



People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
Higher School of Computer Science and Digital Technologies

AI 3rd Mini-Project Report

Theme:

Constraint Satisfaction Problem for Semester 2 Timetable
Scheduling

The Group Members:

OUARI Kenza

GUIR Mohamed Abdelhai

MENAD Abdelmadjid

BAHLOUL Rami

ATOUI Abderahman Yakoub

Supervisors:

Mrs. LEKEHALI Soumia

Mr BECHAR amine

Contents

1	Introduction	2
2	Problem Formulation, Variable Domains, and Constraints	3
2.1	Problem Formulation	3
2.2	Variables	3
2.3	Variable Domains	3
2.4	Constraints	3
2.4.1	Hard Constraints	3
2.4.2	Soft Constraints (Optional for Extra Grades)	4
3	Task	5
4	Problem Description	6
5	Application Development	7
5.1	Approach	7
5.1.1	Problem Formulation	7
5.1.2	Constraints Implementation	7
5.1.3	Algorithm Implementation	7
5.2	Tools and Technologies	7
6	Challenges and Solutions	8
6.1	Challenges	8
6.2	Solutions	8
7	Results and Discussion	9
7.1	Screenshots	9
7.1.1	Application Interface	9
7.1.2	Code Snippets	14
8	Conclusion	18

Chapter 1

Introduction

This project aims to tackle the intricate challenge of scheduling the Semester 2 timetable for first-year Computer Science students. Crafting a timetable that satisfies both the constraints and preferences of students and faculty alike is a daunting task that requires a systematic and rigorous approach.

The primary objective of this project is to formulate the problem in a manner that defines the variables, domains, and constraints, thereby ensuring the feasibility of the generated timetable. To achieve this, we will develop an application using a Constraint Satisfaction Problem (CSP) framework, a well-established approach for solving such problems. This application will integrate backtracking algorithms, such as the AC3 algorithm, which will aid in reducing variable domains by eliminating incompatible options, as well as heuristics such as MRV (Minimum Remaining Values), MCV (Most Constraining Variable), and LCV (Least Constraining Value) to enhance solution efficiency.

The ultimate goal is to generate a timetable that satisfies all constraints imposed by the scheduling problem while striving to adhere to optional constraints as much as possible. Although optional, these constraints are crucial for optimizing aspects such as the equitable distribution of faculty workload.

In this report, we will elaborate on the methodology followed to formulate the problem, implement constraints, and develop the scheduling application. We will also address the challenges encountered along the way and the solutions devised to overcome them. Finally, we will evaluate the performance of our scheduling algorithm and analyze the results obtained in our quest to solve this complex scheduling problem.

Chapter 2

Problem Formulation, Variable Domains, and Constraints

2.1 Problem Formulation

The problem involves scheduling various courses for Semester 2 within a week consisting of five days (Sunday to Thursday). Each day has multiple work slots. The goal is to allocate these slots to different courses while satisfying several constraints.

2.2 Variables

The variables in this problem are the time slots for each course and tutorial. Each course requires a specific number of slots, and these slots must be assigned in a way that satisfies all constraints.

2.3 Variable Domains

The domains for these variables are the available time slots across the week:

- **Days:** Sunday, Monday, Tuesday, Wednesday, Thursday
- **Slots:**
 - Sunday, Monday, Wednesday, Thursday: 5 slots each day
 - Tuesday: 3 morning slots

2.4 Constraints

Constraints ensure that the timetable adheres to the given requirements. They are divided into hard constraints and soft constraints.

2.4.1 Hard Constraints

1. The week consists of five days: Sunday, Monday, Tuesday, Wednesday, and Thursday.

2. Each day has five work slots, except Tuesday which has only three in the morning.
3. A maximum of three successive slots is allowed.
4. Lectures of different courses should not be scheduled in the same slot.
5. Different courses for the same group should have different slot allocations.

2.4.2 Soft Constraints (Optional for Extra Grades)

- Each teacher should have a maximum of two days of work.

Chapter 3

Task

The task is to develop an application that:

- Formulates the problem
- Defines variable domains
- Implements constraints
- Employs backtracking algorithms, AC3, MRV, MCV, and LCV
- Generates a feasible timetable that satisfies the hard constraints and aims to satisfy as many soft constraints as possible

Chapter 4

Problem Description

The scheduling involves the following courses and their corresponding requirements for 1st-year Computer Science students:

- **Sécurité:** one lecture + one TD (Teacher 1)
- **Méthodes formelles:** one lecture + one TD (Teacher 2)
- **Analyse numérique:** one lecture + one TD (Teacher 3)
- **Entrepreneuriat:** one lecture (Teacher 4)
- **Recherche opérationnelle 2:** one lecture + one TD (Teacher 5)
- **Distributed Architecture & Intensive Computing:** one lecture + one TD (Teacher 6)
- **Réseaux 2:** one lecture + one TD (Teacher 7), one TP (Teachers 8, 9, 10)
- **Artificial Intelligence:** one lecture + one TD (Teacher 11), one TP (Teachers 12, 13, 14)

Chapter 5

Application Development

5.1 Approach

5.1.1 Problem Formulation

We identified the variables (time slots for each course) and their respective domains (available time slots in a week). The constraints were then clearly defined to guide the allocation of these slots.

5.1.2 Constraints Implementation

Hard constraints were implemented first to ensure the timetable's feasibility. Soft constraints were implemented to improve the quality of the timetable.

5.1.3 Algorithm Implementation

We used backtracking algorithms enhanced with preprocessing and heuristics:

- **AC3 Algorithm:** Used as a preprocessing step to reduce variable domains by eliminating inconsistent values.
- **Heuristics (MRV, MCV, LCV):** Applied during backtracking to efficiently select the next variable to assign and the value to assign to it.

