INSTITUTO
UNIVERSITÁRIO
DE LISBOA

Room allocation multicriteria optimization in university timetabling

Bruno Miguel Antunes Teixeira Batista

Master in Computer Engineering

Supervisor:
Vítor Basto Fernandes, PhD, Associate Professor with Aggregation,
Iscte - University Institute of Lisbon

Co-Supervisor:
Professor Iryna Yevseyeva, PhD
School of Computer Science and Informatics, Faculty of Technology, De Montfort University, Leicester LE1 9BH, United Kingdom

July, 2025

[This page is intentionally left blank.]

TECHNOLOGY
AND ARCHITECTURE

Department of Information Science and Technology

Room allocation multicriteria optimization in university timetabling

Bruno Miguel Antunes Teixeira Batista

Master in Computer Engineering

Supervisor:
Vítor Basto Fernandes, PhD, Associate Professor with Aggregation,
Iscte - University Institute of Lisbon

Co-Supervisor:
Professor Iryna Yevseyeva, PhD
School of Computer Science and Informatics, Faculty of Technology, De Montfort University, Leicester LE1 9BH, United Kingdom

July, 2025

[This page is intentionally left blank.]

"Education is the most powerful weapon which you can use to change the world." — Nelson Mandela

I dedicate this work to my family and friends, for their unconditional support throughout this academic journey. To those who always believed in me, even during the most challenging moments, my deepest gratitude.

[This page is intentionally left blank.]

Acknowledgment

I would like to express my sincere gratitude to all the people involved throughout the development of this dissertation. Special thanks to my advisor, Professor Vítor Basto, for his guidance, availability, and support. I also thank the academic coordinators and all those who contributed directly or indirectly to the completion of this work.

[This page is intentionally left blank.]

Resumo

Este trabalho aborda o problema de atribuição de salas na gestão de horários no Iscte, atualmente realizado de forma manual, sendo um processo moroso e complexo. Apesar da existência de software, a sua utilização é limitada devido à dificuldade de parametrização e adaptação reduzida aos requisitos institucionais.

Assim foi realizada uma revisão sistemática da literatura com o objetivo de identificar as abordagens mais utilizadas, analisando aplicações, vantagens, desvantagens e tendências emergentes. As abordagens híbridas, que combinam meta-heurísticas com busca local, têm-se mostrado particularmente eficazes na resolução do University Course Timetabling Problem (UCTP). Algoritmos evolutivos multiobjetivo, como Non-dominated Sorting Genetic Algorithm II (NSGAII), Non-dominated Sorting Genetic Algorithm III (NSGAIII) e Multi-Objective Evolutionary Algorithm Based on Decomposition (MOEA/D), destacam-se em contextos com múltiplos objetivos.

Definiram-se seis critérios, quatro objetivos a minimizar: mudança de sala para turmas consecutivas (RCCC), deslocação de alunos (SD), diferença entre o tipo de sala solicitado e atribuído (MCRA) e capacidade da sala (CRA) e duas restrições rígidas: alocação adequada da sala à aula (ROB) e sobreposição de salas (RO).

Com base nestes critérios, foram desenvolvidas versões personalizadas dos algoritmos NSGAII, NSGAIII e MOEA/D, na framework *JMetal* com integração de busca local.

Os testes, com dados reais do Iscte, envolveram cerca de 10,000 alunos, 26,020 turmas e 126 salas. O *benchmarking* comparou heurísticas construtivas [1], algoritmos Multi-Objective Evolutionary Algorithms (MOEA) e programação inteira linear, modelada em Pyomo [2] e resolvida com Gurobi [3]. A validação foi efetuada com o ficheiro `comp02.ctt` do conjunto Curriculum-Based Course Timetabling da competição ITC2007.

Palavras-chave: *Gestão de Horários Universitários*, *Atribuição de Salas*, *Programação Linear Inteira*, *Metaheurísticas*, *Algoritmos Evolutivos Multiobjetivo*, *Busca Local*.

[This page is intentionally left blank.]

Abstract

This work addresses the problem of room assignment in timetable management at Iscte, currently performed manually, being a time-consuming and complex process. Despite the existence of software, its use is limited due to the difficulty of parameterization and reduced adaptation to institutional requirements.

Thus, a systematic literature review was carried out with the objective of identifying the most used approaches, analyzing applications, advantages, disadvantages, and emerging trends. Hybrid approaches, which combine metaheuristics with local search, have proven particularly effective in solving the UCTP. Multi-objective evolutionary algorithms, such as NSGAII, NSGAIII, and MOEA/D, stand out in contexts with multiple objectives.

Six criteria were defined, four objectives to minimize: room change for consecutive classes (RCCC), student displacement (SD), difference between the requested and assigned classroom type (MCRA), and room capacity (CRA); and two hard constraints: proper room allocation for the class (ROB) and room overlapping (RO).

Based on these criteria, customized versions of the algorithms NSGAII, NSGAIII, and MOEA/D were developed, in the *JMetal* framework with local search integration.

The tests, with real data from Iscte, involved approximately 10,000 students, 26,020 classes, and 126 rooms. The *benchmarking* compared constructive heuristics [1], MOEA algorithms, and integer linear programming, modeled in Pyomo [2] and solved with Gurobi [3]. The validation was performed with the file `comp02.ctt` from the Curriculum-Based Course Timetabling dataset of the ITC2007 competition.

Keywords: *University Timetabling*, *Room Assignment*, *Integer Linear Programming*, *Metaheuristics*, *Multi-Objective Evolutionary Algorithms*, *Local Search*.

[This page is intentionally left blank.]

Contents

Acknowledgment iii

Resumo v

Abstract vii

List of Figures xiii

List of Tables xv

List of Acronyms xvii

Chapter 1. Introduction 1
1.1. Research Questions 3
1.2. Objectives 3
1.3. Research Method 3

Chapter 2. Review of the Literature 7
2.1. University Timetable Planning Problem 7
2.1.1. Types of timetable planning 7
2.1.2. Constraints 8
2.2. Approaches to University Course Timetabling Problem 8
2.2.1. Methods Based on Operational Research (OR) 8
2.2.2. Single-Solution Based Metaheuristics (SSBM) 9
2.2.3. Population-Based Metaheuristics (PBM) 10
2.2.4. Evolutionary Algorithms (EA) 10
2.2.5. Other Population-Based Metaheuristics 14
2.2.6. Hyper-heuristics 17
2.3. University Course Timetabling Problem Formulations 18
2.3.1. Single-Objective Optimization (SOO) 18
2.3.2. Multi-Objective Optimization (MOO) 18
2.3.3. Many-Objective Optimization (MaOO) 21
2.3.4. Quality Indicators in Multi-Objective Optimization and Many-Objective Optimization 22
2.3.5. Application of SOO, MOO, and MaOO to the UCTP 24
2.3.6. Algorithms Used to Solve the Room Allocation Multicriteria Optimization Problem 24
2.3.7. Software and Frameworks 31

2.4. State of the Art 32
2.4.1. Systematic Review of Methodologies to Solve the UCTP 32
2.4.2. Review of the Identified Works 34
2.4.3. Conclusions and Future Pathways for Solving the University Course Timetabling Problem 38

Chapter 3. Problem and Proposed Solution 41
3.1. Problem Description 41
3.2. Problem Constraints 42
3.2.1. Soft Constraints 42
3.2.2. Hard Constraints 43
3.3. Proposed Solution 44

Chapter 4. Implementation 47
4.1. Tools Used 47
4.2. Solution Implementation in Java 48
4.2.1. Data Integration 48
4.2.2. Problem Structure and Modeling 49
4.2.3. Algorithms and Hybrid Strategies 49
4.2.4. Integer Linear Programming Model 51
4.2.5. Execution 51

Chapter 5. Experiments and Parameters 53
5.1. Data Used 53
5.2. Experimental Settings 54
5.3. Parameters of the Evolutionary Algorithms 54
5.4. Execution Environment Setup 55

Chapter 6. Results and Discussion 57
6.1. Tests of the Evolutionary Algorithms with the Iscte – Instituto Universitário de Lisboa Dataset 57
6.1.1. Experiment 1 – Without Constructive Population and With Local Search 57
6.1.2. Experiment 2 – Without Local Search, with Constructive Population 59
6.1.3. Experiment 3 – With Local Search and Constructive Population 60
6.1.4. Experiment 4 – Without Local Search and Without Constructive Population 62
6.1.5. Conclusions of the Evolutionary Algorithms Tests 64
6.1.6. Results on the `comp02.ctt` Instance from ITC2007 64

Chapter 7. Conclusion 67
7.1. Achievement of Objectives 69
7.2. Future Work 69

Annex A. Planning 71

Annex B. Participation in the AMiM'24 Event 73

References 75

[This page is intentionally left blank.]

List of Figures

Figure 1.1 DSR Methodology Process Model. Source: Brocke, Hevner, and Maedche [9] 4

Figure 2.1 Basic steps of an Evolutionary Algorithm. Source: Martinelli [18]. 11

Figure 2.2 Flow of a Genetic Algorithm 12

Figure 2.3 Operation of the mutation operator 13

Figure 2.4 How the crossover operator works 14

Figure 2.5 ACO flow 15

Figure 2.6 PSO flow 16

Figure 2.7 Illustration of the Pareto Front 19

Figure 2.8 Publication trend of NSGAII per year. Source: Ma et al. [43] 25

Figure 2.9 Basic flow of NSGAII. Source: Ma et al. [43] 26

Figure 2.10 Selection process in NSGAII. Source: Deb et al. [42]. 27

Figure 2.11 Process of associating solutions to reference points. Source: Vesikar et al. [44]. 28

Figure 2.12 Basic flowchart of MOEA/D. Adapted from Zhang and Li [34]. 30

Figure 2.13 Steps of the systematic literature review 33

Figure 2.14 Distribution of the publication rate of metaheuristic algorithms to solve the UCTP, Source: Bashab et al. [8] 38

Figure 4.1 UML diagram representing the problem modeling in Java using the jMetal framework 48

Figure 6.1 HV values 58

Figure 6.2 Generalized Spread Indicator (GSPREAD) values 58

Figure 6.3 Reference front plotting for six objectives room allocation UCTP – Experiment 1 59

Figure 6.4 HV values 59

Figure 6.5 GSPREAD values 60

Figure 6.6 Reference front plotting for six objectives room allocation UCTP. 60

Figure 6.7 HV values 61

Figure 6.8 GSPREAD values 61

Figure 6.9 Reference front plotting for six objectives room allocation UCTP. 62
Figure 6.10 HV values 63
Figure 6.11 GSPREAD values 63
Figure 6.12 Reference front plotting for six objectives room allocation UCTP. 64

List of Tables

Table 2.1 Example of dominance comparison between two solutions in a minimization problem. 26

Table 2.2 Filtering process 33

Table 5.1 Description of the Iscte Instance 53

Table 5.2 Description of the Benchmark Instance (comp02.ctt – ITC2007) 54

Table 5.3 Summary of configurations used in the different experimental setups 54

Table 5.4 Algorithm configurations 55

Table 6.1 Problem 1 – Mean and standard deviation of NHV and GSPREAD for each algorithm 57

Table 6.2 Problem 1 – Mean and standard deviation of NHV and GSPREAD for each algorithm 59

Table 6.3 Problem 3 – Mean and standard deviation of NHV and GSPREAD for each algorithm 61

Table 6.4 Problem 3 – Mean and standard deviation of NHV and GSPREAD for each algorithm 62

Table 6.5 Objective function values of the best solution (ROB and RO = 0) – Experiment 1 65

Table 6.6 Objective function values of the best solution (ROB and RO = 0) – Experiment 2 65

Table 6.7 Objective function values of the best solution (ROB and RO = 0) – Experiment 3 65

Table 6.8 Objective function values of the best solution (ROB and RO = 0) – Experiment 4 66

Table A.1 Tasks Schedule for 2023 and 2024 71

[This page is intentionally left blank.]

List of Acronyms

ABC: Artificial Bee Colony

ACO: Ant Colony Optimization

AI: Artificial intelligence

BIP: Binary Integer Programming

COP: Combinatorial Optimization Problem

CP: Constraint Programming

CSP: Constraint Satisfaction Problem

CTTP: Curriculum-based Timetabling Problem

DK: Design Knowledge

DSR: Design Science Research

EA: Evolutionary Algorithms

EMO: Evolutionary Multi-Objective Optimization

EP: Evolutionary Programming

ES: Evolution Strategies

ETTP: Examination Timetabling Problem

FF: Fitness Function

GA: Genetic Algorithm

GLPK: GNU Linear Programming Kit

Gurobi: Gurobi Optimizer

GP: Genetic Programming

GSPREAD: Generalized Spread Indicator

GCH: Graph Coloring Heuristics

HGA: Hybrid Genetic Algorithm

HC: Hill Climbing

HS: Harmony Search

HypE: Hypervolume Estimation Algorithm

HV: HyperVolume

ILP: Integer Linear Programming

Iscte: Iscte – Instituto Universitário de Lisboa

ITC: International Timetabling Competition

LP: Linear Programming

LS: Local Search

LTTP: Lecture Timetabling Problem

MA: Memetic Algorithms

MaOO: Many-Objective Optimization

PBM: Population-Based Metaheuristics

MH: metaheuristics

MILP: Mixed Integer Linear Programming

MIP: Mixed-Integer Programming

MOEA: Multi-Objective Evolutionary Algorithms

MOEA/D: Multi-Objective Evolutionary Algorithm Based on Decomposition

MOO: Multi-Objective Optimization

NHV: Normalized HyperVolume

NLP: Nonlinear Programming

NSGA: Non-dominated Sorting Genetic Algorithm

NSGAII: Non-dominated Sorting Genetic Algorithm II

NSGAIII: Non-dominated Sorting Genetic Algorithm III

OR: Operational Research

PATAT: Practice and Theory of Automated Timetabling

PD: Pareto Dominance

PF: Pareto Front

PO: Pareto Optimal

PSO: Particle Swarm Optimization

Pyomo: Python Optimization Modeling Objects

SA: Simulated Annealing

SOO: Single-Objective Optimization

SPREAD: Spread Indicator

SSBM: Single-Solution Based Metaheuristics

TS: Tabu Search

TTTP: Teacher Timetabling Problem

UCTP: University Course Timetabling Problem

UCTTP: University Course and Teacher Timetabling Problem

UTTP: University Timetabling Problem

VNS: Variable Neighborhood Search

CHAPTER 1

Introduction

This work was developed as part of the Master's thesis in Computer Engineering, with the theme "Room allocation multicriteria optimization in university timetabling". The motivation came from the Computational Intelligence and Optimization curricular unit, where the topic of University Course Timetabling Problem (UCTP) was addressed in the context of Iscte – Instituto Universitário de Lisboa, recognized as a Combinatorial Optimization Problem (COP) of high complexity and NP-hard nature [4]–[6].

The efficient management of academic timetables is essential for the smooth running of educational institutions. At Iscte, the scale of the problem and the limitations of the current software in terms of scalability, customization, and performance justify the need for a more robust solution adapted to the institutional reality.

A literature review of the state of the art was carried out presented in Chapter 2. Methods based on Integer Linear Programming (ILP) and Mixed Integer Linear Programming (MILP), although rigorous, face scalability difficulties. On the other hand, population-based metaheuristics such as Genetic Algorithm (GA) and Ant Colony Optimization (ACO) stand out for their effectiveness on large instances [5], [7].

Recent studies, such as those by Chen et al. (2021) and Bashab et al. (2023) [5], [8], point to hybrid algorithms, which combine metaheuristics with local intensification, as the most effective, accounting for 35% of the successful works identified. The Evolutionary Algorithms (EA), such as Non-dominated Sorting Genetic Algorithm II (NSGAII), Non-dominated Sorting Genetic Algorithm III (NSGAIII), and Multi-Objective Evolutionary Algorithm Based on Decomposition (MOEA/D), are considered promising due to their ability to deal with multiple criteria simultaneously.

This thesis therefore proposes the application and comparison of two approaches to solving the UCTP at Iscte: one based on ILP, for its ability to generate optimal solutions, and another based on the EA identified in the literature review, implemented with the JMetal framework and using local search techniques and constructive heuristics. The aim is to evaluate the performance of both, taking into account the realistic dimension of the problem and institutional requirements.

The remaining content of this work is organized as follows: Chapter 2 presents the literature review, where the fundamental concepts associated with the room assignment problem in the context of the UCTP are discussed. This Chapter also includes a systematic review aimed at identifying and analyzing the most commonly used and effective approaches to its resolution. Chapter 3 describes the problem under study and the proposed solution, based on the formulation of specific constraints and objectives. Chapter

4 details the implementation of the evolutionary algorithms and the ILP approach, as well as the tools used. Chapter 6 presents the analysis of the results obtained from the experiments carried out with real data and a benchmarking instance. Finally, Chapter 7 summarizes the main conclusions reached, also identifying the limitations of the study and proposing directions for future work. Appendix A describes the work plan developed throughout the project, while Appendix B presents the participation in the scientific conference.

1.1. Research Questions

The research questions of this work are as follows:

- What are the most widely used approaches to solving the UCTP?
- What are the main advantages, disadvantages, and limitations associated with these approaches?
- What innovations and opportunities for improvement have been identified in the literature?
- What recent trends have emerged in UCTP research?
- How does the performance of metaheuristic algorithms compare to ILP techniques in solving the UCTP?
- How do the algorithms perform when applied to the ITC2007 benchmarks?

1.2. Objectives

This work has the following objectives:

- To conduct a literature review on the subject in order to provide a solid scientific basis to support the developed study.
- To conduct a systematic review to identify and analyze the most widely used approaches to solving the UCTP.
- To benchmark the different algorithms identified in the systematic review using real data from Iscte, in order to solve the room allocation problem.
- To evaluate the performance of evolutionary algorithms in solving the UCTP, compared to approaches based on ILP.
- To test the algorithms with the ITC2007 dataset in order to validate their effectiveness.
- To identify relevant innovations and potential areas for improvement in the analyzed approaches.

1.3. Research Method

This work uses the Design Science Research (DSR) method, as described by Brocke, Hevner, Maedche, et al. [9]. This method is a research paradigm focused on improving knowledge through the creation of artifacts, with the aim of solving real-world problems and contributing to the advancement of technological and scientific knowledge. The results of DSR include both designed artifacts and Design Knowledge (DK), providing a deeper understanding of application contexts.

The DSR process follows six interconnected stages, with feedback loops that allow for adjustments and iterations throughout the project. The ultimate goal is to generate DK and contribute to the advancement of different domains, such as information systems, engineering, architecture, business, economics and others [10].

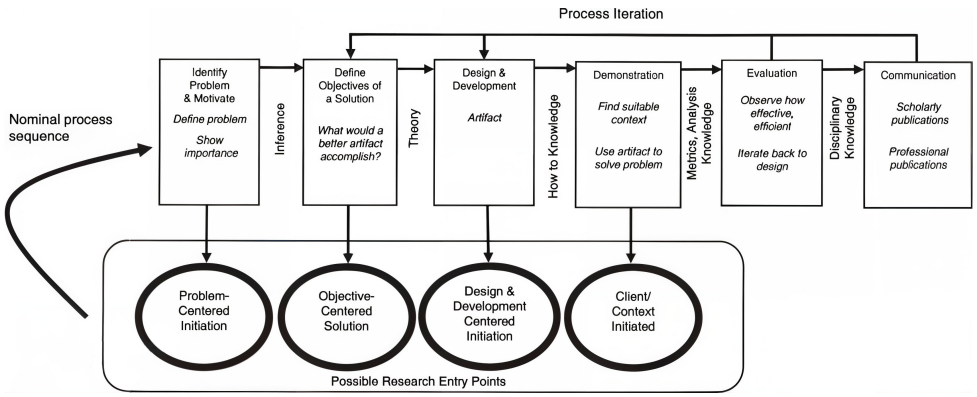

FIGURE 1.1. DSR Methodology Process Model. Source: Brocke, Hevner, and Maedche [9]

Stages of the DSR Process:

1. **Problem Identification and Motivation:** In this step, the specific research problem is defined, justifying the importance of a solution and motivating researchers and stakeholders.
2. **Definition of Solution Objectives:** In this phase, researchers infer goals and objectives from the specific problem definition. These objectives can be quantitative, involving measurable measurements, or qualitative, related to more subjective aspects, establishing clear and achievable goals to guide the development of the solution effectively.
3. **Design and Development:** This stage focuses on creating the proposed solution. It includes drafting the artifact, which can take the form of any designed object incorporating research contributions. Determining the desired functionalities and defining the architecture of the artifact are critical aspects of this phase. The search for innovation, efficiency and effectiveness in implementation is fundamental to the success of this stage.
4. **Demonstration:** Demonstration is the phase in which the artifact is used to solve real instances of the problem. This can include practical experimentation, simulations or case studies. Validating the solution in practical contexts provides valuable information about its performance and effectiveness. The nature of the Demonstration can vary, but its central purpose is to verify the real-world applicability of the proposed solution.
5. **Evaluation:** In this stage, comparisons are made between the objectives defined for the solution and the actual results achieved. Evaluation can involve various forms of analysis, such as quantitative measurements, statistical analysis and qualitative assessments. This critical step provides information on whether the proposed solution has been successful or requires adjustments.
6. **Communication:** In the Communication phase, all aspects of the problem and the artifact are communicated clearly and comprehensively to the stakeholders.

The nature of the communication can vary, from technical reports to more accessible presentations, depending on the objectives and target audience. Ensuring effective communication is essential for disseminating research results, allowing the proposed solution to have a significant impact [9].

[This page is intentionally left blank.]

CHAPTER 2

Review of the Literature

This chapter presents a review of the literature with the aim of framing the work carried out in this dissertation. Initially, the fundamental concepts and main characteristics associated with the UCTP are discussed. This is followed by a systematic review that analyzes the different approaches proposed in the literature for solving this problem.

2.1. University Timetable Planning Problem

The UCTP is an NP-hard optimization problem, which means that it cannot be solved in polynomial time by a deterministic algorithm. It is also classified as a COP, where the objective is to allocate finite resources (classrooms, teachers, and timetables) to events (classes or exams), while respecting a set of constraints. In some cases, the problem consists of finding a timetable that satisfies all the constraints (a constraint satisfaction problem), while in others, the goal is to optimize a function that takes into account both hard and soft constraints. The complexity of the problem arises from the need to fulfill the constraints and the nature of the variables, which can be discrete or continuous [5] [11].

