University Course Timetabling: From Theory to Real-World Implementation

Extending Naderi's Framework with Real-World Constraints

using Google OR-Tools (SCIP)

Group Project

Dev D. Rabadia (NF1005560),

Catherine C. Calantoc (NF1001422),

Rosario D. Torres (NF1001385),

Miko L. Tan (NF1008647)

Master of Data Analytics, University of Niagara Falls Canada

DAMO-610-3: Winter 2025 Operations Analytics

Professor Hany Osman

March 25, 2025

Chapter 1 - Problem Description

1.1 Introduction

Scheduling and assignment problems are critical in operations analytics, as they ensure optimal resource allocation and system efficiency. One such problem in the service sector is university course timetabling, which involves assigning courses to classrooms and instructors while considering constraints such as faculty availability, classroom capacity, and student preferences. Efficient scheduling in higher education minimizes conflicts, reduces resource wastage, and enhances overall educational experiences for both students and faculty.

1.2 Literature Review

University course scheduling has been extensively researched, with various optimization techniques applied to solve the University Course Timetabling Problem (UCTP)**.** One widely used approach involves mathematical modeling techniques**,** such as Linear Integer Programming (LIP)**,** which formally expresses the constraints and objectives of UCTP. Studies such as Naderi (2016) have proposed LIP models to optimize instructor-course assignments, ensuring feasible and efficient scheduling.

Given that UCTP is NP-hard, researchers have explored heuristic and metaheuristic algorithms to find near-optimal solutions efficiently. Techniques such as Genetic Algorithms (GA), Simulated Annealing (SA), and Variable Neighborhood Search (VNS) have been widely applied in university timetabling problems *(Burke et al., 2007; Lü & Hao, 2010)*. These methods offer flexible and scalable solutions when exact optimization techniques become computationally infeasible for large-scale instances.

More recently, artificial intelligence (AI) techniques have been explored to further automate and improve the timetabling process. Machine learning, constraint programming, and reinforcement learning have been investigated as potential solutions to optimize scheduling by learning from historical data and dynamically adjusting constraints *(Shiau, 2011)***.** These AI-driven methods aim to improve adaptability and efficiency, making them promising approaches for future research in university course scheduling.

1.3 Problem Definition

The University Course Timetabling Problem (UCTP) is a well-known combinatorial optimization problem that requires the allocation of limited resources, such as instructors, classrooms, and time slots, to a set of courses in a way that satisfies various constraints. The goal is to generate an optimal schedule that maximizes faculty preferences, minimizes operational costs, and ensures smooth academic operations.

The UCTP consists of two binary decision dimensions:

- **Instructor Assignment**: Determining which faculty members will teach specific courses.
- **Class Scheduling**: Assigning courses to appropriate classrooms and time slots while avoiding conflicts.

These decisions are subject to multiple constraints:

- **Hard Constraints** (must be strictly satisfied):
    - No instructor can be assigned to two courses simultaneously.
    - No classroom can accommodate multiple courses at the same time.

    - Each course must be assigned to a valid classroom with the required resources.
- **Soft Constraints** (preferences that should be optimized):
    - Maximizing instructor preferences for courses and time slots.
    - Distributing instructor workload fairly.
    - Ensuring students have balanced schedules.

1.4 Challenges Faced

University course timetabling presents numerous challenges that make it a complex optimization problem:

- **Scalability Issues:** Large universities have thousands of courses, instructors, and classrooms, making the scheduling problem computationally expensive *(Naderi, 2016).*
- **Multiple Conflicting Constraints:** Balancing faculty availability, student course preferences, and institutional policies can lead to conflicts that are difficult to resolve *(Burke et al., 2007).*
- **Dynamic Changes:** Last-minute changes, such as faculty unavailability or room reassignments, require quick adaptations without disrupting the entire schedule *(Dimopoulou & Miliotis, 2004).*
- **Fairness and Satisfaction:** Ensuring an equitable distribution of courses among instructors while also considering their preferences adds another layer of complexity *(MirHassani, 2006).*
- **Limited Computational Resources:** Exact optimization approaches may be impractical for large instances, necessitating heuristic or metaheuristic techniques *(Lü & Hao, 2010).*

- **Faculty Availability:** Faculty members have specific availability based on their personal schedules, research commitments, and administrative responsibilities. Ensuring that their teaching schedules do not conflict with these commitments is crucial *(Causmaecker et al., 2009).*
- **Classroom Capacity and Availability:** Classrooms vary in size and facilities and must be assigned based on the specific needs of each course (e.g., lab facilities, seating capacity). The goal is to maximize the utilization of available classroom space while avoiding conflicts *(Boland et al., 2008).*
- **Student Preferences:** Students often have preferences for certain courses and instructors, as well as scheduling preferences to avoid time conflicts with other courses they are enrolled in *(Shiau, 2011).*
- **Curriculum Requirements:** Courses must be scheduled in accordance with the curriculum requirements of various programs, ensuring that students can complete their required courses in a timely manner *(Daskalaki & Birbas, 2005).*
- **Institutional Policies:** These include policies related to workload distribution, fair allocation of teaching responsibilities, and adherence to accreditation standards *(Al-Yakoob & Sherali, 2007).*

1.5 Importance of UCTP in the Service Industry

The University Course Timetabling Problem (UCTP) plays a crucial role in the education sector, which functions as a service industry responsible for delivering knowledge and academic resources to students. Effective scheduling in higher education institutions enhances operational efficiency by optimizing faculty time, classrooms, and institutional infrastructure, ensuring that

academic resources are used effectively *(Naderi, 2016).* Proper scheduling also significantly impacts student satisfaction, as well-structured timetables provide flexible course schedules that align with student preferences and academic needs *(Burke et al., 2007).*

Additionally, a well-balanced timetable promotes faculty work-life balance by distributing teaching responsibilities fairly, reducing burnout, and improving job satisfaction *(Causmaecker et al., 2009).* From an administrative perspective, efficient course scheduling minimizes manual efforts, reduces scheduling errors, and leads to administrative cost reductions, ensuring that institutions allocate resources effectively while avoiding last-minute adjustments *(Boland et al., 2008).*