5.2 Tools and Technologies

- **Programming Languages:** Python (Django), HTML, CSS, JavaScript
- **Technologies and Tools:** Postman, Django, VS Code
- **Constraint Package:** python-constraint or similar packages suitable for CSP problems

Chapter 6

Challenges and Solutions

6.1 Challenges

- **Domain Reduction:** Managing the domains of variables to ensure constraints are met and the solution space is feasible.
- **Constraint Propagation:** Efficiently propagating constraints to reduce the search space and improve the algorithm's performance.

6.2 Solutions

- **AC3 Algorithm:** Implemented as a preprocessing step to reduce domains by removing values that are inconsistent with constraints.
- **Heuristics (MRV, MCV, LCV):** Utilized to select variables and values that minimize the remaining search space and handle the most constrained variables first.

Chapter 7

Results and Discussion

The application successfully generated feasible timetables that satisfied all hard constraints and maximized the satisfaction of soft constraints.

Note: The schedule shows one course per group, but in principle, the same time and lecture hall are used for all groups.

Below are screenshots of the application interface and code snippets to illustrate the functionality and workings of the developed application.

7.1 Screenshots

7.1.1 Application Interface

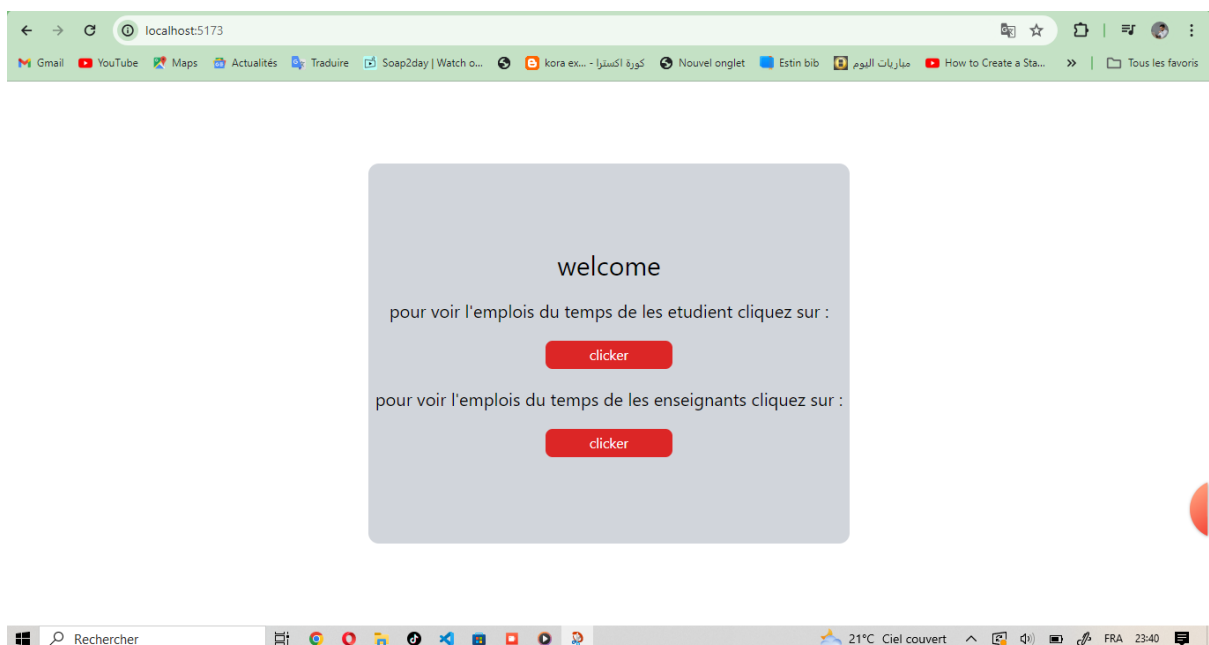


Figure 7.1: Application Interface

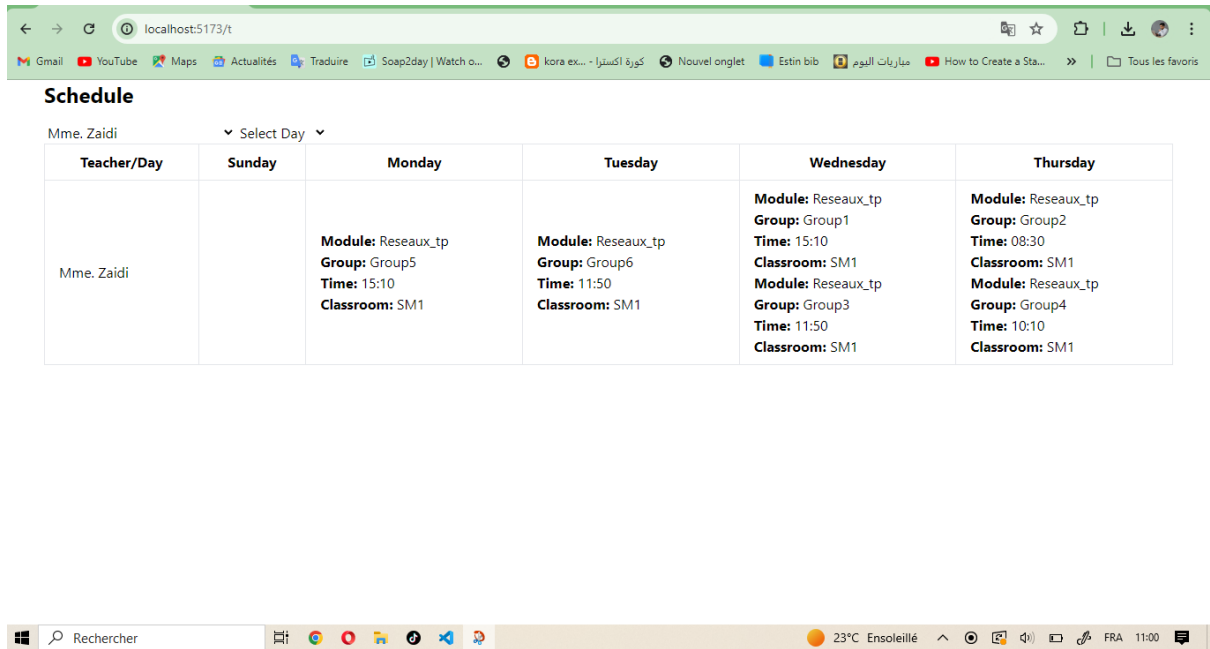


Figure 7.2: Application Interface

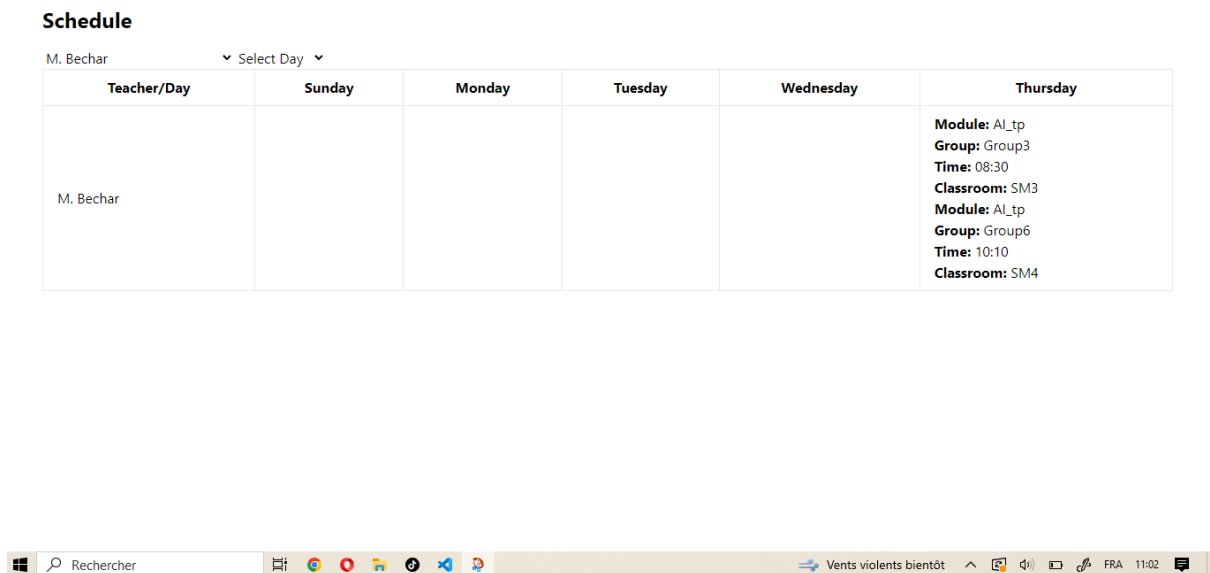


Figure 7.3: Application Interface

Schedule

Day/Group	Group1	Group2	Group3	
Sunday	Module: Securite_lecture Teacher: Dr. Djebari Time: 08:30 Classroom: Amphi7 <hr/> Module: Methodes_formelles_lecture Teacher: Dr. Zedek Time: 10:10 Classroom: Amphi7 <hr/> Module: Securite_td Teacher: Mme. Djenane Time: 11:50 Classroom: S1 <hr/> Module: Methodes_formelles_td Teacher: Dr. Zedek Time: 15:10 Classroom: S4	Module: Securite_lecture Teacher: Dr. Djebari Time: 08:30 Classroom: Amphi7 <hr/> Module: Methodes_formelles_lecture Teacher: Dr. Zedek Time: 10:10 Classroom: Amphi7 <hr/> Module: Securite_td Teacher: Mme. Khelouf Time: 11:50 Classroom: S2 <hr/> Module: Analyse_numerique_td Teacher: Dr. Alkama Time: 15:10 Classroom: S5	Module: Securite_lecture