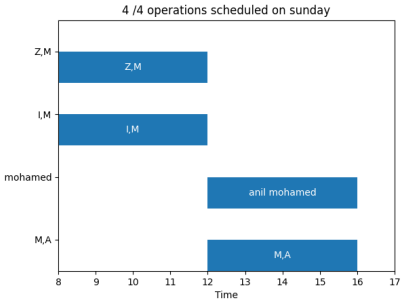
Figure 4.10: Schedule of a week for 50 with min operations on best

4.6.3 Objective 3: Bi-Objective

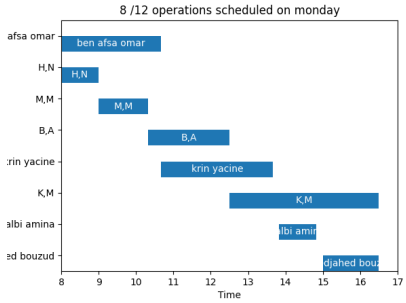
The weekly schedule presented in Figure 4.11 showcases the results obtained for instance 3, where the bi-objective model was utilized to sched-

ule 20 operations. It is noteworthy that the schedule demonstrates the effective allocation of all 20 operations, successfully accommodating them within the first four days. This optimized scheduling approach highlights the ability to balance both objectives and achieve a favorable distribution of operations throughout the week.

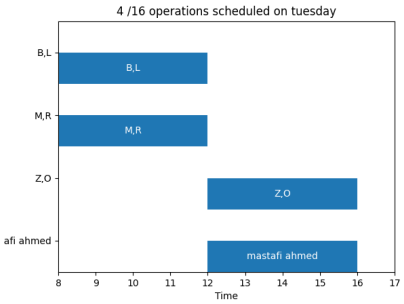
4.6.3.1 with 20 operations:



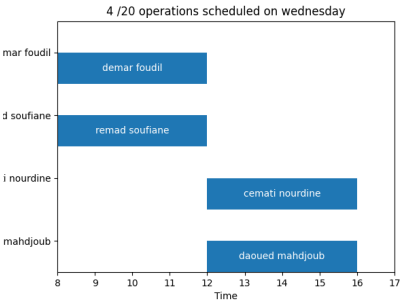
(a) Day one



(b) Day two



(c) Day three

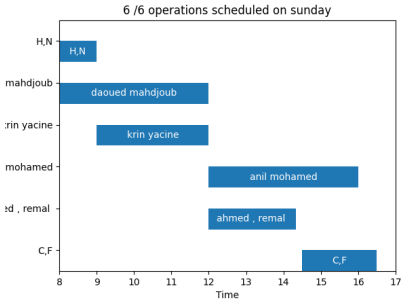


(d) Day four

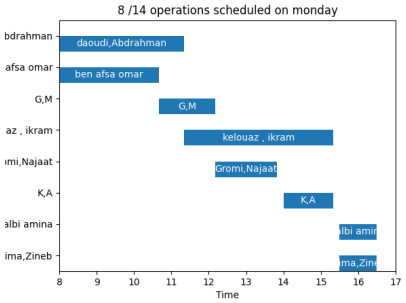
Figure 4.11: Schedule of a week for 20 operations with bi-objectiv on best

4.6.3.2 with 50 operations:

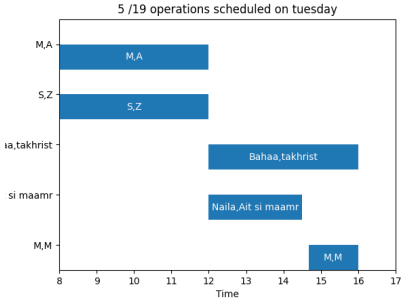
The weekly schedule illustrated in Figure 4.12 showcases the results of instance 3, where the Bi-objective model was employed to schedule 50 operations. Notably, the schedule demonstrates a balanced distribution of operations across all days, with 31 operations scheduled throughout the week. This achievement reflects the effectiveness of the Bi-objective model in achieving a favorable trade-off between the objectives and ensuring an efficient allocation of operations over the course of the week.



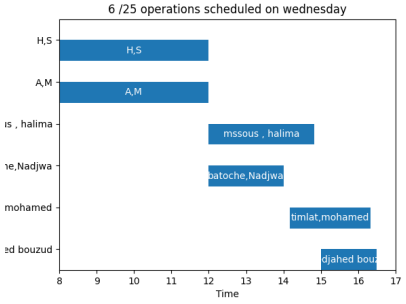
(a) Day one



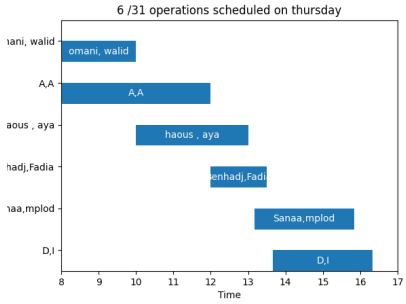
(b) Day two



(c) Day three



(d) Day four



(e) Day five

Figure 4.12: Schedule of a week for 50 with min operations on best