National University of Science and Technology POLITEHNICA Bucharest
Doctoral School of Industrial Engineering and Robotics

DOCTORAL THESIS

OPTIMIZATION OF LEARNING ACTIVITY PLANNING PROCESSES THROUGH ENGINEERING-SPECIFIC ALGORITHMS AND DEVELOPMENT OF AN INTEGRATED INFORMATION PLATFORM

PhD Student,

Ing. CÎTU (TUNSOIU) M. Daniela

PhD Supervisor,

Prof. univ. dr. ing. DOICIN Cristian-Vasile

Bucharest
– 2026–

TABLE OF CONTENTS

	LIST OF ABBREVIATIONS	**VII**
	INTRODUCTION	**XI**
	DEFINITIONS AND TERMINOLOGY	**XIII**
	STRUCTURE AND CONTENT OF THE THESIS	**XXI**
	JUSTIFICATION OF THE TOPIC	**XXIV**
	PART I. CURRENT STATE OF DOCUMENT MANAGEMENT IN A UNIVERSITY	
CHAPTER 1.	**HIGHER EDUCATION SYSTEM AND DOCUMENT WORKFLOWS**	**1**
1.1.	Dynamics of Higher Education: Evolution of the Number of Students	2
1.2.	Organization of the Higher Education System	4
1.3.	Internal Structure of Higher Education Institutions	5
1.4.	University Cycles	6
1.5.	Classification of Documents Generated and Managed within a University/Faculty	7
	1.5.1. Administrative and Academic Organization Documents	8
	1.5.2. Documents Related to Academic Evaluation and Performance	8
	1.5.3 Documents for Organizing Student and Academic Staff Schedules	9
1.6.	Documents Required for Teaching Activity Scheduling	10
	1.6.1 Curriculum	10
	1.6.2 Study Groups	11
	1.6.3 Staff Establishment Plan	12
	1.6.4 Teaching Assignment Order	13
1.7.	Precedence Relationships between Documents Involved in Teaching Activity Scheduling	13
CHAPTER 2.	**STUDY OF EXISTING ONLINE APPLICATIONS FOR UNIVERSITY DOCUMENT MANAGEMENT**	**17**
2.1.	University Document Management	17
2.2.	Types of Applications Used in University Document Management	18
2.3.	Applications for Administrative Activity Management	19
	2.3.1 Moodle E-learning Platform	19
	2.3.2 Oracle PeopleSoft Campus Solutions	20
	2.3.3 University Management System (UMS)	21
	2.3.4 MyGJU Admin Platform	22
2.4.	Applications for Teaching Activity Scheduling	23
	2.4.1 FET (Free Timetabling Software)	24
	2.4.2 aSc TimeTables	25
	2.4.3 Mimosa	26
	2.4.4 Untis Express	26
	2.4.5 TimeTabler	28
	2.4.6 M-Sched	29
	2.4.7 Bullet Time Tabler Education (BTTE)	29

CHAPTER 3.	**ALGORITHMS USED FOR DOCUMENT MANAGEMENT IN A UNIVERSITY**	. . . . .	**30**
3.1.	Search Algorithms	. . . . .	32
	3.1.1 TF-IDF Algorithm	. . . . .	32
	3.1.2 BM 25 Algorithm	. . . . .	33
3.2.	Clustering Algorithms	. . . . .	35
	3.2.1 Hierarchical Clustering Algorithms	. . . . .	36
	3.2.2 Centroid-Based Clustering Algorithms	. . . . .	38
3.3	Classification Algorithms	. . . . .	41
	3.3.1 Decision Tree Algorithm	. . . . .	41
	3.3.2 Support Vector Machine (SVM)	. . . . .	43
	3.3.3 Naïve Bayes	. . . . .	44
3.4	Encryption and Security Algorithms	. . . . .	45
CHAPTER 4.	**ALGORITHMS USED IN TEACHING ACTIVITY MANAGEMENT**	. . . . .	**47**
4.1	Backtracking Algorithm	. . . . .	47
	4.1.1 Operating Principle	. . . . .	47
	4.1.2 Implementation and Pseudocode	. . . . .	48
	4.1.3 Complexity, Advantages, and Disadvantages	. . . . .	49
4.2	Genetic Algorithm	. . . . .	50
	4.2.1 Implementation Principle	. . . . .	50
	4.2.2 Implementation and Optimizations	. . . . .	51
4.3	Machine Learning for Teaching Activity Scheduling	. . . . .	55
	4.3.1 Definition of Extraction Processes	. . . . .	57
	4.3.2 Data Extraction	. . . . .	58
	4.3.3 Data Cleaning and Processing	. . . . .	59
	4.3.4 Data Modeling	. . . . .	60
	4.3.5 Data Analysis	. . . . .	61
	4.3.6 Feature engineering	. . . . .	62
	4.3.7 Study of ML Algorithms	. . . . .	63
	4.3.8 Selection of the ML Algorithm	. . . . .	65
PART II.	**ORIGINAL CONTRIBUTIONS REGARDING MODELING, OPTIMIZATION, AND DEVELOPMENT OF A DEDICATED SOFTWARE PLATFORM**		
CHAPTER 5.	**OBJECTIVES AND RESEARCH METHODOLOGY**	. . . . .	**67**
5.1.	Research Purpose and Objectives	. . . . .	67
5.2.	Research Methodology	. . . . .	68
CHAPTER 6.	**ANALYSIS OF EXISTING SOLUTIONS AND TEACHING ACTIVITY SCHEDULING METHODS**	. . . . .	**71**
6.1.	Comparative Analysis of Existing Teaching Activity Management Applications	. . . . .	71
6.2.	Analysis of Algorithms Used in Existing Research	. . . . .	73
6.3.	Problems in Teaching Activity Scheduling	. . . . .	74

INTRODUCTION

In the current context of education, the efficient management of resources and time has become a critical challenge, given the large volume of documents that support the educational process. Modern education systems, increasingly oriented toward digitalization, require advanced solutions for planning teaching activities that address both institutional needs and the preferences of academic staff.

This doctoral thesis aims to support the digitalization of management processes in higher education institutions by investigating mathematical models that underpin algorithms for the automatic scheduling of teaching activities, thereby significantly reducing the time required for their development.

In recent years, the number of students enrolled in Romanian higher education has experienced significant fluctuations. According to data from the National Institute of Statistics (INS), "in the 2021–2022 academic year, approximately 415,000 students were enrolled in undergraduate programs in Romania, while in the 2022–2023 academic year the number of undergraduate students decreased by 4%" (Cononenco, 2024). Year-to-year fluctuations in student numbers pose challenges in terms of planning and the efficient allocation of resources.

STRUCTURE AND CONTENT OF THE DOCTORAL THESIS

The doctoral thesis is structured into 12 chapters and is supported by 183 figures, 19 tables, and 6 appendices.

Part I: Current State of Document Management in a University

The first chapter, ***Higher Education System and Document Workflows***, provides an overview of the organization of the Romanian higher education system, including the internal structure of a university in the context of integration into the European Higher Education Area (EHEA) and the implementation of the Bologna Process.

The second chapter, ***Study of Existing Online Applications for Document and Teaching Activity Management in Universities***, highlights the concept of digitalization of document management processes within a university and presents a classification of the types of applications currently in use. Additionally, four applications for activity management and seven applications for the automatic generation of teaching activity schedules are analyzed in detail.

The third chapter, ***Algorithms Used for Document Management in a University***, presents in detail four types of algorithms—search, clustering, classification, and encryption—which provide solutions for implementing various problems within document workflows. Each category includes a detailed description of at least two of the most commonly used algorithms for problem solving.

The fourth chapter, ***Algorithms Used in the Management of Teaching Activities***, presents three complex algorithms for solving automatic teaching activity scheduling problems: backtracking, genetic algorithms, and an AI-based algorithm (Random Forest Classifier). For each

algorithm, the operating principle, implementation method, and a pseudocode description are analyzed to facilitate a clearer understanding of the algorithmic stages.

Part II: Original Contributions on Modeling, Optimization, and the Development of a Dedicated Software Platform

The fifth chapter, ***Research Objectives and Methodology***, presents the general objective of this work, which consists of optimizing teaching activity planning processes and developing an integrated software platform.

The sixth chapter, ***Analysis of Existing Solutions and Teaching Activity Planning Methods***, includes two detailed comparative analyses of existing applications and algorithms used in current research. Based on the observations derived from these analyses, the problems related to teaching activity scheduling were defined and the challenges of generating an optimal schedule were identified.

The seventh chapter, ***User Requirements Analysis and Definition of Application Development Requirements***, analyzes user needs, based on which the hardware and software constraints that must be met by the application were defined. A thorough understanding of user requirements forms the foundation for the successful development of the integrated software platform for teaching activity planning.

The eighth chapter, ***Mathematical Model for Teaching Activity Management***, presents two mathematical models that support the algorithms selected for development.

The ninth chapter, ***Application and Database Design***, analyzes the architectural model that is most suitable for the developed application and describes its implementation using containers developed within the Docker Compose framework.

The tenth chapter, ***Analysis of Automatic Generation Algorithms***, presents the logical stages and the structure of the implemented code for the three addressed algorithms: backtracking, genetic, and Random Forest. Each algorithm exhibits specific advantages and disadvantages in the development process.

The eleventh chapter, ***Graphical User Interface Design***, presents each page developed within the application.

The final chapter, Chapter 12, ***Application Analysis and Testing***, presents the results obtained following the implementation of the algorithms.

CHAPTER 1

HIGHER EDUCATION SYSTEM AND DOCUMENT WORKFLOWS

Academic education “has long-term effects on the development of human resources and plays an essential role in the dissemination of knowledge” (Socoliuc, Dolghi & Ciubotariu, 2023).

“Over time, higher education has undergone numerous reforms, either to meet social and economic needs or to enhance the quality of higher education and facilitate adaptation to changes in society. These reforms have focused on various areas, such as course content, teaching techniques, accessibility of education, the quality level of institutions, and adaptability to labor market requirements” (Socoliuc, 2023).

In the field of tertiary education, "significant changes have occurred as a result of the internationalization of the labor market, student and academic staff mobility, increased competitiveness for obtaining employment aligned with one's qualifications, and the guarantee of the free development of young people's personalities" (Bleikli, 2000; Solbrekke & Karseth, 2006).

1.1 Dynamics of Higher Education: Evolution of Student Numbers

Technological advancements have led to an increase in the number of students seeking to pursue higher education across Europe. According to a study conducted by Eurostat in 2022, approximately 18.8 million students are enrolled in higher education in Europe, across the three cycles of tertiary education: bachelor's (59%), master's (36%), and doctoral (5%) programs (Tertiary Education Statistics, 2024).

According to Eurostat statistics for the period 2015–2022, "a substantial increase in the number of students (Fig. 1.2) enrolled in higher education has been recorded in recent years" (except for 2022, when a slight decrease was observed), which has led to an increasing need to streamline and optimize universities' administrative processes.

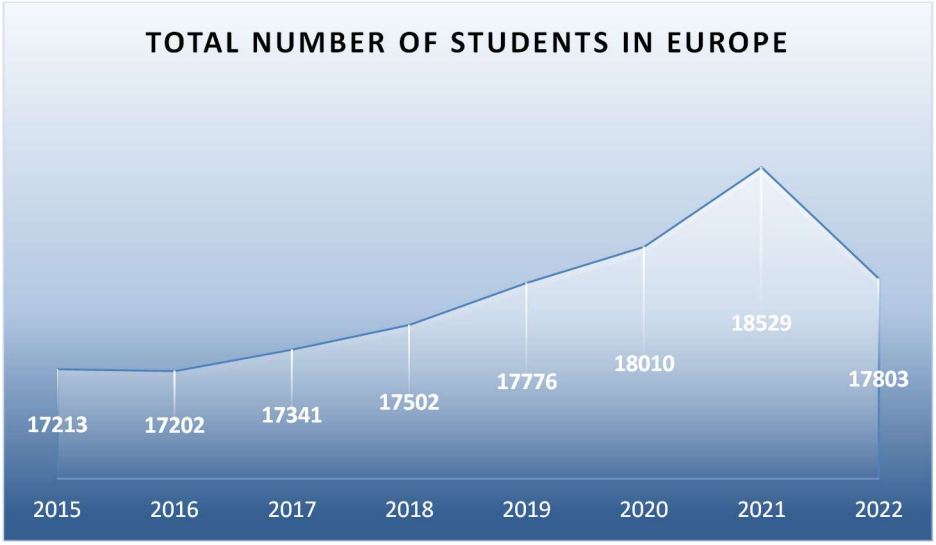


Fig. 1. 1 Number of students in 2015-2022 in UE, milions
(Tertiary education statistics, 2022)

1.2 Organization of the Higher Education System

"Higher education is organized exclusively by education providers and higher education institutions that hold provisional operating authorization or accreditation, in accordance with legal provisions," as stated in Article 7, Chapter II of the Higher Education Law no. 199/2023. Within the Romanian education system, there are 89 higher education institutions, both public and private, located in 23 urban localities:

- ✓ "52 accredited public higher education institutions (46 civilian and 7 military)";
- ✓ "37 private higher education institutions (34 accredited and 3 provisionally authorized)" (Annual Report on the National Meta-Ranking, 2023).