Teacher: Dr. Djebari Time: 08:30 Classroom: Amphi7 <hr/> Module: Methodes_formelles_lecture Teacher: Dr. Zedek Time: 10:10 Classroom: Amphi7 <hr/> Module: Securite_td Teacher: Mme. Kassa Time: 11:50 Classroom: S3 <hr/> Module: Recherche_operationnelle_td Teacher: Dr. Issadi Time: 15:10 Classroom: S6	Module: Securite_lecture Teacher: Dr. Djebari Time: 08:30 Classroom: Amphi7 <hr/> Module: Methodes_formelles_lecture Teacher: Dr. Zedek Time: 10:10 Classroom: Amphi7 <hr/> Module: Securite_td Teacher: Mme. Kassa Time: 11:50 Classroom: S3 <hr/> Module: Recherche_operationnelle_td Teacher: Dr. Issadi Time: 15:10 Classroom: S6
Monday	Module: Analyse_numerique_lecture Teacher: Dr. Alkama Time: 08:30 Classroom: Amphi7 <hr/> Module: Entrepreneuriat_lecture Teacher: Dr. Kaci Time: 10:10 Classroom: Amphi7 <hr/> Module: Analyse_numerique_td Teacher: Dr. Alkama Time: 11:50 Classroom: S2 <hr/> Module: Recherche_operationnelle_td Teacher: Dr. Issadi Time: 15:10 Classroom: S3	Module: Analyse_numerique_lecture Teacher: Dr. Alkama Time: 08:30 Classroom: Amphi7 <hr/> Module: Entrepreneuriat_lecture Teacher: Dr. Kaci Time: 10:10 Classroom: Amphi7 <hr/> Module: Methodes_formelles_td Teacher: Dr. Zedek Time: 11:50 Classroom: S1 <hr/> Module: Distributed_architecture_td Teacher: Dr. Djenadi Time: 15:10 Classroom: S4	Module: Analyse_numerique_lecture Teacher: Dr. Alkama Time: 08:30 Classroom: Amphi7 <hr/> Module: Entrepreneuriat_lecture Teacher: Dr. Kaci Time: 10:10 Classroom: Amphi7 <hr/> Module: Methodes_formelles_td Teacher: Dr. Zedek Time: 15:10 Classroom: S1 <hr/> Module: Distributed_architecture_td Teacher: Dr. Djenadi Time: 11:50 Classroom: S4	Module: Analyse_numerique_lecture Teacher: Dr. Alkama Time: 08:30 Classroom: Amphi7 <hr/> Module: Entrepreneuriat_lecture Teacher: Dr. Kaci Time: 10:10 Classroom: Amphi7 <hr/> Module: Methodes_formelles_td Teacher: Dr. Zedek Time: 15:10 Classroom: S1 <hr/> Module: Distributed_architecture_td Teacher: Dr. Djenadi Time: 11:50 Classroom: S4