2.1.1. Types of timetable planning

University timetable planning covers several subdomains, depending on the need, however the most common are exams and courses, which are defined below:

- **Course planning:** this is characterized as a multidimensional `assignment problem`, where students and teachers are allocated to courses, sections or classes, and events (lessons) are distributed to specific rooms and times. This is an essential component in educational timetabling which involves, in a first stage, assigning students to courses and, in a second stage, allocating these courses to appropriate rooms and time periods.

  This problem can be broken down into several components, such as student scheduling, course allocation, teacher distribution and room allocation.

  There are also significant differences between subject timetables and exam timetables, while subjects require a fixed allocation to a single room, exams can be spread over several rooms [5].
- **Exam planning:** consists of assigning exams to limited time periods, with the aim of avoiding conflicts and overlaps, ensuring that a student does not have two exams at the same time and that only one takes place per room in each interval. Also taken into account is the capacity of the rooms, which must accommodate all students [5].

2.1.2. Constraints

Constraints within the scope of University Course and Teacher Timetabling Problem (UCTTP) planning can be either soft or hard, depending on the requirements of the institution, and are defined as follows [5] [11] [12]:

- **Soft constraints:** These are constraints that should preferably be satisfied; however, their violation does not invalidate the solution. Failing to meet these constraints decreases the quality of the solution and adds a penalty to the objective function. The aim is to minimize this penalty.

  Examples of soft constraints:

  1. A student should not have only one class in a day.
  2. The characteristics of the room should match the requirements of the course.
  3. A student should not have more than two consecutive lessons.
  4. A student should not have a lesson scheduled in the last time slot of the day.
- **Hard constraints:** These are constraints that must be satisfied for a solution to be considered valid in the UCTTP or any other optimization problem. Examples include:
  1. A teacher cannot teach two classes at the same time.
  2. Only one course can be allocated to each room per time slot.
  3. A course can only be assigned to predefined time periods.
  4. Students must not have overlapping classes.

2.2. Approaches to University Course Timetabling Problem

Currently, there are several approaches to solving the UCTP. According to Chen et al. [5], these can be divided into six main categories — from traditional approaches based on Operational Research (OR) to modern hybrid methods that combine different techniques to improve efficiency in solving the problem.

2.2.1. Methods Based on Operational Research (OR)

OR methods are techniques that use mathematical formulations to find optimal or near-optimal solutions. These methods can include [5], [6]:

- **Integer Linear Programming (ILP):** ILP is a mathematical optimization method in which all or some variables are restricted to non-negative integer values. It is widely used in the context of the UCTTP to allocate courses to teachers, students, and timetables, ensuring that constraints such as room availability or timetable conflicts are satisfied. This method is considered mathematically complete but presents scalability challenges, making it computationally expensive for large institutions.

  To model the problem, ILP organizes courses into subject groups and time intervals, enabling the optimization of class distribution. Some approaches, such as Binary Integer Programming (BIP), can improve allocation efficiency by reducing student and teacher dissatisfaction. Although often used in isolation, it

can also be combined with heuristics to facilitate constraint handling and improve the efficiency of finding viable solutions [5], [6].

- **Mixed Integer Linear Programming (MILP):** MILP is a mathematical optimization method in which the objective function and constraints are linear, and some or all of the decision variables are integers.

  This method is widely used in various fields due to its ability to model complex problems involving discrete decisions. However, MILP can only handle linear functions, which often requires the linearization of non-linear equations, increasing the number of variables and making the problem computationally intensive [13], [14].
- **Graph Coloring Heuristics (GCH):** GCH aims to assign the minimum number of colors to the vertices of a graph, ensuring that adjacent vertices have different colors. In the context of the UCTP, events correspond to vertices, conflicts between events represent edges, and time intervals represent colors. GCH heuristics are used to assign classes to time slots. Examples include the highest degree (which prioritizes events with more conflicts), the highest weighted degree (which considers the number of students), the saturation degree (which selects events with fewer available periods), and the color degree (which prioritizes events with more conflicts with already allocated events) [5].
- **Constraint Programming (CP):** CP is an optimization method for solving combinatorial problems, using techniques from Artificial intelligence (AI), computer science, and operations research to find solutions that satisfy a set of user-defined constraints. Problems modeled in CP are often referred to as Constraint Satisfaction Problem (CSP) and are widely applied in domains such as planning, routing, configuration, and bioinformatics [15].

2.2.2. Single-Solution Based Metaheuristics (SSBM)

The Single-Solution Based Metaheuristics (SSBM) are local search algorithms that explore the neighborhood of a single solution to find improvements. These can be classified into three categories: single-stage, where hard and soft constraints are optimized simultaneously; two-stage, where a feasible solution is first found considering only hard constraints and then improved by optimizing soft constraints; and relaxation, which allows adjustments such as ignoring infeasible events or creating fictitious time slots to accommodate activities. Below, we present algorithms from SSBM [5], [6]:

- **Tabu Search (TS):** TS is a metaheuristics (MH) Algorithm that starts the search from an initial solution and selects the best available neighbor. If the new solution is not in the tabu list, the algorithm moves to that solution; otherwise, it checks the aspiration criterion, which allows tabu solutions to be accepted if they are better than the best solution found so far. The tabu list is updated at each iteration to prevent revisiting previous solutions. The duration of a solution's stay

on the tabu list is determined by the Tabu Tenure, and the algorithm continues exploring the solution space until a predefined stopping condition is met [6], [16].

- **Local Search (LS):** LS is an optimization method used to find a solution that maximizes a criterion within a defined search space. The algorithm starts from an initial solution and iteratively moves to neighboring solutions, provided the neighborhood structure is well defined. Hill Climbing (HC) is an example of LS, where each move is selected solely based on neighboring information to achieve local improvement. The algorithm can also stop after a time limit or when no significant improvement is observed over a predefined number of iterations [6], [16], [17].
- **Simulated Annealing (SA):** SA is an LS method inspired by the physical process of heating and cooling solids. This algorithm helps avoid becoming trapped in local optima and enhances the exploration of the solution space. It starts with a random solution, and at each iteration, the current solution may be replaced by another randomly selected one, based on a controlled probability that allows escaping local optima. This process continues until a termination criterion is reached.

2.2.3. Population-Based Metaheuristics (PBM)

Population-Based Metaheuristics (PBM) operate on a set of solutions, applying different operators and rules to generate new ones in their vicinity. These approaches are widely used to solve complex optimization problems, whether Single-Objective Optimization (SOO), Multi-Objective Optimization (MOO), or Many-Objective Optimization (MaOO), where the process begins with a population of solutions and, at each iteration, the best ones are chosen through a selection mechanism. Modifications are then applied according to the meta-heuristic used, replacing the least fit ones. This process is repeated until a stopping criterion is reached. In this type of approach, EA, ACO, Particle Swarm Optimization (PSO), Memetic Algorithms (MA), Harmony Search (HS), and Artificial Bee Colony (ABC) are widely used, which are defined below [5], [6], [16].

2.2.4. Evolutionary Algorithms (EA)

EA are inspired by the process of natural evolution, as proposed by Charles Darwin, operating on a population of solutions, applying operators such as selection, crossover, mutation, and replacement to find optimal or approximate solutions to complex problems. EA are population-based MH because they operate with multiple solutions simultaneously and use the selection of the fittest individuals to create new and increasingly better generations [4], [6], [18].

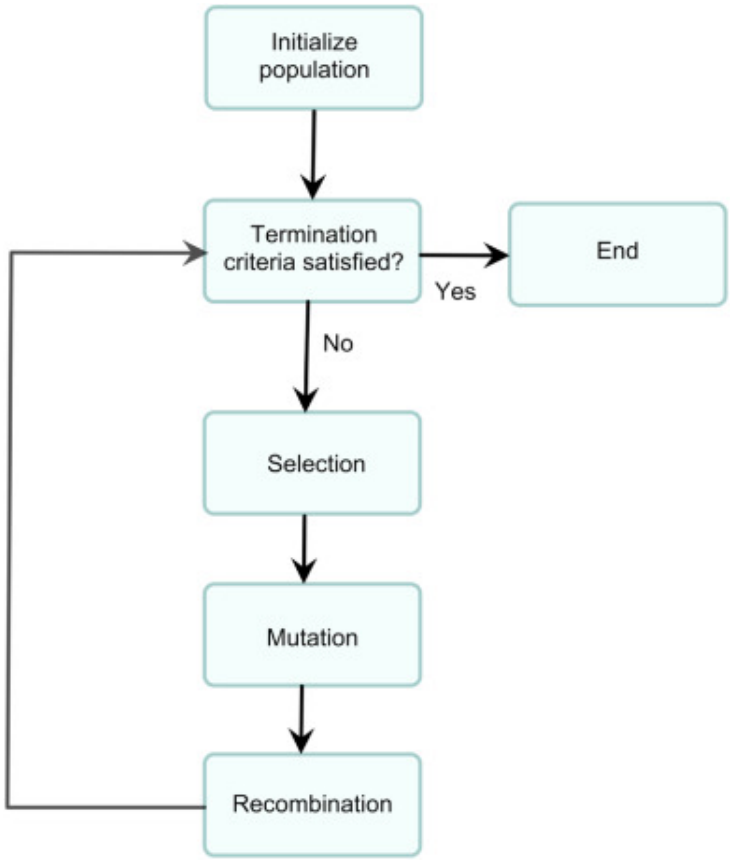

FIGURE 2.1. Basic steps of an Evolutionary Algorithm. Source: Martinelli [18].

As shown in Figure 2.1, an Evolutionary Algorithm (EA) begins with the generation of an initial population of solutions, usually at random.

Next, it checks whether a stopping criterion has been met; if not, the algorithm proceeds with the selection of the best solutions based on a fitness function (Fitness Function (FF)), ensuring that the fittest individuals are more likely to be chosen.

After selection, mutation is applied, introducing small random changes to increase diversity and prevent premature convergence. Recombination then occurs, combining characteristics from different solutions to generate new individuals and improve the exploration of the search space.

This process is repeated until the stopping criterion is satisfied, allowing optimal or near-optimal solutions to be found.

Among the most common EA are Genetic Algorithms (GA), Evolution Strategies (Evolution Strategies (ES)), Genetic Programming (Genetic Programming (GP)), and Evolutionary Programming (Evolutionary Programming (EP)), as well as some learning-based classification systems. In the context of the UCTP, GA stand out due to their widespread application in solving this type of problem [6], [18], [19].

- **Genetic Algorithm:** GA are algorithms inspired by Charles Darwin's theory of natural selection, proposed by John H. Holland in the 1960s [20]. They follow the structure of EA, using an evolutionary process based on genetic operators.

Crossover combines characteristics of the parents to generate offspring, while mutation introduces small random variations, promoting genetic diversity and avoiding premature convergence.

The definition of parameters—such as the number of generations, population size, and crossover and mutation probabilities—directly influences diversity, solution quality, and execution time. Additionally, the tournament size regulates selective pressure, helping to prevent premature convergence [20]–[22].

Figure 2.2 illustrates the functioning of a GA, where a population of solutions is evaluated and iteratively modified by genetic operators until a stopping condition is reached [23].

The main steps are described below:

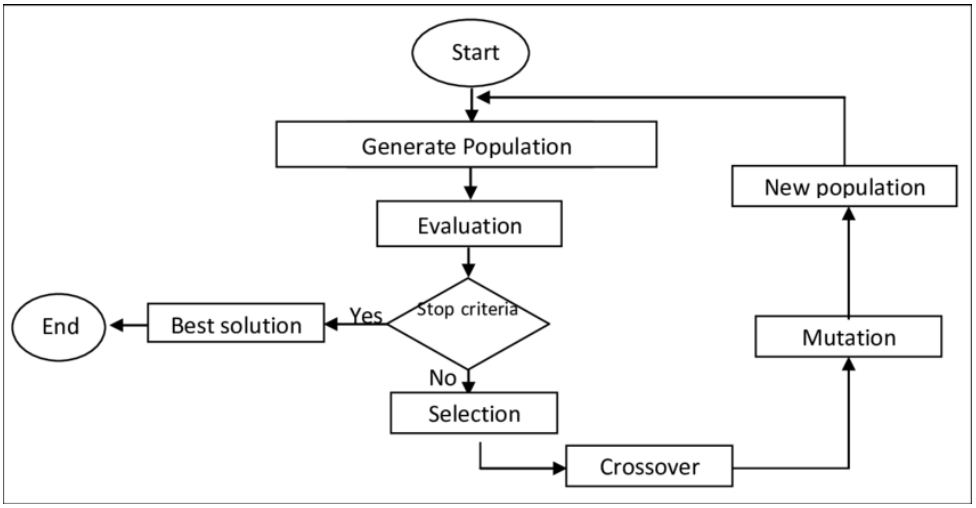

FIGURE 2.2. Basic flow of a GA. Source: Faitouri e Khalafullah [24].

1. The first stage of modeling corresponds to the representation of the individual, in which each one represents a possible solution to the problem. This representation can be binary, integer, real, or permutation-based. In the binary approach, a sequence of bits is used to encode the solutions; the integer representation associates specific elements with each gene; the real representation uses continuous numbers and is suitable for problems with physical quantities. Permutation representation is the most appropriate for sorting problems, such as the Traveling Salesman Problem, as it preserves the relative sequence of the elements.
2. The population corresponds to the set of candidate solutions that evolve throughout the optimization process and can be generated randomly or based on previously evaluated solutions (seeds). According to Diaz-Gomez and Hougen [25], the definition of the initial population should consider factors such as the search space, the evaluation function, the diversity of solutions, the complexity of the problem, the selection method, and the number of individuals. When seeds are used, they are chosen from previously

analyzed random solutions, selecting those with the best performance [21], [25].

3. The fitness function measures the quality of each individual, indicating how well it adapts to the problem. A well-defined function allows good solutions to be effectively distinguished from less suitable ones. Otherwise, promising solutions may be discarded or less useful solutions may be retained. In multi-objective contexts, different weights can be assigned to different evaluation criteria. The fitness value of an individual i is defined as $\frac{f_i}{f_A}$, where f_i represents the evaluation of individual i, and f_A is the average of the evaluations of the population. In other words, fitness is calculated in relation to the other members of the population, ensuring a comparative evaluation [21].
4. Selection methods determine the individuals that will contribute to the next generation, being influenced by the so-called selective pressure. Among the most common are: roulette, which assigns each individual a probability proportional to its fitness value; selection by absolute fitness, which favors the fittest and may lead to premature convergence; tournament, which chooses the best from random subsets, balancing diversity and elitism; and ranking, which orders individuals according to their performance and distributes selection probabilities based on this ordering, promoting greater diversity in the population [25].
5. In this stage, genetic operators are applied, namely crossover and mutation, with the aim of refining and diversifying the search, introducing genetic variability into the population and allowing the generation of new solutions.

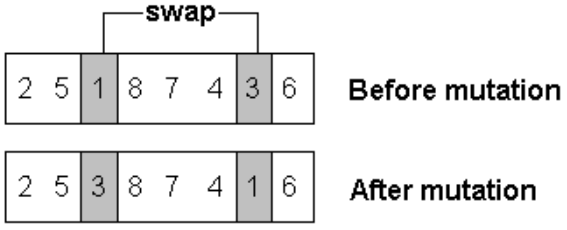

FIGURE 2.3. Operation of the mutation operator. Source: El Majdoubi et al. [26].

The mutation operator, as shown in Figure 2.3, randomly alters the genetic characteristics of an individual based on a probabilistic criterion, using only the parent to generate the descendant. Among the main types are: insertion mutation, which moves a gene after another selected gene, partially maintaining the order; inversion mutation, which reverses the sequence of genes between two random points; and uniform mutation, which replaces a gene with a random value within its possible domain.

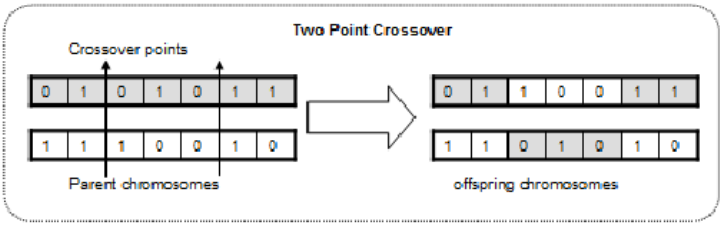

FIGURE 2.4. How the crossover operator works. Source: El Majdoubi et al. [26].

The crossover operator, as shown in Figure 2.4, generates new individuals from the combination of two parents. This can occur in a simple way with a single cut-off point, at multiple points, or uniformly, where each gene of the offspring is chosen randomly from one of the parents, promoting genetic diversity and efficient exploration of the solution space.

The most common crossover methods are: single-point, where a random point is chosen in the parents and the parts before and after that point are combined to form the offspring; multi-point, which is similar to the previous one but uses multiple cut-off points, increasing the variability in recombination; and uniform, where a binary mask is used to define, gene by gene, which parent contributes each piece of information, promoting a more balanced mix of genetic material.

6. Finally, stopping criteria are applied, which determine when the execution of a GA should be terminated. The first criterion occurs when individuals reach a pattern considered optimal. The second is based on previously defined constraints, such as the maximum number of generations, execution time, total number of evaluations, or the absence of significant improvements throughout the iterations. The algorithm ends when a satisfactory solution is found or when one of these conditions is met [21].

   Next, we present EA developed for optimization problems with multiple objectives (MOO) and with many objectives (MaOO), highlighting the algorithms NSGAII, NSGAIII, and MOEA/D.

2.2.5. Other Population-Based Metaheuristics

In addition to EA, other approaches such as ACO, PSO, MA, HS, and ABC are noteworthy and will be described below due to their relevance and applicability to complex optimization problems.

2.2.5.1. *Ant Colony Optimization (ACO)*

ACO is inspired by the collective behavior of real ant colonies, particularly their ability to find efficient paths between the colony and food sources. Proposed by Marco Dorigo in the early 1990s, this approach uses artificial agents (ants) that build solutions based on local information and indirect signals simulated through the deposition of pheromones.

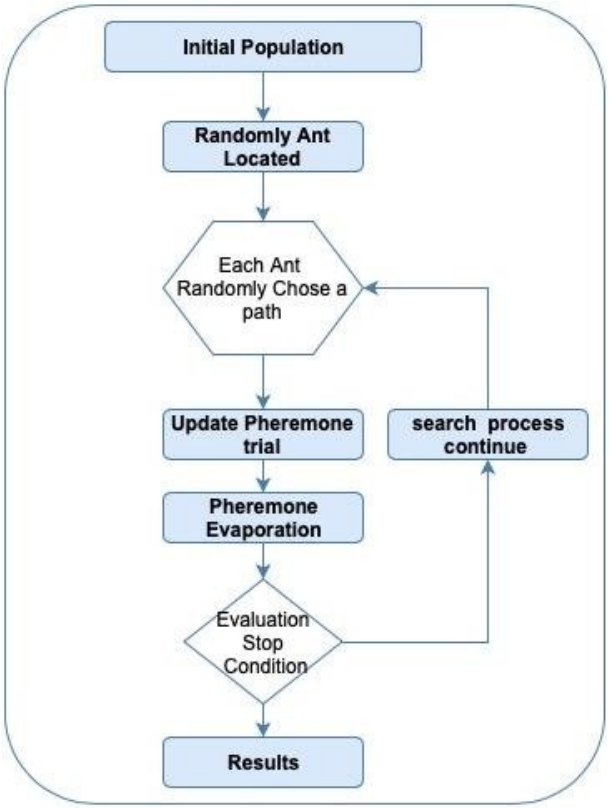

FIGURE 2.5. Illustrative flow of the functioning of ACO. Source: Almufti [27].

The general operation of the algorithm is illustrated in Figure 2.5, which represents the complete cycle of construction, evaluation, and updating of solutions.

Initially, the ants are randomly distributed and begin to build solutions by traversing the graph, guided by two factors: the amount of pheromone present on each path (τ_{ij}) and the heuristic visibility (η_{ij}), usually defined as the inverse of the distance between nodes.

During each iteration, the ants build complete solutions based on a probabilistic choice of the next node to visit. The decision of an ant k to move from a node i to a node j is modeled by the following formula:

$$p_{ij}^{k} = \frac{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{l \in N_k(i)} [\tau_{il}(t)]^{\alpha} \cdot [\eta_{il}]^{\beta}}$$

Where:

- $\tau_{ij}(t)$ is the amount of pheromone present on path (i, j) at time t;
- η_{ij} is the heuristic visibility (for example, $\eta_{ij} = \frac{1}{d_{ij}}$);
- α and β are parameters that control the relative influence of the pheromone trail and the heuristic;
- $N_k(i)$ represents the set of nodes not yet visited by ant k.

After each iteration, the amount of pheromone is updated to reflect the quality of the solutions found, applying an evaporation rule:

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \Delta\tau_{ij}(t)$$

where $\rho \in [0,1]$ represents the evaporation rate, and $\Delta\tau_{ij}(t)$ is the amount of pheromone deposited, proportional to the quality of the solution found.

This mechanism allows the artificial colony to efficiently explore the search space, promoting convergence to high-quality solutions in problems such as the Traveling Salesman Problem, task scheduling, and route optimization [28].

2.2.5.2. *Particle Swarm Optimization (PSO)*

PSO is a stochastic optimization MH inspired by the collective behavior of animals, such as flocks of birds or schools of fish. Proposed by Kennedy and Eberhart in 1995, the algorithm simulates a population of particles (potential solutions) that move through the search space, influenced by their own experience and that of the group.

Figure 2.6 illustrates how PSO works. Initially, the particles are randomly distributed, each with a position $\vec{x}_i$ and a velocity $\vec{v}_i$. Each particle stores its best-known position ($\vec{p}_i$) and is aware of the best global position found by the population ($\vec{g}$).

