Automated Timetable Generation for FSTM using Metaheuristic Optimization

Progress Report : Data Processing Phase

Group Members :
Youssef Ait Bahssine
Mustapha Zmirli
Mohamed Bajadi

Module :
Metaheuristics - Master in Artificial Intelligence

31 décembre 2025

Table des matières

1 Introduction and Problematic

The University Timetabling Problem (UTP) is a highly complex combinatorial optimization problem, proven to be NP-complete. At the Faculty of Sciences and Techniques of Marrakech (FSTM), the challenge involves assigning a set of courses, teachers, and student groups to specific time slots and rooms while respecting a diverse set of institutional requirements.

The core problematic of this project lies in the idiosyncratic nature of these constraints ; a **"feasible"** timetable must satisfy all mandatory requirements, while an **"optimal"** one must further minimize the violation of soft preferences to ensure the satisfaction of both students and faculty.

2 Problem Definition

The project objective is to design and implement an automated system that generates a weekly timetable by assigning five primary entities :

- **Courses/Modules :** The academic units to be scheduled.
- **Time Slots :** Available periods within the academic week.
- **Classrooms / Laboratories :** The physical locations for events.
- **Teachers :** The academic staff delivering the modules.
- **Student Groups :** The cohorts attending the sessions.

3 Constraints and Objective Function

Following the taxonomy provided by Lewis (2007), we classify our constraints into two categories :

3.1 Hard Constraints (Feasibility)

These are mandatory conditions. A solution is only valid if zero hard constraints are violated :

- **Teacher Conflict :** A teacher cannot teach two courses simultaneously.
- **Student Group Conflict :** A student group cannot attend more than one course at a time.
- **Room Conflict :** A room cannot host multiple courses at the same time.
- **Room Capacity :** The assigned room must accommodate the group size.
- **Room Type :** Courses requiring labs or amphitheaters must be assigned to appropriate room types.

3.2 Soft Constraints (Quality)

These represent preferences that improve the timetable's utility :

- **Schedule Gaps :** Minimizing idle periods (windows) for both teachers and students.

- **Load Balancing :** Distributing teaching hours evenly across the week.
- **Session Timing :** Minimizing very early or very late sessions.

3.3 Objective Function

We will use a Weighted Cost Function $f(S)$ to evaluate a solution S :

$$f(S) = \sum_i w_H \cdot H_i(S) + \sum_j w_S \cdot S_j(S) \tag{1}$$

where H_i are hard constraint violations, S_j are soft constraint violations, and $w_H \gg w_S$ to ensure feasibility is prioritized.

4 Methodology

Our approach follows these development stages :

1. **Formal Modeling :** This stage involves translating the real-world FSTM timetabling requirements into a rigorous mathematical framework. We define the sets of entities (Teachers, Rooms, Groups) and formulate the constraints as mathematical functions. This provides a clear "blueprint" for the optimization process.
2. **Solution Encoding :** To enable the metaheuristic to search for solutions, we use a *direct representation*. A timetable is encoded as a data structure where each entry represents a specific assignment. This allows the algorithm to efficiently explore the search space.
3. **Constraint Handling :** We adopt a **"One-Stage" strategy**. Instead of solving hard constraints first and soft constraints later, we combine both into a single objective function. By using high penalty weights (w_H) for hard constraints, the algorithm naturally prioritizes feasibility.
4. **Experimental Evaluation :** The final stage involves testing the algorithm using the specific data instance extracted from the FSTM Excel file. Since no external benchmarks are available, this real-world dataset serves as our primary test case.

5 Potential Metaheuristic Algorithms

Based on the survey of state-of-the-art techniques, we evaluate :

- **Simulated Annealing :** Useful for escaping local optima by allowing occasional "worse" moves early in the process.
- **Evolutionary Algorithms (Genetic Algorithms) :** Population-based search to explore large solution spaces.

6 Mathematical Formulation of Constraints

To ensure clarity and facilitate implementation, we formalize the constraints using mathematical notation.

6.1 Notation (Parameters)

- **Sets :** C (Courses), T (Teachers), G (Groups), R (Rooms), S (Slots).
- **Variables :** $x_{c,t,g,r,s} \in \{0,1\}$ (Assignment binary variable).
- **Derived Parameters :**
  - $type_r$: Room type (e.g., 'A' for Amphi, 'S' for Classroom).
  - $type_c$: Required room type (e.g., 'Cours' → Amphi, 'TP' → Lab).

6.2 Hard Constraints Formulas

1. **Teacher Conflict (H_1) :** $\sum_{t\in T}\sum_{s\in S}\max(0, \sum_{c,g,r} x_{c,t,g,r,s} - 1)$
   Ensures no teacher is assigned to multiple courses at the same time.
2. **Group Conflict (H_2) :** $\sum_{g\in G}\sum_{s\in S}\max(0, \sum_{c,t,r} x_{c,t,g,r,s} - 1)$
   Ensures no student group attends multiple sessions simultaneously.
3. **Room Conflict (H_3) :** $\sum_{r\in R}\sum_{s\in S}\max(0, \sum_{c,t,g} x_{c,t,g,r,s} - 1)$
   Ensures physical exclusivity of rooms.
4. **Room Capacity (H_4) :** $\sum_{c,t,g,r,s} x_{c,t,g,r,s} \cdot \max(0, size_g - cap_r)$
   Penalizes if the group size exceeds the room capacity.
5. **Room Type (H_5) :** $\sum_{c,t,g,r,s} x_{c,t,g,r,s} \cdot |type_c - type_r|$
   Ensures sessions are assigned to appropriate room types (e.g., Labs for TPs).

6.3 Soft Constraints Formulas

1. **Schedule Gaps (S_1) :** $\sum_{g\in G}\sum_{d\in Days} \mathrm{Gaps}(g,d)$
   Minimizes idle periods (windows) for student groups.
2. **Load Balancing (S_2) :** $\mathrm{Var}(\sum_{c,g,r,s} x_{c,t,g,r,s} \text{ for each } t)$
   Distributes teaching hours evenly across the faculty.
3. **Session Timing (S_3) :** $\sum_{c,t,g,r,s} x_{c,t,g,r,s} \cdot penalty_s$
   Avoids undesirable early or late slots.

7 Implementation Progress

7.1 Data Extraction Pipeline

We developed a Python-based pipeline using `openpyxl` to process the official Excel occupation file.

- **Room Processing :** Extracted 34 rooms with capacities and types.
- **Assignment Extraction :** Successfully parsed 183 sessions, handling complex merged cells.
- **Time Normalization :** Implemented a 30-minute offset (e.g., 09 :00 → 08 :30).

7.2 Data Structuring

The extracted data is organized into structured CSV files : `assignments.csv`, `rooms.csv`, and `groups.csv`.