1.6 Documents Required for Teaching Activity Scheduling

The development of a teaching activity schedule within a higher education institution in Romania requires a set of documents that provide the necessary information. The main documents from which relevant data are extracted for the implementation of teaching activity scheduling are:

1. Curriculum (Study Plan)
2. Study Groups
3. Staff Establishment (Staffing Plan)
4. Classroom Availability
5. Course Syllabi



Fig.1. 2 Documents required for academic scheduling

1.7 Precedence Relationships among Documents Involved in Teaching Activity Scheduling

In any public institution, there are operational procedures governing the logical flow of documents required for carrying out the teaching process. Each faculty manages multiple study programs internally. Therefore, for the scheduling of teaching activities within a faculty (covering all years of study and all academic programs), the precedence relationships among the required documents are of critical importance.

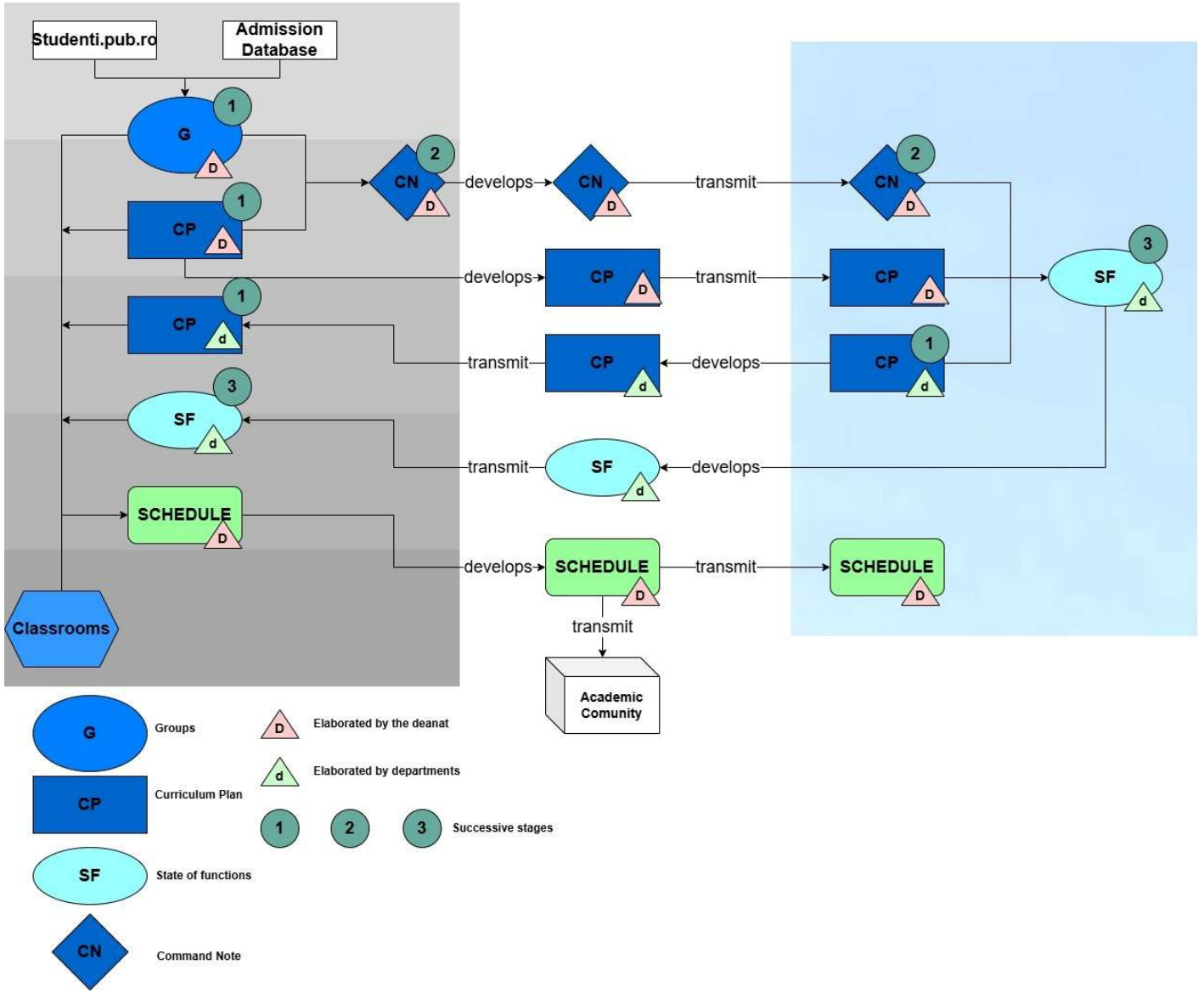


Fig.1. 3 Document workflow for the development of teaching activity scheduling (Tunsoiu et al, 2025)

CHAPTER 2
STUDY OF EXISTING ONLINE APPLICATIONS FOR DOCUMENT MANAGEMENT IN UNIVERSITIES

2.1 Introduction to University Document Management

The concept of the document has undergone continuous changes, primarily driven by the evolution of information technologies. “Traditional document management methods, characterized by the handling of paper-based files, are not suitable for the dynamic and digitalized environment in which modern education is developing” (Imeri, Memeti, Sadiki & Idrizi, 2024).

The global market for document management systems was valued at over USD 7.52 billion in 2020, and estimates indicate that it will reach USD 14.82 billion by 2029 (Mondor, 2022). The Association for Intelligent Information Management (AIIM) states that document management, also referred to as a Document Management System (DMS), is “the process of storing, organizing, and tracking electronic documents in a controlled and structured manner, based on predefined workflows” (Imeri, 2024).

2.2 Types of Applications Used in University Document Management

The utilization of information from various existing document management systems leads to the development of software applications intended to replace traditional work procedures. The available solutions are illustrated in Fig. 2.1.

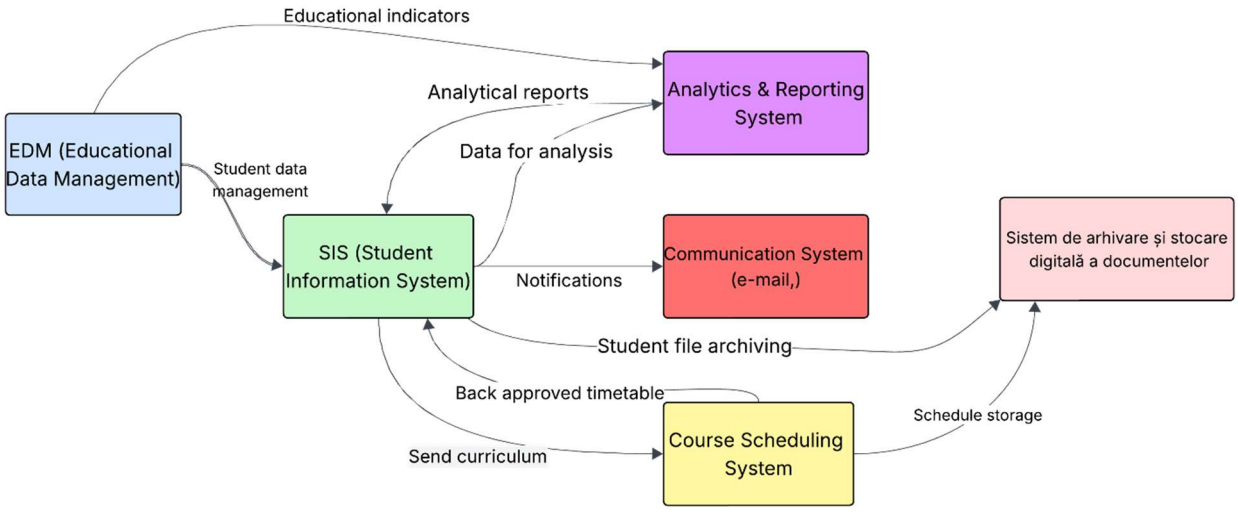


Fig. 2. 1 Types of applications used

CHAPTER 3
ALGORITHMS USED FOR DOCUMENT MANAGEMENT IN A UNIVERSITY

In the field of education, key objectives include the transition to electronic document management, improving the quality of educational processes and academic staff performance, and developing new approaches to secure data storage and rapid data processing. Addressing these challenges requires "the creation and implementation of an electronic document management system, taking into account the specific characteristics of the domain" (Krasnyansk, Ostroukh, Karpushkin & Obukhov, 2016).

Effective document management must consider aspects such as document digitalization, storage and organization, search functionality, data security, archiving, and process automation (Struyf, Hubert & Rousseeuw, 1997).

Based on these requirements, various algorithms have been developed to facilitate the efficient implementation of software applications:

1. Search algorithms, such as TF-IDF (Term Frequency–Inverse Document Frequency) and BM25 (Best Matching 25), which assign weights to keywords in documents based on their frequency and importance;
2. Clustering algorithms, such as K-Means and K-Medoids, which can be used to group similar documents into clusters, thereby facilitating the organization and retrieval of related documents;
3. Classification algorithms, including decision trees, Support Vector Machines (SVM), and Naïve Bayes, which can be used to classify documents into categories or topics based on their content;
4. Encryption and security algorithms, such as AES (Advanced Encryption Standard), which can be used to ensure the security of sensitive document data by allowing access only to authorized users.

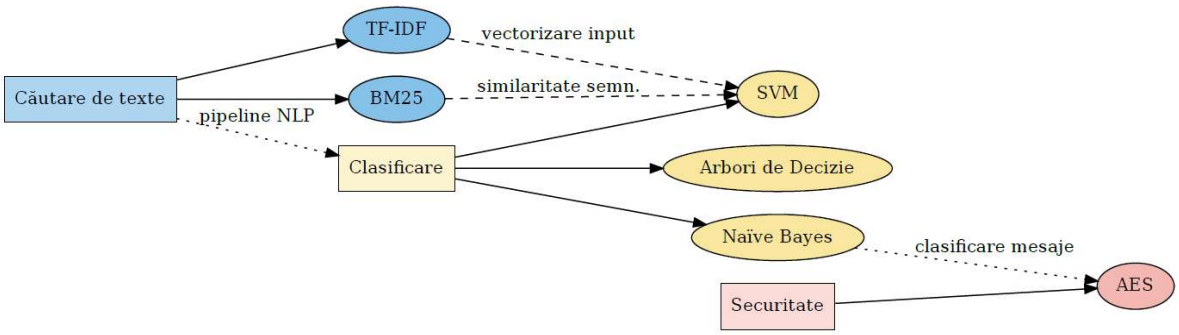


Fig. 3. 1 Order of use of the implemented algorithms

CHAPTER 4
ALGORITMI UTILIZAȚI ÎN MANAGEMENTUL ACTIVITĂȚILOR DIDACTICE

4.1 Backtracking Algorithm

Backtracking involves finding a solution incrementally by attempting different options and discarding them if they lead to an invalid solution. It is commonly used in situations where multiple possibilities must be explored to solve a problem, such as finding a path in a maze, solving puzzles like Sudoku, or generating a schedule of activities. When an invalid solution is reached, the algorithm returns to the previous decision point and explores an alternative path until a valid solution is found or all possibilities are exhausted (WsCube Tech, 2024).

4.2. Genetic Algorithm

The Genetic Algorithm (GA) is an optimization algorithm inspired by the principles of natural selection. It is a population-based search algorithm that employs the concept of survival of the fittest (Katoch, Chauhan & Kumar, 2021).

4.2.2 Implementation and optimizations

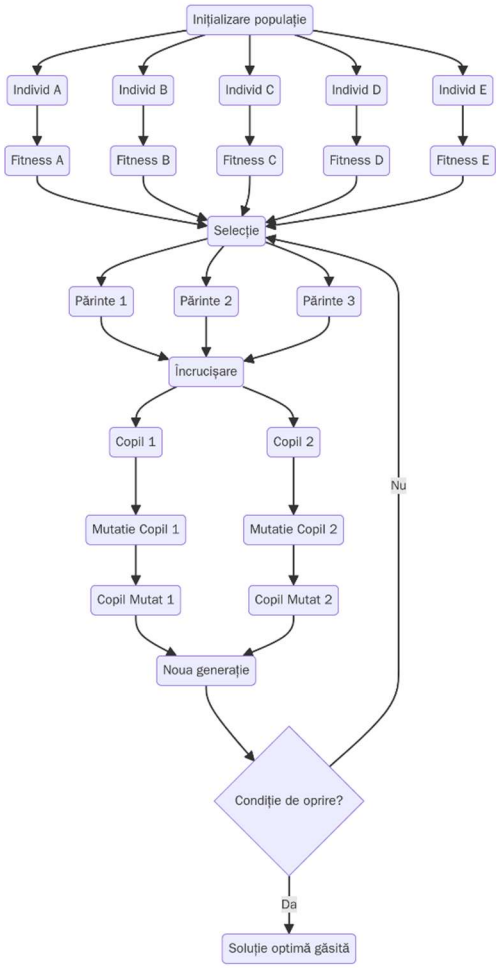


Fig. 4. 1 Stages of GA implementation

4.3 Machine Learning for Teaching Activity Scheduling

Machine learning represents a fundamental paradigm of artificial intelligence, referring to a system's ability to extract knowledge and patterns from historical data to improve performance on future tasks. The essence of this field lies in the development and application of methods capable of generalizing based on prior experience, without requiring explicit human intervention (Burkart & Huber, 2021). Machine learning is a field of "artificial intelligence that enables algorithms to learn from initial data and make predictions without being explicitly programmed" (Mahesh, 2020). ML algorithms are divided into several categories, depending on the type of learning and the nature of the problem they address (Mahesh, 2020).