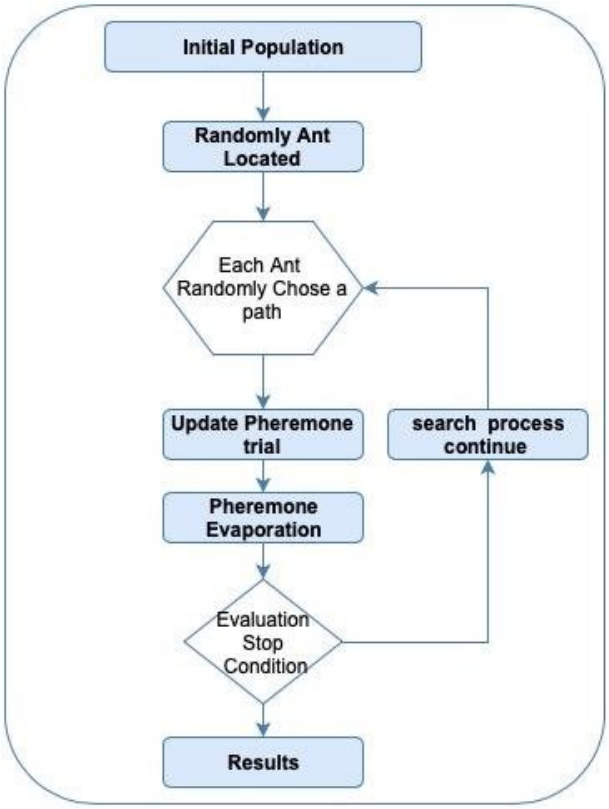

FIGURE 2.6. Illustrative flow of the functioning of PSO. Source: Almufti [27].

The velocity and position of the particles are updated using the following equations:

$$\vec{v}_i(t+1) = \omega \cdot \vec{v}_i(t) + c_1 \cdot r_1 \cdot (\vec{p}_i - \vec{x}_i(t)) + c_2 \cdot r_2 \cdot (\vec{g} - \vec{x}_i(t))$$

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t+1)$$

where:

- ω is the inertia weight;
- c_1 and c_2 are acceleration coefficients (cognitive and social);
- r_1, $r_2 \sim U(0,1)$ are random numbers;
- $\vec{p}_i$ is the best position of particle i;
- $\vec{g}$ is the best global position of the population.

Each particle is evaluated using the objective function, which allows updating both the individual and the global best positions:

$$\vec{p}_i(t+1) = \begin{cases} \vec{x}_i(t+1), & \text{if } f(\vec{x}_i(t+1)) < f(\vec{p}_i(t)) \\ \vec{p}_i(t), & \text{otherwise} \end{cases}$$

Optionally, a constriction factor χ can be used to ensure stability:

$$\vec{v}_i(t+1) = \chi \cdot [\vec{v}_i(t) + \phi_1 \cdot r_1 \cdot (\vec{p}_i - \vec{x}_i(t)) + \phi_2 \cdot r_2 \cdot (\vec{g} - \vec{x}_i(t))]$$

In this way, PSO is able to efficiently explore and refine the search space, being successfully applied to continuous and discrete problems, function optimization, AI, and timetabling [29].

2.2.6. Hyper-heuristics

Hyper-heuristics are search techniques that select, generate, or sequence heuristics or MH to solve complex optimization problems. They operate at a higher level than traditional MH, aiming to improve generalization and reduce dependence on problem-specific knowledge. The most common types are:

- **Selection Hyper-heuristics:** These select, among various heuristics (or meta-heuristics), the most appropriate one to apply based on past performance. They may include machine learning techniques such as reinforcement learning or Q-learning.
- **Generation Hyper-heuristics:** Instead of selecting, these create new heuristics by combining components of existing ones. For example, generating a new neighborhood function by combining simple operators.
- **Sequencing Hyper-heuristics:** These aim to define the best order or sequence of heuristics to apply in order to achieve better results. For instance, first applying mutation, then crossover, and finally local search.
- **Parallel Hyper-heuristics:** These exploit parallel architectures (such as GPUs or clusters) to execute multiple heuristics simultaneously, with information exchange between them [30].

2.3. University Course Timetabling Problem Formulations

This section addresses the main optimization approaches used in the formulation and resolution of the UCTP, grouped according to the number of objectives considered.

2.3.1. Single-Objective Optimization (SOO)

Single-Objective Optimization deals with problems in which the goal is to minimize or maximize a single objective function, subject to constraints. It is used in situations where the aim is to optimize one criterion, such as cost, time, or efficiency, with the goal of finding a single optimal solution that represents the best possible value.

The general formulation of an optimization problem can be expressed as:

$$g_i(x) \leq 0, \quad i = 1, \ldots, m \text{ and } h_j(x) = 0, \quad j = 1, \ldots, p, \quad x \in \Omega$$

where:

- $x = (x_1, \ldots, x_n)$ is the decision variable vector of dimension n;
- Ω is the search space of feasible solutions;
- $f(x)$ is the scalar objective function to be minimized or maximized;
- $g_i(x)$ are the inequality constraints; and
- $h_j(x)$ are the equality constraints [4], [31].

2.3.2. Multi-Objective Optimization (MOO)

MOO refers to an area of optimization in which two or more objectives are considered simultaneously. This type of problem is common in real-world contexts, where multiple criteria must be optimized together, for example, minimizing cost and maximizing the quality of a product. Unlike single-objective optimization (SOO), where a single optimal solution is sought, MOO problems involve trade-offs between conflicting objectives.

In these cases, the goal is not to find a single ideal solution, but rather a set of equally valid solutions that represent different trade-offs between the objectives. This set is known as the **Pareto Front**, composed of *non-dominated* solutions — that is, solutions for which there is no other that is simultaneously better in all objectives.

2.3.2.1. *Mathematical Formulation*

The mathematical formulation of a MOO problem can be expressed as:

$$f : \mathcal{X} \to \mathbb{R}^m, \quad x \mapsto [f_1(x), f_2(x), \ldots, f_m(x)]^\top$$

where:

- $\mathcal{X}$ is the decision space, which contains all feasible solutions;
- $f(x)$ represents the vector of objective functions to be minimized or maximized;
- $f_i(x)$ is the i-th objective function, with $i = 1, \ldots, m$;
- $\mathbb{R}^m$ is the objective space, where each solution x is evaluated according to the m criteria.

The goal of MOO is to minimize (or maximize) all functions $f_i(x)$ simultaneously. Since there is generally no single global optimal solution, the aim is to find the **Pareto-efficient solution set**, which corresponds to the best possible alternatives given the problem constraints [4], [31].

2.3.2.2. *Pareto Dominance and Front*

The Pareto Front (PF) is a central concept in MOO, where the aim is to simultaneously optimize multiple, often conflicting, objective functions. In this context, the classical concept of optimal solution is replaced by that of Pareto Optimal (PO), which represents the best possible trade-off among the considered objectives. A solution is said to be PO given a vector objective function:

$$f : X \to \mathbb{R}^m, \quad f(x) = [f_1(x), f_2(x), \ldots, f_m(x)],$$

a solution $x^* \in X$ is said to be Pareto optimal if there is no other solution $x \in X$ such that:

- $f_i(x) \leq f_i(x^*)$ for all $i \in \{1, \ldots, m\}$, and
- $f_j(x) < f_j(x^*)$ for at least one index $j \in \{1, \ldots, m\}$.

The set of PO solutions forms the Pareto set, composed of non-dominated alternatives in the decision space. Its image in the objective space is the PF, which graphically represents the best possible trade-offs between conflicting objectives. Each point on the PF expresses this balance, where improving one objective necessarily worsens another.

To ensure the usefulness of the PF in decision-making, it is essential that the solutions are well distributed along the front. This allows the decision-maker to consider a variety of alternatives with different preferences and priorities. A concentrated distribution may neglect important areas of the objective space, limit the available options, and indicate that the algorithm was not able to efficiently explore the entire front [4], [31].

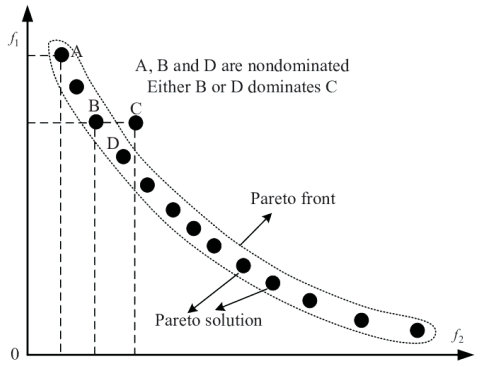

FIGURE 2.7. Illustration of the PF. Source: Cai et al. [32]

In Figure 2.7, the dashed curve represents the PF, composed of non-dominated solutions such as points A, B, and D, in which it is not possible to improve one objective without worsening another. Point C, on the other hand, is dominated by B and D and therefore does not belong to the PF. Identifying the PF is an essential step to support

decision-making, as it provides the decision-maker with a complete view of the available trade-offs. The choice of the most appropriate solution within the front will depend on specific preferences or additional criteria, often not mathematically modeled [32].

In MOO, the selection of solutions directly influences the convergence and diversity of the population over generations. Techniques such as Pareto Dominance (PD) and diversity mechanisms like niching and crowding distance are widely used. The latter, mainly adopted in NSGAII, measures the local density between neighboring solutions in the objective space, favoring those located in less dense regions and promoting a more balanced distribution along the PF [31].

2.3.2.3. *Characterization of MOO Problems*

The complexity of MOO problems depends on several structural characteristics, such as the type of variables involved (continuous, discrete, or mixed), which directly influences the search operators and the ease of exploring the decision space. Other important factors include the geometry of the PF (convex, non-convex, or disconnected), the presence of multiple local optima, and the topology of the decision space (with complex or disconnected feasible regions).

In addition, problems involving uncertainty or dynamic data require robust strategies, and in many cases, it is necessary to incorporate the decision-maker's preferences to guide the search toward the most relevant solutions [31].

2.3.2.4. *Classification of MOO Solution Methods*

The choice of resolution strategy in MOO problems is directly related to how the decision-maker's (DM) preferences are considered in the process. This classification can be divided into four main categories [31]:

- **A priori methods:** Incorporate preferences before optimization, usually through weights or utility functions.
- **A posteriori methods:** Generate the PF without considering preferences, allowing the decision-maker to choose after execution.
- **Interactive methods:** Adjust preferences during the process, with continuous feedback from the decision-maker.
- **No-preference methods:** Seek solutions close to the ideal point without decision-maker intervention.

2.3.2.5. *Computational Challenges*

Solving MOO problems with Multi-Objective Evolutionary Algorithms (MOEA) involves relevant computational challenges, especially in complex contexts or with many objectives [31]. The main challenges are:

- **Computational cost:** Evaluating multiple objectives simultaneously requires more time and computational resources.
- **Scalability:** As the number of objectives increases, PD becomes less effective, making solution selection more difficult.

- **Execution time:** The complexity of MOEA increases with the problem dimension, which may compromise its applicability on a large scale.

2.3.3. Many-Objective Optimization (MaOO)

MaOO is a subfield of MOO that deals with problems involving four or more conflicting objectives to be considered simultaneously. This type of problem is common in complex real-world contexts, such as multidisciplinary engineering, bioinformatics, sustainable product design, and urban systems planning, where multiple criteria must be optimized together.

MaOO is conceptually based on the fundamentals of MOO, but it introduces additional challenges mainly due to the high dimensionality of the objective space. As the number of objectives increases, difficulties arise such as:

- An exponential growth in the number of non-dominated solutions, reducing the effectiveness of PD;
- Difficulties in visualizing the PF and articulating preferences;
- High computational complexity in classical algorithms, such as non-dominated sorting and hypervolume calculations;
- The decision-maker faces cognitive challenges in understanding the trade-offs between solutions;
- Need for new strategies such as decomposition, objective space reduction, or interactive methods [31].

2.3.3.1. *Mathematical Formulation*

The mathematical formulation of a MaOO problem is analogous to that of a MOO problem, but with $m \geq 4$ objectives:

$$f : \mathcal{X} \to \mathbb{R}^m, \quad f(x) = (f_1(x), \ldots, f_m(x)) \text{ with } x \in \mathcal{X}$$

Where:

- $\mathcal{X}$ represents the decision space, i.e., the set of all feasible solutions;
- $\mathbb{R}^m$ is the objective space, in which solutions are evaluated based on m distinct criteria;
- $f(x)$ is the vector-valued objective function, composed of m functions $f_i(x)$, each corresponding to an objective to be minimized or maximized;
- each solution $x \in \mathcal{X}$ is mapped to a vector $f(x) \in \mathbb{R}^m$ that expresses its performance with respect to all objectives;
- the goal is to find a set of non-dominated solutions that constitutes the PF, representing the best possible trade-offs among conflicting objectives [4].

2.3.3.2. *Strategies for MaOO*

To address the specific challenges of MaOO, dedicated algorithms have been developed to overcome the limitations of traditional approaches. The NSGAIII [33] uses reference

points to ensure diversity in high-dimensional spaces, while the MOEA/D [34] adopts a decomposition strategy that breaks the problem into scalar subproblems.

The Hypervolume Estimation Algorithm (HypE) [35] focuses on hypervolume maximization through Monte Carlo sampling, making it suitable for scenarios where exact computation becomes infeasible due to a large number of objectives.

In addition, interactive methods such as the Nautilus method [36] and the Pareto Navigator [37] involve the decision-maker in the process, dynamically adjusting the search direction based on their preferences. These techniques are especially useful when the aim is to integrate human knowledge into the solution selection process.

These strategies represent significant advances, particularly in contexts with many objectives, where conventional techniques lose effectiveness [4].

2.3.3.3. *Future of the MaOO Field*

Initiatives such as the construction of the MyCODA ontology [4] aim to structure knowledge about MaOO. This ontology, presented in Chapter 13 of [38], classifies concepts, methods, metrics, and applications, promoting a common vocabulary and collaboration among researchers.

2.3.4. Quality Indicators in Multi-Objective Optimization and Many-Objective Optimization

Quality indicators play a central role in evaluating the performance of algorithms in MOO and MaOO, especially with regard to convergence to the Pareto front and the diversity of the solutions generated. Among the main indicators, the following stand out:

- **HyperVolume (HV):** measures the volume of the objective space covered by non-dominated solutions, bounded by a reference point, simultaneously reflecting the convergence and diversity of the obtained front. The larger the hypervolume, the better the coverage of the PF [31].
- **Normalized HyperVolume (NHV):** is used to evaluate the quality of approximated fronts. It is based on the HV metric, however the NHV is computed in the normalized objective space, which allows for a fairer comparison between algorithms, especially when objectives have different scales or units. This normalization ensures that each objective contributes equally to the performance evaluation, mitigating the bias caused by differing value ranges. NHV values range from 0 to 1, with values closer to 1 indicating fronts that are closer to and more representative of the ideal Pareto front [39].

  In MOO problems with up to three objectives, HV is widely used due to its computational efficiency. However, as the number of objectives increases, as in MaOO problems, computing the HV becomes more demanding, which can limit its direct use during algorithm execution. Nevertheless, HV remains an effective metric for post-processing evaluations. During execution, it is often

replaced by more computationally efficient indicators, such as IGD+, R2, or the ε-indicator [4].

- **Inverted Generational Distance (IGD) and IGD+:** evaluate the quality of the obtained front based on its proximity to a reference Pareto front. IGD computes the average distance from reference points to the generated solutions, reflecting both convergence and diversity. However, it may favor dominated fronts. IGD+ corrects this limitation by employing a modified metric that respects the direction of Pareto dominance, thus being more suitable in the context of MaOO [4], [31].
- **Spread Indicator (SPREAD) / Spacing:** assesses how evenly solutions are distributed along the front. A lower value of this indicator indicates good coverage and balance among solutions [31]. This indicator is widely used in MOO, but its direct application in MaOO is limited due to the geometric complexity of high-dimensional fronts and the difficulty in defining meaningful spacing metrics. In MaOO, the dispersion of solutions is generally assessed using metrics more suitable for multiple objectives, such as HV, R2, and IGD+, which incorporate this analysis more robustly [4].
- **Generalized Spread Indicator:** The Generalized Spread (GS) is a diversity metric that evaluates the uniformity and extent of non-dominated solutions in problems with more than two objectives. It measures the distance between each point and its nearest neighbors, as well as to the extreme points of the front. The lower the value, the better the distribution. Although useful, it is not Pareto-compliant, which requires careful interpretation [40], [41].
- **Epsilon Indicator:** indicates the smallest value needed to shift (or translate) all solutions of one front so that it dominates another. This enables quantitative comparisons between different solution fronts. It is widely used in MOO and also applicable in MaOO due to its good computational scalability. Although it is only weakly compatible with Pareto dominance, it remains a useful metric for assessing convergence and coverage, especially when used alongside other indicators [4], [31].
- **R2 Indicator:** applied in both MOO and MaOO, uses weighted utility functions to assess solution quality based on the decision maker's preferences, being particularly effective in multi-criteria decision-making contexts. With few objectives (2 or 3), it is simple to configure representative weight vectors and geometrically interpret the influence of each direction, making it a lightweight alternative to hypervolume and compatible with interactive or preference-based algorithms.

  In the MaOO context, R2 evaluates the quality of a set of solutions based on a finite set of scalarization functions, such as the weighted Chebyshev function. It is considered a comprehensive metric, as it incorporates aspects of convergence, dispersion, uniformity, and cardinality. Furthermore, it stands out for

its computational efficiency in problems with four or more objectives, in which hypervolume becomes infeasible to calculate accurately.

However, this indicator is only weakly compatible with Pareto dominance, being sensitive to the quality of the weight vector distribution, the choice of the utopian point, and the geometry of the Pareto front. For example, uniformly distributed weight vectors tend to perform better on linear fronts [4], [31].

These metrics are fundamental not only for guiding the evolution of the population during the execution of the algorithms but also for enabling rigorous comparisons between different optimization approaches.

2.3.5. Application of SOO, MOO, and MaOO to the UCTP

The UCTP can be formulated and solved through different optimization paradigms, depending on the number of objectives considered and the degree of conflict between them.

In the context of SOO, according to Chen et al. [5], ILP and MILP are frequently used to solve small-scale problems due to the NP-hard complexity of the problem. Although these approaches can guarantee the achievement of optimal solutions, their application becomes computationally infeasible in large-scale problems due to the exponential growth of the search space.

On the other hand, in the MOO and MaOO formulations, metaheuristic-based algorithms or hybrid approaches are often used, given the need to balance multiple conflicting objectives and ensure diversity in the generated solutions. Although the number of studies is still limited, the authors highlight these approaches as a promising trend for future research, especially in high-complexity scenarios.

2.3.6. Algorithms Used to Solve the Room Allocation Multicriteria Optimization Problem

This section presents the algorithms used to solve the problem within the scope of this work.

2.3.6.1. *Non-dominated Sorting Genetic Algorithm II*

NSGAII is a multi-objective evolutionary algorithm developed by Deb et al. [42], as a modification of the original Non-dominated Sorting Genetic Algorithm (NSGA). Its objective is to solve problems involving the simultaneous optimization of multiple conflicting criteria, aiming to find an efficient approximation of the PF.

According to Ma et al. [43], and as illustrated in Figure 2.8, the number of publications using NSGAII has increased substantially from 2013 to 2022.

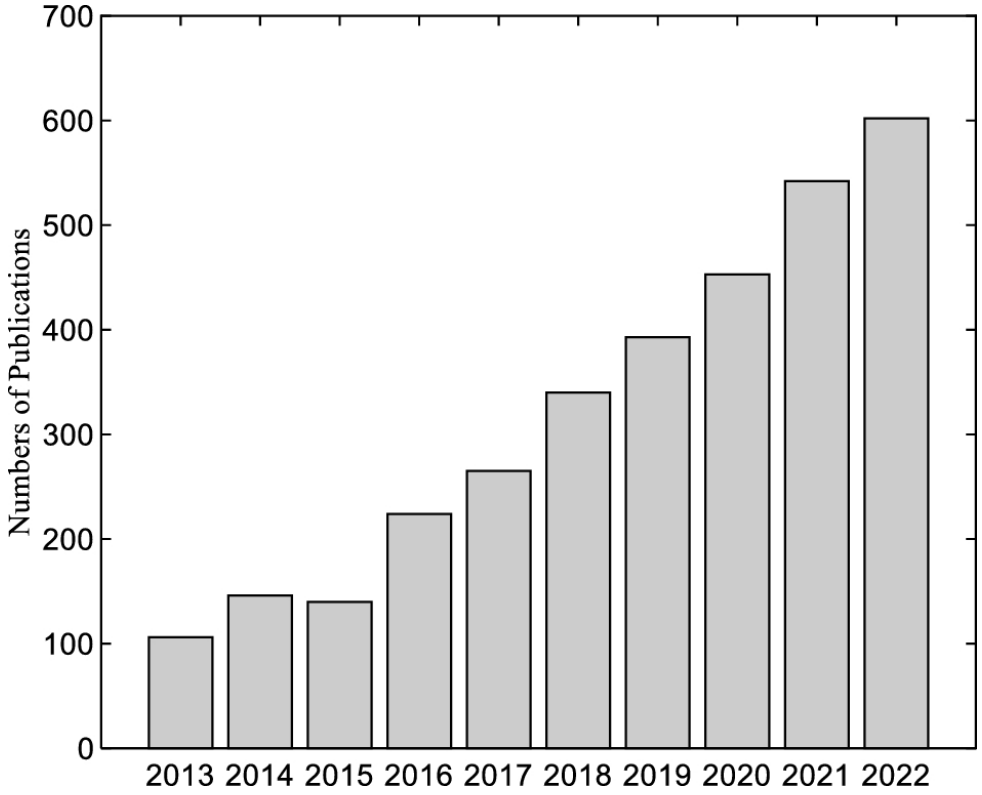

FIGURE 2.8. Publication trend of NSGAII per year. Source: Ma et al. [43]

The algorithm emerged as an evolution of the first generations of dominance-based sorting algorithms, introducing improvements to several limitations identified in earlier approaches, such as high computational complexity, the absence of elitism, and the dependency on external parameters for diversity control.