On the other hand, inefficient scheduling can have significant negative consequences. Resource wastage occurs when classrooms are underutilized, leading to inefficient faculty time allocation and higher operational costs *(Daskalaki & Birbas, 2005).* Additionally, conflicts and overlaps in course schedules can prevent students from enrolling in required courses, delaying their graduation and negatively affecting academic progress *(Shiau, 2011).* Poor scheduling may also result in *faculty and student dissatisfaction*, as inconvenient class times, excessive faculty workloads, and suboptimal learning experiences create frustration and inefficiencies *(Lü & Hao, 2010).* Finally, operational inefficiencies arise when universities must frequently make manual adjustments to resolve scheduling conflicts, diverting administrative resources from other academic priorities *(Al-Yakoob & Sherali, 2007).*

Thus, addressing the UCTP through efficient optimization techniques is essential for ensuring smooth academic operations, reducing conflicts, and enhancing the overall educational experience for all stakeholders.

Chapter 2 - Initial Solution

2.1 Problem Addressed in the Selected Article

The article "Modeling and Scheduling University Course Timetabling Problems" by B. Naderi (2016) presents an integer linear programming (ILP) model for solving the UCTP. The specific problem addressed in Naderi's (2016) article is the optimization of university course timetabling. This involves scheduling a set of courses, lecturers, and classrooms in a way that maximizes the overall preferences of lecturers and the organization while adhering to various constraints. The complexity of this problem arises from the need to balance multiple factors, such as lecturer availability, course requirements, and classroom capacities, to create a feasible and efficient timetable.

2.2.1 Decision Variables

The mathematical model incorporates the following binary decision variables:

- $Z_{l,c,d,r,t} = \begin{cases} 1 \\ 0 \end{cases}$

  1 if lecturer l teaches course c on school day per week d in room r during timeslot per day t, 0 otherwise.

- $X_{l,d} = \begin{cases} 1 \\ 0 \end{cases}$

  1 if lecturer l is scheduled to teach on school day per week d, 0 otherwise.

2.2.2 Objective Function

The model aims to **maximize** the following objective function:

$$Max \quad Z = \sum_{l=1}^{L}\sum_{d=1}^{D} X_{l,d} \cdot p2_{l,d}$$

$$+ \sum_{l=1}^{L}\sum_{c=1}^{C}\sum_{d=1}^{D}\sum_{r=1}^{R}\sum_{t=1}^{T} Z_{l,c,d,r,t} \cdot p1_{l,c} + \sum_{l=1}^{L}\sum_{c=1}^{C}\sum_{d=1}^{D}\sum_{r=1}^{R}\sum_{t=1}^{T} Z_{l,c,d,r,t} \cdot p3_{c,d}$$

where:

L The number of lecturers (professors) C The number of courses

D The number of schooldays (days) per week

R The number of rooms (classrooms)

T The number of timeslots per day

l Index for lecturers where l = {1,2,… ,L}

c Index for courses where c = {1,2,… ,C}

d Index for days where d = {1,2,… ,D}

r Index for rooms where r = {1,2,… ,R}

t Index for timeslots where t = {1,2,… ,T}

$p1_{l,c}$ The preference of lecturer l for teaching course c.

$p2_{l,d}$ The preference of lecturer l for being invited on school day d.

$p3_{c,d}$ The preference of course c for being presented on school day d.

$a1_{l,d}$ Parameter taking value 1 if lecturer l can be invited on school day d, and 0 otherwise.

$a2_{l,c}$ Parameter taking value 1 if lecturer l can teach course c, and 0 otherwise.

$a3_{c,r}$ Parameter taking value 1 if course c can be presented in classroom r, and 0 otherwise.

2.2.3 Constraints

The model is subject to the following constraints:

1. **Constraint 1: One Schedule per Course** - Purpose: Ensures each course is scheduled exactly once across all lecturers, days, rooms, and timeslots.

$$\sum_{l=1}^{L}\sum_{d=1}^{D}\sum_{r=1}^{R}\sum_{t=1}^{T} Z_{l,c,d,r,t} = 1, \qquad \forall c$$

2. **Constraint 2: No Lecture Overlap** - Purpose: Prevents a lecturer from being assigned to multiple courses at the same time (day and timeslot).

$$\sum_{c=1}^{C}\sum_{r=1}^{R} Z_{l,c,d,r,t} \leq X_{l,d} \quad \forall l,d,t$$

3. **Constraint 3: Lecturer Course Feasibility on Scheduled Days** - Purpose: Ensures a lecturer is only scheduled on a day if they can teach at least one course assigned on that day, respecting their course feasibility.

$$X_{l,d} \leq a1_{l,d}, \quad \forall l,d$$

4. **Constraint 4: No Room Overlap** - Purpose: Ensures a room hosts at most one course at any given day and timeslot.

$$\sum_{l=1}^{L} \sum_{c=1}^{C} Z_{l,c,d,r,t} \leq 1, \quad \forall d, r, t$$

5. **Constraint 5: Lecturer -Availability -** Purpose: Limits a lecturer's assignments on a day to their availability (hard constraint).

$$\sum_{d=1}^{D} \sum_{r=1}^{R} \sum_{t=1}^{T} Z_{l,c,d,r,t} \leq a2_{l,c}, \quad \forall l, c$$

6. **Constraint 6: Course-Room Fit** - Purpose: Restricts courses to rooms where they can be held (hard constraint).

$$\sum_{l=1}^{L} \sum_{d=1}^{D} \sum_{t=1}^{T} Z_{l,c,d,r,t} \leq a3_{c,r}, \quad \forall c, r$$

7. **Constraint 7: Binary Constraint for Lecturer Assignment** Purpose: A lecturer assignment is a binary decision.

$$X_{l,d} \in \{0, 1\}, \quad \forall l, d$$

8. **Constraint 8: Binary Constraint for Course Assignment** Purpose: A course assignment is a binary decision.

$$Z_{l,c,d,r,t} \in \{0,1\}, \quad \forall l,c,d,r,t$$

2.3 Solution Approach

The University Course Timetabling Problem (UCTP) is a well-known NP-hard combinatorial optimization problem, making exact methods impractical for large-scale instances. The original article proposes three metaheuristic algorithms to solve such instances:

2.3.1 Imperialist Competitive Algorithm (ICA)

ICA is a population-based metaheuristic inspired by sociopolitical imperialism. It operates as follows:

- A set of elite solutions (imperialists) is selected from the population.
- Weaker solutions (colonies) are attracted to their imperialists and iteratively improved.
- Competition occurs between empires, with stronger imperialists taking over colonies from weaker ones.
- If a colony outperforms its imperialist, they swap roles.