Tuesday	Module: Recherche_operationnelle_lecture Teacher: Dr. Issadi Time: 08:30 Classroom: Amphi7 <hr/> Module: Distributed_architecture_lecture Teacher: Dr. Djenadi Time: 10:10 Classroom: Amphi7 <hr/> Module: Distributed_architecture_td Teacher: Dr. Djenadi Time: 11:50 Classroom: S4	Module: Recherche_operationnelle_lecture Teacher: Dr. Issadi Time: 08:30 Classroom: Amphi7 <hr/> Module: Distributed_architecture_lecture Teacher: Dr. Djenadi Time: 10:10 Classroom: Amphi7 <hr/> Module: Recherche_operationnelle_td Teacher: Dr. Issadi Time: 11:50 Classroom: S3	Module: Recherche_operationnelle_lecture Teacher: Dr. Issadi Time: 08:30 Classroom: Amphi7 <hr/> Module: Distributed_architecture_lecture Teacher: Dr. Djenadi Time: 10:10 Classroom: Amphi7 <hr/> Module: Analyse_numerique_td Teacher: Dr. Alkama Time: 11:50 Classroom: S2	Module: Recherche_operationnelle_lecture Teacher: Dr. Issadi Time: 08:30 Classroom: Amphi7 <hr/> Module: Distributed_architecture_lecture Teacher: Dr. Djenadi Time: 10:10 Classroom: Amphi7 <hr/> Module: Recherche_operationnelle_td Teacher: Dr. Issadi Time: 11:50 Classroom: S3
Wednesday	Module: Reseaux_lecture Teacher: Dr. Zenadji Time: 08:30 Classroom: Amphi7 <hr/> Module: AI_lecture Teacher: Dr. Lekehali Time: 10:10 Classroom: Amphi7 <hr/> Module: Reseaux_td Teacher: Mr. Sahli Time: 11:50 Classroom: S4 <hr/> Module: Reseaux_tp Teacher: Mme. Zaidi Time: 15:10 Classroom: SM1	Module: Reseaux_lecture Teacher: Dr. Zenadji Time: 08:30 Classroom: Amphi7 <hr/> Module: AI_lecture Teacher: Dr. Lekehali Time: 10:10 Classroom: Amphi7 <hr/> Module: Reseaux_td Teacher: Dr. Zenadji Time: 11:50 Classroom: S5 <hr/> Module: AI_td Teacher: Dr. Lekehali Time: 15:10 Classroom: S4	Module: Reseaux_lecture Teacher: Dr. Zenadji Time: 08:30 Classroom: Amphi7 <hr/> Module: AI_lecture Teacher: Dr. Lekehali Time: 10:10 Classroom: Amphi7 <hr/> Module: Reseaux_td Teacher: Mr. Sahli Time: 15:10 Classroom: S3 <hr/> Module: Reseaux_tp Teacher: Mme. Zaidi Time: 11:50 Classroom: SM1	Module: Reseaux_lecture Teacher: Dr. Zenadji Time: 08:30 Classroom: Amphi7 <hr/> Module: AI_lecture Teacher: Dr. Lekehali Time: 10:10 Classroom: Amphi7 <hr/> Module: Reseaux_td Teacher: Mr. Sahli Time: 11:50 Classroom: S3 <hr/> Module: Reseaux_tp Teacher: Mme. Zaidi Time: 15:10 Classroom: SM1
	Module: Distributed_architecture_td Teacher: Dr. Djenadi Time: 15:10 Classroom: S4	Module: Distributed_architecture_td Teacher: Dr. Djenadi Time: 15:10 Classroom: S4	Module: Distributed_architecture_td Teacher: Dr. Djenadi Time: 15:10 Classroom: S4	Module: Distributed_architecture_td Teacher: Dr. Djenadi Time: 15:10 Classroom: S4