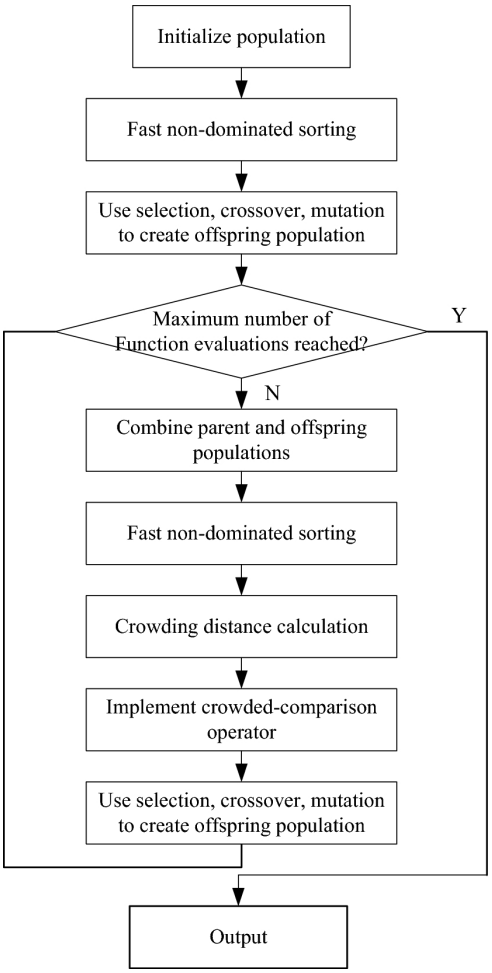

FIGURE 2.9. Basic flow of NSGAII. Source: Ma et al. [43]

Initially, an initial population is generated, followed by the application of the three main improvements introduced by NSGAII, which are defined as follows:

- **Fast non-dominated sorting:** NSGAII introduced a ranking method based on non-dominated fronts with reduced complexity ($O(MN^2)$, where M is the number of objectives and N is the population size). This method allows the population to be organized into dominance layers, favoring solutions that are closer to the optimal Pareto front.

TABLE 2.1. Example of dominance comparison between two solutions in a minimization problem.

Solution	**Objective 1**	**Objective 2**	**Objective 3**
A	30	40	40
B	20	30	35

It can be observed that solution B presents better values in all objectives compared to solution A, which implies that B dominates A.

- **Elitism:** The algorithm preserves the best solutions from each generation by combining the parent and offspring populations before selection. This mechanism ensures that high-quality solutions are not lost during the evolutionary process, increasing the robustness of the search.
- **Diversity preservation:** To maintain a balanced distribution of solutions along the PF, the concept of crowding distance was introduced. This metric evaluates the density of neighboring solutions and favors individuals located in less densely populated regions, promoting diversity without requiring additional parameters such as the sharing parameter.

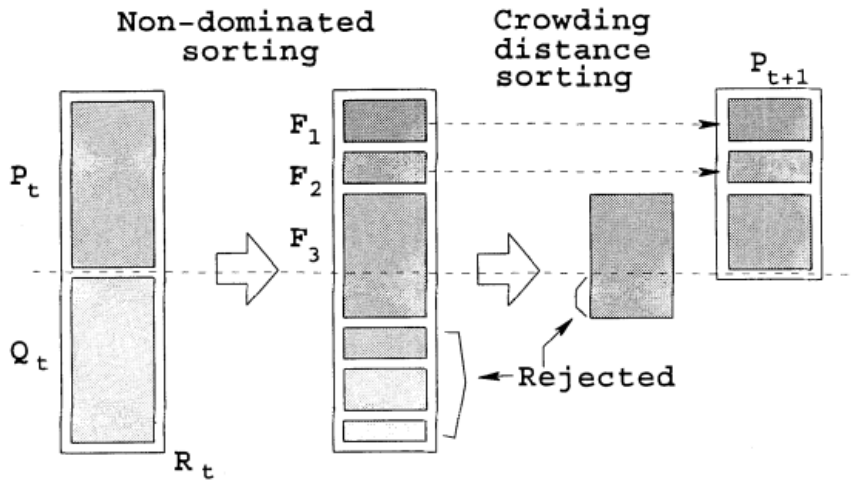

FIGURE 2.10. Selection process in NSGAII. Source: Deb et al. [42].

Figure 2.10 represents the selection process in NSGAII, where the parent and offspring populations are combined and organized into dominance fronts. The fronts are sequentially added to the new population until the desired size is reached. If only part of a front needs to be selected, the crowding distance is applied to prioritize more diverse solutions. The result is the new population P_{t+1} for the next generation.

To generate the offspring population, genetic operators such as selection, crossover, and mutation are applied, as described in Item 5.. After the creation of the new offspring population, the parent and offspring populations are combined to form an expanded set. Then, fast non-dominated sorting is performed, organizing the solutions into non-dominated fronts, followed by the calculation of the crowding distance to maintain diversity.

The solutions are then selected based on their dominance front and crowding distance, forming the new population P_{t+1}. This process of selection, crossover, and mutation is repeated iteratively, generation after generation, until the stopping criterion—defined as the maximum number of objective function evaluations—is reached.

2.3.6.2. *Non-dominated Sorting Genetic Algorithm III*

NSGAIII is a many-objective evolutionary algorithm proposed by Deb et al. [33], aimed at addressing optimization problems involving a large number of simultaneous objectives.

It emerges as an evolution of NSGAII, introducing a selection mechanism based on predefined reference points, rather than relying solely on PD sorting and crowding distance, as in NSGAII. This approach enables a better distribution of solutions along the PF, even in high-dimensional spaces where PD becomes less discriminative.

Maintaining the concept of elitism, NSGAIII combines the parent and offspring populations into an expanded population and sorts the solutions into dominance fronts. However, the final selection of individuals for the next generation is based on a process of association with uniformly distributed reference points, maximizing diversity among solutions.

The algorithm begins with a random population and applies the fast non-dominated sorting process, as described in Subsection 2.3.6.1. Next, solutions are associated with the reference points to guide the selection of the new generation.

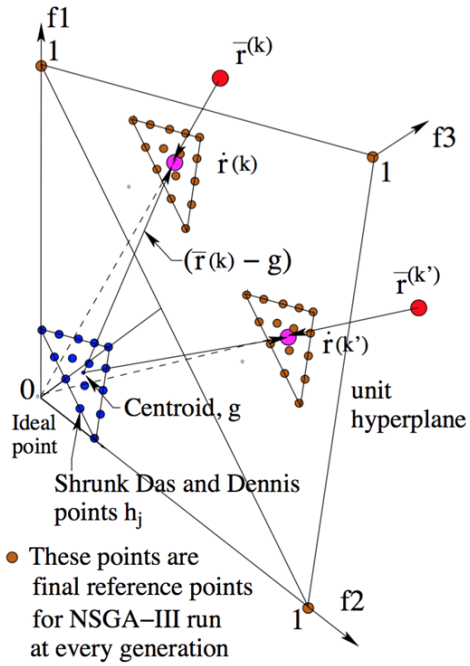

FIGURE 2.11. Process of associating solutions to reference points. Source: Vesikar et al. [44].

The process of associating solutions to reference points in NSGAIII is based on several fundamental concepts. The Ideal Point 0 represents the theoretical point at which all objectives are minimized simultaneously, although in practice it is unattainable. The centroid g corresponds to the center of mass of the solutions in the objective space and serves as a reference for data normalization.

The points are "shrunk" toward the centroid to better fit the actual distribution of the Pareto front. These adjusted points form the final reference points (shown in orange), which are used in each generation to guide solution selection.

The association between solutions and reference points occurs in the normalized hyperplane (defined, for example, by the equation $f_1 + f_2 + f_3 = 1$), ensuring a balanced distribution in the objective space. The vectors $r(k)$ represent the connection between a solution and its nearest reference point, considering normalization with respect to the centroid.

Next, the algorithm proceeds similarly to NSGAII, applying the selection process based on dominance fronts. However, the choice of individuals for the new generation is guided by the reference points, prioritizing solutions associated with less represented regions in the objective space. This mechanism ensures not only convergence to the Pareto front but also a diverse and balanced distribution of solutions along the front [33], [44].

2.3.6.3. *Multi-Objective Evolutionary Algorithm Based on Decomposition*

MOEA/D was proposed by Zhang and Li [34] as an alternative to algorithms based on PD. Instead of operating directly on a population based on dominance between solutions, MOEA/D decomposes the multi-objective problem into multiple scalar optimization subproblems, which are solved simultaneously.

Each subproblem is associated with a weight vector and maintained by a single individual in the population. The evolution of each solution depends only on its neighbors, defined by the proximity between the weight vectors. The update occurs locally, replacing only the individuals in the neighborhood if a new solution performs better according to the scalar decomposition function.

The algorithm can also be used for MaOO, showing good scalability and computational performance. Its modular architecture also allows the incorporation of various genetic operators and different decomposition strategies, such as Tchebycheff, weighted aggregation, or PBI (Penalty Boundary Intersection).

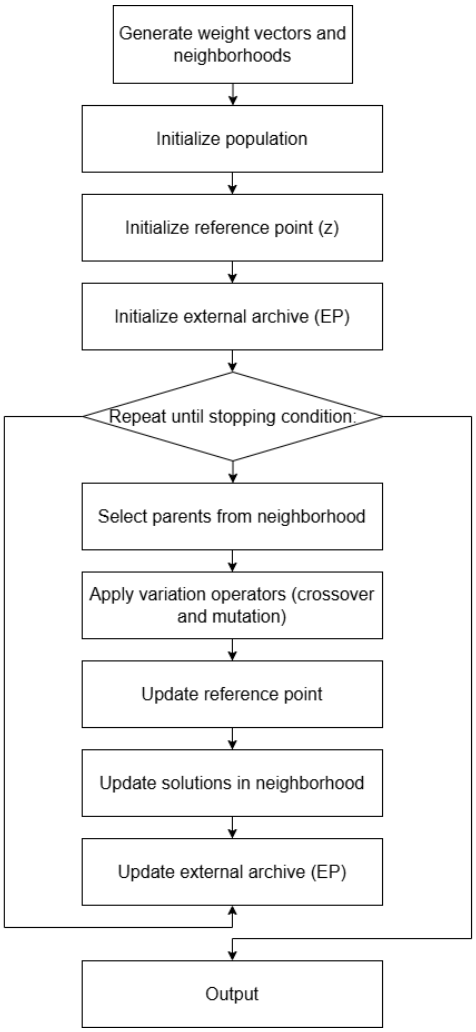

FIGURE 2.12. Basic flowchart of MOEA/D. Adapted from Zhang and Li [34].

Figure 2.12 represents the main steps of the algorithm, which can be described as follows:

- **Generate weight vectors and neighborhoods:** The weight vectors are uniformly distributed, and for each one, the set of closest neighbors is calculated based on Euclidean distance.
- **Initialize population:** Each weight vector is associated with a random initial solution. The reference vector z is defined with the best values found so far.
- **Run until stopping criterion:**
  - Select parents within the neighborhood;
  - Apply crossover and mutation operators;
  - Update the reference vector based on the new solution;
  - Update the neighborhood solutions based on the decomposition function;
  - (Optional) Update the external archive with non-dominated solutions.

- **Return solutions:** The final solutions correspond to the best found for each subproblem, or to the non-dominated solutions stored in an external archive (if used), which maintains a global set of solutions.

2.3.6.4. *ILP as a SOO Approach*

As discussed in Section 2.2.1, ILP is widely used in solving the UCTP when it is formulated as a single-objective problem. In this approach, different criteria are combined into a single, usually linear, objective function, allowing the application of exact optimization techniques. Although it provides optimal solutions, ILP is limited by scalability and is more suitable for small-sized instances or as a benchmark for comparison with heuristic approaches, as exemplified by the review conducted by Abdipoor et al. [12].

2.3.7. Software and Frameworks

Solving the UCTP involves not only defining mathematical models and algorithms but also using appropriate computational tools that enable their implementation, experimentation, and analysis. The choice of well-established *software* and *frameworks* contributes to the consistency of results and facilitates development, given the support for reusable components, standardized indicators, and extensive documentation.

The tools used in this dissertation are described below:

2.3.7.1. *jMetal*

jMetal is an object-oriented Java framework developed to facilitate the implementation and evaluation of metaheuristics applied to multi-objective optimization [41]. It includes a wide range of algorithms such as NSGAII, NSGAIII, MOCell, SMPSO, MOEA/D, as well as several benchmark problems (ZDT, DTLZ, WFG, CEC2009) and quality indicators like Hypervolume, Epsilon, SPREAD, GD, and IGD.

Its modular architecture allows for the creation and customization of new problems, algorithms, and operators, promoting code reuse and development flexibility. jMetal also supports systematic experimental studies, with automatic generation of LaTeX tables, boxplots, statistical tests (e.g., Wilcoxon), and a graphical interface to simplify usage. It is widely used in the scientific community due to its robustness, extensibility, and comprehensive support for the experimentation cycle.

2.3.7.2. *Python Optimization Modeling Objects*

Python Optimization Modeling Objects (Pyomo) is an open-source library developed for the Python programming language, designed for the formulation, resolution, and analysis of optimization models. It integrates directly with a high-level programming language, allowing symbolic modeling flexibility and the use of auxiliary libraries.

It supports a wide variety of problems, including linear, nonlinear, mixed-integer, stochastic, bilevel, and disjunctive programming. It offers scripting capabilities, parallelization, and integration with various solvers, such as Gurobi [45].

2.3.7.3. *GNU Linear Programming Kit*

GNU Linear Programming Kit (GLPK) is an open-source solver developed by the GNU project to solve Linear Programming (LP) and Mixed-Integer Programming (MIP) problems. Written in C, it is provided as a reusable library and includes methods such as simplex, interior-point, and branch-and-cut.

It supports the GNU MathProg language, compatible with AMPL, and can be integrated with various programming languages, such as Python via Pyomo. It is used in many academic and research contexts, especially when open-source solutions are preferred [46].

2.3.7.4. *Gurobi Optimizer*

Gurobi Optimizer (Gurobi) is a commercial solver used to solve LP, ILP, MIP, and Nonlinear Programming (NLP) problems. It implements algorithms such as primal and dual simplex, barrier (interior-point), branch-and-bound, and branch-and-cut.

It is provided as an optimized library with APIs available for several programming languages, including Python, Java, C++, C#, and R. In academic and research contexts, its integration with libraries such as Pyomo allows flexible modeling [3].

2.4. State of the Art

This study followed the guidelines proposed by Kitchenham & Charters in 2007 [47] for conducting a systematic literature review, ensuring a rigorous and transparent methodological approach. A systematic literature review is a structured research method that involves the identification, evaluation, and synthesis of relevant evidence available in the literature on a given topic.

Following the guidelines of Kitchenham & Charters (2017) [47], the study adopted a robust and reliable approach to analyze and synthesize the existing knowledge in the field of this dissertation.

To ensure a solid knowledge base and a comprehensive analysis, online repositories were selected that align with the objectives of this investigation, guaranteeing a complete and relevant literature review. The selected repositories were:

- IEEE Xplore (`https://ieeexplore.ieee.org/Xplore/home.jsp`)
- Scopus (`https://www.scopus.com/home.uri`)
- Web of Science (`https://www.webofscience.com/wos/woscc/basic-search`)
- ScienceDirect (`https://www.sciencedirect.com/`)
- ABSCO (`https://www.ebsco.com/`)

2.4.1. Systematic Review of Methodologies to Solve the UCTP

This subsection presents the first systematic review with the objective of identifying studies that analyze and compare the different approaches present in the literature, as well as the most widely used methodologies to solve the UCTP.

The steps carried out to conduct the literature review are detailed next:

The following keywords were used:

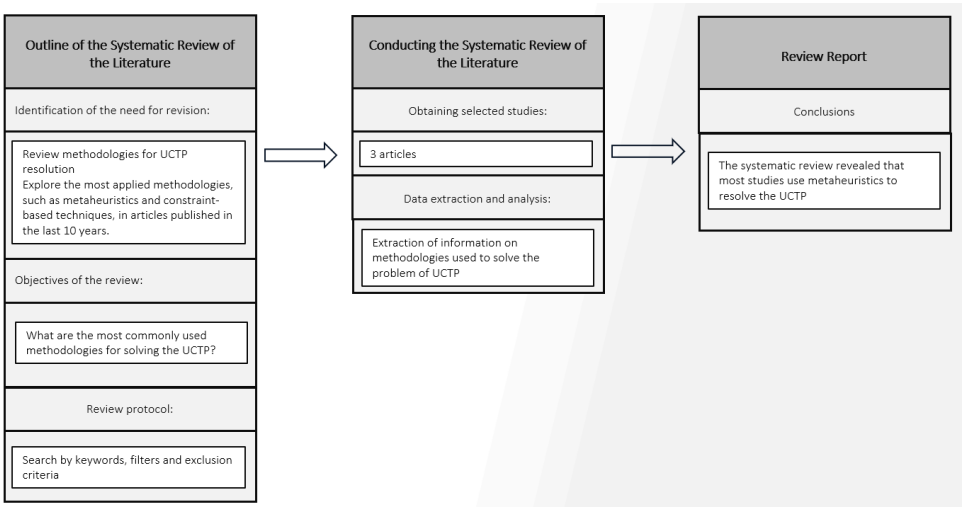

FIGURE 2.13. Steps of the systematic literature review

(("Abstract":University timetabling OR "Document Title":University timetabling) OR ("Abstract":UCTP OR "Document Title":UCTP)) AND (("Abstract":"methodologies" OR "Document Title":"methodologies")).

The keywords "University timetabling", "UCTP" and "methodologies" were used, enabling a broad search for articles analyzing various methodologies applied to the UCTP problem. All repositories used supported keywords without double quotation marks, which allowed the search to return results such as "University timetabling" or "University course timetabling." The absence of quotation marks means that the space is interpreted as an "AND".

TABLE 2.2. Filtering process

Database	**Without Filter**	**First Filter**	**Second Filter**	**Third Filter**
IEEE	6	5	0	2
Web of Science	14	9	2	1
Scopus	71	46	3	0
ScienceDirect	19	13	2	0
EBSCO	87	58	0	0
Total	197	131	7	3

2.4.1.1. *Filtering Process*

The filtering process consisted of four stages:

1. **Initial Search:** A total of 197 articles were retrieved from five scientific databases (IEEE, Web of Science, Scopus, ScienceDirect, and EBSCO), using the specified keywords applied to titles and abstracts.

2. **First Filter:** A temporal criterion was applied, restricting the analysis to articles published between 2014 and 2024, which reduced the total to 131.
3. **Second Filter:** An analysis of the remaining 131 articles was conducted, removing those that were not aligned with the objectives of the present systematic review. After this filter, 7 articles remained.
4. **Third Filter:** Duplicates among the databases were eliminated, resulting in a final total of 3 articles, which are analyzed in detail in the following section.

2.4.2. Review of the Identified Works

2.4.2.1. *Pandey and Sharma (2016)*

In 2016, Pandey and Sharma [7] classified the timetabling problem as NP-hard, emphasizing the difficulty of finding an optimal solution for large instances. The article points out that hard constraints, such as room and schedule allocation, must be satisfied at all costs, while soft constraints, such as teachers' preferences, allow greater flexibility.

The authors also highlight that the timetabling problem varies between universities due to the large number of constraints, making manual resolution difficult. In response, researchers have explored several techniques, such as metaheuristics, hyper-heuristics, adaptive methods, graph coloring, and particle swarm optimization. Constraint-based reasoning is also mentioned as an effective approach. Furthermore, the timetabling problem is categorized into variants such as Examination Timetabling Problem (ETTP), Curriculum-based Timetabling Problem (CTTP), Lecture Timetabling Problem (LTTP) (lecture timetabling problem), and Teacher Timetabling Problem (TTTP) (tournament timetabling problem).

Section II discusses the various variants of the timetabling problem, such as University Timetabling Problem (UTTP) (course and exam timetabling), CTTP (weekly course timetabling), LTTP (assigning one lecture per day), and ETTP (exam scheduling with constraints). The authors underline the importance of clear classification to effectively address the problem and present mathematical formulations of the main constraints to ensure feasible and optimized solutions.

Section III outlines the history of the timetabling problem, illustrating how the assignment of events to time slots while satisfying constraints increases computational complexity as problem size grows. Evolutionary algorithms and constraint-based methods were among the first proposed solutions, with significant contributions from Colorni et al. (1991, 1992, 1997) and Deris et al. (1997). Reviews by Carter (1986) and Corne et al. (1994) explored the application of genetic algorithms and linear programming. More recently, bio-inspired approaches such as ABC and ACO have gained prominence.

Conferences like Practice and Theory of Automated Timetabling (PATAT) and competitions such as International Timetabling Competition (ITC) have played crucial roles in motivating research and establishing benchmarks. In 2006, RongQu and Di Gaspero, and later Andrea Schaerf in 2007, facilitated result validation through the creation of websites that promote research in this area.

Section IV reviews various approaches to solving the timetabling problem, including linear programming (Carter, 1986; Deris et al., 1997), integer programming (Lajos, 1996), and logic programming with backtracking (Kang and White, 1992). More recent work by Schimmelpfeng and Helber (2007) using CPLEX for mixed-integer programming demonstrated success with real-world instances.

Constraint satisfaction problem (CSP) approaches are also discussed, with graph coloring emerging as a popular method, where events are represented as vertices and constraints as edges. Metaheuristics such as ACO, PSO, and genetic algorithms (GA), which combine local search procedures with learning strategies, are highlighted for their effectiveness in exploring the search space.

The conclusion reinforces the NP-hard complexity of the problem, emphasizing that evolutionary algorithms are preferred for generating feasible approximate solutions, with the ITC serving as a standard platform for result validation.

2.4.2.2. *Chen et al. (2021)*

In 2021, Chen et al. [5] identified the timetabling problem as a common combinatorial optimization challenge in academic institutions, emphasizing the relevance of studying recent methodologies applied to the UCTP. The article classifies these approaches and highlights the associated challenges, distinguishing between hard constraints, which must be strictly satisfied, and soft constraints, which are desirable but adaptable depending on institutional requirements.

The authors discuss methodologies for solving the UCTP using benchmark datasets such as Socha, ITC-02, ITC-07, and ITC-19. These methodologies are categorized into operational research-based techniques, single-solution metaheuristics, population-based metaheuristics, hyper-heuristics, multi-criteria approaches, and hybrid approaches. The article provides a detailed analysis of each category, discussing their advantages and disadvantages, with examples of studies that applied these methodologies. The application of these methods in real-world scenarios is also examined, evaluating their practical effectiveness and adaptability to the specific needs of the institutions studied. The variability of hard and soft constraints across different institutions is also emphasized.