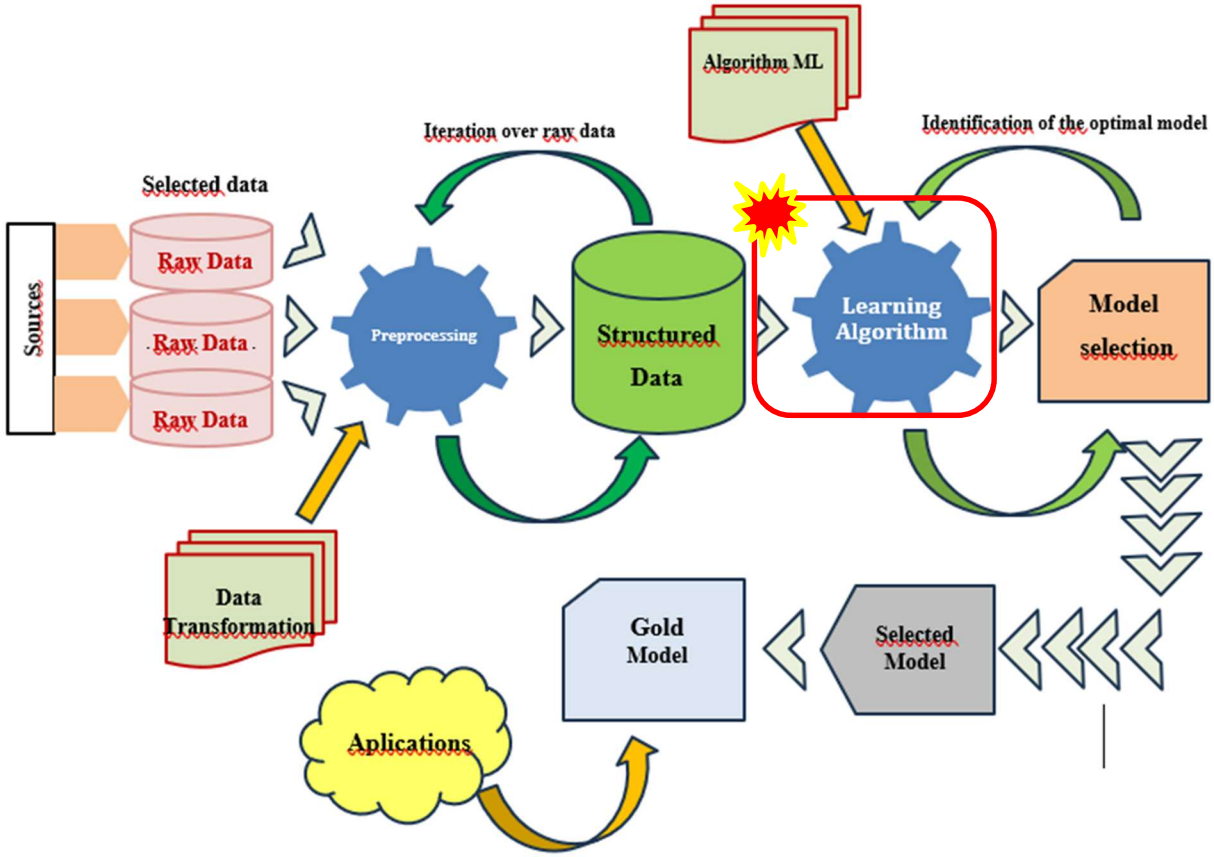


Fig. 4. 2 The machine learning process (adaptat după Ojha, 2019)

CHAPTER 5
RESEARCH OBJECTIVES AND METHODOLOGY

The study on modeling and optimizing the teaching activity management process, as well as the development of an application for document management and academic activity scheduling within a higher education institution, is primarily based on the research, development, and testing of algorithms for generating an optimal solution that satisfies both hard and soft constraints. This approach supports the staff involved in the manual, semester-based preparation of teaching activity schedules.

In Part II of the doctoral thesis, Original Contributions on Modeling, Optimization, and the Development of a Dedicated Software Platform, the objectives proposed in this chapter are validated.

CHAPTER 6
ANALYSIS OF EXISTING SOLUTIONS AND TEACHING ACTIVITY SCHEDULING METHODS

The automatic generation of teaching activity schedules represents an important component in the modernization of educational management, considering the continuous growth in data volume and the increasing complexity of constraints. This chapter provides both an applied and theoretical synthesis of existing solutions, the algorithms used, and the difficulties associated with the problem, with the aim of substantiating subsequent research directions.

Additionally, based on the three analyzed algorithms, a multicriteria analysis is performed, allowing the ranking of alternatives according to the results obtained through the weighting of the established criteria.

6. 1. Comparative Analysis of Existing Applications for Teaching Activity Management

This section presents a comparative analysis of existing applications in order to identify how well each product meets user needs. For the six applications presented in Chapter 2, the following characteristics are analyzed: graphical user interface, customization, automatic generation, resource management, reporting, export functionality, support and documentation, and budget, as presented in Table 6.1.

Following the analysis of 17 scientific articles that investigated various approaches to identifying the most effective solutions for the automatic scheduling of teaching activities, it can be concluded that there is no universal algorithm capable of fully satisfying the requirements of a faculty with a specific size, diverse types of constraints, and curricula that vary from year to year. It was observed that different combinations of algorithms (e.g., HGA, ILS + HC) can achieve better performance compared to individual methods.

For large-scale problems, the use of probabilistic metaheuristics (SA, TS, MOSA) is recommended; however, these methods require careful tuning of generation parameters. For smaller problems, deterministic methods (HC, MIP) are recommended, as they provide fast and efficient results.

All algorithms used in the analyzed studies are optimization algorithms, classified into six categories: local search, metaheuristics, multi-objective optimization, evolutionary algorithms, exact methods, and hybrid approaches (Fig. 2.22).

6.3 Problems in Teaching Activity Scheduling

6.3.1 Problem Definition and Constraints

The teaching activity scheduling problem consists of organizing activities (e.g., lectures, seminars) in such a way as to optimize the use of available resources while complying with various constraints. These constraints are divided into two categories: hard constraints and soft constraints (Burke & Petrovic, 2002).

A valid teaching activity schedule must satisfy the following **hard constraints**:

- ❖ The activities of a course must be scheduled and assigned to distinct time slots;
- ❖ All courses within the same curriculum taught by the same instructor must be scheduled in different time periods (Abdullah, Turabieh, McCollum & McMullan, 2012);
- ❖ An instructor cannot teach two courses simultaneously (Chen, 2021).

Requirements that may be relaxed within a valid teaching activity schedule are referred to as soft constraints and may include:

- ✓ Even distribution of courses for both instructors and students throughout the week;
- ✓ Avoiding the scheduling of courses at unsuitable or non-preferred times, such as early morning or late evening hours (Chen, 2021);
- ✓ Avoiding the allocation of classes during lunch breaks (Chen, 2021).

6.5 Multicriteria Analysis

"Multicriteria Analysis (MCA) describes any structured approach used to determine overall preferences among multiple alternative options, where these options contribute to the achievement of a set of different objectives" (Roman, 2012). This method is used to rank available options based on a predefined set of criteria and weights that reflect their relative importance. "MCA allows the simultaneous consideration of multiple criteria in complex situations" (Roman, 2012). For the MCA of the selected algorithms, five criteria are used in conducting the analysis.

6.5.3 Performance Matrix

Within the performance matrix, presented in Table 6.5, all criterion values for each analyzed alternative are included. Each row of the performance matrix corresponds to an alternative, while the columns represent the performance of each alternative with respect to each criterion. Table X presents the performance matrix used for the multicriteria analysis of the three analyzed algorithms.

Tabel 6. 1 Performance Matrix

Algorithm	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**
Backtracking	9.0	95	1	2	4	**10.56**
Genetic Algorithm	3.5	85	4	4	3	**24.29**
Random Forest	5.0	75	5	3	2	**15.00**

6.5.4 Normalization

Normalization is the method by which all criterion values presented in Table 6.7, expressed in different units of measurement, are transformed onto a common scale (Triantaphyllou, 2000). In multicriteria analysis, normalization is an essential step that enables the comparison of alternatives expressed in different units. It transforms the raw criterion values into a common scale, typically [0, 1], while preserving proportional relationships (Triantaphyllou, 2000).

The Min–Max method is used for value normalization. This method provides a specific formula for each type of normalization (maximization or minimization). Accordingly, minimization is applied to criteria C1 and C5, while maximization is applied to criteria C2, C3, C4, and C6.

Tabel 6. 2 Normalization

Algorithm	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**
Backtracking	0	1	0	0	0	0
Genetic Algorithm	0,15	0,5	0,109	1	0,5	1
Random Forest	0,727	0	1	0,5	1	0,312

6.5.5 Ranking of Alternatives

In the first stage of ranking the alternatives, the normalized value of each criterion is multiplied by its corresponding weight. Subsequently, the ranking is obtained by summing the weighted values resulting from the first stage. For criterion C2, the assigned weight is 0.25, according to Table X, and the values used for ranking are as follows:

$I_{21} = 1 * 0{,}25 = 0{,}25 \qquad I_{22} = 0{,}5 * 0{,}25 = 0{,}125 \qquad I_{23} = 0 * 0{,}25 = 0 \qquad (6.4)$

In Table 6.8 presents the values obtained as a result of the performed calculations:

Tabel 6. 3 Ranking of alternatives

Algorithm	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**	**Total**
Backtracking	0	0,25	0	0	0	0	0,25
Genetic Algorithm	0,15	0,125	0,1125	0,15	0,05	0,2	**0,7875**
Random Forest	0,109	0	0,15	0,075	0,1	0,0624	0,4964

Following the multicriteria analysis performed using the Min–Max method, the Genetic Algorithm achieved the highest overall score (0.7875), confirming its balance between efficiency, quality, and adaptability. Although the Backtracking algorithm provides high-quality solutions, its execution time and scalability limitations make it less suitable for practical applications. In contrast, the Random Forest algorithm obtained an intermediate score, standing out particularly in terms of scalability and ease of implementation.

CHAPTER 7
USER REQUIREMENTS ANALYSIS AND DEFINITION OF APPLICATION DEVELOPMENT REQUIREMENTS

The development of a customized application involves configuring operations according to the needs of clients or users. Integrating user preferences and requirements into the architectural design of the application is essential, as the final product is evaluated based on ease of use and the efficiency of the results obtained (Verma, Thompson & Louviere, 1999).

7. 3. Data Collection Methods

Data collection methods are important because the way in which the collected information is used can influence the analytical approach adopted by the researcher in identifying the most suitable solutions for the identified needs. Therefore, the following methods are employed (Fig. 7.2):

1. **Surveys or questionnaires**, using various types of questions to document perceptions, attitudes, beliefs, or knowledge within a clearly defined sample of individuals;
2. **Interviews**, used to collect information from individuals through face-to-face interactions. Interviews are ideal for documenting participants' accounts, perceptions, or narratives regarding attitudes toward and responses to specific situations or phenomena. Interview data are used to define the clear structure of the application and the essential elements required for its proper functioning (Paradis et al., 2016);

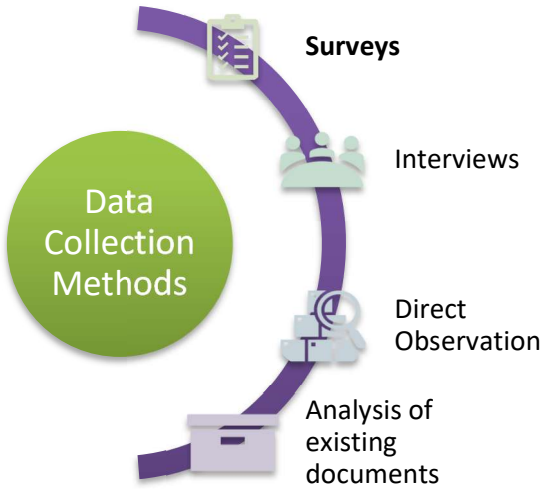


Fig. 7. 1 Data collection methods

3. **Direct observation**, used to observe the current process of timetable generation and the techniques employed by the individuals involved, in order to identify the most feasible solutions that are accessible to both students and academic staff.
4. **Analysis of existing documents**, conducted to study the regulations and procedures that may influence the application requirements.
5. Focus groups, which are "used to collect information within a group setting, through predetermined interview questions addressed by moderators to participants, in order to stimulate group discussions" (Paradis et al., 2016).

7. 4. Data Collection

Within the doctoral thesis, the first four data collection methods were used to gather all the information required for application development. The primary method for identifying critical requirements was the interview conducted with the faculty dean, the person directly responsible for organizing academic staff and students.