2.3.2 Simulated Annealing (SA)

SA is a local search technique modeled after the physical annealing process:

- It starts from a feasible initial solution.
- Slight modifications are made to explore neighboring solutions.
- Worse solutions can be accepted probabilistically to escape local optima.

- The acceptance probability decreases over time using a cooling schedule.

2.3.3 Variable Neighborhood Search (VNS)

VNS systematically explores multiple neighborhoods to avoid being trapped in local optima:

- Two primary neighborhood structures are used—one for course reassignment and another for rescheduling lecturer invitations.
- Each neighborhood is explored in turn, and the best solution found is used to restart the search.
- The process continues until no further improvements can be found.

2.3.4 Our Implementation: CP-SAT Solver

While the original paper emphasizes metaheuristics, **our approach diverges** by using **exact optimization solvers** for model-based solutions:

- **Base Model:** Implemented using the **CP-SAT solver** from **Google OR-Tools**
    - Designed to handle the rich set of constraints efficiently.
    - Well-suited for small- to medium-scale problems.
    - Fast solution time for feasibility and optimization of hard constraints.

Our solution was implemented using **Google OR-Tools in Python**, utilizing **the CP-SAT Solver** to model and solve the base University Course Timetabling Problem. The following steps summarize our modelling strategy:

1. **Data Loading and Preprocessing** - The model inputs were sourced from an Excel file titled *uctp-base-model-data.xlsx*, which contained separate sheets for Lecturers, Courses, Rooms, Days, Time Periods, and various Preference Tables. These datasets provided the necessary structure for constructing a comprehensive timetabling model. During preprocessing, missing preference values were automatically filled with default constants to ensure completeness and consistency. Specifically, a value of 2 was used to represent a neutral preference, 1 indicated a preferred option, and 0 signified unavailability or incompatibility. This normalization allowed for uniform handling of soft and hard constraints throughout the model.
2. **Index and Parameter Definitions** - The next step involved extracting the key sets and dimensions required for the model. These included the number of lecturers (L), courses (C), rooms (R), days (D), and time periods (T). These sets were derived dynamically based on the contents of the Excel file. To facilitate efficient lookups during constraint formulation, we constructed dictionaries in Python representing the soft constraint preference matrices: p1 for lecturer-course preferences, p2 for lecturer-day preferences, and p3 for course-day preferences. Similarly, we defined three availability mappings (a1, a2, a3) to represent whether a lecturer is available on a given day, qualified to teach a course, or whether a course is compatible with a particular room.
3. **Decision Variable Creation** - Two primary binary decision variables were introduced to model the problem: Z[l,c,d,r,t] and X[l,d]. The variable Z[l,c,d,r,t] equals 1 if lecturer *l* is scheduled to teach course *c* on day *d* in room *r* at time *t*; otherwise, it is 0. This captures the full granularity of a teaching assignment. The auxiliary variable X[l,d] equals 1 if lecturer *l* is scheduled to teach on day *d*, and 0 otherwise. Both variables were created

using model.NewBoolVar(...), enabling the constraint programming engine to reason over binary decision spaces.

4. **Constraint Modelling** - The model incorporated all core hard constraints defined in the mathematical formulation using OR-Tools' AddExactlyOne, AddAtMostOne, and standard logical constraints. These constraints ensure: (1) each course is assigned exactly once across all lecturer-room-day-time combinations; (2) no lecturer is assigned to more than one course during the same time slot; (3) lecturers are only scheduled on days they are available; and (4) courses are only scheduled in rooms that meet their compatibility requirements. These logical conditions collectively guarantee the feasibility of the timetable with respect to institutional policies and physical limitations.
5. **Objective Function** - The optimization goal of the model was to maximize total scheduling satisfaction by incorporating three soft constraint preferences: lecturer-course (p1), lecturer-day (p2), and course-day (p3). This was expressed through a weighted summation of the binary decision variables and their associated preference scores. The objective was implemented using OR-Tools' model.Maximize(...) function, ensuring the solver prioritized high-preference assignments wherever feasible.
6. Solution and Output - Once all variables and constraints were defined, the model was solved using CpSolver(). The solver produced a feasible and optimized timetable that was then exported to uctp-base-model-output.xlsx for analysis and presentation. The final solution achieved an objective score of 105—the maximum preference score possible for the given data instance. Impressively, the solver reached this optimal solution in under 9 seconds on a standard laptop, demonstrating both the scalability and efficiency of the CP-SAT approach for real-world academic scheduling problems.

This approach ensures that the resulting schedule satisfies all constraints while optimizing preferences, with minimal manual intervention.

Chapter 3 - Extended Model Design and Implementation

3.1 Introduction

While the base model successfully addressed fundamental scheduling requirements such as course assignments, instructor feasibility, and room-time allocations, it lacked several operational complexities typically encountered in real-world university environments. In practice, academic institutions must manage hybrid delivery formats, variable session durations, faculty workload contracts, room-type compatibility, and accessibility considerations—factors that are essential for generating implementable academic schedules.

In this chapter, we extend the base model by introducing a series of enhancements rooted in the institutional realities of academic scheduling. These enhancements were inspired by actual practices at the University of Niagara Falls Canada, where course timetabling must balance instructional quality, equity, and logistics across multiple campuses, buildings, and program requirements. Our objective is to evolve the theoretical model into a practical tool that reflects both operational policies and pedagogical priorities.

The extended model remains faithful to the mathematical foundations introduced in Naderi (2016) but introduces a greater level of structural depth and flexibility. By modeling multi-section courses, accommodating variable teaching patterns, enforcing instructor-specific

constraints, and supporting infrastructural constraints such as building transfers and room accessibility, the model is better equipped to serve as a decision-support system in real academic settings. Ultimately, this chapter bridges the gap between theoretical optimization and institutional feasibility, aligning the model more closely with the lived experience of academic administrators, instructors, and students.