Schedule

Group3	Group4	Group5	Group6
ule: Securite_lecture 1er: Dr. Djebari : 08:30 room: Amphi7	Module: Securite_lecture Teacher: Dr. Djebari Time: 08:30 Classroom: Amphi7	Module: Securite_lecture Teacher: Dr. Djebari Time: 08:30 Classroom: Amphi7	Module: Securite_lecture Teacher: Dr. Djebari Time: 08:30 Classroom: Amphi7
ule: odes_formelles_lecture 1er: Dr. Zedek : 10:10 room: Amphi7	Module: Methodes_formelles_lecture Teacher: Dr. Zedek Time: 10:10 Classroom: Amphi7	Module: Methodes_formelles_lecture Teacher: Dr. Zedek Time: 10:10 Classroom: Amphi7	Module: Methodes_formelles_lecture Teacher: Dr. Zedek Time: 10:10 Classroom: Amphi7
ule: Securite_td 1er: Mme. Kassa : 11:50 room: S3	Module: Securite_td Teacher: Mme. Djenane Time: 15:10 Classroom: S1	Module: Securite_td Teacher: Mme. Khelouf Time: 15:10 Classroom: S2	Module: Securite_td Teacher: Mme. Kassa Time: 15:10 Classroom: S3
ule: erche_operationnelle_td 1er: Dr. Issadi : 15:10 room: S6	Module: Methodes_formelles_td Teacher: Dr. Zedek Time: 11:50 Classroom: S4	Module: Analyse_numerique_td Teacher: Dr. Alkama Time: 11:50 Classroom: S5	Module: Recherche_operationnelle_td Teacher: Dr. Issadi Time: 11:50 Classroom: S6
ule: se_numerique_lecture 1er: Dr. Alkama : 08:30 room: Amphi7	Module: Analyse_numerique_lecture Teacher: Dr. Alkama Time: 08:30 Classroom: Amphi7	Module: Analyse_numerique_lecture Teacher: Dr. Alkama Time: 08:30 Classroom: Amphi7	Module: Analyse_numerique_lecture Teacher: Dr. Alkama Time: 08:30 Classroom: Amphi7
ule: Entrepreneuriat_lecture 1er: Dr. Kaci : 10:10 room: Amphi7	Module: Entrepreneuriat_lecture Teacher: Dr. Kaci Time: 10:10 Classroom: Amphi7	Module: Entrepreneuriat_lecture Teacher: Dr. Kaci Time: 10:10 Classroom: Amphi7	Module: Entrepreneuriat_lecture Teacher: Dr. Kaci Time: 10:10 Classroom: Amphi7
ule: Methodes_formelles_td 1er: Dr. Zedek : 15:10 room: S1	Module: Analyse_numerique_td Teacher: Dr. Alkama Time: 15:10 Classroom: S2	Module: Reseaux_td Teacher: Mr. Sahli Time: 11:50 Classroom: S5	Module: Reseaux_td Teacher: Dr. Zenadji Time: 11:50 Classroom: S6
ule: buted_architecture_td 1er: Dr. Djenadi : 11:50 room: S4	Module: Recherche_operationnelle_td Teacher: Dr. Issadi Time: 11:50 Classroom: S3	Module: Reseaux_tp Teacher: Mme. Zaidi Time: 15:10 Classroom: SM1	Module: AI_td Teacher: Dr. Lekehali Time: 15:10 Classroom: S5
ule: erche_operationnelle_lecture 1er: Dr. Issadi : 08:30 room: Amphi7	Module: Recherche_operationnelle_lecture Teacher: Dr. Issadi Time: 08:30 Classroom: Amphi7	Module: Recherche_operationnelle_lecture Teacher: Dr. Issadi Time: 08:30 Classroom: Amphi7	Module: Recherche_operationnelle_lecture Teacher: Dr. Issadi Time: 08:30 Classroom: Amphi7
ule: buted_architecture_lecture 1er: Dr. Djenadi : 10:10 room: Amphi7	Module: Distributed_architecture_lecture Teacher: Dr. Djenadi Time: 10:10 Classroom: Amphi7	Module: Distributed_architecture_lecture Teacher: Dr. Djenadi Time: 10:10 Classroom: Amphi7	Module: Distributed_architecture_lecture Teacher: Dr. Djenadi Time: 10:10 Classroom: Amphi7
ule: Analyse_numerique_td 1er: Dr. Alkama : 11:50 room: S2	Module: Reseaux_td Teacher: Dr. Zenadji Time: 11:50 Classroom: S5	Module: Methodes_formelles_td Teacher: Dr. Zedek Time: 11:50 Classroom: S1	Module: Reseaux_tp Teacher: Mme. Zaidi Time: 11:50 Classroom: SM1
ule: Reseaux_lecture 1er: Dr. Zenadji : 08:30 room: Amphi7	Module: Reseaux_lecture Teacher: Dr. Zenadji Time: 08:30 Classroom: Amphi7	Module: Reseaux_lecture Teacher: Dr. Zenadji Time: 08:30 Classroom: Amphi7	Module: Reseaux_lecture Teacher: Dr. Zenadji Time: 08:30 Classroom: Amphi7
ule: AI_lecture 1er: Dr. Lekehali : 10:10 room: Amphi7	Module: AI_lecture Teacher: Dr. Lekehali Time: 10:10 Classroom: Amphi7	Module: AI_lecture Teacher: Dr. Lekehali Time: 10:10 Classroom: Amphi7	Module: AI_lecture Teacher: Dr. Lekehali Time: 10:10 Classroom: Amphi7
ule: Reseaux_td 1er: Mr. Sahli : 15:10 room: S3	Module: Distributed_architecture_td Teacher: Dr. Djenadi Time: 11:50 Classroom: S3	Module: Recherche_operationnelle_td Teacher: Dr. Issadi Time: 11:50 Classroom: S2	Module: Methodes_formelles_td Teacher: Dr. Zedek Time: 11:50 Classroom: S1
ule: Reseaux_tp 1er: Mme. Zaidi : 11:50 room: SM1	Module: AI_tp Teacher: Mme. Hamma Time: 15:10 Classroom: SM2	Module: Distributed_architecture_td Teacher: Dr. Djenadi Time: 15:10 Classroom: S2	Module: Analyse_numerique_td Teacher: Dr. Alkama Time: 15:10 Classroom: S1
ule: Reseaux_tp	Module: AI_td	Module: AI_td	Module: AI_td Teacher: Dr. Lekehali