The review included 35 articles, of which six explored operational research (OR)-based methodologies, ten focused on single-solution metaheuristics, eight on population-based metaheuristics, two on hyper-heuristics, one on a multi-criteria approach, and eight on hybrid approaches. In terms of benchmarks, single-solution and population-based metaheuristics were the most commonly used, with seven approaches each. Five of the seven single-solution approaches used SA, while two of the population-based approaches applied Ant Colony Optimization (ACO). In contrast, hyper-heuristics and multi-criteria approaches were less popular, although they present opportunities for future studies.

For real-world UCTTP applications, hybrid approaches were the most frequently used, followed by single-solution metaheuristics, operational research techniques, population-based metaheuristics, and hyper-heuristics.

In Section IV, the authors present several practical examples of real-world applications. One such study applied a hybrid approach combining Variable Neighborhood Search (Variable Neighborhood Search (VNS)) and TS to solve the UCTTP at the Fluminense Federal University. This hybrid method was developed using the FINESS framework and outperformed the individual performance of VNS and TS. Another example comes from the University of Bahrain, where a clustering and color-mapping algorithm was used to generate conflict-free schedules for 1,270 students and 83 courses, leveraging data mining techniques. Additionally, a Hybrid Genetic Algorithm (HGA) with four neighborhood operators was applied to manage teaching workloads in higher education institutions in the Philippines, generating feasible solutions for 118 classes and optimizing teachers' workloads. These practical examples highlight the adaptability and success of hybrid and metaheuristic methods in complex real-world scenarios.

Section VI of the article analyzes the most commonly used benchmark datasets and state-of-the-art methodologies for addressing the UCTP. The authors highlight that datasets from international competitions such as Socha, ITC 2002, ITC 2007 (Tracks 2 and 3), Hard Benchmark, and ITC 2019 are the most popular among researchers.

In Section VII, the authors report that the 35 approaches analyzed indicate that OR techniques generate feasible but low-quality solutions. Single-solution metaheuristics, such as SA, are effective but require precise parameter tuning. Population-based metaheuristics, such as GA and ACO, are superior in exploring the solution space but tend to be computationally intensive. Hyper-heuristics offer fast and adaptable algorithms but face difficulties in effectively exchanging information, while multi-criteria approaches demand substantial computational effort. Hybrid methods are noted for their complexity and high costs. The section concludes with a call for the development of effective algorithms that are simple to use and adaptable to various institutional contexts.

The conclusion emphasizes that the UCTP remains an active and important research area, largely motivated by international competitions. The authors also detail the characteristics of the benchmark datasets and discuss the limitations of each category of methodology. Opportunities for future research are identified, focusing on the development of more flexible and adaptable solutions for real academic institutions.

2.4.2.3. *Bashab et al. (2023)*

In 2023, Bashab et al. [8] explored methodologies used to address the University Course Timetabling Problem (UCTP), classified as NP-hard and belonging to the class of Combinatorial Optimization Problems (COP). The article highlights the application of various optimization techniques, particularly metaheuristics, in multi-objective optimization to find robust solutions that account for both hard and soft constraints. It reviews recent studies discussing key concepts, algorithms, benchmarks, and open questions, emphasizing the growing need for automation, while recognizing that human intervention is still required in certain aspects of the problem.

The study explains that the timetabling problem involves the allocation of exams and classes, aiming to prevent scheduling conflicts and ensure that rooms are suitable for the number of students. Course timetabling is further divided into curriculum-based and enrollment-based approaches, both involving the allocation of teachers, students, and rooms.

Bashab et al. [8] classify metaheuristic optimization methodologies into four main categories: Sequential Methods, Clustering Methods, Constraint-Based Methods, and Metaheuristic Methods, with extensions such as Hybrid Evolutionary Algorithms and Hyper-heuristics. The study also details simpler heuristic approaches like the LS algorithm, along with more advanced metaheuristics such as Simulated Annealing (SA), TS, and population-based methods like GA, PSO, ACO, and ABC, which iteratively seek optimal solutions.

Regarding algorithm evaluation, the article focuses on balancing exploration and exploitation, using evolutionary operators such as crossover, mutation, and selection. To assess performance, metrics like accuracy, number of evaluations, and execution times are compared.

In terms of benchmarks, the study references datasets from ITC2007 and Purdue University for course timetabling, as well as datasets from Toronto, Nottingham, and Melbourne for exam timetabling. These benchmarks are essential for evaluating the effectiveness of applied approaches.

In the discussion, the research presents a comprehensive review of the university timetabling problem, divided into four main stages: understanding the problem, classifying constraints, analyzing applied approaches, and defining criteria to select the best method. The main concern is satisfying all constraints, which become more complex as they are added, making it more difficult to find the optimal solution. Suggested timetables should be generalized, satisfying all hard constraints while minimizing the soft ones.

Metaheuristics have been widely explored to optimize this type of problem and are divided into single-solution methods and population-based methods. Figure 2.14 highlights that Genetic Algorithms (GA) had a significant presence (16%) compared to other methods, while hybrid algorithms stood out with a 35% rate due to their effectiveness in final results. Additionally, swarm intelligence algorithms, which use a decentralized and self-organizing approach, also demonstrated efficiency and reliability.

In terms of publication types, most research was published in scientific journals (86) and conferences (58), indicating strong academic interest and discussion among researchers. However, there is a lack of performance measures to compare different algorithms, and the difficulty of making fair comparisons is emphasized due to the variability of constraints and objectives among different institutions.

For future directions, the article suggests exploring more modern optimization algorithms, such as multi-objective evolutionary algorithms, which are promising for obtaining robust solutions for the timetabling problem. The hybridization of algorithms and the

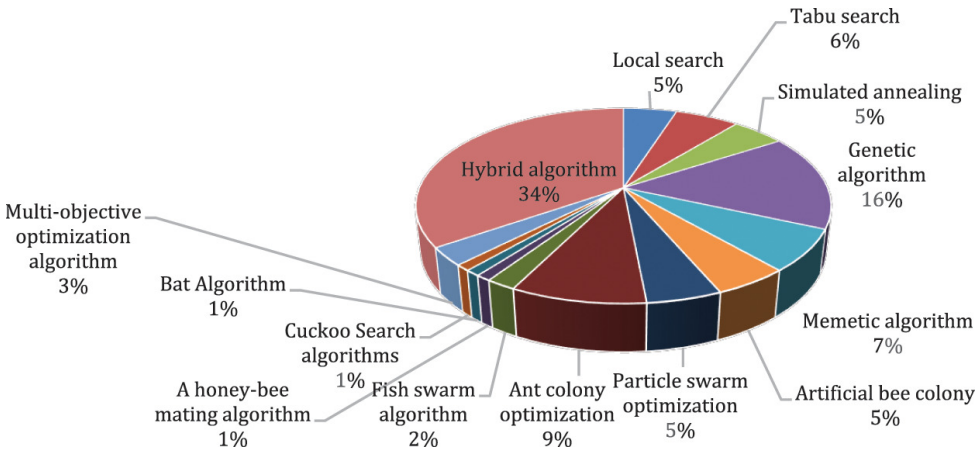

FIGURE 2.14. Distribution of the publication rate of metaheuristic algorithms to solve the UCTP, Source: Bashab et al. [8]

exploration of new techniques are seen as important ways to achieve greater efficiency in solving this type of complex problem.

Finally, the conclusion highlights the importance of hybrid algorithms in achieving robust and effective solutions and suggests that there is room to improve performance metrics and create more realistic benchmarks. Additionally, the increase in the number of publications in the last decade underscores the growing relevance of the university timetabling problem in the field of combinatorial optimization.

2.4.3. Conclusions and Future Pathways for Solving the University Course Timetabling Problem

The analysis of the works by Pandey and Sharma (2016) [7], Chen et al. (2021) [5], and Bashab et al. (2023) [8] indicates that the UCTP remains one of the main challenges in combinatorial optimization. The complexity of the problem results from the need to simultaneously satisfy several hard and soft constraints.

The review shows that despite the historical application of methods based on ILP and MILP, these face scalability limitations when applied to large instances of the problem. On the other hand, approaches based on MH, especially population-based ones such as GA and ACO algorithms, have demonstrated effectiveness in exploring the solution space.

Both Chen et al. (2021) [5] and Bashab et al. (2023) [8] highlight hybrid algorithms as the most effective and promising approaches. The combination of techniques, such as GA with SA or VNS, allows the integration of global exploration with local refinement, resulting in more robust and adaptable solutions to different institutional scenarios. Bashab et al. (2023) [8] reinforce that hybrid algorithms accounted for 35% of the successful approaches identified in the recent literature, significantly outperforming isolated approaches.

Multi-objective evolutionary algorithms, such as NSGAII, and more recently, NSGAIII and MOEA/D, are pointed out as highly promising alternatives, especially due to their

ability to handle multiple evaluation criteria simultaneously, which is an inherent feature of the UCTP.

Additionally, approaches such as hyper-heuristics and swarm intelligence-based methods have also gained relevance due to their adaptability and simplicity of implementation, although they still require further exploration in real-world contexts.

Thus, based on the systematic review conducted, it can be concluded that:

- Hybrid approaches that combine the exploratory power of metaheuristics with local intensification strategies currently represent the most effective line for solving the UCTP;
- Evolutionary algorithms such as NSGAII, NSGAIII, and MOEA/D are highly recommended for future research, especially in contexts with multiple objectives;
- The automation of the scheduling generation process continues to evolve but still requires enough flexibility to consider human preferences and institutional specificities;
- It is essential to validate these approaches using recognized benchmarks, such as those from the ITC, and to continue improving comparison metrics to effectively evaluate different algorithms.

The literature review, as well as the systematic review carried out, will serve as the foundation for the practical approach to the UCTP problem at Iscte, guiding the selection of the most appropriate methods and tools for its resolution.

[This page is intentionally left blank.]

CHAPTER 3

Problem and Proposed Solution

3.1. Problem Description

The problem addressed in this dissertation concerns room allocation to existing timetables, which is a specific subset of the UCTP, where the main objective is to optimize previously defined schedules by minimizing a set of soft constraints.

To this end, a real dataset was used, corresponding to the calendar of one academic semester at Iscte – Instituto Universitário de Lisboa, consisting of approximately 10,000 students, 26,020 classes, and 126 classrooms of 23 distinct types.

Each classroom is described by properties such as capacity, building, and specific characteristics (e.g., computer lab), while each class includes information about enrolled students, the course, the group, the schedule, and the required classroom features. The following sections formally present the sets of rooms and classes, as well as their respective properties:

- Let $\mathcal{R} = \{r_1, r_2, \ldots, r_{|R|}\}$ be the set of rooms, where each room $r_j \in R$ is described by the following properties:
  - **Name**: room identifier;
  - **Normal capacity**: number of available seats;
  - **Number of features**: total number of features, such as computer lab, journalism lab, regular classroom, etc.;
  - **Features**: set of room features, such as computer lab, journalism lab, regular classroom, etc.;
  - **Building**: name of the building to which the room belongs.
- Let $C = \{c_1, c_2, c_3, \ldots, c_{|C|}\}$ be the set of classes, where each class $c_i \in C$ is described by the following properties:
  - **Course**: set of courses separated by comma, where each course has its identifier, for example: ME, LETI, LEI, etc.;
  - **Shift**: class shift;
  - **Class**: class name;
  - **Enrolled students**: number of enrolled students;
  - **Day of week**: day of the week;
  - **Time**: execution time, consisting of the day, start and end time of the class;
  - **Requested room features**: requested room characteristics, such as computer lab, journalism lab, regular classroom, etc.;
  - **Actual room features**: set of features of the assigned room;

3.2. Problem Constraints

As indicated in the literature review, the constraints applied to the UCTP can be classified as hard or soft. In this dissertation, this distinction is concretely applied, defining which constraints must be strictly fulfilled (hard) and which represent preferences to be satisfied whenever possible (soft).

3.2.1. Soft Constraints

The soft constraints considered are four:

- **S1 Student relocation:** aims to minimize the change of building between classes within the same shift, same course unit, and on the same day; that is, students should not have to move between different buildings within short time intervals;
- **S2 Incompatibility between requested room type and assigned room:** evaluates the compatibility between the requested and the actually assigned room, penalizing cases where the assigned room does not meet the requested requirements, e.g., the need for a computer lab;
- **S3 Room capacity waste:** penalizes the waste of seats, accounting for cases where the assigned room has significantly more capacity than the number of enrolled students, with an acceptable waste threshold defined (e.g., 40%);
- **S4 Room change in consecutive classes:** penalizes when the same room is not assigned to classes that belong to the same course unit and shift, and take place on the same day.

The mathematical formulation of these constraints is as follows:

S1 Student relocation between buildings (SD)

$$\min f_1 = \sum_{\substack{i<j \\ c_i,c_j \in \mathcal{C} \\ shift(c_i)=shift(c_j) \\ unit(c_i)=unit(c_j) \\ day(c_i)=day(c_j)}} \delta(building(r_i) \neq building(r_j))$$

Where:

- $\delta(\cdot)$ is an indicator function that returns 1 if rooms r_i and r_j belong to different buildings, and 0 otherwise;
- r_i and r_j are the rooms assigned to classes c_i and c_j, respectively;
- The function penalizes the allocation of classes with the same shift, course unit, and day to different buildings.

S2 Incompatibility between requested and assigned room type (MCRA)

$$\min f_2 = \sum_{i \in \mathcal{C}} incompatibility(c_i, r_i)$$

Where:

- $incompatibility(c_i, r_i)$ represents the number of features requested by class c_i that are not met by the assigned room r_i;
- Penalizes the allocation of rooms that do not meet the functional requirements of the class (e.g., type of lab, technical resources, etc.).

S3 Waste of room capacity (CRA)

$$\min f_3 = \sum_{i \in \mathcal{C}} \delta \left(\frac{capacity(r_i) - enrolled(c_i)}{capacity(r_i)} \times 100 > \theta \right)$$

Where:

- $capacity(r_i)$: capacity of the room assigned to class c_i;
- $enrolled(c_i)$: number of students enrolled in class c_i;
- θ: maximum acceptable waste percentage (e.g., $\theta = 40\%$);
- $\delta(\cdot)$: indicator function that returns 1 if the waste exceeds θ, 0 otherwise;
- Penalizes assignments where the room has significantly more capacity than required.

S4 Room change for consecutive classes of the same course unit and shift (RCCC)

$$\min f_6 = \sum_{\substack{i<j \\ c_i, c_j \in \mathcal{C} \\ unit(c_i)=unit(c_j) \\ shift(c_i)=shift(c_j)}} \delta \left(room(c_i) \neq room(c_j) \right)$$

Where:

- $unit(c_i)$: course unit associated with class c_i;
- $shift(c_i)$: shift of class c_i;
- $room(c_i)$: room assigned to class c_i;
- $\delta(\cdot)$: indicator function that returns 1 if the rooms are different, 0 otherwise;
- Penalizes assignments where different rooms are allocated to classes of the same course unit and shift.

3.2.2. Hard Constraints

The hard constraints considered are:

- **H1:** aims to ensure that no class is assigned to a room with insufficient capacity for the number of students;
- **H2:** avoids class overlap in the same room, i.e., prevents two classes from taking place simultaneously in the same physical space;

The mathematical formulation of these constraints is presented below:

H1 Insufficient room capacity (ROB)

$$\min f_4 = \sum_{i \in \mathcal{C}} \delta \left(capacity(r_i) < enrolled(c_i) \right)$$

Where:

- $capacity(r_i)$: normal capacity of the room assigned to class c_i;
- $enrolled(c_i)$: number of students enrolled in class c_i;
- $\delta(\cdot)$: indicator function that returns 1 if the condition is true, 0 otherwise;
- Penalizes allocations where the room does not have enough seats for all enrolled students.

H2 Overlapping classes in the same room (RO)

$$\min f_5 = \sum_{\substack{i<j \\ c_i,c_j \in \mathcal{C} \\ day(c_i)=day(c_j) \\ room(c_i)=room(c_j)}} \delta\left(overlap(c_i, c_j)\right)$$

Where:

- $room(c_i)$: room assigned to class c_i;
- $day(c_i)$: day of the week in which the class occurs;
- $overlap(c_i, c_j)$: function that returns 1 if the class times overlap, 0 otherwise;
- Penalizes cases in which two classes are assigned to the same room on the same day with overlapping schedules.

3.3. Proposed Solution

The strategy defined to solve the UCTP at Iscte was structured based on the literature review and the conclusions of the systematic review. The approach plan integrates multi-objective optimization techniques, hybrid methods, and Integer Linear Programming ILP, as outlined below:

1. **Evolutionary Multi-Objective Optimization (EMO):** The multi-objective evolutionary algorithms that stood out in the systematic review of Subsection 2.4.3 will be applied, namely:
   - NSGA-II
   - NSGA-III
   - MOEA/D

   These algorithms are suitable for dealing with problems involving multiple conflicting criteria, such as the UCTP at Iscte.
2. **Hybrid Approaches with Local Intensification:** Hybrid versions of the evolutionary algorithms will be developed, incorporating local intensification mechanisms (local search). These strategies aim to refine and improve the solutions generated during the evolutionary process by performing small targeted adjustments to each individual in the population.
3. **Integer Linear Programming:** ILP will be used to solve small-scale instances of the problem. This approach enables the generation of optimal solutions, which will be used as a reference to validate the effectiveness of the approximate solutions produced by the evolutionary and hybrid algorithms.

4. **Heuristic Initial Population (Constructive Population):** In addition to the random generation of initial solutions, a constructive strategy based on simple heuristics and allocation rules will be used. This initial population will take into account the criteria defined in the problem modeling presented in Section 3.1, and will be compared to the randomly generated populations. Tests will also include variations with and without local search applied to this constructive population.

All proposed variants (base algorithms, hybrids, ILP, and constructive populations) will be evaluated using real data from Iscte and the benchmark instance **comp02.ctt** from the ITC dataset. The comparison of results will be based on the quality metrics described in Section 2.3.4.

This approach aims to assess the impact of heuristic construction and local intensification on evolutionary algorithms and to demonstrate the robustness of hybrid approaches in the practical and realistic resolution of complex scheduling problems and large instances such as that of Iscte.

[This page is intentionally left blank.]

CHAPTER 4

Implementation

The development involved the use of the NSGAII, NSGAIII, and MOEA/D algorithms, integrated into the jMetal framework, and an approach based on ILP using the Pyomo library. Regarding the EMO, customized versions with local search integration were implemented.

The real data from ISCTE were integrated and used to validate the performance of the approaches in the scenarios defined in Section 3.3, reflecting the complexity and specific constraints of the institution.

To support the execution and validation of the implemented algorithms, a Java-based solution was developed for data reading, solution evaluation, and comparison between algorithms. This solution was used to test the EMO algorithms on the comp02.ctt instance from the ITC2007 benchmark, enabling a comparative evaluation with a standardized instance from the literature.

4.1. Tools Used

The main tools used for the implementation were:

- **jMetal Framework:** For the application of evolutionary algorithms, this framework provided the infrastructure to define problems, solutions, genetic operators, objective functions, and to conduct experimental studies.
- **Pyomo:** For the mathematical modeling of the problem, the Pyomo library was used, which allows defining optimization models declaratively.
- **Gurobi Optimizer:** The Gurobi solver was used to solve the instances formulated with ILP. Initially, the use of GLPK was considered, as it is an open-source solution commonly used in academic contexts. However, it only supports single-objective problems and does not allow the simultaneous definition of multiple objective functions, as required in this approach.

  Since the developed model uses a structure of separated objectives (multi-objective), it was necessary to use a solver with native support for this type of modeling. After research, it was found that Gurobi is available free of charge for academic purposes, making it a suitable alternative for solving ILP problems with multiple objectives.

The programming languages used were Java and Python, chosen based on compatibility with the tools used. Java was selected because version 6.1 of the jMetal framework was developed in this language. Python was used for the mathematical modeling of the problem, as it is compatible with the Pyomo library and the solver.

4.2. Solution Implementation in Java

The implementation was developed based on object-oriented programming (OOP) principles, using a modular architecture to facilitate maintenance, extension, and reuse of components. Figure 4.1 presents the UML diagram developed, representing the main classes involved in modeling the room and timetable allocation problem.