7.4.1 Interview

The purpose of the interview was to establish the essential requirements for application development, identify current issues in the manual teaching activity scheduling process, understand the rules and constraints that must be respected, and analyze opportunities for process optimization and automation.

The interview was structured into three sections, each addressing a key aspect of the teaching activity scheduling application.

7.4.2 Direct Observation

The manual preparation of teaching activity schedules involves four academic staff members and one vice-dean responsible for coordination. Academic staff members prepare a draft schedule for each department within the Faculty of Industrial Engineering and Robotics. These drafts are reviewed and assigned to classrooms appropriate to the type of activity by the vice-dean. Final validation of the schedule is carried out after all academic staff involved in the specialization program confirm the allocated time slots, ensuring that there are no overlaps with their personal schedules.

7.4.3 Analysis of Existing Documents

In the development process of the document management and teaching activity scheduling application, the analysis of existing documents plays a crucial role in understanding academic regulations and administrative norms that must be observed. This stage enables the alignment of application functionalities with the legal and operational framework of the university, the Ministry of Education and Research, and ARACIS.

A series of relevant documents were analysed, including internal university regulations related to the organization of the academic year and academic staff norms, as well as curricula, which provide relevant information on course titles, types of activities, and the number of hours allocated to each course. Additional documents analysed included the Higher Education Law and ARACIS standards, from which constraints regarding the number of students per group or subgroup were derived.

7.4.4 Surveys / Online Questionnaire

Another stage of the application requirements analysis involved the development of an online questionnaire implemented using Google Forms, with the aim of analysing the opinions of academic staff, department heads, and students regarding teaching activity scheduling and of formulating hard and soft constraints. The questionnaire was completed by 112 respondents: 69 students, 37 academic staff members, and 6 department heads.

The information obtained through the four data collection methods forms the basis for defining the functional requirements and the hard and soft constraints. Based on these, the software architecture of the application will be designed to address the real needs of the faculty and to reduce the time and effort required for teaching activity scheduling.

CHAPTER 8
MATHEMATICAL MODEL FOR TEACHING ACTIVITY MANAGEMENT

The classical timetabling problem, which involves assigning curricular activities to academic staff and identifying appropriate time slots and classrooms while complying with various faculty-imposed constraints, is supported by several mathematical models that have been extensively studied. The problem has a standard mathematical formulation: it is a multi-resource allocation problem that combines graph colouring, integer linear programming (ILP – Integer Linear Programming / MIP – Mixed Integer Programming), constraint programming (CP), and Boolean satisfiability (SAT) (Even, Itai & Shamir, 1976).

8. 1. ILP Mathematical Model

8.1.1 Definition of Data and Notation

In the following, the sets and input data required to define the objective function are presented.

Data sets

- D: the set of days (Monday–Friday);
- For each day $d \in D$, $H_d = \{1, \dots, m_d\}$: the set of positions (time periods) within day d(which may differ from one day to another);
- T: the set of elementary time slots (5 days × 5 slots/day). Let $T_d \subseteq T$denote the set of slots corresponding to day $d \in D$;
- R: the set of classrooms.

  The input data are defined based on the previously introduced sets.

In order to define the imposed constraints, it is necessary to introduce set equations that will be used throughout the model. Thus, using the input data D, H_d, d_e, and A_e, the following are computed:

- **Valid start times**: the set that provides all possible starting positions for event e that can be fully allocated within day d(Daskalaki, Birbas & Housos, 2004);
- **Slots covered by a start**: this set enumerates the elementary time slots (d, h')occupied by event $e$if it starts at (d, h), i.e., the continuous time interval corresponding to that activity (Daskalaki, Birbas & Housos, 2004).

8.1.3 Hard and Soft Constraints

The **hard constraints** that must be satisfied by the model are defined mathematically.

1. **Unique assignment**- each event e is scheduled exactly once:

$$\sum_{(d,h)\in S_e^{valid}} \sum_{r\in R} x_{e,d,h,r} = 1, \forall\, e \in \mathrm{E} \tag{8.10}$$

2. Verification of Instructor and Classroom Availability and Compliance with Classroom Capacity:

$$x_{e,d,h,r} \leq avail^{P}_{p(e),d,h} avail^{R}_{r,d,h}\, 1[enr_e \leq xcap_r] qual_{e,r}\,, \forall e,d,h,r \tag{8.11}$$

8.1.5 Objective Function

The ILP model fully satisfies the hard constraints, which guarantees that the solution is feasible; however, to select the best possible solution, an objective function, or optimization criterion, is introduced. The objective function is a mathematical equation that evaluates the quality of a solution based on the imposed constraints.

8.2 Decision Tree

The third algorithm selected for teaching activity scheduling is the **Random Forest Classifier**, which is supported by a collection of decision trees combined using bagging techniques and majority voting to produce a robust and accurate model.

Based on the information presented in Chapter 3 (decision trees: ID3 and C4.5) and Chapter 4 (Selection of the ML Algorithm), the following sections present the mathematical model underlying the Random Forest Classifier: the decision tree, splitting criteria for a tree, bootstrap aggregating (bagging), and voting mechanisms.

8.2.1 Dataset

The construction of a decision tree begins with a training dataset (Wang & Chen, 2016), denoted as

$$D= \{(x_i,y_i) \mid i=1,\ldots,n \},\ x_i \in R^p,\ y_i \in Y, \tag{25}$$

where:

$x_i \in R^p$ is the feature vector associated with instance i, $x_i = (x_{i1}, x_{i2},\ldots, x_{ip})$

$y_i \in Y$ is the label corresponding to observation i, and the set Ycontains all possible class values (Wang, & Chen, 2016).

8.2.2 Splitting Criteria

For each node, there exists a subset of data $S \subseteq D$. A feature $x_i$and a threshold $\theta$are selected to divide the instances into two subsets (Wang & Chen, 2016). The purpose of this partitioning is to obtain subsets that are as pure as possible at the leaves of the tree, meaning that each leaf ideally contains instances belonging to a single class.

The mathematical measure of class separation is impurity, which is computed using the **Gini index** and **entropy** (Breiman, 2001).

Gini Index

$$\text{Gini(S)} = 1 - \sum_{c \in Y} p_c^2 \text{ , } p_c = \frac{|\{(x,y) \in S: y = c\}|}{|S|}, \quad (27)$$

where p_c represents the proportion of instances in the set Sthat belong to class c

$p_c = 1 \rightarrow Gini(S) = 0$, which corresponds to the case where all instances belong to the same class (perfect purity).

Entropy

$$\text{Entropy(S)} = -\sum_{c \in Y} p_c \log p_c \text{ , } \quad (28)$$

where p_c is the proportion of instances in the set Sthat belong to class c.

Entropy(S) = 0 when there is only a single class present in the set.

Average Impurity after Splitting (Breiman, 2001):

$$I_{split}(S,j,\theta) = \frac{|S_L|}{|S|} I(S_L) + \frac{|S_R|}{|S|} I(S_R) \text{ , } \quad (29)$$

where S_R and S_L are the two resulting subsets after the split, and $I(\cdot)$may represent either the Gini index or Entropy.

The feature and threshold that minimize the impurity after splitting are selected in order to obtain leaf nodes that are as homogeneous as possible. (Breiman, 2001):

$$\text{Gain}(S,j,\theta) = I(S) - I_{split}(S,j,\theta) \quad (30)$$

8.2.3 Bagging and Voting

The basic idea behind bagging is “to train multiple models using random subsets of the training data and then combine their predictions in order to reduce variance and improve the overall accuracy of the model” (Brown, 2023).

CHAPTER 9
APPLICATION AND DATABASE DESIGN

Scheduling teaching activities for both students and academic staff has become more than merely complicated - it has evolved into a highly complex problem. Each schedule must include a

set of interconnected information elements: the classroom in which the classes are held, the taught subject, the type of activity (lecture, laboratory, seminar, project), the responsible academic staff member, the time interval required for delivering the subject, as well as the day of the week.

The development of microservices typically requires an extended team that distributes tasks across the different application development layers. In the case of teaching activity scheduling, such microservices may include: user management (login), management of academic staff preferences, course management and assignment of academic staff to courses, activity scheduling, classroom management, notifications, and the graphical user interface. A similar principle is adopted in the implementation of the application, with each element having separate components on the graphical interface side, while the backend is developed using a unified programming language that addresses all functional requirements.

The software architecture selected for application development is a **hybrid architecture**, combining a monolithic approach with modern technologies such as containerization of the database, PHP services, and Python services (Fig. 9.3). The rationale behind this decision is based on the following considerations:

- **Service separation through containers** enables parallel processing, intensive computation, and independent updates;
- **Functional requirements**: Python is used for algorithm development, PHP for backend data processing, and the database for handling complex data structures;
- **Performance and scalability**: algorithmic processes do not affect user interface accessibility, and each service can be scaled independently;
- **Fault tolerance**: failure of the Python service does not compromise the entire application.

For automatic scheduling, a web application will be developed, primarily accessed through a web browser. The implementation of the application involves multiple layers (Fig. 9.4) and components that work together to provide functionality and user interaction.

9.2. Application Flowchart – Administrator

The application administrator is responsible for managing the data required for the automation of teaching activity scheduling. The administrator's activities within the application are strictly related to data entry that supports the management of academic staff and their teaching activities, classroom management, study group management, and the academic calendar structured according to the academic year. All this information is stored in the database and is fully utilized by the automatic teaching activity generation algorithm.

9.3 Application Flowchart – Academic Staff

Academic staff members have limited operational rights within the application and can influence teaching activity scheduling only by specifying time intervals during which they are

unavailable to conduct teaching activities. According to Fig. 9.10, the activity flow for an academic staff member within the application is structured into several logical steps:

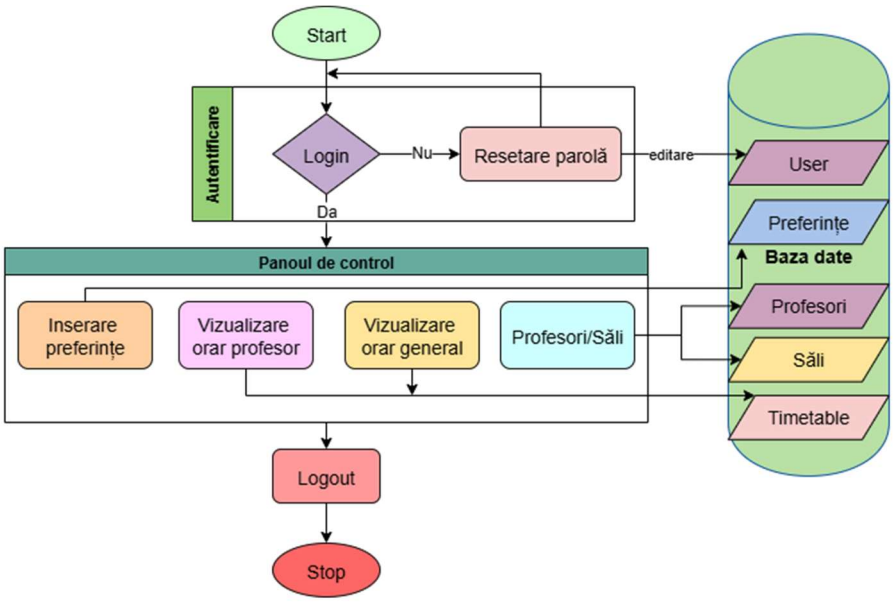


Fig. 9. 1 Application flowchart – academic staff member

9.4 Application Flowchart – Head of Department

Another role that a user may hold within the application is that of Head of Department. The Head of Department is also an academic staff member and therefore has the same rights within the platform, supplemented by two additional main components in the control panel: **Curriculum Management** and **Assignment of Academic Staff to Courses**.

Within the **Curriculum Management** module, the user can insert or edit existing curricula, while in the second module, in accordance with the regulations defined in the staff establishment plans, the user assigns the responsible academic staff members to each course (**Staff Assignment**). These are the only modifications to the application structure compared to the role of a standard academic staff member, namely *Curriculum Management* and *Staff Assignment*.

These functionalities are part of the automation of teaching activity scheduling, allowing modifications to curricula without requiring a new application version. Furthermore, staff assignment represents an additional benefit for the user, enabling the semi-automatic allocation of academic staff to their corresponding courses.