Schedule

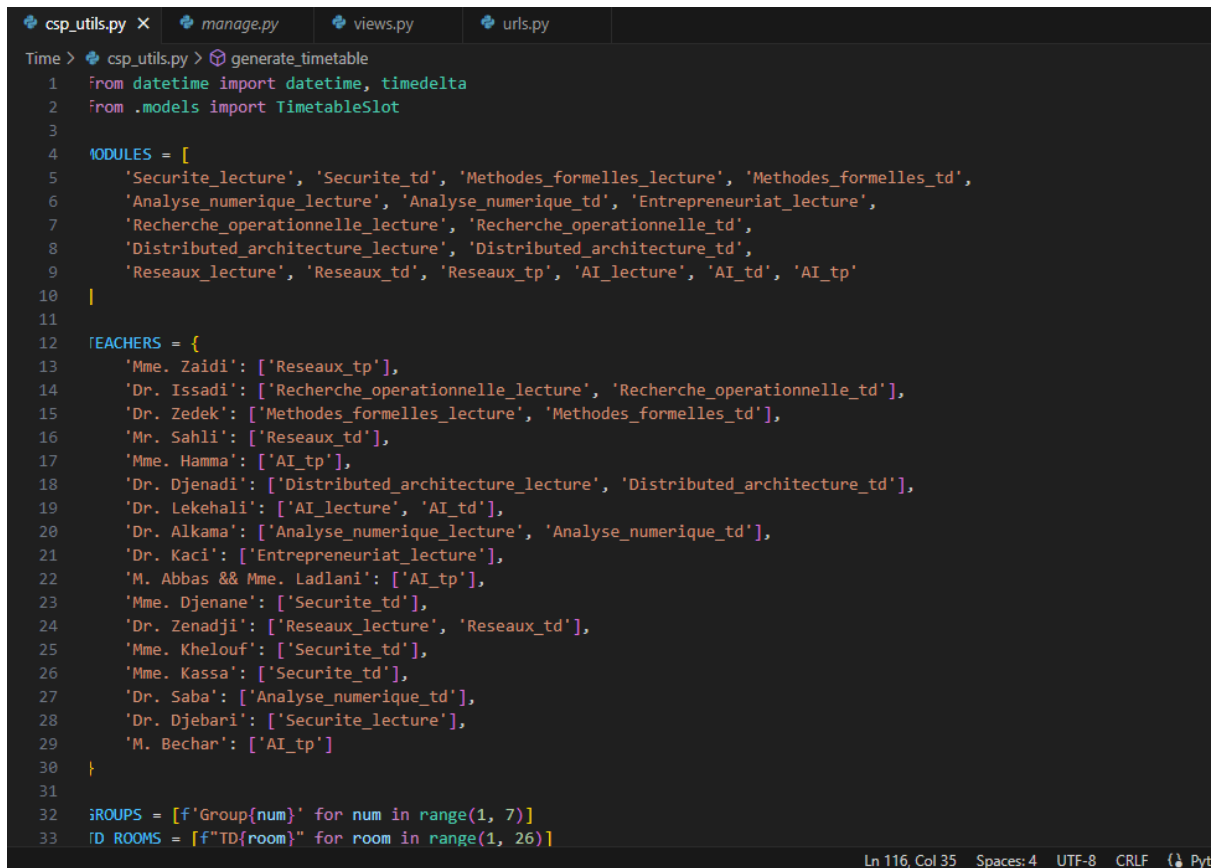
Dr. Lekehali

▼ Select Day ▼

Teacher/Day	Sunday	Monday	Tuesday	Wednesday	Thursday
Dr. Lekehali		Module: AI_td Group: Group6 Time: 15:10 Classroom: S5		Module: AI_lecture Group: Group1 Time: 10:10 Classroom: Amphi7 Module: AI_lecture Group: Group2 Time: 10:10 Classroom: Amphi7 Module: AI_lecture Group: Group3 Time: 10:10 Classroom: Amphi7 Module: AI_lecture Group: Group4 Time: 10:10 Classroom: Amphi7 Module: AI_lecture Group: Group5 Time: 10:10 Classroom: Amphi7 Module: AI_lecture Group: Group6 Time: 10:10 Classroom: Amphi7 Module: AI_td Group: Group2 Time: 15:10 Classroom: S4	Module: AI_td Group: Group1 Time: 08:30 Classroom: S2 Module: AI_td Group: Group3 Time: 10:10 Classroom: S1 Module: AI_td Group: Group4 Time: 11:50 Classroom: S1 Module: AI_td Group: Group5 Time: 15:10 Classroom: S1