3.2 Extended Model Objectives

The primary objective of the extended model is to generate an optimal university course timetable that:

- **Maximizes preference** satisfaction across multiple dimensions (lecturer-course, day, room, location)
- **Ensures the feasibility** of assignments based on hard institutional constraints.
- **Enforces fairness in faculty workload** distribution and session spacing.
- **Incorporates real-world constraints** such as building transitions and contingency room planning.

By integrating these dimensions, the model aims to support decision-making that aligns with both academic goals and operational efficiency.

3.3 Evolution of Methodology

The development of the extended model transitioned from a code-first approach to a structured, math-first methodology. Initially, constraints were directly hard-coded using Google

OR-Tools, which led to long runtimes—averaging 23 minutes per execution—and limited scalability.

To improve performance and maintainability, the team adopted a model-first approach, where constraints were first formulated mathematically before being implemented in Python. This shift enabled cleaner logic, better modularity, and easier debugging. With the help of AI tools for formulation validation and optimization guidance, the model was progressively refined.

As a result, runtime was reduced from 23 minutes to 8 minutes, and finally to 2 minutes in the latest version. This evolution significantly improved both the performance and real-world applicability of the model while preserving constraint integrity.

The extended model is implemented using the **SCIP solver**, a mixed-integer linear programming (MILP) solver accessed through **Google OR-Tools' PyWrapLP interface**. SCIP (Solving Constraint Integer Programs) supports efficient resolution of binary and integer variables under linear constraints, making it well-suited for complex scheduling models like UCTP. The choice of SCIP allows fine-tuning of hard constraints and objective coefficients while scaling to moderately large problem instances. Although the CP-SAT solver was initially considered, the final version utilizes SCIP due to its robustness with linear formulations and integration compatibility with Pywraplp's modeling constructs.

3.4 Extended Model Features and Enhancements

The features introduced in this section represent the practical enhancements implemented in the final version of the model. These are directly aligned with the mathematical constraints defined in Section 3.9. While the list below includes 10 grouped enhancements, certain

features—such as faculty workload control or session frequency patterns—are operationalized through multiple constraints. Conversely, some structural constraints (e.g., room or instructor overlaps) are standard scheduling requirements and are not separately listed as features.

Each feature was developed to ensure the model captures the academic institution's core scheduling requirements while maintaining alignment with the real-world operations of university timetabling systems.

1. **Session Frequency with Pattern Enforcement:** Multi-session courses are assigned either a Monday-Wednesday-Friday (MWF) or Tuesday-Thursday (TTh) schedule, ensuring consistent delivery across the week. This supports standard course delivery patterns and is enforced via pattern assignment constraints.
2. **Fixed Session Times Across Days:** For courses with multiple sessions per week, all instances are scheduled at the same time of day (e.g., 10:00 AM across MWF), aligning with student and instructor expectations for consistency.
3. **Multiple Course Sections:** Courses with high demand may be offered in several sections, each independently scheduled in terms of time, room, and instructor. Sections are treated as unique entities in the model.
4. **Room-Type and Accessibility Assignment:** Courses are restricted to rooms that meet their physical and pedagogical requirements, including accessibility needs (e.g., PWD-compliant rooms) and room type compatibility.
5. **Lecturer Assignment Limits:** Each lecturer has a predefined minimum and maximum number of course sections they may teach, as well as a cap on weekly teaching hours.

6. **Break Requirements Between Sessions:** For lecturers flagged with break preferences, a minimum 30-minute buffer is enforced between sessions to avoid back-to-back assignments.
7. **Single Daily Teaching Location per Lecturer:** Each lecturer is assigned to one location (e.g., campus) per day, minimizing travel between distant locations.
8. **Building Transfer Breaks:** If sessions are assigned to different buildings at the same campus location, a mandatory 30-minute transition period is enforced to allow time for movement.
9. **Reserved Capacity for Contingency:** A fixed percentage (e.g., 10%) of room-time capacity is left unscheduled to support future adjustments or unforeseen circumstances.
10. **Weighted Preference Optimization:** The model prioritizes schedules that maximize lecturer-course fit, preferred teaching days, room conditions, and institutional space usage while penalizing the use of contingency rooms.

3.5 Dataset Structure and Preprocessing

To enable the extended model, the project dataset was modified and expanded using synthetic data representative of real institutional scheduling. These datasets represent the academic elements and operational constraints necessary for scheduling and are pre-processed to transform raw institutional data into model-ready formats.

Three main data files support the modelling process:

- **uctp-extended-model-data-min.xlsx** – This file contains the core input data, structured into multiple sheets:

    - **Lecturers** – Instructor names, employment type, availability, teaching hour caps, minimum and maximum course loads, and break preferences.
    - **Courses** – Course codes, total duration per week, number of sessions, session length, number of sections, and PWD accessibility requirements.
    - **Rooms** – Room IDs, types (e.g., Lecture, Lab), accessibility flags, building IDs, and campus location identifiers.
    - **Days and Time Periods** – A 5-day academic week (Monday to Friday) with 30-minute intervals starting from 08:00 up to 17:00.
- **uctp-extended-model-preprocessed-data.xlsx** – This is a derived file created during preprocessing, where raw data is expanded into:
    - Course sections (e.g., CPSC-500-1, CPSC-500-2)
    - Set indices and mappings required by the model.
    - A flattened structure to simplify constraint iteration.
- **uctp-extended-model-output.xlsx** – This file contains the final optimized schedule in tabular format. It includes columns for Lecturer, Course Section, Day, Room, and Time Block. The solution respects all hard constraints while optimizing based on preferences.

The preprocessing was implemented in Python using the pandas library. It ensures:

- All courses are expanded by the number of sections.
- Time is broken down into uniform 30-minute blocks.
- Binary mapping matrices (e.g., room accessibility compatibility) are constructed for use in constraint programming.
- Default preferences or feasibility flags are applied where data is missing.