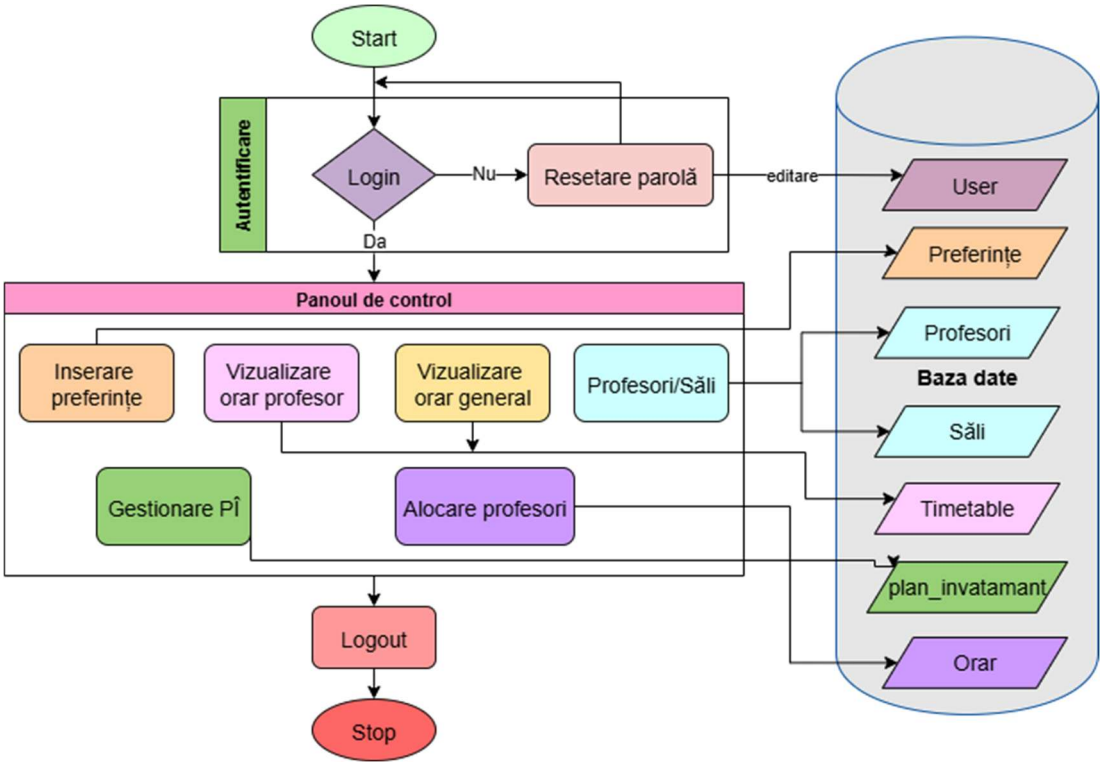


Fig. 9. 2 Application Flowchart – Head of Department

9.5 Selection of the Database Type

The development of an algorithm for teaching activity scheduling begins with the design and population of databases containing essential information, as well as the definition of relationships between tables to ensure efficient data management. The first stage of the design process involves identifying entities and the relationships among them (Scripcariu, 2011). The selected database management system for this application is **MariaDB**.

9.5.1 Main Database Design

Following the analysis of application requirements, the selection of entities represents a key element for identifying the components that play an important role in application development. Accordingly, the core entities for the development of the teaching activity scheduling generator are: **users**, **classrooms**, **academic staff**, **curricula**, and **course–instructor associations**. Each database table entity includes specific attributes, which are described in Appendix 1.

9.5.2 Optimizations Resulting from Development

The development of the application led to several modifications in the database structure. These changes stem from the automation of the entire process of entering the information required for teaching activity scheduling.

The first major modification involved the rooms table, which was transformed into Sali_materii, with the introduction of the fields Disciplina_id, sala_obligatorie, and sala_pc. These fields are necessary to indicate whether a classroom is specific to a particular course. The need for these fields arose from the fact that certain courses (e.g., Chemistry, Physics, Materials Technology) can only be conducted in specific classrooms, where other courses cannot be assigned. Additionally, classrooms equipped with computers were identified, where laboratories or project-based activities are assigned when they are not held in predefined classrooms.

Another significant modification was applied to the course_activity table, which was transformed into plan_invatamant_specializare_abreviere_an_Inceput_anSfarsit_semestrul. This table is deleted after the completion of the study cycle for a given academic program, in order to prevent unnecessary data accumulation. This change was required due to the annual modifications of curricula. It also improves the efficiency of inserting and retrieving curricula data for a specific specialization and academic year.

CHAPTER 10
IMPLEMENTATION OF AUTOMATIC GENERATION ALGORITHMS

Automatic teaching activity scheduling represents a major challenge for developers involved in algorithm design, due to the large number of customized requirements related to academic staff and teaching activities. This chapter analyzes three algorithms that exhibit distinct characteristics and advantages in obtaining optimal solutions and solving complex allocation problems under imposed constraints. The three analyzed algorithms are: **backtracking**—one of the classical methods for obtaining valid solutions; the **genetic algorithm**—an artificial intelligence method based on the natural selection process; and **machine learning**—a rapidly evolving method that involves extensive data processing and automatic learning to generate new solutions based on learned patterns.

The implementation of the algorithms is carried out based on the initial requirements and the imposed constraints.

10.1 Input Data and Desired Outcomes

Most of the information required for developing an automatic teaching activity scheduling algorithm is stored in the application database. Some data are manually introduced through the graphical interface, while others are already recorded in the system.

The main information required by a scheduling algorithm (Fig. 10.1) includes:

- **Curriculum** – courses, activities associated with each course, and the number of hours allocated for teaching;
- **Staff establishment plans** – academic staff associated with each course activity;
- **Availability of academic staff**;

- **Number of groups and subgroups** for each specialization and academic year;
- **Classrooms**, including room type and capacity.

The desired outcome of running an application that implements a scheduling algorithm is a teaching activity timetable for each study program (SP), covering all course activities defined in the curriculum (12 SP × 8 courses/SP × 2 activities/course = 192 activities).

The resulting teaching activity schedule must include information on the course, activity type, parity, responsible academic staff member, and assigned classroom. The schedule is intended to be displayed in a tabular, matrix-like format with the structure presented in Table 10.1, where colors represent the type of activity based on the generated solutions.:

Tabel 10. 1 Structură orar conform cerințe

<table>
<tr><td>Day/Time slot</td><td colspan="2">Group 1
Semi group 1</td><td colspan="2">Group 1
Semi group 2</td></tr>
<tr><td>Luni</td><td colspan="4"></td></tr>
<tr><td>8-9
9-10</td><td>Course
Classroom
Teacher</td><td>Course
Classroom
Teacher</td><td>Course
Classroom
Teacher</td><td>Course
Classroom
Teacher</td></tr>
<tr><td>10-11
11-12</td><td colspan="2">Course
Classroom
Teacher</td><td colspan="2">Course
Classroom
Teacher</td></tr>
<tr><td>12-13</td><td colspan="4" rowspan="3">Course
Classroom
Teacher</td></tr>
<tr><td>13-14</td></tr>
<tr><td>14-15</td></tr>
<tr><td>15-16</td><td colspan="2"></td><td></td><td></td></tr>
</table>

10.2. Validation of Constraints in the Teaching Activity Scheduling Process

Constraints can be divided into two categories: **hard constraints** (which must be strictly satisfied) and **soft constraints** (optional constraints that influence the quality of the solution). Based on the requirements analysis conducted within the Faculty of Industrial Engineering and Robotics, a set of constraints was formulated, which serves as the foundation for implementing the two selected algorithms.

10.3 Backtracking Algorithm

The backtracking algorithm is a method used to find solutions to complex problems. Compared to other algorithms, backtracking can be more efficient, as it eliminates a large number of infeasible solutions through a single traversal of the data. This algorithm operates by decomposing the main problem into smaller subproblems, which are solved sequentially. After finding a solution for the first subproblem, the algorithm proceeds to the next one, building upon

the previous solution. If a subproblem cannot be solved, the algorithm returns to earlier steps (backtracks) and searches for an alternative solution that satisfies both the current subproblem and the previously solved ones, after which it continues with the remaining subproblems.

10.3.1 Backtracking Algorithm Logic Applied to Teaching Activity Scheduling

It is well known that the Backtracking algorithm performs a systematic search through all possible combinations of assignments, backtracking whenever a selection leads to a conflict. Based on the technical details of the algorithm presented in Chapter 4, the following section explains the logic of the Backtracking algorithm as applied to teaching activity scheduling:

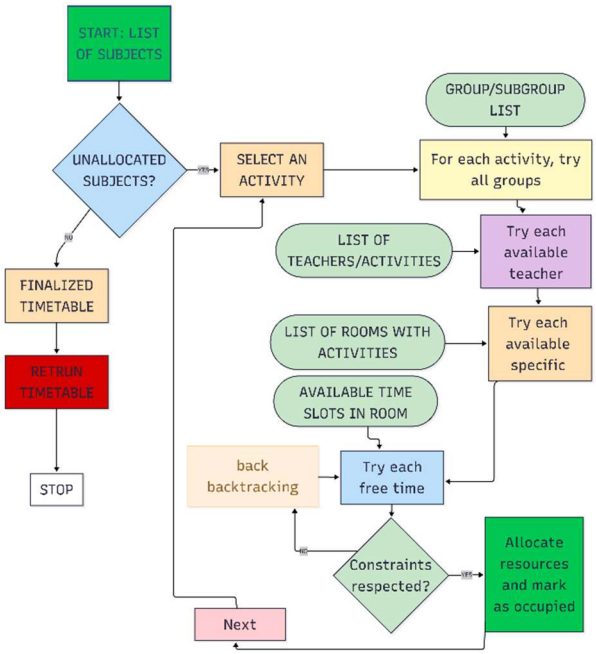


Fig. 10. 1 Schema logică algoritm Bactracking

10.4 Genetic Algorithm

Following extensive research on multiple methods for developing automatic teaching activity scheduling, it was found that genetic algorithms are among the most widely used approaches for implementing scheduling applications. Genetic algorithms can generate a sufficiently large number of solutions that satisfy hard constraints, while avoiding conflicts related to classrooms, instructors, and student groups.

10.4. Genetic Algorithm Logic Applied to Teaching Activity Scheduling

As presented in Chapter 4, genetic algorithms represent a heuristic optimization method inspired by the process of natural selection and biological evolution. The algorithm operates by iteratively modifying a population of individual solutions, randomly selecting individuals based on their fitness values to become parents, and using them to generate offspring in subsequent generations.

There are five key phases analyzed in genetic algorithms: **initial population**, **fitness function**, **selection**, **crossover**, and **mutation** (Fig. 10.9).

The design of the fitness function is essential, as the algorithm's results depend heavily on how it is formulated. The fitness value of each individual is determined by applying the fitness function, which is problem-specific and evaluates the performance of candidate solutions.

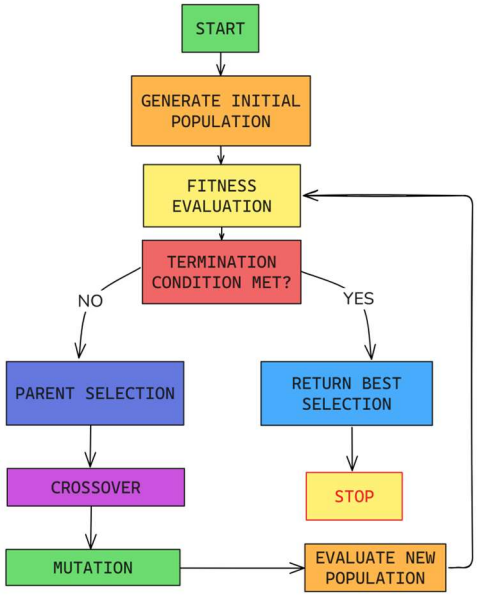


Fig. 10. 2 Schema logică algoritm Genetic

10.5 Machine Learning (ML) Algorithm

Based on the existing documentation presented in Chapter 4, which describes the steps required for developing a machine learning algorithm, an implementation methodology was designed to extract, process, and prepare data for the final prediction. After completing the data extraction process, several stages of the machine learning algorithm are executed using the Python programming language, with each stage implemented as a separate subprogram, as follows:

- **OrariiRun.py** – the orchestrator of all created subprograms;
- **ConfigReader** – the subprogram responsible for loading the JSON configurations defined in the configurator;
- **DataReader** – the subprogram that reads all data from the raw_data folder;
- **DataProcessor** – processes and standardizes the data;
- **DataWriter** – generates output files with the desired content;
- **DataPredictor** – predicts new data using a selected algorithm;
- **DataConflictSolver** – detects existing conflicts and resolves them.