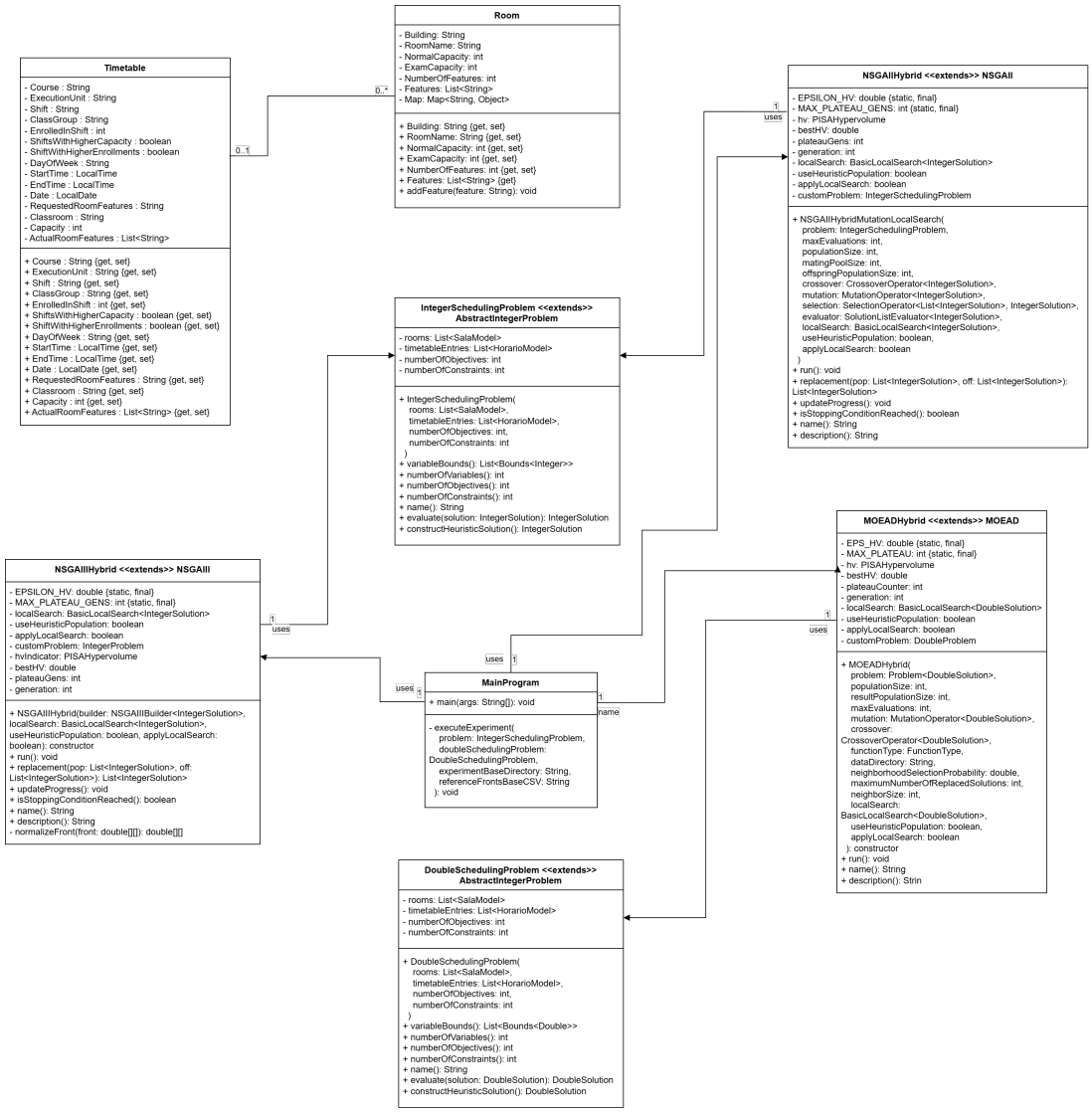

FIGURE 4.1. UML diagram representing the problem modeling in Java using the jMetal framework

The solution was developed following object-oriented programming (OOP) principles, adopting a modular architecture with the aim of facilitating the maintenance, extension, and reuse of components. Figure 4.1 presents the UML diagram with the main entities involved in the modeling of the room and timetable allocation problem.

4.2.1. Data Integration

The data provided were encoded in CSV format. Therefore, parsing was performed and modeling was carried out through the `RoomModel` and `TimetableModel` classes, which

represent the rooms and schedules, respectively. The former stores information such as room capacity and specific features, while the latter contains data related to the curricular unit such as class, number of students, and schedule. The attributes and properties are defined as shown in Figure 4.1.

4.2.2. Problem Structure and Modeling

The `IntegerSchedulingProblem` class, which extends `AbstractIntegerProblem`, and the `DoubleSchedulingProblem` class, which extends `AbstractDoubleProblem` from the jMetal framework, model and define the instance of the optimization problem in their respective types. It is in these components that the objective functions and evaluation logic are implemented, including the calculation of penalties associated with the constraints defined in Section 3.2.

4.2.3. Algorithms and Hybrid Strategies

Customized versions of the NSGAII, NSGAIII, and MOEA/D algorithms were developed, integrating the BasicLocalSearch intensification technique native to the jMetal framework. Each version extends the respective class of each algorithm, allowing for the incorporation of additional strategies such as the use of a heuristic initial population and the optional application of local search, as well as the implementation of the same stopping criterion for all algorithms.

4.2.3.1. *Local Search*

The local search technique was integrated into the EMO algorithms using the BasicLocalSearch class provided by the jMetal framework, configured with an Integer Polynomial Mutation operator, a standard dominance comparator (DefaultDominanceComparator), and the reference to the modeled problem.

The polynomial mutation was parameterized with a probability of $\frac{1}{n}$, where n is the total number of genes per chromosome, and a distribution index of 20.0. The dominance comparator was responsible for evaluating whether the locally generated new solution dominates the current one. The local search was configured to generate a single neighbor per iteration. If this neighbor shows better quality according to the dominance criteria, it replaces the current solution, making the evolutionary process more effective.

4.2.3.2. *Constructive Population*

To increase the effectiveness of the initial solutions and provide higher quality heuristics to the chromosome, a constructive method (constructHeuristicSolution) was implemented. This method assigns each class to a room based on the hard constraints.

The implemented logic can be formally described as follows:

Let $C = \{c_1, c_2, \ldots, c_n\}$ be the set of classes and $R = \{r_1, r_2, \ldots, r_m\}$ the set of available rooms. For each class $c_i \in C$, the heuristic tries to find the first room $r_j \in R$ such that:

$$\text{capacity}(r_j) \geq \text{students}(c_i)$$
$$\neg\text{overlap}(c_i, r_j, D_i)$$

where:

- capacity(r_j) is the capacity of room r_j,
- students(c_i) is the number of students in class c_i,
- D_i represents the day on which class c_i occurs,
- overlap(c_i, r_j, D_i) is a function that checks if there are already classes assigned to room r_j on day D_i with overlapping times.

If no viable room exists, a random room is chosen:

$$r_j \leftarrow \text{rand}(R)$$

4.2.3.3. *Stopping Criterion Based on HyperVolume Plateau*

Instead of using a fixed number of iterations as a stopping criterion, a mechanism based on the hypervolume HV quality indicator was introduced. That is, at each generation, the HV of the current PF is calculated and compared with the best value obtained so far. If there is no significant improvement over a predefined number of consecutive generations, the algorithm terminates early, avoiding unnecessary iterations.

The operation follows these steps:

- **Initialization:**
  - The hypervolume of the initial population is calculated: bestHV = HV(Generation 0).
- **During each generation:**
  - The current hypervolume of the front is calculated: hvNow.
  - It is compared with the best known value bestHV.
  - If the improvement is greater than a small threshold ε, that is:

$$\text{hvNow} > \text{bestHV} + \varepsilon \quad \text{with } \varepsilon = 10^{-6}$$

    then:
    - Update the best value: bestHV $\leftarrow$ hvNow;
    - Reset the counter: plateauGens $\leftarrow 0$.

    Otherwise:
    - Increment the counter: plateauGens $++$.
- **Stopping criterion:**
  - The algorithm stops when:
    1. The maximum number of iterations is reached; **or**
    2. The number of generations without improvement reaches the limit:

$$\text{plateauGens} \geq \text{MAX\_PLATEAU\_GENS} \quad \text{(e.g., 100)}$$

4.2.4. Integer Linear Programming Model

This Integer Linear Programming model aims to allocate schedules to rooms in an optimal way, ensuring a single strict constraint: each schedule must be allocated to exactly one room (ROB). To ensure the total feasibility of the model, a *dummy* room was introduced, whose use is heavily penalized in the objective function.

The need for this fictitious room arose due to the existence of some schedules for which no real one meets the minimum requirements. A practical example occurs with the course *Data Analysis in Human Resource Management III*, which has 59 students enrolled in a shift that requires a space with the characteristics of a computer lab. However, all such rooms, in any building, have a maximum capacity of only 50 seats. Thus, without the introduction of the *dummy* room, the model would become infeasible.

For this reason, unlike the implementation with jMetal, constraints H1 (ROB) and H2 (RO) were not enforced as hard in the ILP model. Their enforcement would render the model infeasible in scenarios with physical limitations, such as the unavailability of rooms with sufficient capacity. Therefore, they were modeled as strongly penalized soft constraints, enabling the model to produce feasible solutions even under exceptional circumstances.

The objective function minimizes penalties associated with the following soft constraints:

- **RCCC** — Change of room between consecutive classes.
- **SD** — Relocation of students due to inadequate room capacity.
- **MCRA** — Incompatibility between the type of room requested and the type of room assigned.
- **CRA** — Assignment of rooms with capacity lower than the number of students enrolled.
- **EDIF** — Use of multiple buildings by the same group.
- **DUMMY** — Use of the *dummy* room, heavily penalized so that it is avoided whenever possible.

The model was implemented in `Python`, using the `Pyomo` library, and the ILP solver `Gurobi`.

4.2.5. Execution

Execution is orchestrated by the `Main` class, which performs data reading, problem configuration, parameter definition as per Table 5.4, and algorithm execution. This structure enables experiments with different approaches and instances.

[This page is intentionally left blank.]

CHAPTER 5

Experiments and Parameters

This section describes the details of the experimental setup adopted to evaluate the proposed approaches for solving the room and timetable allocation problem at Iscte.

5.1. Data Used

To validate the effectiveness of the proposed algorithms, two categories of instances were used:

- **Iscte Instance:** The data provided by the institution represent a realistic scenario of one semester, with characteristics and specific constraints of the institution, as shown in Table 5.1.

Table 5.1. Description of the Iscte Instance

Element	Description
Source	University Institute of Lisbon (ISCTE)
Period	One full academic semester
Number of students	Approximately 10,000
Number of classes	26,020
Number of rooms	126
Room types	23 distinct types (e.g., auditorium, laboratory, standard classroom, etc.)

- **Benchmark Instance (ITC2007 – comp02.ctt):** This instance was used to compare the results obtained and validate the performance of the EA and ILP methods. Table 5.2 indicates the characteristics of the dataset.

TABLE 5.2. Description of the Benchmark Instance (comp02.ctt – ITC2007)

Element	**Description**
Source	International Timetabling Competition 2007 (ITC2007)
Instance	comp02.ctt
Number of courses	72
Number of classes	522
Number of days	5
Total number of periods	30
Number of rooms	16

5.2. Experimental Settings

The experiments conducted were based on two distinct approaches for solving the room and timetable allocation problem: EMO and Integer Linear Programming (ILP).

The EMO-based approach used the same three algorithms in all configurations: NSGAII, NSGAIII, and MOEA/D. Four different experimental setups were carried out, varying based on the use of hybrid strategies (with local search) and the application of a heuristic (constructive) initial population.

Table 5.3 summarizes the main settings adopted in each experiment.

TABLE 5.3. Summary of configurations used in the different experimental setups

Experiment	**Hybrid (Local Search)**	**Constructive Population**	**Runs**
`Experiment 1`	Yes	No	25
`Experiment 2`	No	No	25
`Experiment 3`	Yes	Yes	25
`Experiment 4`	No	Yes	25

5.3. Parameters of the Evolutionary Algorithms

The parameters used in the configuration of the algorithms and experiments of the Evolutionary Algorithms are presented in Table 5.4, which summarizes the values for the main operators and stopping criteria.

TABLE 5.4. Algorithm configurations

Parameter	NSGA-II	NSGA-III	MOEA/D
Population size	200	200	200
Max evaluations	80,000	N/A	80,000
Max iterations	N/A	400	N/A
Mutation operator	Integer Polynomial Mutation	Integer Polynomial Mutation	Polynomial Mutation
Mutation probability	$\frac{1}{n}$, where n is the total number of genes per individual	$\frac{1}{n}$, where n is the total number of genes per individual	$\frac{1}{n}$, where n is the total number of genes per individual
Distribution index (mutation)	20.0	20.0	20.0
Crossover operator	Integer SBX Crossover	Integer SBX Crossover	Differential Evolution Crossover
Crossover probability	0.9	0.9	$CR = 0.5$
Distribution index (crossover)	18.0	18.0	$F = 1.0$
Selection operator	Ranking And Crowding Distance Comparator	Ranking And Crowding Distance Comparator	N/A
Divisions (NSGA-III)	N/A	4	N/A
Decomposition type	N/A	N/A	FunctionType. TCHE

* NSGA-II and MOEA/D were limited by a maximum number of evaluations (80,000), while NSGA-III was limited by a maximum number of iterations (400 generations), with population size 200.

5.4. Execution Environment Setup

The experiments were conducted in an environment with the following specifications:

- **Processor:** AMD Ryzen 7 7840HS with 8 physical cores and 16 logical threads;
- **RAM:** 40 GB;
- **Operating System:** Windows 11 64-bit;

- **IDE:** IntelliJ IDEA (Java) and VS Code (Python);
- **Software:** For the evolutionary algorithms, the jMetal 6.1 framework (Java version) was used. Python 3.12 was used with Pyomo and Gurobi for the ILP model;

CHAPTER 6

Results and Discussion

This chapter presents the results obtained from the experiments conducted with the EA and the ILP, using data from Iscte and the *comp02.ctt* instance from ITC2007.

6.1. Tests of the Evolutionary Algorithms with the Iscte – Instituto Universitário de Lisboa Dataset

As previously mentioned, the objective of this work is to carry out a benchmarking process to solve the room allocation problem for schedules at the Iscte institution. The tests for the variants described in Section 5.2 are presented below.

6.1.1. Experiment 1 – Without Constructive Population and With Local Search

This experiment followed the setup defined in Section 5.4, using local search but without a heuristic initial population. The execution time was 16 hours and 40 minutes with 25 runs per algorithm.

Table 6.1 summarizes the mean and standard deviation values of the NHV and Generalized Spread Indicator (GSPREAD) indicators.

TABLE 6.1. Problem 1 – Mean and standard deviation of NHV and GSPREAD for each algorithm

Algorithm	**NHV (mean ± std)**	**GSPREAD (mean ± std)**
MOEA/D	0.998 ± 0.005	1.002 ± 0.067
NSGA-II	0.550 ± 0.128	0.405 ± 0.030
NSGA-III	0.591 ± 0.090	0.636 ± 0.073

Figure 6.1 presents the NHV values, where the MOEA/D algorithm achieved the best result, with a mean of 0.998, close to the ideal (1.0), and with low variation between runs, as the standard deviation was 0.005, demonstrating high stability and solution quality. The NSGAIII showed intermediate performance with 0.591 ± 0.090, while the NSGAII had the worst result with 0.550 ± 0.128, indicating lower PF coverage and higher dispersion.

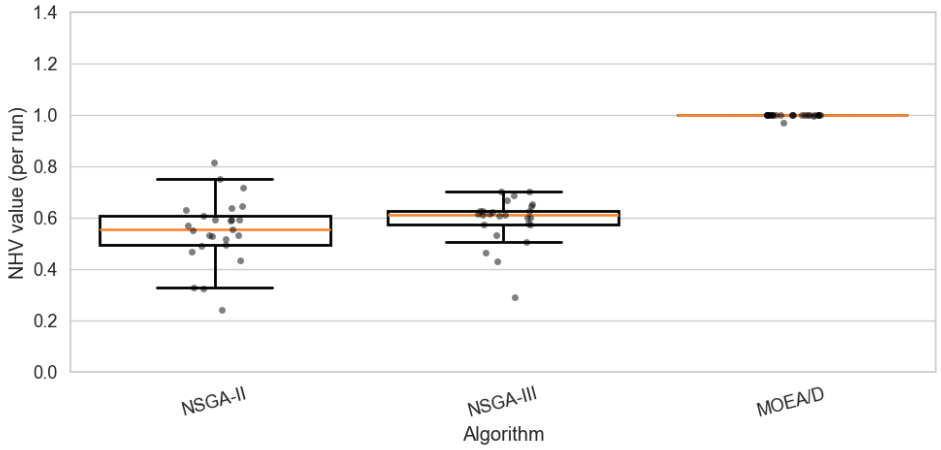

FIGURE 6.1. HV values

In Figure 6.2, which displays the GSPREAD values, it is observed that NSGAII achieved the lowest mean value (0.405 ± 0.030), indicating a good distribution of solutions. NSGAIII showed intermediate performance, with greater dispersion and some outliers. Meanwhile, MOEA/D, despite its excellent NHV result, presented the highest value (1.002 ± 0.067), reflecting difficulties in maintaining diversity along the PF.

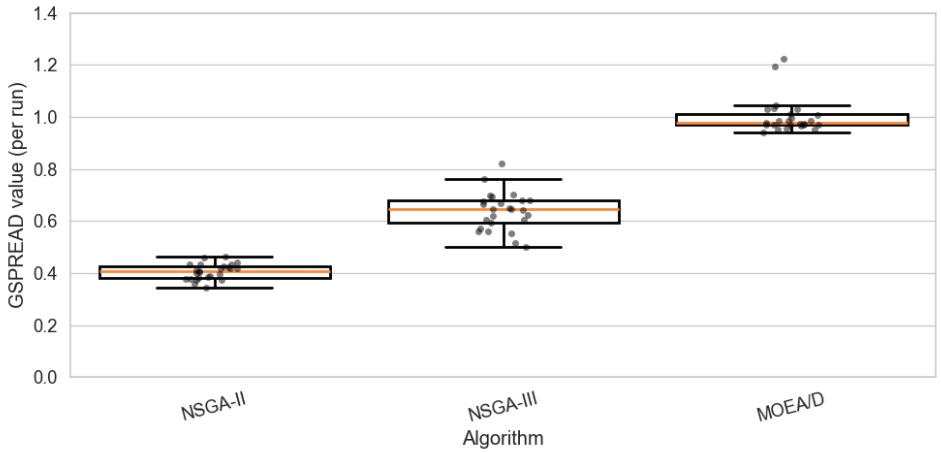

FIGURE 6.2. GSPREAD values

Figure 6.3 shows the parallel coordinates plot for the 25 runs, where it can be observed that MOEA/D exhibits greater variability, especially in the hard constraints (ROB and RO). On the other hand, NSGAII and NSGAIII display more clustered curves, demonstrating greater consistency across runs. Although MOEA/D reaches favorable extreme solutions, its global instability may be associated with the randomness in the generation of the initial population.

Regarding the ILP, a very stable behavior is observed, with good results on hard constraints, very close to zero. Despite the good performance on soft constraints as well, the generation of a single solution limits the exploration of trade-offs among objectives.

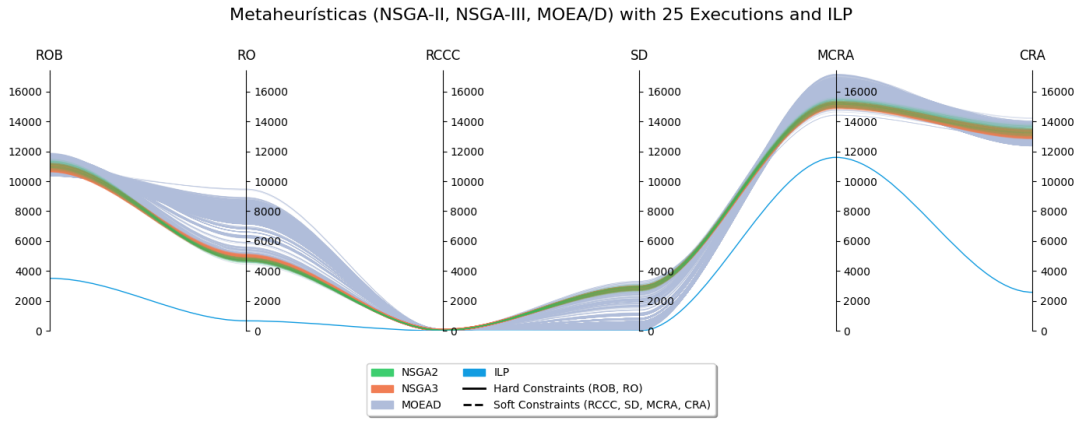

FIGURE 6.3. Reference front plotting for six objectives room allocation UCTP – Experiment 1

6.1.2. Experiment 2 – Without Local Search, with Constructive Population

This experiment used a heuristic initial population without applying local search. The execution time was 14 hours and 10 minutes, with 25 runs per algorithm.

Table 6.2 presents the mean and standard deviation values for the NHV and GSPREAD indicators.

TABLE 6.2. Problem 1 – Mean and standard deviation of NHV and GSPREAD for each algorithm

Algorithm	**NHV (mean ± std)**	**GSPREAD (mean ± std)**
MOEA/D	0.999 ± 0.001	0.911 ± 0.127
NSGA-II	0.798 ± 0.046	0.404 ± 0.026
NSGA-III	0.376 ± 0.107	0.371 ± 0.023

As shown in Figure 6.4, MOEA/D once again achieved the best NHV (0.999 ± 0.001), with nearly unitary results across all runs, almost entirely covering the PF. NSGAII demonstrated good performance (0.798 ± 0.046), with low variability. Meanwhile, NSGAIII registered the worst result (0.376 ± 0.107), with high dispersion and difficulty in reaching regions near the reference point.

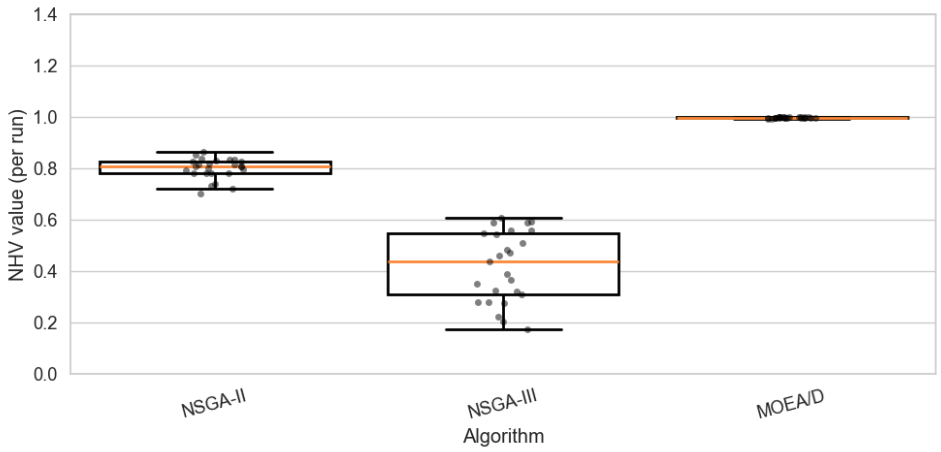

FIGURE 6.4. HV values

Figure 6.5 shows that NSGAIII achieved the lowest mean GSPREAD value (0.371 ± 0.023), reflecting a uniform distribution of solutions. NSGAII maintained similar performance to the previous experiment, also displaying good diversity. Regarding MOEA/D, although efficient in convergence, it produced the worst result (0.911 ± 0.127), indicating a less balanced front.