This structured data pipeline ensures both scalability and flexibility, allowing the model to be easily updated for future scheduling periods or adapted for larger institutions. The data structure was mapped to an updated schema, and the Entity Relationship Diagram (ERD) was revised accordingly to reflect these additions. The revised ERD is available in Appendix C.

3.6 Indices and Parameters

The following sets and input parameters are used in the formulation of the extended university course timetabling model:

3.6.1 Indices:

- $\boldsymbol{L}$: Set of lecturers.
- $\boldsymbol{CS}$: Set of course sections.
- $\boldsymbol{R}$: Set of rooms.
- $\boldsymbol{D}$: Set of days (Monday to Friday).
- $\boldsymbol{T}$: Set of time periods (30-minute increments).
- $\boldsymbol{P}$: Set of day patterns (0 = MWF, 1 = TTh).
- $\boldsymbol{LOC}$: Set of locations (e.g., UNF NF, UNF Fort Erie).

3.6.2 Parameters:

- $\boldsymbol{FreqD_{cs}}$: Frequency of course section *cs* per week (≥ 1).
- $\boldsymbol{Dur\_T_{cs}}$: Duration of course section *cs* in 30-minute blocks.
- $\boldsymbol{MinC_l, MaxC_l}$ **:** Minimum and maximum number of courses a lecturer *l* must/can teach.
- $\boldsymbol{MaxT_l}$: Maximum total weekly teaching time for lecturer *l*.
- $\boldsymbol{BreakT_l}$: 1 if lecturer *l* requires a 30-minute break between sessions; 0 otherwise.

3.6.3 Preference Score:

- $\boldsymbol{p1_{l,cs}}$: Lecturer–Course preference.
- $\boldsymbol{p2_{l,d}}$: Lecturer–Day preference.
- $\boldsymbol{p3_{cs,d}}$: Course–Day preference.
- $\boldsymbol{p4_{l,loc}}$: Lecturer–Location preference.
- $\boldsymbol{p5_{cs,r}}$: Lecturer–Time of day preference.
- $\boldsymbol{p6_{r}}$: Room condition score.

3.6.4 Binary Compatibility Matrices:

- $\boldsymbol{a1_{l,cs}}$: 1 if lecturer *l* is allowed to teach course *cs*.
- $\boldsymbol{a4}_{cs,r}$: 1 if room *r* matches course type requirements.
- $\boldsymbol{a5}_{cs,r}$: 1 if room *r* meets accessibility (PWD) needs for course *cs*.
- $\boldsymbol{a6}_{r}$: 1 if room *r* is a contingency room.
- $\boldsymbol{a7}_{r}$: 1 if room *r* is available.

3.6.4 Other Sets:

- $\boldsymbol{MWF\_Days}$: Days used for MWF pattern (Mon, Wed, Fri).
- $\boldsymbol{TTh\_Days}$: Days used for TTh pattern (Tue, Thu).
- $\boldsymbol{Building}_{r}$: Building of room r.
- $\boldsymbol{Location}_{r}$: Campus location of room r.
- $\boldsymbol{ReservePct}$: Percentage of room-time slots reserved for contingencies.

3.7 Model Decision Variables

The decision variables represent the core binary choices that the solver must make during optimization. These variables form the foundation of the extended model and are designed to reflect real-world scheduling decisions in a mathematically consistent format.

- $\boldsymbol{Z_{l,cs,d,r,t}} \in \{\mathbf{0}, \mathbf{1}\}$: 1 if lecturer l is assigned to teach course section cs on day d, in room r, at time block t; 0 otherwise. This is the primary variable that defines the full schedule.
- $\boldsymbol{X_{l,d}} \in \{\mathbf{0}, \mathbf{1}\}$: 1 if lecturer l is scheduled to teach on day d; 0 otherwise. Used to manage teaching day preferences and availability.
- $\boldsymbol{Pattern_{cs,p}} \in \{\mathbf{0}, \mathbf{1}\}$: 1 if course section cs follows pattern p, where p=0 represents MWF (Monday-Wednesday-Friday), and p=1 represents TTh (Tuesday-Thursday); 0 otherwise. Ensures consistent weekly delivery for multi-session courses.
- $\boldsymbol{FixedTime_{cs,t}} \in \{\mathbf{0}, \mathbf{1}\}$: 1 if all sessions of course section cs are scheduled at time block t; 0 otherwise.

  Used to enforce consistent timing across different days.
- $\boldsymbol{assign_{l,cs}} \in \{\mathbf{0}, \mathbf{1}\}$: 1 if lecturer l is assigned to course section cs; 0 otherwise.

  Ensures that each course section is assigned to a single instructor.
- $\boldsymbol{teaches_{l,cs}} \in \{\mathbf{0}, \mathbf{1}\}$: 1 if lecturer l teaches course section cs; 0 otherwise.

  Used to enforce workload constraints such as minimum and maximum course load per lecturer.
- $\boldsymbol{uses\_location_{l,d,loc}} \in \{\mathbf{0}, \mathbf{1}\}$: 1 if lecturer l is scheduled to teach at location loc on day d; 0 otherwise. Ensures each lecturer is assigned to only one location per day.

These decision variables serve as the foundation for the constraint formulations and objective function discussed in the following sections.

3.8 Objective Function

The objective of the extended model is to generate a timetable that maximizes preference satisfaction across multiple dimensions while penalizing undesirable assignments such as the use of contingency rooms. The optimization is based on six preference types and a penalty weight, all aggregated over valid teaching assignments.

The objective function is defined as follows:

$$\boldsymbol{Maximize\ Z} = \sum_{\boldsymbol{l \in L, cs \in CS, d \in D, r \in R, t \in T}} \boldsymbol{Z_{l,cs,d,r,t}} \cdot \left(\boldsymbol{p1_{l,cs} + p2_{l,d} + p3_{cs,d} + p4_{l,loc} + p5_{cs,r} + p6_{r}}\right) - \boldsymbol{\beta} \cdot \boldsymbol{a6_{r}}$$

Where:

- $\boldsymbol{Z_{l,cs,d,r,t}}$: Binary assignment variable (lecturer-course-day-room-time).
- $\boldsymbol{p1_{l,cs}}$: Lecturer–Course preference score.
- $\boldsymbol{p2_{l,d}}$ **:** Lecturer–Day preference score.
- $\boldsymbol{p3_{cs,d}}$: Course–Day preference score.
- $\boldsymbol{p4_{l,loc}}$: Course–Room type compatibility score.
- $\boldsymbol{p5_{cs,r}}$: Lecturer–Time of day preference score.
- $\boldsymbol{p6_{r}}$: Room condition quality score.
- $\boldsymbol{a6_{r}}$: Binary flag indicating whether room r is a contingency room.