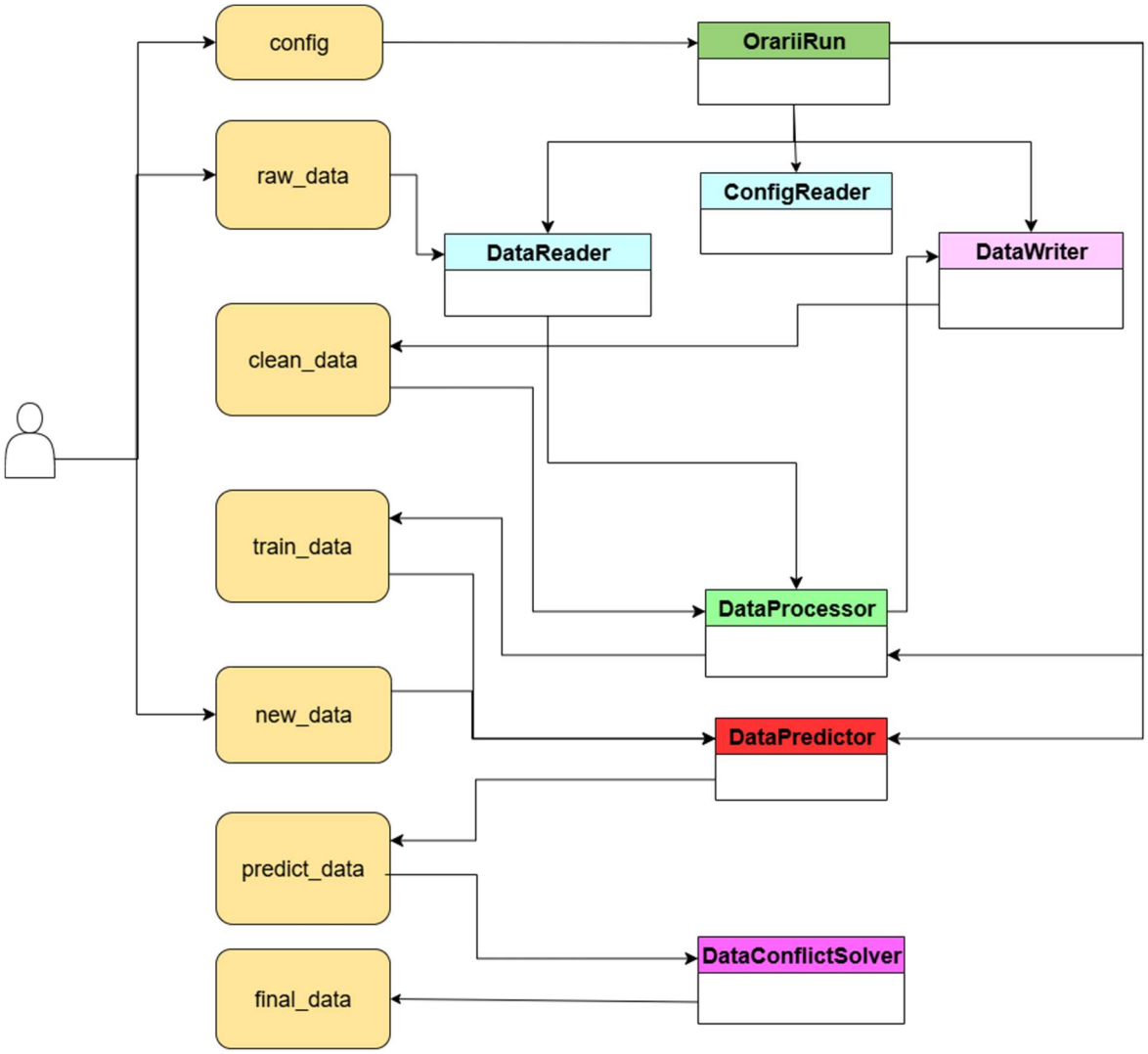


Fig. 10. 3 Metodologie de implementare ML

10.5.4. Conflict Resolution

As with any solution generation process that employs various machine learning algorithms, the resulting solutions must be “refined” to eliminate inconsistencies introduced by the values in the training dataset. For this purpose, the ConflictSolver class is used, in which four critical cases are analyzed and resolved before generating the final solution: ensuring that classroom capacity is not smaller than the number of students in a subgroup/group or specialization, and that there are no overlaps in classroom usage or in the schedules of instructors or student groups.

To achieve this, a private method, detect_conflicts(), was implemented. Within this method, each classroom assigned to an activity is compared against the number of students recorded in the groups table. If the capacity condition is not satisfied, the corresponding record is stored in a conflict variable. If the condition is satisfied, the process proceeds to the next checks, namely the identification of duplicates resulting from grouping by classroom, instructor, or group/subgroup. If duplicates are detected, they are likewise stored in the conflict variable to be resolved later (Fig. 10.26).

Conflict resolution is handled as a separate method within the ConflictSolver class, in which the two key criteria—classroom capacity and the existence of overlaps—are re-evaluated. The previously generated list of conflicts is traversed, and each criterion is analyzed individually. If a conflict is identified as being caused by insufficient classroom capacity, an available classroom that satisfies the capacity requirement is sought and replaces the original assignment in the predicted data file. If a conflict results from overlaps, additional conditions must be checked to determine an appropriate resolution strategy.

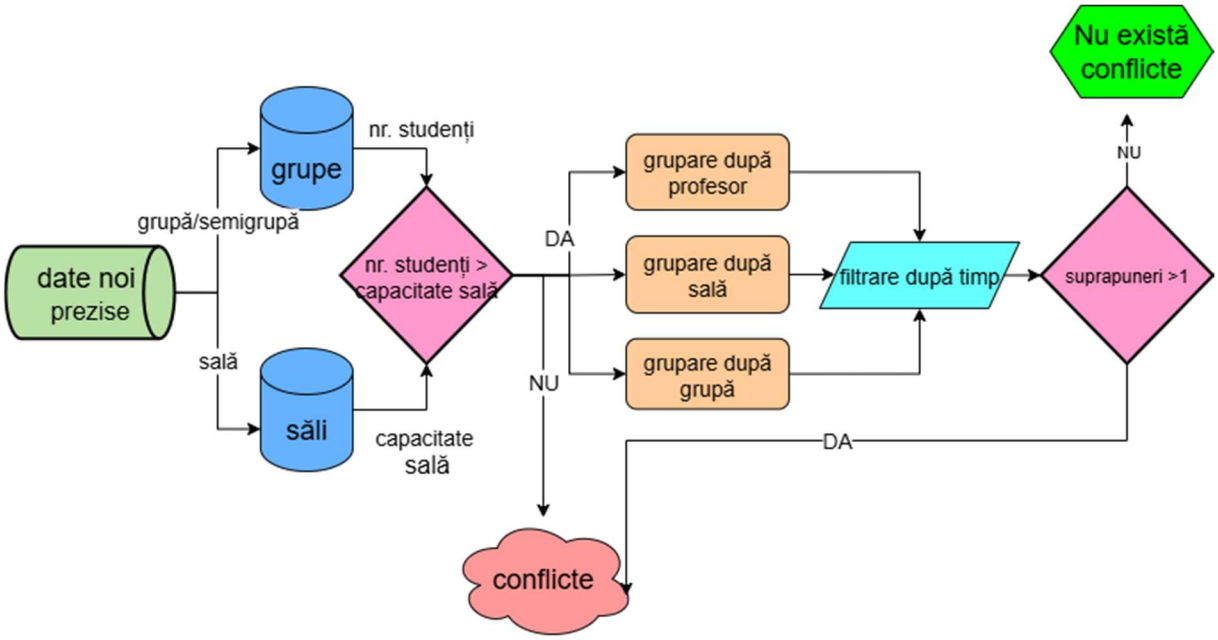


Fig. 10. 4 Conflict identification

If an overlap is identified at the classroom level, an available classroom with an appropriate capacity is searched for and replaces the original assignment in the data file. If an overlap is detected at the group level, the corresponding course is identified and an alternative available time slot is sought for reassignment. The final case, involving overlaps in instructors' schedules, is resolved by verifying the mapping between courses and instructors, identifying instructor availability, and subsequently updating the assignment in the data file.

CHAPTER 11
GRAPHICAL USER INTERFACE DESIGN

One of the three main components of application development is the Graphical User Interface (GUI), which facilitates interaction between the user and the application.

The graphical user interface was structured to present information related to teaching activity scheduling and to allow users to perform rapid and well-structured actions, such as completing forms, navigating between pages, selecting options, and more. The main components of the developed application include: authentication/login, the main page, instructors, classrooms,

instructor preferences, curricula, the page for assigning instructors to courses, and the teaching activity scheduling generator.

The organization of the graphical user interface code was implemented using the Angular framework. Pages were structured as components, with each component consisting of three main files that work together to define the layout, style, and functionality of the component: HTML, CSS, and TypeScript.

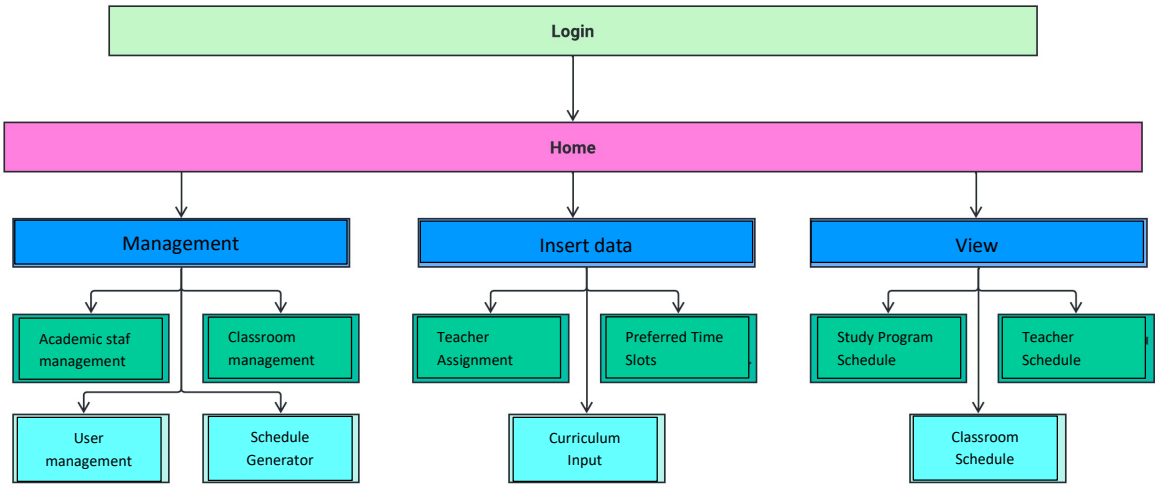


Fig. 11. 1 Structure of the graphical user interface pages

11.1. Login

The teaching activity scheduling management application is used by academic staff within a faculty, application administrators, and heads of departments. Managing private information and securing access to the application's resources and functionalities required the development of an authentication/login and registration interface. Within the application, the need for a login component arises from the private nature of the information that enables the automatic scheduling of teaching activities.

Following successful authentication, based on the user's name and role retrieved from the user_register table, the main page displays different fields and options (Fig. 11.5), as follows:

- **Administrator:** Instructors/Classrooms, ILSA (Intelligent Logic Scheduling Algorithm), Users, Curricula, Instructor Management, Classroom Management;

- **Head of Department**: My Preferences, Instructors/Classrooms, Instructor Assignment, Curricula, Personal Schedule, and Schedule Visualization;
- **Academic Staff**: My Preferences, Personal Schedule, Schedule Visualization, and Instructors/Classrooms.

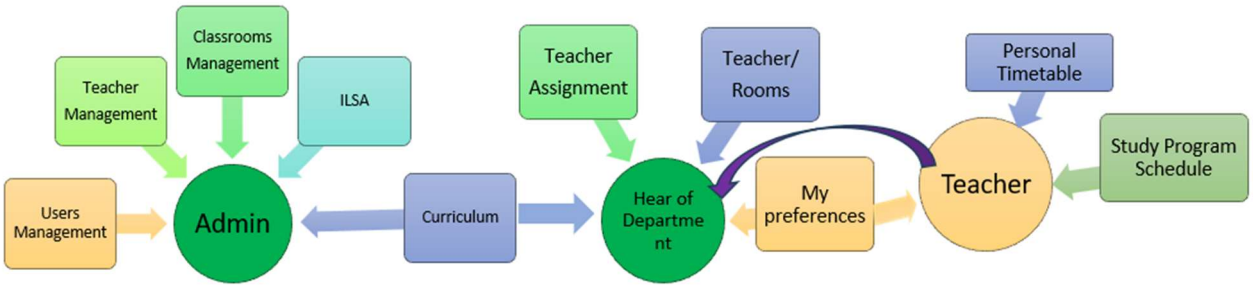


Fig. 11. 2 Structure of pages according to user roles

A page dedicated to the management of academic staff teaching courses within the Faculty of Industrial Engineering and Robotics is required to facilitate efficient and rapid administration (Fig. 11.13).

All requests sent from the frontend (GET, POST, or PUT) are processed by the server, which executes SQL operations of type INSERT/UPDATE/DELETE (4), storing the information in the database. Upon successful completion of these operations, the database returns a success message (5) to the server, which then responds with an HTTP 200 status code (operation successfully completed) to the frontend. Consequently, the user is informed of the operation status through a pop-up message, such as “Data have been successfully saved” (7) or “The user has been deleted/modified” (8), as illustrated in Fig. 11.14.