Figure 7.6: Application Interface

7.1.2 Code Snippets



```
Time > csp_utils.py > generate_timetable
1 from datetime import datetime, timedelta
2 from .models import TimetableSlot
3
4 MODULES = [
5     'Securite_lecture', 'Securite_td', 'Methodes_formelles_lecture', 'Methodes_formelles_td',
6     'Analyse_numerique_lecture', 'Analyse_numerique_td', 'Entrepreneuriat_lecture',
7     'Recherche_operationnelle_lecture', 'Recherche_operationnelle_td',
8     'Distributed_architecture_lecture', 'Distributed_architecture_td',
9     'Reseaux_lecture', 'Reseaux_td', 'Reseaux_tp', 'AI_lecture', 'AI_td', 'AI_tp'
10 ]
11
12 TEACHERS = {
13     'Mme. Zaidi': ['Reseaux_tp'],
14     'Dr. Issadi': ['Recherche_operationnelle_lecture', 'Recherche_operationnelle_td'],
15     'Dr. Zedek': ['Methodes_formelles_lecture', 'Methodes_formelles_td'],
16     'Mr. Sahli': ['Reseaux_td'],
17     'Mme. Hamma': ['AI_tp'],
18     'Dr. Djenadi': ['Distributed_architecture_lecture', 'Distributed_architecture_td'],
19     'Dr. Lekehali': ['AI_lecture', 'AI_td'],
20     'Dr. Alkama': ['Analyse_numerique_lecture', 'Analyse_numerique_td'],
21     'Dr. Kaci': ['Entrepreneuriat_lecture'],
22     'M. Abbas & Mme. Ladlani': ['AI_tp'],
23     'Mme. Djenane': ['Securite_td'],
24     'Dr. Zenadji': ['Reseaux_lecture', 'Reseaux_td'],
25     'Mme. Khelouf': ['Securite_td'],
26     'Mme. Kassa': ['Securite_td'],
27     'Dr. Saba': ['Analyse_numerique_td'],
28     'Dr. Djebari': ['Securite_lecture'],
29     'M. Bechar': ['AI_tp']
30 }
31
32 GROUPS = [f'Group{num}' for num in range(1, 7)]
33 TD_ROOMS = [f"TD{room}" for room in range(1, 26)]
```

Ln 116, Col 35 Spaces: 4 UTF-8 CRLF (Py)

Figure 7.7: Implementation

```

File Edit Selection View Go Run ...
Backend-class-scheduling-CSP

EXPLORER
BACKEND-CLASS-SCHEDU...
  > env
  > Time
  > __pycache__
  > migrations
  > __pycache__
  > _init_.py
  > 0001_initial.py
  > _init_.py
  > admin.py
  > apps.py
  > csp_utils.py
  > models.py
  > serializers.py
  > tests.py
  > urls.py
  > views.py
  > TimeTable
  > .gitignore
  > db.sqlite3
  > manage.py
  > README.md
  > requirements.txt

csp_utils.py X manage.py views.py urls.py
Time > csp_utils.py > generate_timetable
31
32 GROUPS = [f'Group{num}' for num in range(1, 7)]
33 TD_ROOMS = [f'TD{room}' for room in range(1, 26)]
34 TP_ROOMS = [f'TP{room}' for room in range(1, 12)]
35 LECTURE_ROOMS = [f'Amphi{room}' for room in range(7, 8)]
36 CLASSROOMS = TD_ROOMS + TP_ROOMS + LECTURE_ROOMS
37 DAYS = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday']
38
39 def generate_time_slots():
40     start_time = datetime.strptime("08:30", "%H:%M").time()
41     slots = []
42     for i in range(5): # 5 sessions per day
43         end_time = (datetime.combine(datetime.today(), start_time) + timedelta(minutes=90)).time()
44         slots.append((start_time.strftime("%H:%M"), f'{start_time.strftime("%H:%M")} - {end_time.strftime("%H:%M")}'))
45         start_time = (datetime.combine(datetime.today(), end_time) + timedelta(minutes=10)).time()
46     return slots
47
48 TIME_SLOTS = [slot[0] for slot in generate_time_slots()]
49
50 def generate_timetable(constraints=None):
51     slots = []
52     lecture_modules = [module for module in MODULES if 'lecture' in module]
53     other_modules = [module for module in MODULES if 'lecture' not in module]
54
55     # Track the sessions assigned to each group and teacher
56     sessions_assigned_per_group = {group: [] for group in GROUPS}
57     sessions_assigned_per_teacher = {teacher: [] for teacher in TEACHERS.keys()}
58     lectures_per_day_per_group = {group: {day: 0 for day in DAYS} for group in GROUPS}
59     lectures_per_day_per_teacher = {teacher: {day: 0 for day in DAYS} for teacher in TEACHERS.keys()}
60
61     # Track classroom usage

```

Figure 7.8: Implementation

```

File Edit Selection View Go Run ...
Backend-class-scheduling-CSP

EXPLORER
BACKEND-CLASS-SCHEDU...
  > env
  > Time
  > __pycache__
  > migrations
  > __pycache__
  > _init_.py
  > 0001_initial.py
  > _init_.py
  > admin.py
  > apps.py
  > csp_utils.py
  > models.py
  > serializers.py
  > tests.py
  > urls.py
  > views.py
  > TimeTable
  > .gitignore
  > db.sqlite3
  > manage.py
  > README.md
  > requirements.txt

csp_utils.py X manage.py views.py urls.py
Time > csp_utils.py > generate_timetable
50 def generate_timetable(constraints=None):
51     # Process constraints
52     if constraints:
53         teacher_availability = constraints.get('teacher_availability', {})
54     else:
55         teacher_availability = {}
56
57     def is_teacher_available(teacher, day, slot_index):
58         if teacher in teacher_availability:
59             unavailable_slots = teacher_availability[teacher].get(day, [])
60             if slot_index in unavailable_slots:
61                 return False
62             return True
63
64     # Assign lectures and other sessions
65     for day in DAYS:
66         for slot_index, slot in enumerate(TIME_SLOTS):
67             if day == 'Tuesday' and slot_index >= 3:
68                 break # Stop generating slots after the third session on Tuesday
69
70             # Assign lectures
71             if lecture_modules:
72                 module = lecture_modules.pop(0)
73                 teacher = None
74                 for t, modules in TEACHERS.items():
75                     if module in modules:
76                         teacher = t
77                         break
78             if teacher and is_teacher_available(teacher, day, slot_index):
79                 for classroom in LECTURE_ROOMS:
80                     if classroom not in classroom_usage[day][slot]:
81                         if not any(s[0] == day and s[1] == slot_index for s in sessions_assigned_per_teacher[teacher]):
82                             # Assign lecture to classroom
83                             sessions_assigned_per_teacher[teacher].append((day, slot_index, classroom))
84                             sessions_assigned_per_group[teacher_group].append((day, slot_index, classroom))
85                             lectures_per_day_per_teacher[teacher][day] += 1
86                             lectures_per_day_per_group[teacher_group][day] += 1
87                             lecture_modules.append(module)
88
89     # Assign other sessions
90     for day in DAYS:
91         for slot_index, slot in enumerate(TIME_SLOTS):
92             if day == 'Tuesday' and slot_index >= 3:
93                 break
94             # Assign other sessions
95             for module in other_modules:
96                 # Assign session to classroom
97                 sessions_assigned_per_teacher[teacher].append((day, slot_index, classroom))
98                 sessions_assigned_per_group[teacher_group].append((day, slot_index, classroom))
99                 other_modules.append(module)