- β: Penalty weight for using contingency rooms (configurable).

The objective encourages the solver to assign teaching sessions that align with both institutional and instructor preferences, while discouraging the use of suboptimal rooms reserved for contingencies. All preferences and penalties are preprocessed and stored as input dictionaries.

This formulation ensures that the resulting timetable is not only feasible but also optimal in terms of quality and institutional policy alignment.

3.9 Model Constraints

The extended model enforces a comprehensive set of hard constraints to ensure feasibility, fairness, and alignment with real-world university scheduling policies. Each constraint is implemented based on institutional rules and encoded directly in the model using Google OR-Tools CP-SAT. The following list summarizes all constraints with their descriptions and mathematical formulations:

Constraint 1: Course Section Frequency and Day Patterns

$$\sum_{l \in L, d \in D, r \in R, t \in T} Z_{l,cs,d,r,t} = FreqD_{cs} \quad \forall cs \in CS$$

For $FreqD_{cs} > 1$**:**

$$\sum_{p \in P} Pattern_{cs,p} = 1 \quad \forall cs \in CS$$

$$\sum_{l\in L, d\in MWF\_Days, r\in R, t\in T} Z_{l,cs,d,r,t} = FreqD_{cs} \cdot Pattern_{cs,0} \quad \forall cs \in CS$$

$$\sum_{l\in L, d\in TTh\_Days, r\in R, t\in T} Z_{l,cs,d,r,t} = FreqD_{cs} \cdot Pattern_{cs,1} \quad \forall cs \in CS$$

Ensures each course section is scheduled according to its required frequency. Multi-session courses follow either the MWF or TTh day pattern.

Constraint 2: Consistent Time Period

For $FreqD_{cs} > 1$**:**

$$\sum_{t\in T} FixedTime_{cs,t} = 1 \quad \forall cs \in CS$$

$$Z_{l,cs,d,r,t} \leq FixedTime_{cs,t} \quad \forall l \in L, d \in D, r \in R, cs \in CS, t \in T$$

All sessions of a multi-session course must be scheduled at the same time block.

Constraint 3: No Lecturer Overlap

$$\sum_{cs\in CS, r\in R} Z_{l,cs,d,r,t} \leq 1 \quad \forall l \in L, d \in D, t \in T$$

A lecturer cannot be assigned to more than one course section in the same time slot.

Constraint 4: Lecturer-Course Feasibility

$$\sum_{d\in D, r\in R, t\in T} Z_{l,cs,d,r,t} \leq a1_{l,cs} \cdot FreqD_{cs} \quad \forall l \in L, cs \in CS$$

Only qualified lecturers can be assigned to course sections.

Constraint 5: No Room Overlap

$$\sum_{l \in L, cs \in CS} Z_{l,cs,d,r,t} \leq 1 \quad \forall r \in R, d \in D, t \in T$$

Each room can be used for at most one session at a time.

Constraint 6: Lecturer Availability (Max Weekly Teaching Hours)

$$\sum_{cs \in CS, d \in D, r \in R, t \in T} Z_{l,cs,d,r,t} \cdot Dur\_T_{cs} \leq MaxT_l \quad \forall l \in L$$

The total weekly assigned hours per lecturer cannot exceed their teaching load.

Constraint 7: **Course–Room Compatibility**

$$\sum_{l \in L, d \in D, t \in T} Z_{l,cs,d,r,t} \leq a4_{cs,r} \cdot a5_{cs,r} \cdot a7_r \quad \forall cs \in CS, r \in R$$

Courses must only be scheduled in rooms that meet requirements.

Constraint 8: Lecturer Breaks

$$\boldsymbol{Z_{l,cs_1,d,r_1,t_1} + Z_{l,cs_2,d,r_2,t_2} \leq 1 \; \forall l \in L}$$

Where $\boldsymbol{BreakT_l = 1, cs_1, cs_2 \in CS, d \in D, r_1, r_2 \in R, t_1, t_2 \in T}$

Where $\boldsymbol{0 < |t_2 - t_1| < 2}$ ($\boldsymbol{30\ minutes = 1\ time\ period, check\ up\ to\ 2\ periods}$)

Lecturers with a break preference must have at least a 30-minute break between consecutive sessions.

Constraint 9: Workload Limits (Min/Max Course Load)

$$\boldsymbol{MinC_l} \leq \sum_{\boldsymbol{cs \in CS}} \boldsymbol{teaches_{l,cs}} \leq \boldsymbol{MaxC_l} \quad \forall \boldsymbol{l \in L}$$

Where:

$$\boldsymbol{teaches_{l,cs}} \geq \frac{\sum_{\boldsymbol{d \in D, r \in R, t \in T}} \boldsymbol{Z_{l,cs,d,r,t}}}{\boldsymbol{FreqD_{cs}}} \quad \forall \boldsymbol{l \in L, cs \in CS}$$

$$\boldsymbol{teaches_{l,cs}} \leq \sum_{\boldsymbol{d \in D, r \in R, t \in T}} \boldsymbol{Z_{l,cs,d,r,t}} \quad \forall \boldsymbol{l \in L, cs \in CS}$$

Each lecturer must be assigned to a number of course sections within their allowed range.