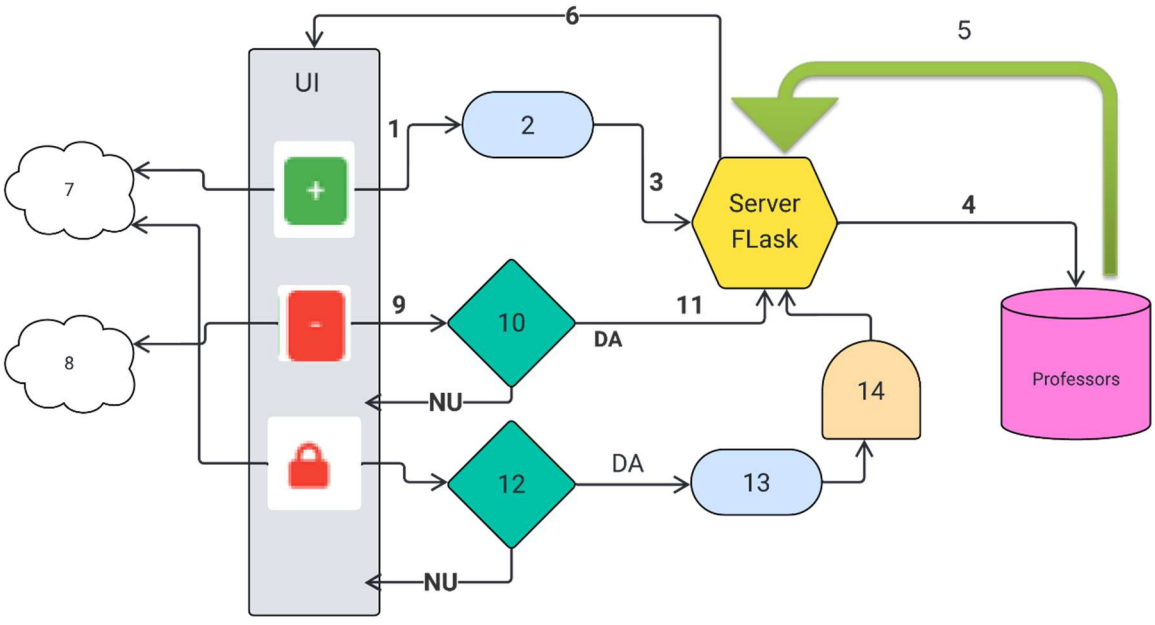


Fig. 11. 3 Workflow for adding, editing, and deleting an academic staff member

11.6. Rooms

The classrooms page is intended for academic staff and the head of department and serves to display information related to classrooms, namely the classroom name, capacity, and the type of activity that can be conducted (lecture, seminar, project, technical laboratory, or computer laboratory) (Fig. 11.16).

To display the information within the component's TypeScript file, a GET request is sent to the Flask server, which connects to the rooms table, retrieves all existing records, and inserts them into a list that is returned to the frontend (Fig. 11.15).

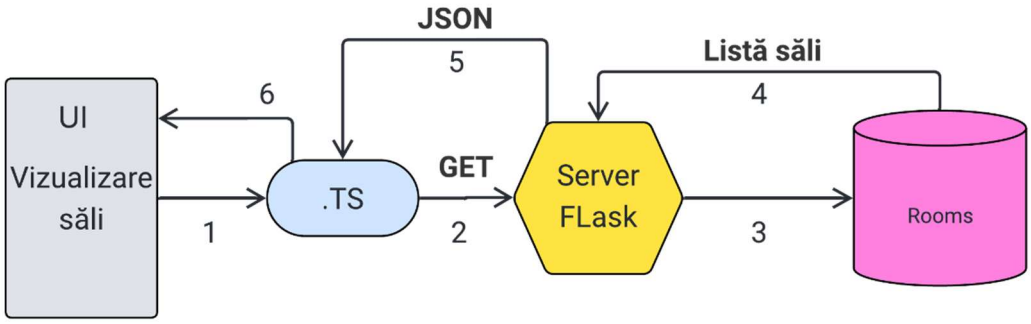


Fig. 11. 4 GET procedure for retrieving the list of academic staff

The development of an interface for managing relevant information from curricula significantly improves the efficiency of data entry into the database. Access to this interface is restricted to administrators and heads of departments. Within the interface, a visualization area was created to display previously entered curricula, along with sections for uploading, validating, and saving data for each specialization individually (Fig. 11.21).

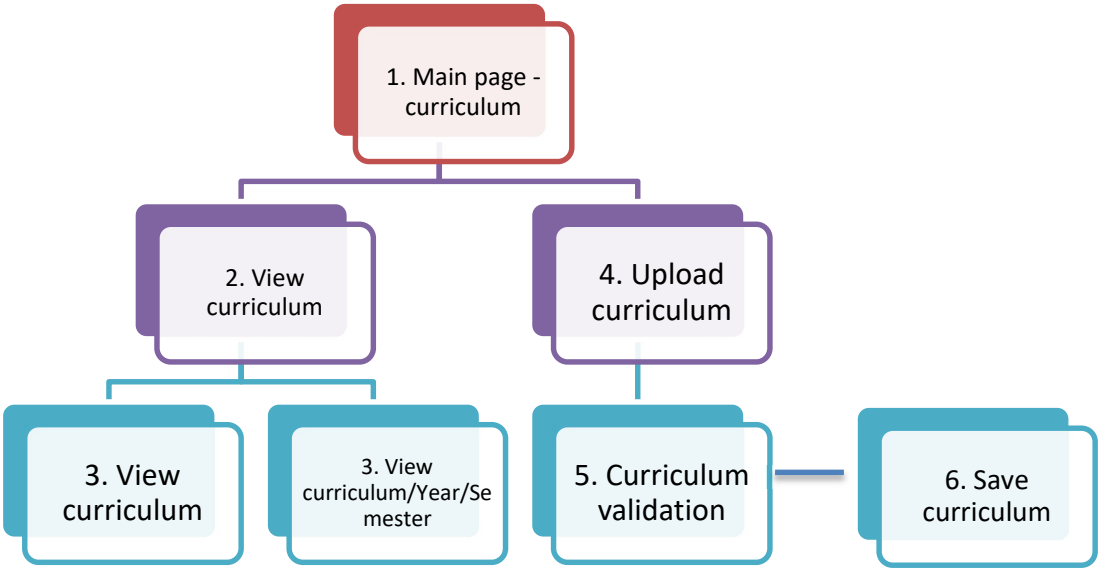


Fig. 11. 5 Structure of the Curriculum subpages

CAPITOLUL 12
ANALYSIS AND TESTING OF THE APPLICATION

This chapter presents the results obtained and the issues encountered during the development of the three types of algorithms. For the genetic algorithms and machine learning (ML) approaches, detailed reports and charts were generated for various components using advanced Python libraries such as matplotlib, seaborn, pandas, and scikit-learn. These libraries enabled efficient analysis of the results, providing a comprehensive overview of algorithm performance.

For the backtracking algorithm, the generation of automated reports is limited, as it relies on exhaustive exploration of possible solutions. Consequently, the analysis was conducted manually by interpreting the resulting data and execution times, without the use of specialized visualization tools.

12.1 Selection of specializations for result observation

At present, within the Faculty of Industrial Engineering and Robotics, there are 10 study programs (SPs) with a duration of four years, one SP with a duration of three years, and two SPs with a duration of two years, resulting in a total of 47 student cohorts across all study programs.

Given the strong objective of generating a valid teaching activity schedule for all study programs within the faculty, automatic scheduling was initially attempted using all three algorithms. However, numerous issues were encountered, which led to a restructuring of the number of selected specializations in order to obtain a smaller set of results that could be meaningfully compared.

During the implementation of the generation algorithms (backtracking and genetic), the first major issue encountered was the lack of available classroom time slots for allocating all seminars and project activities across all specializations. To verify this, using the concrete data provided by the Faculty of Industrial Engineering and Robotics, the required number of time slots needed to support all course activities was calculated based on the curricula and the existing number of groups and subgroups.

12.2 Backtracking Algorithm

Within the backtracking algorithm, the obtained results were generated in the form of .csv files. In order to verify constraint satisfaction and data correctness, part of the information was manually processed. The data were sorted by specialization, year of study, day, and time interval. Subsequently, a web-based AI content generation application, ChatGPT, was used to arrange the data into the desired teaching activity schedule format.

The resulting schedule was exported as an Excel file and has the structure illustrated in Fig. 12.1.

Zi	Interval	611		612	dual
		Sg.1	Sg.2	Sg. 1	Sg.2
Monday	08:00–10:00	Programarea calculatoarelor È™i limbaj de programare 1 curs, Tom Savu, CD 016	Programarea calculatoarelor È™i limbaj de programare 1 curs, Tom Savu, CD 016	Programarea calculatoarelor È™i limbaj de programare 1 curs, Tom Savu, CD 016	Programarea calculatoarelor È™i limbaj de programare 1 curs, Tom Savu, CD 016
	10:00–12:00	Chimie curs, Mariana Prodana, CD 016	Chimie curs, Mariana Prodana, CD 016	Chimie curs, Mariana Prodana, CD 016	Chimie curs, Mariana Prodana, CD 016
	12:00–14:00		Chimie laborator, George Burc, EG 005 - par	Limba engleza seminar, Alexandra Salcianu, CB 201	Limba engleza seminar, Alexandra Salcianu, CB 201
	14:00–16:00			Chimie laborator, George Burc, EG 005 - par	
	16:00–18:00				
	18:00–20:00				
Tuesday	08:00–10:00	Programarea calculatoarelor È™i limbaj de programare 1 laborator, Vlad GheorghiÈ›Ã, CB 202	Programarea calculatoarelor È™i limbaj de programare 1 laborator, Bogdan Jugravu, CB 203		D esen tehnic È™i infograficÃƒ 1 laborator, Iulius Stroe, BN 323
	10:00–12:00	Bazele ingineriei industriale 1 proiect, Corina Botez, BN 323	Bazele ingineriei industriale 1 proiect, Corina Botez, BN 323	Desen tehnic È™i infograficÃƒ 1 laborator, Iulius Stroe, BN 323	limbaj de programare 1 laborator, Bogdan Jugravu, CB 203
	12:00–14:00	Desen tehnic È™i infograficÃƒ 1 laborator, Iulius Stroe, BN 323		Mecanica 1 seminar, Costel Popa, CB 021	Mecanica 1 seminar, Costel Popa, CB 021
	14:00–16:00	Limba engleza seminar, Alexandra Salcianu, BN 027	D esen tehnic È™i infograficÃƒ 1 laborator, Iulius Stroe, BN 323	Programarea calculatoarelor È™i limbaj de programare 1 laborator, Bogdan Jugravu, CB 203	
	16:00–18:00	Mecanica 1 curs, Prunau Maria, CD 016	Mecanica 1 curs, Prunau Maria, CD 016	Mecanica 1 curs, Prunau Maria, CD 016	Mecanica 1 curs, Prunau Maria, CD 016

Fig. 12. 1 Teaching activity schedule obtained using backtracking

Under these conditions, teaching activity schedules were generated for all selected specializations, covering Years I–IV, and compliance with the proposed constraints was analyzed. The following results were obtained (Table 12.2).

12.3 Genetic Algorithm

Following the execution of the genetic algorithm, viable solutions were generated and stored in a database. The achievement of the proposed objectives materialized through the implementation of a function that generates a matrix in HTML format, enabling faster visualization of the data produced by the algorithm (Fig. X).

Based on the analysis of the results obtained for the four selected specializations, the following observations were made:

- Allocation of certain courses to time slots without assigning a classroom;
- Allocation of certain courses to time slots without assigning an instructor;
- Introduction of idle time gaps between classes;
- Uneven distribution of activities across days.

For each specialization, the generated teaching activity schedule can be downloaded in .html format, with the following naming convention: orar_specializare_serie_an.html, and its graphical interface is illustrated in Fig. 12.2.

Orar An 1 IAII

Zi/Interval	611_1	611_2	612_1	612_2	613_1	613_2
Monday 08:00-10:00		Educație fizică 1 (CB-005) Seminar (para)				
Monday 10:00-12:00		Algebră liniară, geometrie analitică și diferențială (CE-301bis) Seminar (para)			Limbi moderne 1 - Franceză (CB-301) Curs (impara)	Limbi moderne 1 - Franceză (CB-301) Curs (impara)
Monday 12:00-14:00	Chimie (AN-017) Curs	Chimie (AN-017) Curs	Chimie (AN-017) Curs	Chimie (AN-017) Curs	Chimie (AN-017) Curs	Chimie (AN-017) Curs
Monday 14:00-16:00	Limbi moderne 1 - Engleză (CE-301bis) Seminar	Limbi moderne 1 - Engleză (CE-301bis) Seminar				Desen tehnic și infografică 1 (CK-006) Laborator (3 ore)
Monday 16:00-18:00		Desen tehnic și infografică 1 (CK-007) Laborator (3 ore)	Chimie (CA-013) Laborator (para)			
Monday 18:00-20:00						
Tuesday 08:00-10:00			Programarea calculatoarelor și limbaje de programare 1 (CB-109) Laborator		Fizică (CK-009d) Laborator (para)	Algebră liniară, geometrie analitică și diferențială (CB-005) Seminar (para)
Tuesday 10:00-12:00	Limbi moderne 1 - Engleză (CD-016) Curs (impara)	Limbi moderne 1 - Engleză (CD-016) Curs (impara)	Limbi moderne 1 - Engleză (CD-016) Curs (impara)	Limbi moderne 1 - Engleză (CD-016) Curs (impara)		Educație fizică 1 (CB-008) Seminar (para)
Tuesday 12:00-14:00	Algebră liniară, geometrie analitică și diferențială (CB-006) Seminar (para)			Desen tehnic și infografică 1 (CB-009) Laborator (3 ore)		
Tuesday 14:00-16:00			Educație fizică 1 (CB-005) Seminar (para)			
Tuesday 16:00-18:00	Desen tehnic și infografică 1 (CD-008) Curs (3 ore)	Desen tehnic și infografică 1 (CD-008) Curs (3 ore)	Desen tehnic și infografică 1 (CD-008) Curs (3 ore)	Desen tehnic și infografică 1 (CD-008) Curs (3 ore)	Desen tehnic și infografică 1 (CD-008) Curs (3 ore)	Desen tehnic și infografică 1 (CD-008) Curs (3 ore)
Tuesday 18:00-20:00						
Wednesday 08:00-10:00	Educație fizică 1 (CB-005) Seminar (para)		Algebră liniară, geometrie analitică și diferențială (CB-008) Seminar (para)			
Wednesday 10:00-12:00	Programarea calculatoarelor și limbaje de programare 1 (AN-028) Curs (3 ore)	Programarea calculatoarelor și limbaje de programare 1 (AN-028) Curs (3 ore)	Programarea calculatoarelor și limbaje de programare 1 (AN-028) Curs (3 ore)	Programarea calculatoarelor și limbaje de programare 1 (AN-028) Curs (3 ore)	Programarea calculatoarelor și limbaje de programare 1 (AN-028) Curs (3 ore)	Programarea calculatoarelor și limbaje de programare 1 (AN-028) Curs (3 ore)

Fig. 12. 2 Teaching activity schedule obtained using genetic algorithms

According to the fundamental principle of genetic algorithms, Eiben and Smith state that "the quality of the best solution found by an evolutionary algorithm depends on the population size and the number of generations. Larger populations tend to produce greater diversity and require fewer generations to converge, whereas smaller populations may need more generations to explore the search space effectively" (Eiben & Smith, 2003).