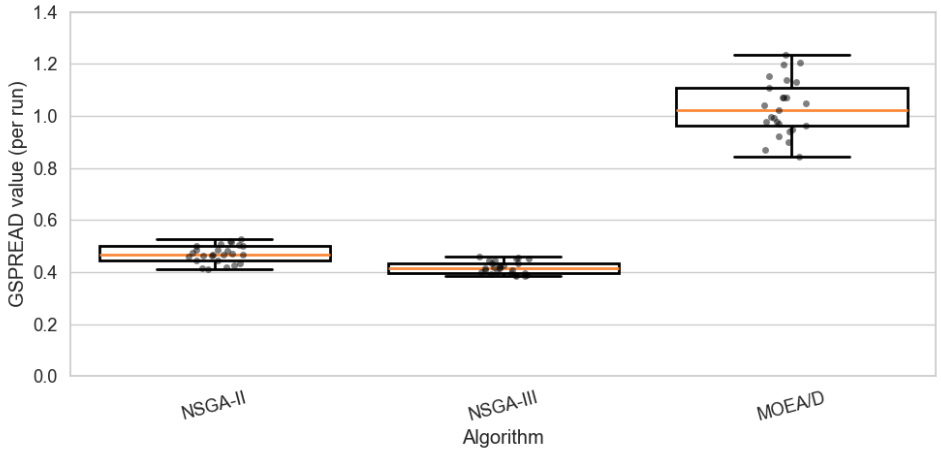

FIGURE 6.5. GSPREAD values

In Figure 6.6, the parallel coordinates plot reinforces the observed patterns, that is, MOEA/D shows greater variability, especially on the ROB and RO constraints. In contrast, NSGAII and NSGAIII maintain greater consistency across runs. MOEA/D continues to obtain high-performing extreme solutions but with pronounced instability.

In this second experiment, the heuristic population proved effective, guiding all algorithms to near-zero minimizations of the hard constraints, even surpassing the ILP. For the soft constraints, the results were similar, suggesting that a good initial population significantly impacts solution feasibility.

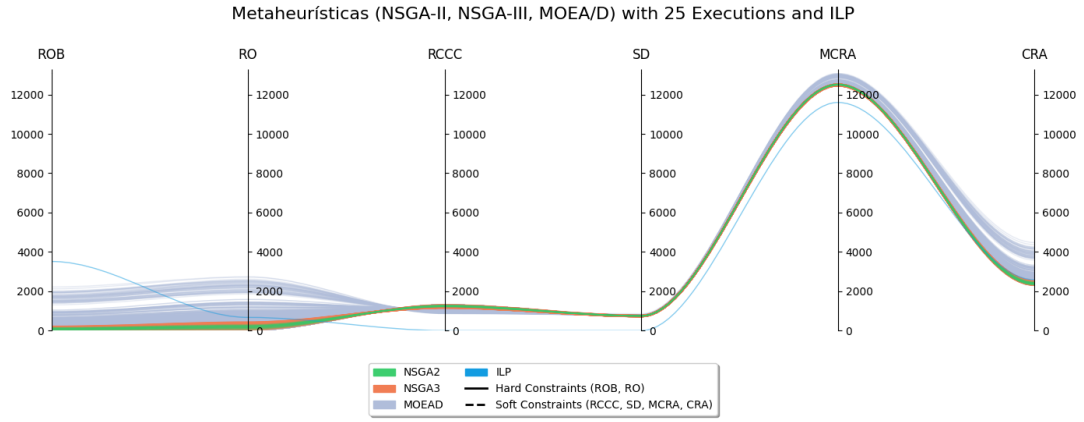

FIGURE 6.6. Reference front plotting for six objectives room allocation UCTP.

6.1.3. Experiment 3 – With Local Search and Constructive Population

This experiment combined the heuristic initial population with local search. The execution time was 15 hours and 48 minutes, with 25 runs per algorithm.

Table 6.3 presents the mean and standard deviation values for the NHV and GSPREAD indicators.

TABLE 6.3. Problem 3 – Mean and standard deviation of NHV and GSPREAD for each algorithm

Algorithm	**NHV (mean ± std)**	**GSPREAD (mean ± std)**
MOEA/D	0.998 ± 0.001	1.031 ± 0.105
NSGA-II	0.800 ± 0.040	0.469 ± 0.033
NSGA-III	0.418 ± 0.137	0.417 ± 0.024

Figure 6.7 shows that MOEA/D again led in NHV (0.998 ± 0.001), repeating the pattern of strong coverage of the PF. NSGAII once again obtained a high value (0.800 ± 0.040), with low variation. NSGAIII showed the lowest NHV (0.418 ± 0.137), with high dispersion and instability.

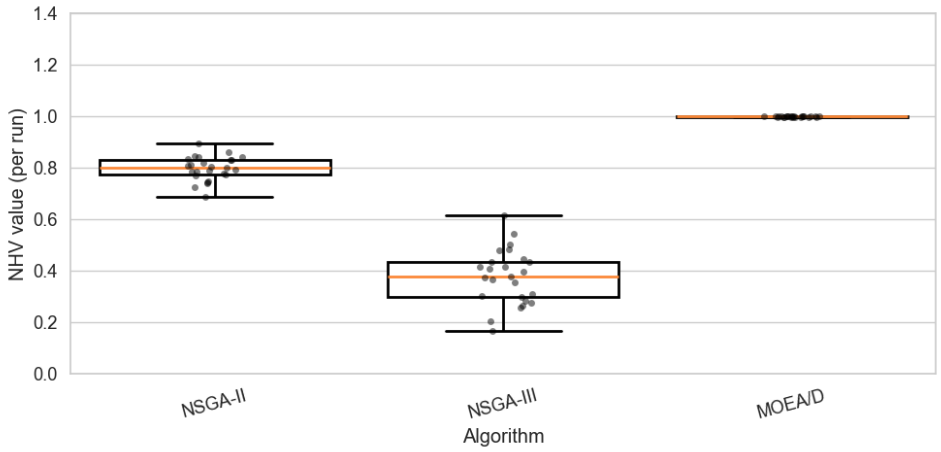

FIGURE 6.7. HV values

Regarding GSPREAD, Figure 6.8 shows that NSGAIII stood out with the best value (0.417 ± 0.024), suggesting well-distributed solutions. NSGAII maintained consistency (0.469 ± 0.033). Meanwhile, MOEA/D again produced the worst result (1.031 ± 0.105), reflecting difficulty in maintaining diversity.

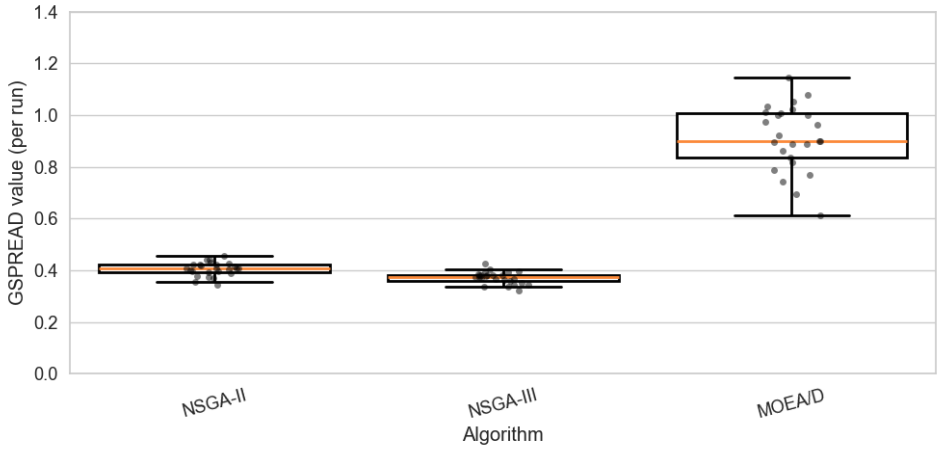

FIGURE 6.8. GSPREAD values

The plot in Figure 6.9 confirms the previous patterns where MOEA/D shows greater variability, mainly in the ROB and RO constraints. Although with some fluctuations, NSGAII and NSGAIII exhibit more cohesive curves in the objective functions.

The combination of constructive population with local search did not yield significant improvements. The high quality of the initial population appears to have reduced the utility of local intensification.

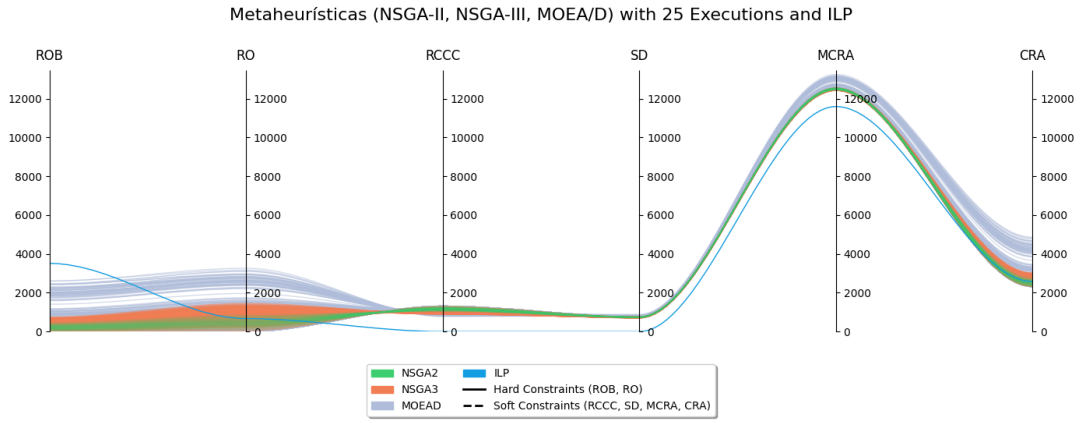

FIGURE 6.9. Reference front plotting for six objectives room allocation UCTP.

6.1.4. Experiment 4 – Without Local Search and Without Constructive Population

This experiment did not use any form of intensification, nor a heuristic initial population. The execution time was 13 hours and 30 minutes, with 25 runs per algorithm. Table 6.4 summarizes the results obtained.

TABLE 6.4. Problem 3 – Mean and standard deviation of NHV and GSPREAD for each algorithm

Algorithm	**NHV (mean ± std)**	**GSPREAD (mean ± std)**
MOEA/D	1.000 ± 0.000	1.026 ± 0.098
NSGA-II	0.946 ± 0.040	0.447 ± 0.032
NSGA-III	0.675 ± 0.219	0.455 ± 0.036

In Figure 6.10, MOEA/D reached the maximum value of NHV (1.000 ± 0.000), indicating that the generated solutions fully cover the dominated region of the PF. NSGAII also showed high performance (0.946 ± 0.040) and low variability, demonstrating good exploration capability and consistency between runs. On the other hand, NSGAIII had the worst performance (0.675 ± 0.219) with high dispersion, including extreme values, highlighting difficulty in reaching regions close to the reference point.

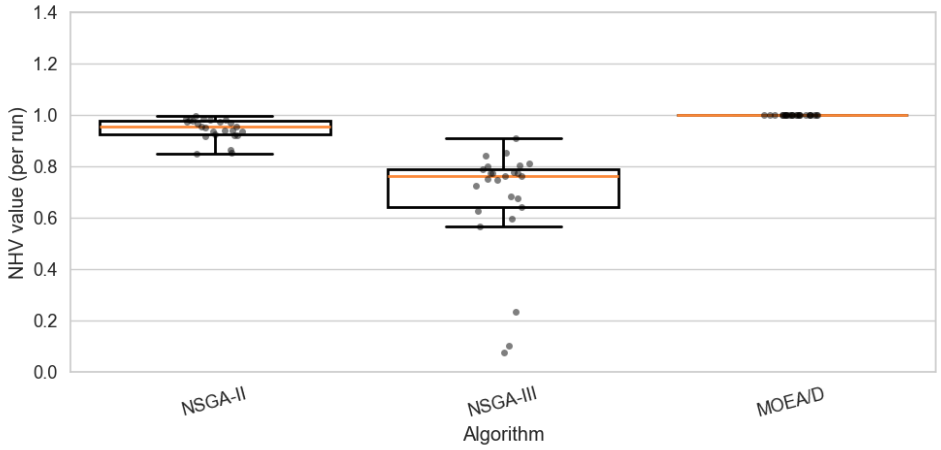

FIGURE 6.10. HV values

Regarding GSPREAD, as shown in Figure 6.11, NSGAII obtained the best distribution (0.447 ± 0.032). NSGAIII had a slightly higher value (0.455 ± 0.036), yet still with stable distribution among runs. On the other hand, MOEA/D, despite the excellent NHV performance, again presented the highest value (1.026 ± 0.098), confirming its pattern of low diversity.

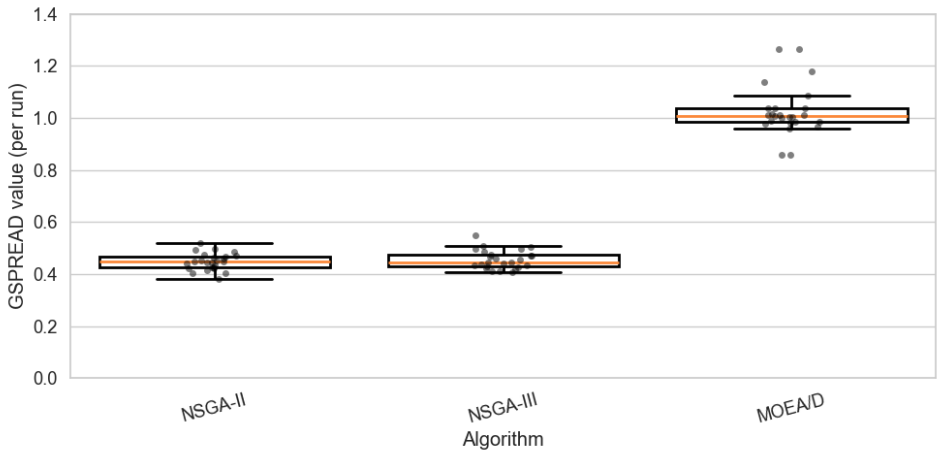

FIGURE 6.11. GSPREAD values

In relation to Figure 6.12, it can be observed that MOEA/D continues to show high dispersion in the hard constraints, while NSGAII and NSGAIII maintain greater stability. Nevertheless, MOEA/D continues to reach solutions with extreme performance, as evidenced by its maximum NHV.

Even without intensification mechanisms, this experiment revealed that MOEA/D is capable of effectively exploring the search space. However, the absence of a well-structured initial population compromised the diversity of the solutions.

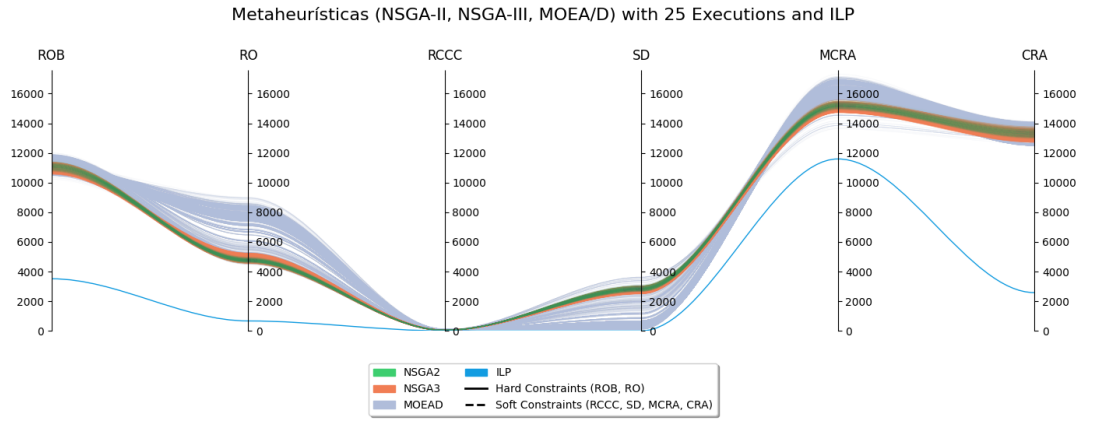

FIGURE 6.12. Reference front plotting for six objectives room allocation UCTP.

6.1.5. Conclusions of the Evolutionary Algorithms Tests

The analysis of the tests revealed clear patterns in performance across different configurations. MOEA/D demonstrated high convergence capability, as expressed by the elevated NHV indicator values. However, this performance was accompanied by limitations in terms of diversity and stability in the hard constraints. This behavior suggests that although it effectively explores the objective space, MOEA/D tends to generate solutions concentrated in specific regions, with high variability between runs.

NSGAII showed good consistency, demonstrating a balance between convergence and diversity. Its performance remained stable even in scenarios without additional mechanisms, indicating strong robustness. NSGAIII, on the other hand, although less effective in terms of convergence, revealed good capability for preserving diversity, as shown by the low GSPREAD values, especially when a heuristic initial population was used.

The introduction of a constructive population had a direct impact on the feasibility and quality of solutions from the first generations, allowing better fulfillment of hard constraints. In contrast, the application of local search had a limited effect, especially when the initial population already guided the evolution toward promising regions of the search space.

Overall, the results indicate that the initialization strategy plays a more decisive role than local intensification in the global effectiveness of the EA in this experiment. These tests reinforce the importance of good heuristics for the initial population in problems with many constraints.

Another relevant point to highlight is that the algorithms developed for MaOO problems—i.e., with four or more objectives—such as MOEA/D and NSGAIII, did not demonstrate significantly superior performance compared to NSGAII, which was originally designed for MOO problems. Despite their advanced strategies based on decomposition and reference directions, the MaOO algorithms did not show a clear advantage in this specific problem, which involves six objectives and a substantially large population of around 26,020 classes. This result suggests that, in certain contexts, classic MOO algorithms may remain competitive, also offering high-quality solutions with lower computational complexity and greater stability across executions.

6.1.6. Results on the `comp02.ctt` Instance from ITC2007

This section presents the results obtained with the NSGAII, NSGAIII, and MOEA/D algorithms on the benchmark instance `comp02.ctt` from the ITC2007 competition. According to Di Gaspero et al. [48], the objective is to validate the ability of the algorithms to generate feasible solutions in accordance with the

criteria defined by the benchmark, i.e., solutions with null or negative penalties in the hard constraints (ROB and RO).

6.1.6.1. *Experiment 1 – Without Constructive Population and With Local Search*

In this experiment, no constructive initial population was used, but local search intensification was applied. The total execution time was approximately 16 minutes.

TABLE 6.5. Objective function values of the best solution (ROB and RO = 0) – Experiment 1

Algorithm	ROB	RO	RCCC	SD	MCRA	CRA
NSGA-II	-30	-53	72	4	35	446
NSGA-III	-28	-43	67	0	55	482
MOEA/D	-79	-130	158	96	150	552

6.1.6.2. *Experiment 2 – Without Local Search, With Constructive Population*

In this case, a constructive heuristic population was used, but without applying local search. The total execution time was approximately 9 minutes and 30 seconds.

TABLE 6.6. Objective function values of the best solution (ROB and RO = 0) – Experiment 2

Algorithm	ROB	RO	RCCC	SD	MCRA	CRA
NSGA-II	0	0	75	0	20	510
NSGA-III	-7	-9	69	0	20	502
MOEA/D	-72	-123	168	345	135	492

6.1.6.3. *Experiment 3 – With Local Search and Constructive Population*

This experiment combined the use of a heuristic constructive population with local search. The total execution time was approximately 11 minutes and 20 seconds.

TABLE 6.7. Objective function values of the best solution (ROB and RO = 0) – Experiment 3

Algorithm	ROB	RO	RCCC	SD	MCRA	CRA
NSGA-II	-16	-34	72	1	35	460
NSGA-III	-32	-52	70	0	60	422
MOEA/D	-69	-132	155	233	135	448

6.1.6.4. *Experiment 4 – Without Local Search and Without Constructive Population*

In this final scenario, neither local search nor constructive initial population was used. The execution time was approximately 9 minutes and 20 seconds.

TABLE 6.8. Objective function values of the best solution (ROB and RO = 0) – Experiment 4

Algorithm	ROB	RO	RCCC	SD	MCRA	CRA
NSGA-II	-13	-11	76	0	50	470
NSGA-III	-19	-48	70	0	10	458
MOEA/D	-76	-135	151	279	115	422

It is observed that all algorithms were able to generate feasible solutions according to the ITC2007 criteria, with null or negative penalties in the hard constraints. NSGAIII presented the best overall results among the pure multi-objective approaches, with lower aggregate values in the soft constraints and good feasibility. On the other hand, MOEA/D, although demonstrating superior robustness in hard constraints, produced solutions with significantly higher penalties in the soft constraints.

These results reinforce the viability of using multi-objective evolutionary algorithms in the context of the UCTP, and validate the compliance of the solutions with the standards defined by the ITC2007 competition.

CHAPTER 7

Conclusion

This dissertation presented a comparative analysis of approaches for solving the timetabling problem at Iscte – Instituto Universitário de Lisboa, with a specific focus on classroom allocation. Currently, the construction and management of timetables is performed manually, since the available software does not meet the institutional requirements. This process is time-consuming and inefficient, highlighting the need for automated, scalable solutions that are more suited to the complexity of Iscte academic context.

The research was conducted based on the initially defined questions, which were answered as follows:

- **Q1: What are the most widely used approaches to solving the UCTP?**
  To answer this question, a systematic literature review was conducted using five scientific repositories aligned with the objectives of the study. The analysis revealed that the most recurring approaches involve metaheuristics, with emphasis on evolutionary algorithms and swarm intelligence methods. Among these, hybrid methods that combine such algorithms with local search techniques proved to be the most effective, representing 35% of the successful approaches.
- **Q2: What are the main advantages, disadvantages, and limitations associated with these approaches?**
  The review analysis showed that metaheuristics approaches are highly effective in generating near-optimal solutions and are highly adaptable to different institutional contexts. However, these approaches require careful parameter tuning and may present high computational costs. On the other hand, exact methods such as Integer Linear Programming offer greater precision and guarantee optimal solutions, but face serious scalability limitations when applied to large problem instances.
- **Q3: What innovations and improvement opportunities have been identified in the literature?**
  Based on the systematic review carried out, the following innovations stand out:
  - The use of hybrid algorithms that combine the exploratory capacity of metaheuristics with local intensification techniques such as Simulated Annealing or Variable Neighborhood Search.
  - The growing use of multi-objective evolutionary algorithms (Evolutionary Algorithms), such as NSGAII, NSGAIII, and MOEA/D, which are particularly effective in solving problems with multiple simultaneous criteria, as well as their integration with exact approaches like Integer Linear Programming.
  - The introduction of clustering techniques and data mining to avoid conflicts and optimize resource management in large volumes of information.