```

Figure 7.9: Implementation


```

50 def generate_timetable(constraints=None):
51     # Process constraints
52     if constraints:
53         teacher_availability = constraints.get('teacher_availability', {})
54     else:
55         teacher_availability = {}
56
57     def is_teacher_available(teacher, day, slot_index):
58         if teacher in teacher_availability:
59             unavailable_slots = teacher_availability[teacher].get(day, [])
60             if slot_index in unavailable_slots:
61                 return False
62         return True
63
64     # Assign lectures and other sessions
65     for day in DAYS:
66         for slot_index, slot in enumerate(TIME_SLOTS):
67             if day == 'Tuesday' and slot_index >= 3:
68                 break # Stop generating slots after the third session on Tuesday
69
70             # Assign lectures
71             if lecture_modules:
72                 module = lecture_modules.pop(0)
73                 teacher = None
74                 for t, modules in TEACHERS.items():
75                     if module in modules:
76                         teacher = t
77                     break
78                 if teacher and is_teacher_available(teacher, day, slot_index):
79                     for classroom in LECTURE_ROOMS:
80                         if classroom not in classroom_usage[day][slot]:
81                             if not any(s[0] == day and s[1] == slot_index for s in sessions_assigned_per_teacher[teacher]):

```

Figure 7.10: Implementation

```

94         if not any(s[0] == day and s[1] == slot_index for s in sessions_assigned_per_teacher[teacher]):
95             all(lectures_per_day_per_group[group][day] < 2 for group in GROUPS) and \
96             lectures_per_day_per_teacher[teacher][day] < 2:
97                 slots.append(TimetablesSlot.objects.create(
98                     day=day,
99                     start_time=slot,
100                     module_name=module,
101                     teacher_name=teacher,
102                     group_name=":".join(GROUPS), # All groups
103                     classroom_name=classroom
104                 ))
105                 for group in GROUPS:
106                     sessions_assigned_per_group[group].append((day, slot_index))
107                     lectures_per_day_per_group[group][day] += 1
108                 sessions_assigned_per_teacher[teacher].append((day, slot_index))
109                 lectures_per_day_per_teacher[teacher][day] += 1
110                 classroom_usage[day][slot].append(classroom)
111                 break
112
113     # Assign TD and TP sessions
114     for module in other_modules:
115         if module.endswith('_td') or module.endswith('_tp'):
116             teacher = None
117             for t, modules in TEACHERS.items():
118                 if module in modules:
119                     teacher = t
120                 break
121             if teacher and is_teacher_available(teacher, day, slot_index):
122                 for classroom in (TD_ROOMS if module.endswith('_td') else TP_ROOMS):
123                     if classroom not in classroom_usage[day][slot]:
124                         for group in GROUPS:
125                             # Check if the group already has a session at this time and day

```

Figure 7.11: Implementation

```

Time > csp_utils.py > generate_timetable
50 def generate_timetable(constraints=None):
134     group_name=group,
135     classroom_name=classroom
136 ))
137 sessions_assigned_per_group[group].append((day, slot_index))
138 sessions_assigned_per_teacher[teacher].append((day, slot_index))
139 classroom_usage[day][slot].append(classroom)
140 break
141
142 # Check if a teacher or group has more than three consecutive sessions
143 if slot_index >= 2:
144     for teacher, sessions in sessions_assigned_per_teacher.items():
145         if len(sessions) >= 3 and all((day, i) in sessions for i in range(slot_index - 2, slot_index + 1)):
146             # Remove the last session from the assigned sessions
147             last_session = sessions[-1]
148             slots = [s for s in slots if not (s.day == last_session[0] and s.start_time == TIME_SLOTS[last_session[1]])]
149             for group in GROUPS:
150                 if (last_session[0], last_session[1]) in sessions_assigned_per_group[group]:
151                     sessions_assigned_per_group[group].remove((last_session[0], last_session[1]))
152             sessions_assigned_per_teacher[teacher].remove((last_session[0], last_session[1]))
153
154     for group, sessions in sessions_assigned_per_group.items():
155         if len(sessions) >= 3 and all((day, i) in sessions for i in range(slot_index - 2, slot_index + 1)):
156             # Remove the last session from the assigned sessions
157             last_session = sessions[-1]
158             slots = [s for s in slots if not (s.day == last_session[0] and s.start_time == TIME_SLOTS[last_session[1]])]
159             sessions_assigned_per_group[group].remove((last_session[0], last_session[1]))
160             for teacher in TEACHERS.keys():
161                 if (last_session[0], last_session[1]) in sessions_assigned_per_teacher[teacher]:
162                     sessions_assigned_per_teacher[teacher].remove((last_session[0], last_session[1]))
163
164     return slots

```

Figure 7.12: Implementation

Chapter 8

Conclusion

This assignment successfully demonstrated the application of CSP techniques to real-world scheduling problems. The developed application efficiently generated feasible timetables using backtracking algorithms with AC3 preprocessing and heuristics, ensuring adherence to hard constraints and optimizing for soft constraints.