The Python programming language is used to compute the obtained values and to display the generated 3D graph. The library used for score calculation is NumPy (Numerical Python), which is employed to create the Mahfound model function, while matplotlib is used for data visualization, and Axes3D, a module of matplotlib, is used to generate 3D visualizations of the computed values. Based on the data contained in the matrix, a 3D plot is generated for the number of generations, population size, and the estimated fitness score.

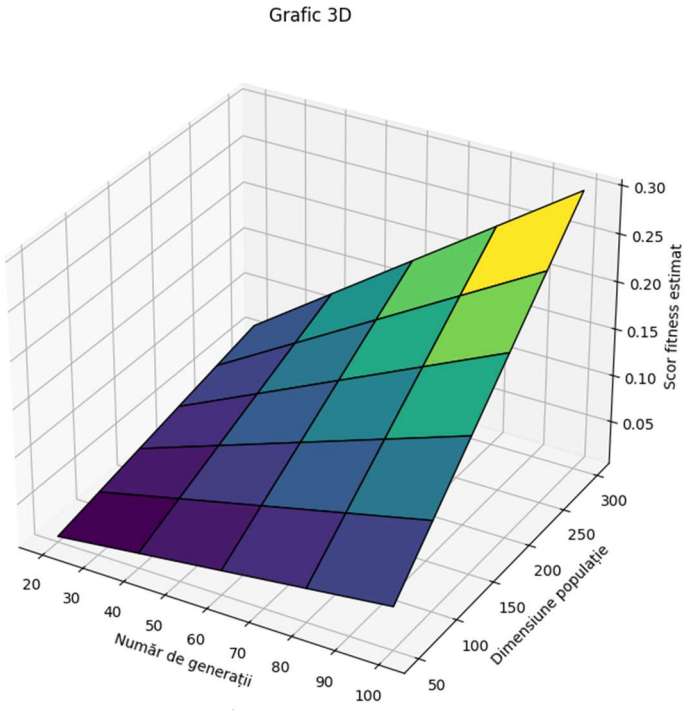


Fig. 12. 3 Grafic 3D Generații-Populație-Fitness

As illustrated in Fig. 12.5, the fitness score increases rapidly during the initial phases but reaches a plateau once a threshold of approximately 200–250 individuals and more than 80 generations is reached. This behavior reflects a natural saturation of the algorithm, in which increasing computational resources no longer leads to significant performance improvements.

12.4 Machine Learning

Based on the processing of data extracted from existing teaching activity schedules over the past 10 years, training data were obtained for predicting new scheduling data. Accordingly, the **Random Forest Classifier** algorithm was applied, configured with standard parameters: 100 decision trees (*n_estimators* = 100), the Gini splitting criterion (*criterion* = 'gini'), and no depth limitation (*max_depth* = None). The model was used to predict three parameters required for scheduling, namely the **day**, **time slot**, and **classroom**. To ensure reproducibility, a fixed random seed was set (*random_state* = 42).

During the training stage, the dataset was split using the train_test_split method from the **scikit-learn** library, allocating 80% of the sample for training and 20% for testing. For each scheduling parameter—day, time slot, and classroom—a separate model was trained, treating each prediction as a multi-class classification problem. The trained models were stored in an internal structure (self.models) and were built using data extracted from the train_data.csv file, allowing individual access and evaluation for each parameter.

All this information was partially processed manually and, with the assistance of an AI-based tool, was transformed into a matrix representation with days and time slots. As a result, a teaching activity schedule was obtained that closely resembles the existing format and can be validated only after being reviewed and verified by academic staff. Based on the analyses performed, it can be stated that this represents a stable version of an automatically generated timetable that satisfies the imposed soft constraints.

Ziua /	611-1	611-2	612	dual
Luni - 08			Educație fizică / Pricop Adrian / sala sport (par)	Educație fizică / Pricop Adrian / sala sport (par)
Luni - 10	Educație fizică / Pricop Adrian / sala sport (impar)	Educație fizică / Pricop Adrian / sala sport (impar)	Chimie / George Burcă / eg005 (par)	Chimie / Mariana Prodana / eg005 (par)
Luni - 12	Limba engleză / Alexandra Salcianu / bn027 (impar)	Limba engleză / Alexandra Salcianu / bn027 (impar)	Bazele ing. / Corina Botez / bn323 (sapt. 1-7)	Bazele ing. / Corina Botez / bn323 (sapt. 1-7)
Luni - 14	Bazele ing. / Corina Botez / bn323 (sapt. 1-7)	Bazele ing. / Corina Botez / bn323 (sapt. 1-7)		
Marți - 08	desen tehnic / Sorin Botez / bn308 (saptamanal)	desen tehnic / Iulius Stroe / bn323 (saptamanal)		
Marți - 12	Algebra lin. / Cristina Calineata / cb006 (par)	Algebra lin. / Cristina Calineata / cb006 (par)		
			Mecanica / Costel Popa / bn023 (impar)	Mecanica / Costel Popa / bn023 (impar)
Marți - 14			Algebra lin. / Cristina Calineata / cb007 (par)	Algebra lin. / Cristina Calineata / cb007 (par)
Marți - 16			Limba engleză / Alexandra Salcianu / cb201 (par)	Limba engleză / Alexandra Salcianu / cb201 (par)
Miercuri - 10	Mecanica / Costel Popa / cb021 (impar)	Mecanica / Costel Popa / cb021 (impar)	Știința materialelor / Marius Vasilescu / jk205 (par)	
Miercuri - 12	desen tehnic / Sorin Botez / cd016 (impar)	desen tehnic / Sorin Botez / cd016 (impar)	desen tehnic / Sorin Botez / cd016 (impar)	desen tehnic / Sorin Botez / cd016 (impar)
	Engleză / Alexandra Salcianu / cd016 (par)	Engleză / Alexandra Salcianu / cd016 (par)	Engleză / Alexandra Salcianu / cd016 (par)	Engleză / Alexandra Salcianu / cd016 (par)

Fig. 12. 4 Teaching activity schedule generated using machine learning

12.6 Comparison of the Three Algorithms

The three algorithms analyzed in this thesis—**backtracking**, the **genetic algorithm**, and the **Random Forest Classifier**—offer different approaches to the problem of generating university teaching activity schedules.

The **Backtracking algorithm**, being an exact method, can find a valid solution; however, it incurs a high computational cost, which makes it difficult to use and apply in scenarios involving multiple constraints. The **genetic algorithm** (an evolutionary method) provides a good compromise between solution quality and execution time, successfully generating an activity schedule that is optimal with respect to the imposed constraints, although it does not guarantee a fully valid solution in all cases. The **Random Forest Classifier**, as a machine learning algorithm, operates efficiently in terms of moderate resource usage, but it relies on labeled historical data and lacks the ability to perform optimization or to validate the logical consistency of a teaching activity schedule. Thus, each method exhibits distinct strengths and can be selected depending on the application context or combined with another algorithm to obtain optimal solutions.

For a graphical comparison of the three algorithms, a script was developed that assigns scores on a scale from 0 to 10 to the main criteria, and the obtained results were represented graphically.

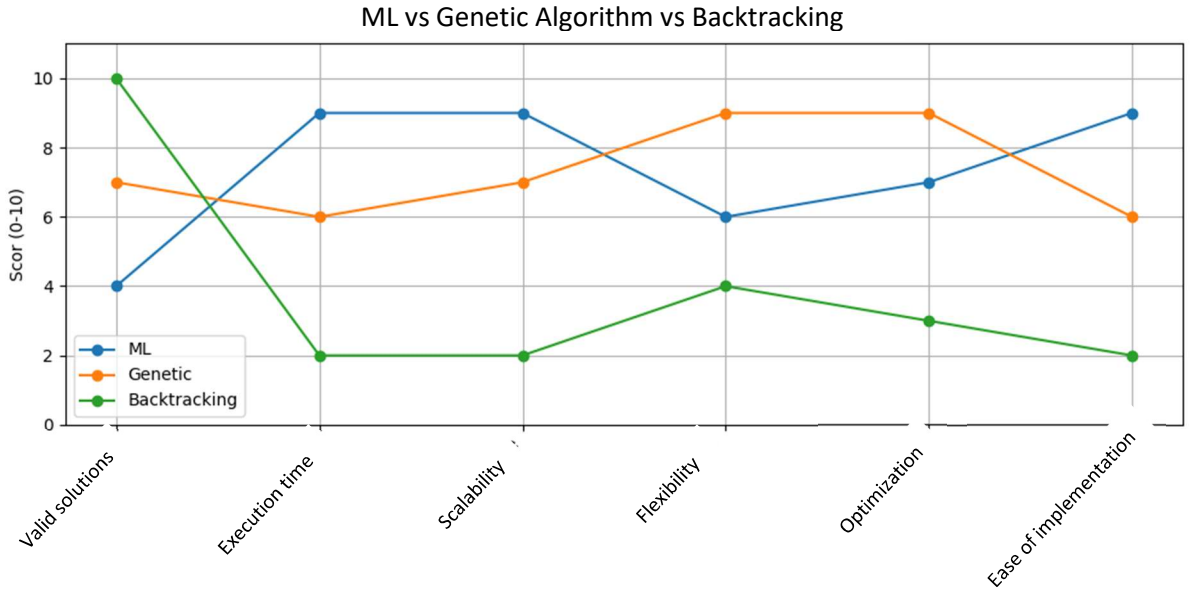


Fig. 12. 5 Algorithm comparison chart

CONCLUSIONS

One of the most challenging decisions in the automatic generation of teaching activity schedules is the selection of the algorithm to be implemented. The detailed analysis of numerous existing applications used in universities and available online supported the choice of the three algorithms implemented in this thesis. Three relevant algorithms for addressing this problem were analyzed—**Backtracking**, **Genetic Algorithm**, and **Random Forest Classifier**—focusing on their accuracy, efficiency, and adaptability to real-world scenarios. The implementation of these

algorithms proved to be a viable approach, with each method exhibiting specific advantages, disadvantages, and limitations.

ORIGINAL CONTRIBUTIONS

This work has both a theoretical and an applied character and presents a series of original contributions related to modeling, algorithm implementation, and the development of an application for teaching activity scheduling within the Faculty of Industrial Engineering and Robotics.

4. *Practical Contributions Related to Application Development*
    a. Design of the software architecture for application development;
    b. Design, development, and optimization of the database required for implementing the software application;
    c. Implementation of the Backtracking, Genetic Algorithm, and Random Forest Classifier using the Python programming language;
    d. Design, development, and optimization of the graphical user interface (frontend) of the software application using Angular;
    e. Testing of the algorithms and the graphical user interface, and development of comparative analyses of the results obtained for each implemented algorithm.

FUTURE RESEARCH DIRECTIONS

Based on the developments carried out and the analysis of the three algorithms, the possibility of optimization or hybrid implementation can be confirmed. The complexity of the automatic teaching activity scheduling problem, together with the existence of a wide range of optimization algorithms, significantly expands the scope for multiple research directions. Developing a scalable, efficient, and fast solution in which artificial intelligence, graph theory, and various algorithms converge to address the real needs of higher education institutions represents a promising research avenue.

One important future direction involves the optimization of genetic algorithm parameters (mutation, crossover, and population size). Selecting appropriate values for population size, mutation rate, or selection criteria can have a decisive impact on solution quality. Future research may focus on self-tuning algorithms, which automatically adjust their parameters based on the evolution of the solution. Such approaches can reduce human intervention in parameter selection and enable adaptive parameter choice based on objective criteria.