  The main improvement opportunities identified include:
  - The development of more flexible and adaptable models, capable of accommodating both specific institutional constraints and human preferences, in order to ensure more personalized and realistic solutions.

  - The need to improve the evaluation metrics used, focusing on more robust indicators and benchmarks that better represent institutional reality, allowing for more rigorous, fair, and reproducible comparisons between different approaches.
  - The encouragement of the development of algorithms that combine simplicity, computational efficiency, and adaptability, aiming at practical applications in educational environments with varying levels of complexity.
- **Q4: What recent trends have emerged in UCTP research?**
  A growing interest in multi-objective and many-objective evolutionary algorithms has been observed, due to their ability to handle multiple evaluation criteria simultaneously — a central characteristic of UCTP — as well as the increasing use of hybrid approaches.
  There is also a growing emphasis on empirical validation, with the use of standardized benchmarks such as those provided by ITC competitions. Furthermore, the need to develop simpler, more efficient, and adaptable algorithms is highlighted, focusing on practical applicability and scalability in real institutional contexts.
- **Q5: How does the performance of metaheuristic algorithms compare to ILP techniques in solving the UCTP?**
  To answer this question, customized versions of the NSGAII, NSGAIII, and MOEA/D algorithms were implemented using the *jMetal* framework. These versions were enhanced with constructive initialization strategies, local intensification operators, and a stopping criterion based on the Hypervolume (HyperVolume) quality indicator.
  In parallel, an Integer Linear Programming model was developed using the Pyomo library, enabling a direct comparison between evolutionary algorithms and exact methods.
  Using real data from Iscte, the problem was modeled with two hard constraints — room occupation and overlap — and four objective functions to be minimized: room changes between consecutive classes (RCCC), student displacements (SD), room type incompatibility (MCRA), and capacity waste (CRA).
  The results showed that all approaches incorporating a heuristic initial population were able to generate feasible solutions with respect to hard constraints. NSGAIII demonstrated the best overall performance on the soft objectives, maintaining a good balance between convergence and diversity. MOEA/D obtained the best NHV values, demonstrating strong coverage of the Pareto Front (PF), although with lower solution diversity. NSGAII stood out for its consistency and balanced performance.
  As for Integer Linear Programming, it demonstrated high precision. However, it faced difficulties in solving the problem when the input data did not fully meet the constraints defined in the model. For example, if a course unit requires a computer lab with a capacity for 59 students, and the institution only has such rooms with a maximum of 50 seats, the model becomes infeasible.
  Overall, the results confirmed the effectiveness of Multi-Objective Optimization and Many-Objective Optimization algorithms in solving the University Course Timetabling Problem, especially in its multi-objective formulation.
- **Q6: What is the performance of the algorithms when applied to ITC2007 benchmarks?**
  All algorithms were evaluated using the `comp02.ctt` file from the ITC2007 competition. The results obtained demonstrated compliance with the standards established by the competition, as described by Di Gaspero et al. [48], validating the robustness and effectiveness of the developed solutions.

It should be noted that this type of work provided an enriching experience, allowing for the consolidation of practical and theoretical knowledge in a consistent and structured manner, as well as aligned with the practical challenges within this field.

7.1. Achievement of Objectives

The objectives defined at the beginning of this work were successfully achieved, namely:

- A literature review and a systematic review were carried out according to the guidelines of Kitchenham & Charters (2007), allowing the identification of the most commonly used approaches to solve the University Course Timetabling Problem, providing a solid scientific foundation for the study.
- The detailed analysis of the identified methodologies made it possible to outline the main advantages, disadvantages, and trends.
- A benchmarking was developed between evolutionary algorithms and techniques based on Integer Linear Programming, using real data from Iscte – Instituto Universitário de Lisboa, allowing the evaluation of the performance of different approaches.
- Customized versions of the Non-dominated Sorting Genetic Algorithm II, Non-dominated Sorting Genetic Algorithm III, and Multi-Objective Evolutionary Algorithm Based on Decomposition algorithms were implemented, as well as an Integer Linear Programming model using the Python Optimization Modeling Objects library, enabling a comprehensive comparison between exact methods and metaheuristics.
- The algorithms were tested with a recognized international benchmark, the ITC2007, validating the effectiveness of the proposed solutions in a standardized scenario.
- Finally, relevant innovations and opportunities for improvement were identified, such as: development of more flexible models, more accurate evaluation metrics and benchmarks more representative of reality, and encouragement of the creation of simpler, more efficient, and adaptable algorithms.

7.2. Future Work

Despite the success achieved, some points must be considered. The high number of genes per chromosome makes the Integer Linear Programming approach more difficult, since it is enough for there not to be a room that meets all the requirements of a schedule for the model to become unfeasible, thus increasing the effort required in the application of this approach. Regarding the parameter tuning of the Evolutionary Algorithms, it was performed manually, requiring high computational cost.

Additionally, a graphical user interface (GUI) has been developed to facilitate interaction with the proposed approach.

As future studies, the following points are highlighted:

- Exploration of metaheuristics approaches, combining Integer Linear Programming in order to refine the solutions;
- Dynamic parameter tuning during the execution of Evolutionary Algorithms, in order to enhance convergence improvement;
- Application of the solution to the complete timetabling problem, not only to room allocation, that is, integrating all dimensions of the University Course Timetabling Problem;
- Optimization with preferences using methods such as interactive and a priori techniques;
- Improvement of this GUI to support real-time feedback and integration with institutional platforms.

In summary, this research reinforces the potential of the applied approaches, especially when constructive heuristics are applied. The solutions and benchmarkings contribute with relevant advances

to the development of optimization strategies applied to academic planning as well as to the scientific community.

ANNEX A

Planning

TABLE A.1. Tasks Schedule for 2023 and 2024

Tasks	October	November	December	January	February	March	April	June	July	August	September
1											
2											
3											
4											
5											
6											
7											
8											

1. In this task, a detailed literature review will be carried out to support the dissertation topic.
2. This task consists of researching and exploring works related to the dissertation topic and will be conducted continuously throughout the project. The goal is to ensure a comprehensive and ongoing analysis, contributing to a solid and updated foundation.
3. The state of the art will be compiled and analyzed based on the works identified previously, aiming to determine the most suitable approaches to address the problem under study.
4. Customized algorithms will be implemented using the jMetal framework, along with the modeling of the problem using ILP.
5. Benchmarking will be performed with the customized algorithms, evaluating their performance in different scenarios.
6. The algorithms will be tested using the internationally recognized ITC2007 benchmark, validating the effectiveness of the proposed solutions in a standardized setting.
7. Dissertation revision.
8. Dissertation writing.

[This page is intentionally left blank.]

ANNEX B

Participation in the AMiM'24 Event

As part of the validation and dissemination of the work developed in this dissertation, an abstract of this dissertation was submitted and accepted for presentation at the *Autumn Meeting of Industrial Mathematics 2024 (AMiM'24)* [49], held on October 4 and 5, 2024, in Foz do Arelho, Portugal. This event brought together researchers and professionals interested in the application of Mathematics to solve real-world problems in industrial and social contexts.

The paper entitled *"University Timetabling Multicriteria Optimization Benchmarking Using Metaheuristics & ILP"*, was presented on October 5 by the author of this dissertation, with the presence of the advisor Professor Dr. Vítor Basto Fernandes.

During the presentation, the approaches proposed in this work were discussed, with a central focus on solving the room allocation problem in the UCTP. The participation allowed not only the sharing of the obtained results but also contact with researchers in the field, reinforcing the scientific and practical relevance of the study.

Participation in AMiM'24 was an enriching experience, providing an opportunity for external validation of the work. The event fostered the exchange of ideas, academic networking, and the dissemination of the developed research within the scientific community [49].

[This page is intentionally left blank.]

References

[1] D. Salem, "University examination timetable scheduling using constructive heuristic compared to genetic algorithm," *Fayoum University Journal of Engineering*, vol. 6, pp. 39–46, Jan. 2023. DOI: 10.21608/fuje.2022.157195.1018.

[2] W. Hart, C. Laird, J.-P. Watson, and D. Woodruff, *Pyomo - Optimization Modeling in Python*. Jan. 2012, vol. 67, ISBN: 978-1-4614-3225-8. DOI: 10.1007/978-1-4614-3226-5.

[3] Gurobi Optimization, LLC, *Gurobi optimizer reference manual*, https://www.gurobi.com, Gurobi Optimization, LLC, 2025.

[4] D. Brockhoff, M. Emmerich, B. Naujoks, and R. Purshouse, "Introduction to many-criteria optimization and decision analysis," in *Many-Criteria Optimization and Decision Analysis: State-of-the-Art, Present Challenges, and Future Perspectives*, D. Brockhoff, M. Emmerich, B. Naujoks, and R. Purshouse, Eds. Cham: Springer International Publishing, 2023, pp. 3–28, ISBN: 978-3-031-25263-1. DOI: 10.1007/978-3-031-25263-1_1. [Online]. Available: https://doi.org/10.1007/978-3-031-25263-1_1.

[5] M. C. Chen, S. N. Sze, S. L. Goh, N. R. Sabar, and G. Kendall, "A survey of university course timetabling problem: Perspectives, trends and opportunities," *IEEE Access*, vol. 9, pp. 106 515–106 529, 2021. DOI: 10.1109/ACCESS.2021.3100613.

[6] H. Babaei, J. Karimpour, and A. Hadidi, "A survey of approaches for university course timetabling problem," *Computers & Industrial Engineering*, vol. 86, pp. 43–59, 2015, Applications of Computational Intelligence and Fuzzy Logic to Manufacturing and Service Systems, ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2014.11.010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360835214003714.

[7] J. Pandey and A. K. Sharma, "Survey on university timetabling problem," in *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, 2016, pp. 160–164.

[8] A. Bashab, A. O. Ibrahim, I. A. Tarigo Hashem, K. Aggarwal, F. Mukhlif, F. A. Ghaleb, and A. Abdelmaboud, "Optimization techniques in university timetabling problem: Constraints, methodologies, benchmarks, and open issues," *Computers, Materials and Continua*, vol. 74, no. 3, pp. 6461–6484, 2022, ISSN: 1546-2218. DOI: https://doi.org/10.32604/cmc.2023.034051. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S154622182200340X.

[9] J. v. Brocke, A. Hevner, and A. Maedche, "Introduction to design science research," in Sep. 2020, pp. 1–13, ISBN: 978-3-030-46780-7. DOI: 10.1007/978-3-030-46781-4_1.

[10] H. Ma, Y. Zhang, S. Sun, T. Liu, and Y. Shan, "A comprehensive survey on nsga-ii for multi-objective optimization and applications," *Artificial Intelligence Review*, vol. 56, no. 12, pp. 15 217–15 270, Dec. 2023, ISSN: 1573-7462. DOI: 10.1007/s10462-023-10526-z. [Online]. Available: https://doi.org/10.1007/s10462-023-10526-z.

[11] M. Assi, B. Halawi, and R. A. Haraty, "Genetic algorithm analysis using the graph coloring method for solving the university timetable problem," *Procedia Computer Science*, vol. 126, pp. 899–906, 2018, Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 22nd International Conference, KES-2018, Belgrade, Serbia, ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2018.08.024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050918313024.

[12] S. Abdipoor, R. Yaakob, S. L. Goh, and S. Abdullah, "Meta-heuristic approaches for the university course timetabling problem," *Intelligent Systems with Applications*, vol. 19, p. 200 253, 2023, ISSN: 2667-3053. DOI: https://doi.org/10.1016/j.iswa.2023.200253. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2667305323000789.

[13] T. Castillo-Calzadilla, M. Cuesta, C. Quesada, C. Olivares-Rodriguez, A. Macarulla, J. Legarda, and C. Borges, "Is a massive deployment of renewable-based low voltage direct current microgrids feasible? converters, protections, controllers, and social approach," *Energy Reports*, vol. 8, pp. 12 302–12 326, 2022, ISSN: 2352-4847. DOI: https://doi.org/10.1016/j.egyr.2022.09.067. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352484722017917.

[14] E. Rappos, E. Thiémard, S. Robert, and J.-F. Hêche, "A mixed-integer programming approach for solving university course timetabling problems," *Journal of Scheduling*, vol. 25, no. 4, pp. 391–404, Aug. 2022, ISSN: 1099-1425. DOI: 10.1007/s10951-021-00715-5. [Online]. Available: https://doi.org/10.1007/s10951-021-00715-5.

[15] F. Rossi, P. van Beek, and T. Walsh, "Chapter 1 - introduction," in *Handbook of Constraint Programming*, ser. Foundations of Artificial Intelligence, F. Rossi, P. van Beek, and T. Walsh, Eds., vol. 2, Elsevier, 2006, pp. 3–12. DOI: https://doi.org/10.1016/S1574-6526(06)80005-2. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574652606800052.

[16] R. Lewis, "A survey of metaheuristic-based techniques for university timetabling problems," *OR Spectrum*, vol. 30, no. 1, pp. 167–190, 2008, ISSN: 1436-6304. DOI: 10.1007/s00291-007-0097-0. [Online]. Available: https://doi.org/10.1007/s00291-007-0097-0.

[17] J. H. Obit, "Developing novel meta-heuristic, hyper-heuristic and cooperative search for course timetabling problems," PhD thesis, University of Nottingham, 2010. [Online]. Available: https://eprints.nottingham.ac.uk/13581/.

[18] D. D. Martinelli, "Generative machine learning for de novo drug discovery: A systematic review," *Computers in Biology and Medicine*, vol. 145, p. 105 403, 2022, ISSN: 0010-4825. DOI: https://doi.org/10.1016/j.compbiomed.2022.105403. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010482522001950.

[19] P. A. Vikhar, "Evolutionary algorithms: A critical review and its future prospects," in *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*, 2016, pp. 261–265. DOI: 10.1109/ICGTSPICC.2016.7955308.

[20] C. A. Coello Coello, "An introduction to evolutionary algorithms and their applications," in *Advanced Distributed Systems*, F. F. Ramos, V. Larios Rosillo, and H. Unger, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 425–442, ISBN: 978-3-540-31674-9.

[21] F. Busetti, *Genetic algorithms overview*, https://www.researchgate.net/publication/2383112_Genetic_Algorithms_Overview, Accessed on: March 30, 2025, May 2001.

[22] R. B. Kato, V. de Almeida Paiva, and S. Izidoro, "Algoritmos genéticos," in *BIOINFO – Revista Brasileira de Bioinformática*. BIOINFO, Jul. 2021, N/A, ISBN: 9786599275326. DOI: 10.51780/978-6-599-275326-13.

[23] N. AL-Madi and A. T. Khader, "De jong's sphere model test for a social-based genetic algorithm (sbga)," *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 8, no. 3, pp. 212–218, Mar. 2008, Available on ResearchGate. [Online]. Available: https://www.researchgate.net/publication/242468271_De_Jong's_sphere_model_test_for_a_Social-Based_Genetic_Algorithm_SBGA.

[24] F. Faitouri and W. Khalafullah, "Generating maintenance table of power stations using genetic algorithm," *Majallat al-'ulūm wa-al-dirāsāt al-insānīyah*, vol. 1, pp. 1–7, Jan. 2016. DOI: 10.37376/1571-000-010-007.

[25] P. Diaz-Gomez and D. Hougen, "Initial population for genetic algorithms: A metric approach," in *Proceedings of the 8th Mexican International Conference on Artificial Intelligence (MICAI 2007)*, ser. Lecture Notes in Artificial Intelligence, vol. 4827, Springer, 2007, pp. 43–49, ISBN: 978-3-540-76891-6.

[26] O. El Majdoubi, F. Abdoun, N. Rafalia, and O. Abdoun, "Artificial intelligence approach for multi-objective design optimization of composite structures: Parallel genetic immigration," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 9, no. 3, pp. 2863–2870, 2020. DOI: 10.30534/ijatcse/

2020/04932020. [Online]. Available: https://doi.org/10.30534/ijatcse/2020/04932020.

[27] S. Almufti, R. Marqas, and V. Saeed, "Taxonomy of bio-inspired optimization algorithms," *Journal of Advanced Computer Science& Technology*, vol. 8, p. 23, Aug. 2019. DOI: 10.14419/jacst.v8i2.29402.

[28] A. Rezvanian, S. M. Vahidipour, and A. Sadollah, "An overview of ant colony optimization algorithms for dynamic optimization problems," in *Optimization Algorithms*, M. Andriychuk and A. Sadollah, Eds., Rijeka: IntechOpen, 2023, ch. 12. DOI: 10.5772/intechopen.111839. [Online]. Available: https://doi.org/10.5772/intechopen.111839.

[29] D. Wang, D. Tan, and L. Liu, "Particle swarm optimization algorithm: An overview," *Soft Computing*, vol. 22, no. 2, pp. 387–408, 2018, ISSN: 1433-7479. DOI: 10.1007/s00500-016-2474-6. [Online]. Available: https://doi.org/10.1007/s00500-016-2474-6.

[30] T. Dokeroglu, T. Kucukyilmaz, and E.-G. Talbi, "Hyper-heuristics: A survey and taxonomy," *Computers & Industrial Engineering*, vol. 187, p. 109 815, 2024, ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2023.109815. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360835223008392.

[31] C. Coello, D. Veldhuizen, and G. Lamont, "Moea theory and issues," in *Evolutionary Algorithms for Solving Multi-Objective Problems: Second Edition*. Boston, MA: Springer US, 2007, pp. 283–337, ISBN: 978-0-387-36797-2. DOI: 10.1007/978-0-387-36797-2_6. [Online]. Available: https://doi.org/10.1007/978-0-387-36797-2_6.

[32] Q. Cai, M. Lijia, and M. Gong, "A survey on network community detection based on evolutionary computation," *International Journal of Bio-Inspired Computation*, vol. 8, Jan. 2014. DOI: 10.1504/IJBIC.2016.076329.

[33] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 577–601, 2014. DOI: 10.1109/TEVC.2013.2281535.

[34] Q. Zhang and H. Li, "Moea/d: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007. DOI: 10.1109/TEVC.2007.892759.

[35] J. Bader and E. Zitzler, "Hype: An algorithm for fast hypervolume-based many-objective optimization," *Evolutionary computation*, vol. 19, pp. 45–76, Mar. 2011. DOI: 10.1162/EVCO_a_00009.

[36] K. Miettinen, P. Eskelinen, F. Ruiz, and M. Luque, "Nautilus method: An interactive technique in multiobjective optimization based on the nadir point," *European Journal of Operational Research*, vol. 206, no. 2, pp. 426–434, 2010. DOI: 10.1016/

j.ejor.2010.02.041. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221710001785.

[37] P. Eskelinen, K. Miettinen, K. Klamroth, and J. Hakanen, "Pareto navigator for interactive nonlinear multiobjective optimization," *OR Spectrum*, vol. 32, no. 2, pp. 211–227, 2010. DOI: 10.1007/s00291-008-0151-6. [Online]. Available: https://link.springer.com/article/10.1007/s00291-008-0151-6.

[38] V. Basto-Fernandes, D. Salvador, I. Yevseyeva, and M. Emmerich, "Many-criteria optimisation and decision analysis ontology and knowledge management," in *Many-Criteria Optimization and Decision Analysis: State-of-the-Art, Present Challenges, and Future Perspectives*, ser. Natural Computing Series, D. Brockhoff, M. Emmerich, B. Naujoks, and R. Purshouse, Eds., Cham: Springer International Publishing, 2023, ch. 13, pp. 299–316, ISBN: 978-3-031-25263-1. DOI: 10.1007/978-3-031-25263-1_13. [Online]. Available: https://doi.org/10.1007/978-3-031-25263-1_13.

[39] M. Asadzadeh and B. Tolson, "Hybrid pareto archived dynamically dimensioned search for multi-objective combinatorial optimization: Application to water distribution network design," *Journal of hydroinformatics*, vol. 14, pp. 192–205, Jan. 2012. DOI: 10.2166/hydro.2011.098.

[40] M. Á. Domínguez-Ríos, F. Chicano, and E. Alba, "Effective anytime algorithm for multiobjective combinatorial optimization problems," *Information Sciences*, vol. 565, pp. 210–228, 2021, ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2021.02.074. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025521002310.

[41] J. J. Durillo and A. J. Nebro, "Jmetal: A java framework for multi-objective optimization," *Advances in Engineering Software*, vol. 42, no. 10, pp. 760–771, 2011, ISSN: 0965-9978. DOI: https://doi.org/10.1016/j.advengsoft.2011.05.014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0965997811001219.

[42] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002. DOI: 10.1109/4235.996017.

[43] H. Ma, Y. Zhang, S. Sun, T. Liu, and Y. Shan, "A comprehensive survey on nsga-ii for multi-objective optimization and applications," *Artificial Intelligence Review*, vol. 56, no. 12, pp. 15 217–15 270, 2023, ISSN: 1573-7462. DOI: 10.1007/s10462-023-10526-z. [Online]. Available: https://doi.org/10.1007/s10462-023-10526-z.

[44] Y. Vesikar, K. Deb, and J. Blank, "Reference point based nsga-iii for preferred solutions," in *Proceedings of the 2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2018, pp. 1587–1594. DOI: 10.1109/SSCI.2018.8628819.

[45] Pyomo Project, *About pyomo*, Acedido em 10 de maio de 2025, 2025. [Online]. Available: https://www.pyomo.org/about.

[46] GNU Project, *Introduction to glpk*, Acedido em 10 de maio de 2025, 2025. [Online]. Available: https://www.gnu.org/software/glpk/.

[47] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," vol. 2, Jan. 2007.

[48] L. Di Gaspero, B. Mccollum, and A. Schaerf, "The second international timetabling competition (itc-2007): Curriculum-based course timetabling (track 3)," Jan. 2007. [Online]. Available: https://www.researchgate.net/publication/215777351_The_Second_International_Timetabling_Competition_ITC-2007_Curriculum-based_Course_Timetabling_Track_3.

[49] AMiM'24 Scientific Committee, *Autumn meeting of industrial mathematics 2024 (amim'24)*, Foz do Arelho, Portugal. Acesso em Maio 2025, Oct. 2024. [Online]. Available: https://www.spm.pt/PT-MATHS-IN/amim24.