Constraint 10: Single Teaching Location per Day

$$\sum_{\boldsymbol{loc \in LOC}} \boldsymbol{uses\_location_{l,d,loc}} = \boldsymbol{1} \quad \forall \boldsymbol{l \in L, d \in D}$$

Where:

$$\boldsymbol{loc\_usage_{l,d,loc}} = \sum_{\boldsymbol{r \in R_{loc,}, cs \in CS, t \in T}} \boldsymbol{Z_{l,cs,d,r,t}} \ \forall \boldsymbol{l \in L, d \in D, loc \in LOC}$$

$$\boldsymbol{uses\_location_{l,d,loc}} \cdot (|\boldsymbol{CS}| \cdot |\boldsymbol{T}|) \geq \boldsymbol{loc\_usage_{l,d,loc}} \ \forall \boldsymbol{l \in L, d \in D, loc \in LOC}$$

A lecturer can teach in only one campus location per day.

Constraint 11: Contingency Room Reservation

$$\sum_{l \in L, cs \in CS, d \in D, r \in R, t \in T} Z_{l,cs,d,r,t} \leq (1 - ReservePct) \cdot |R| \cdot |D| \cdot |T|$$

Reserves a percentage of room-time slots for emergencies.

Constraint 12: Building Transfer Break

$$Z_{l,cs_1,d,r_1,t_1} + Z_{l,cs_2,d,r_2,t_2} \leq 1 \quad \forall l \in L, cs_1, cs_2 \in CS, d \in D, r_1, r_2 \in R, t_1, t_2 \in T$$

Where:

$Location_{r1} = Location_{r2}$ (same location)

$Building_{r1} \neq Building_{r2}$ (different buildings)

$0 < |t_2 - t_1| < 2$ $(30 - minute\ gap)$

If sessions are in different buildings on the same campus, insert a 30-minute buffer.

Constraint 13: Single Lecturer per Course Section

$$\sum_{l \in L} assign_{l,cs} = 1 \ \forall cs \in CS$$

Each course section is assigned to exactly one lecturer.

Constraint 14: One Session per Day (for Multi-Session Courses)

$$\sum_{l \in L, r \in R, t \in T} Z_{l,cs,d,r,t} \leq 1 \; \forall cs \in CS \; where \; FreqD_{cs} > 1, d \in D$$

Multi-session courses cannot be scheduled more than once per day.

3.10 Assumptions and Limitations

To ensure tractability and focus during model development, the following assumptions were made:

3.7.1 Model Assumptions:

- **Uniform Lecturer Qualification:** All lecturers are assumed to be qualified to teach all course sections, i.e. $a1_{1,cs} = 1$
- **All Rooms Are Classrooms:** Each room is compatible with any course in terms of room type, i.e. $a4_{cs,r} = 1$
- **Full Room Availability:** All rooms are assumed to be available and not under maintenance or renovation, i.e. $a7_{r} = 1$
- **No Contingency Room Labelling:** Although contingency capacity is reserved in the model, no rooms were specifically marked as contingency; thus, $a6_{r}$ was not applied.
- **Fixed Time Period Duration:** Each time slot represents 30 minutes. Course duration is allocated as $Dur\_T_{cs} = \frac{TotalWeeklyMinutes_{cs}}{30 \cdot FreqD_{cs}}$
- **Default Preferences:** Missing values in preference matrices were set to a default value of 1.

- **Employment Status Use:** Lecturer employment type (Full-Time or Part-Time) was included for reporting only and not used as a scheduling constraint beyond maximum hours.

3.7.2 Model Limitations:

- **Scalability:** The model's performance degrades with larger datasets due to the exponential increase in variables and constraints.
- **Uniform Preference Values:** Most preference inputs were set to 1 or 2, which reduces the granularity and differentiation in optimization.
- **Simplified Room Constraints:** Delivery mode (e.g., in-person, hybrid) and specific pedagogical needs were not enforced beyond generic room type compatibility.
- **No Student Constraints:** The model currently does not account for student course loads or prevent schedule clashes at the student level.
- **Time Resolution:** The use of 30-minute blocks increases the number of variables, which may not be necessary for coarser timetabling requirements.
- **Solver Time Limit:** A 2-minute time cap was applied to ensure fast turnaround. While the model yields feasible solutions, it may not always reach global optimality within the time limit.

These assumptions allowed the project to focus on lecturer-centric feasibility and institutional priorities, while identifying opportunities for refinement in future work.

Chapter 4 – Summary and Future Recommendations

4.1 Summary and Conclusion

This project tackled the University Course Timetabling Problem (UCTP) using an extended, constraint-rich optimization model developed in Python with Google OR-Tools. Building upon the foundational work of Naderi (2016), our extended model incorporated operationally realistic constraints including instructor workload limits, session patterns (MWF/TTh), consistent time slots, room type and accessibility requirements, single-location teaching policies, and building transfer buffers.

The implementation evolved from a code-first prototype to a mathematically structured, AI-assisted solution. Through iterative refinement, model runtime was reduced significantly—from 23 minutes in early versions to just 2 minutes in the final model. The integration of preference-based soft constraints ensured that the resulting timetable was not only feasible but also optimized for quality and institutional alignment.

The project demonstrates that real-world academic timetabling can be effectively modelled as a mixed constraint satisfaction and optimization problem. The resulting solution is scalable, adaptable, and capable of supporting decision-making in higher education scheduling.

4.2 Future Work & Recommendations

To further enhance the model's applicability and performance, we recommend the following extensions and areas of research:

1. Student-Centric Constraints

Incorporate student enrolment data to avoid scheduling conflicts and improve satisfaction. Adding student-level constraints would enhance the realism of the model and broaden its institutional value.

2. Dynamic Time Slot Flexibility

Introduce variable-length time blocks (e.g., 1-hour sessions) to reduce complexity and align with actual academic schedules.

3. Advanced Preference Weighting

Refine the objective function by assigning customizable weights to different preference types (e.g., room quality vs. time-of-day) to better reflect institutional priorities.

4. Heuristic Initialization

Use a greedy algorithm to generate an initial feasible solution, which could help the solver converge faster and improve quality under strict time limits.

5. Parallel and Modular Solving

Decompose the model by day, location, or department and solve sub-problems in parallel before integrating results with conflict resolution logic.

6. Room Type Enforcement

Extend room compatibility rules to enforce specific room types based on course delivery mode (e.g., lab, seminar, online ready).

7. Solver Diagnostics and UI

Incorporate solver statistics (e.g., branches explored, time per constraint) and build a user interface for easier data input, scenario testing, and visualization of results.

8. Real-World Pilot Testing

Apply the model to actual institutional data from the University of Niagara Falls Canada to validate effectiveness, refine assumptions, and engage stakeholders.

These recommendations represent the next step in transforming the model from an academic exercise into a robust, institution-ready scheduling solution. By addressing these enhancements, the model can evolve into a comprehensive decision-support tool capable of meeting the dynamic and complex demands of real-world university operations.

References

Naderi, B. (2016). "*Modeling and Scheduling University Course Timetabling Problems*." International Journal of Research in Industrial Engineering, 5(1–4), 1–15.

Burke, E. K., Elliman, D. G., & Weare, R. F. (2004). *A genetic algorithm-based university timetabling system*. In *Proceedings of the 2nd International Conference on the Practice and Theory of Automated Timetabling*.

Burke, E.K., McCollum, B., Meisels, A., Petrovic, S., & Qu, R. (2007). "A Graph-Based Hyper-Heuristic for Educational Timetabling Problems." *European Journal of Operational Research*, 176, 177-192.

Lü, Z., & Hao, J.-K. (2010). *Adaptive Tabu Search for course timetabling*. *European Journal of Operational Research*, 200(1), 235–244.

Shiau, D.F. (2011). "A Hybrid Particle Swarm Optimization for a University Course Scheduling Problem with Flexible Preferences." *Expert Systems with Applications*, 38, 235-248.

Appendix A. Python Script Implemented in the Project

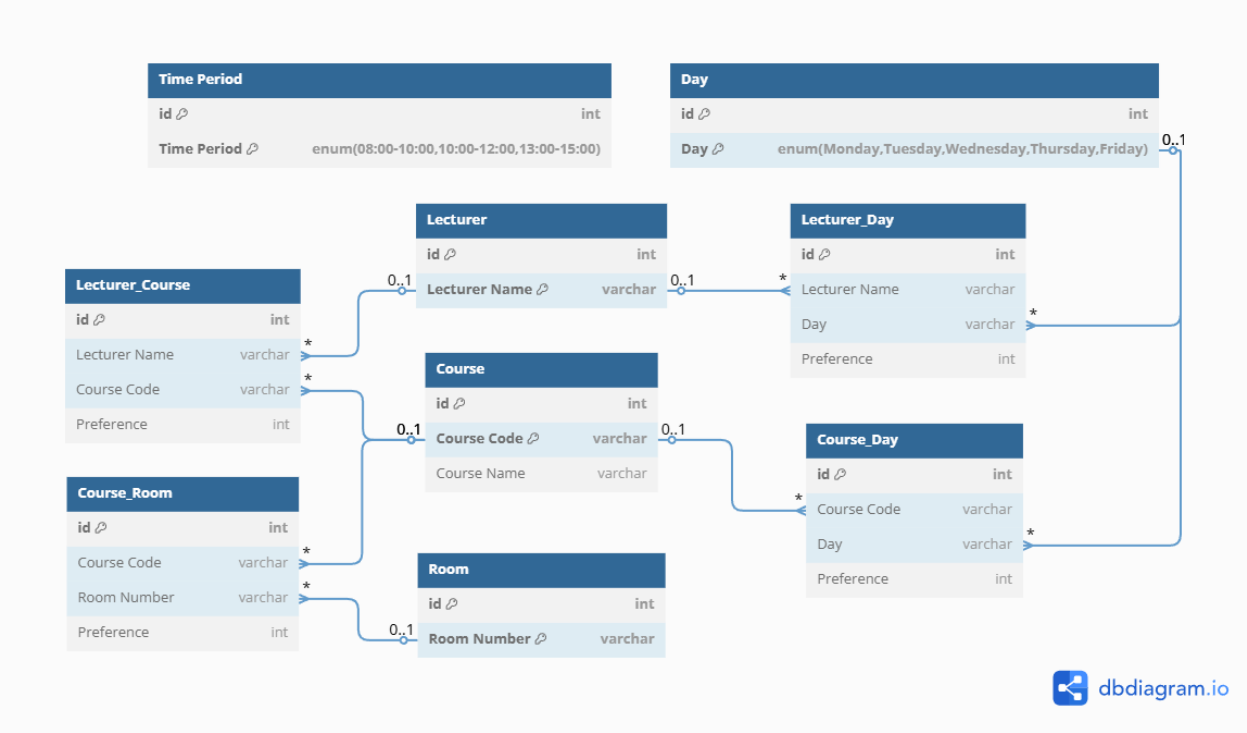
Base Model:

See uctp-base-model_v4.ipynb Python script submission for full implementation using Google OR-Tools CP-SAT Solver.

Extended Model:

See uctp-extended-model_v5.ipynb Python script submission for full implementation using Google OR-Tools with the SCIP MILP Solver (via Pywraplp).

Appendix B-1. ERD of the Base Model



Appendix B-2. ERD of the Extended Model

Appendix C. Individual Contribution and AI Usage Report

Each member should describe their contribution and specify any AI tools used, such as ChatGPT, Deepseek, Copilot, code generation tools, or data analysis assistants. Provide details on how AI supported the work, such as summarization, debugging, ideation, or automation.

Student Name	Role/Responsibility	Contribution Details	Use of AI Tools (if applicable)
Miko Tan	Team Lead / Code Development	Coded the mathematical model using Google OR-Tools	Used AI tools to get coding suggestions, improve optimization logic, and debug complex parts of the model when necessary. Manual validation and implementation remained the core approach.
Dev Rabadia	Web Developer/ Quality Assurance	Web Development/ Researcher	Consulted AI tools to explore best practices in UI/UX and conduct supplementary research on university scheduling systems. All development decisions were made based on manual testing and team feedback.
Catherine Calantoc	Quality Assurance/ Technical Writer	Testing the Python Code and Documentation	Used AI tools to gather relevant background material and enhance clarity in documentation. AI-assisted queries

			were cross-verified with course materials and project requirements.
Rosario Torres	Quality Assurance/ Technical Writer	Testing the Python Code and Documentation	Leveraged AI tools to streamline phrasing and formatting in documentation. Some AI-generated suggestions were selectively adopted after manual review for relevance and accuracy.