Class Scheduling & Instructor Assignment in Higher Education

A Group Project for DAMO-610-3: Operations Analytics (Winter 2025)

Presenters:

1. Catherine Calantoc (NF1001422)
2. Dev Rabadia (NF1005560)
3. Miko Tan (NF1008647)
4. Rosario Torres (NF1001385)

University of Niagara Falls Canada

Agenda

- Project Overview
- University Course Lifecycle
- Model Development Process
- Base Model
- Extended Model
- Model Comparison
- Simulation
- Conclusion
- Q&A

Project Overview

Scheduling and assignment optimize resources, reduce inefficiencies, and improve performance by managing personnel, equipment, and time.

Optimization models and analytics techniques lead to **cost reduction**, **better service quality**, and **increased productivity**.

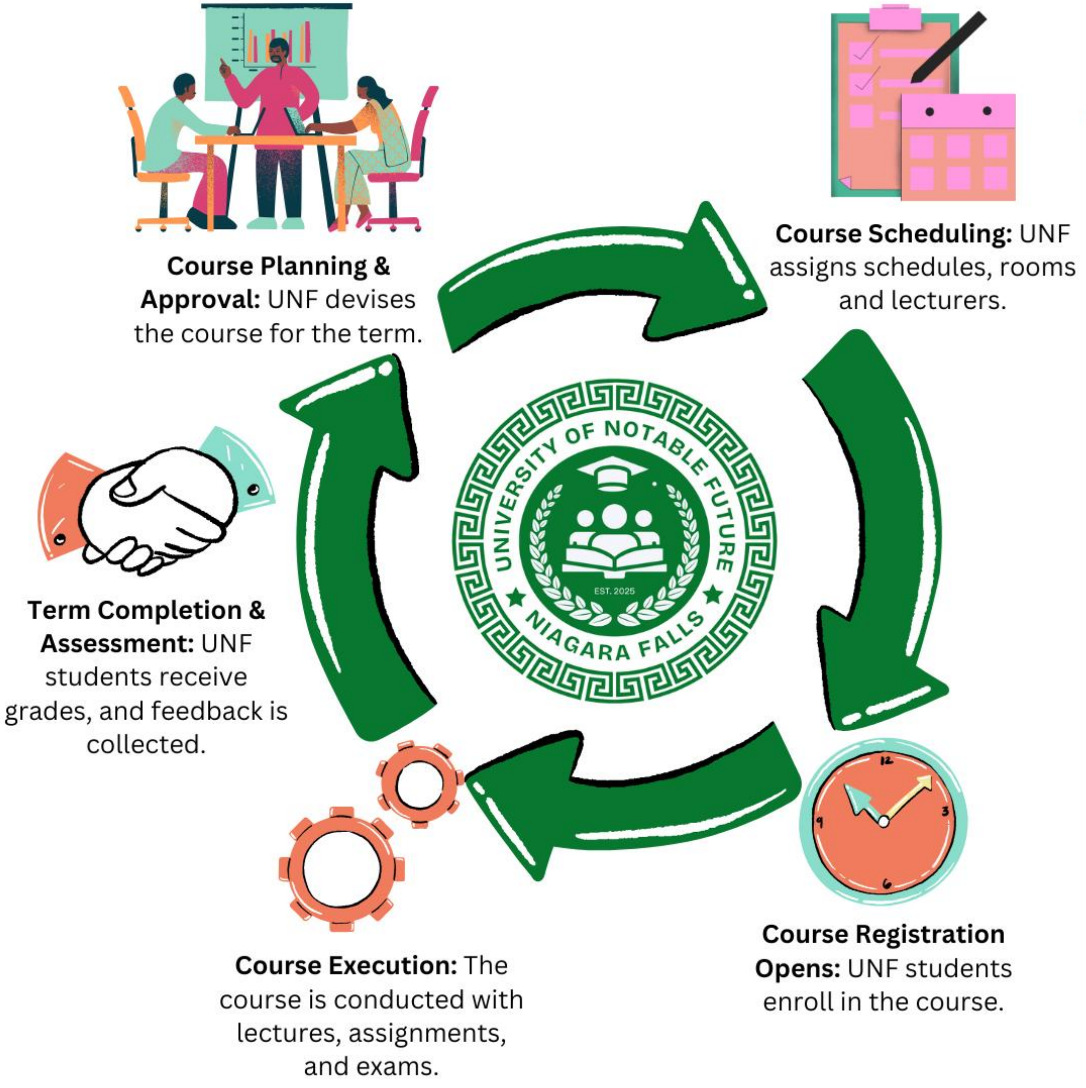
Project Objectives

1. Define the problem and its significance to the relevant service industry.
2. Implement the proposed mathematical model using Google OR-Tools and solve it with synthetic datasets.
3. Extend the model by incorporating additional business objectives and applying it to new synthetic datasets.

Class Scheduling & Instructor Assignment Problem

- **Problems Faced:** Balancing faculty availability, classroom capacity, student preferences, and the curriculum in class scheduling and instructor assignment.
- **Challenges:** Inefficient scheduling leads to wasted resources, conflicts, and dissatisfaction among students and instructors.
- **Expected Benefits:** An optimal scheduling system can maximize faculty preferences, minimize operational costs, and ensure smooth academic operations.

University Course Lifecycle



Course Scheduling Lifecycle

- [x] Collect Course Requirements
- [x] Assign Lecturers
- [x] Allocate Rooms
- [x] Define Time Periods and Resolve Conflicts
- [x] Generate & Validate Timetable
- [x] Finalize & Publish Schedule
- [ ]
- [ ]

Model Development Process

Dataset Preparation

1. Define Entities and Attributes
2. Define Relationships
3. Mock Test Data

Data Definition

1. Load Dataset
2. Data Preprocessing
3. Build Course Sections *(for extended model only)*
4. Output Preprocessed Data *(for extended model only)*

Model Definition

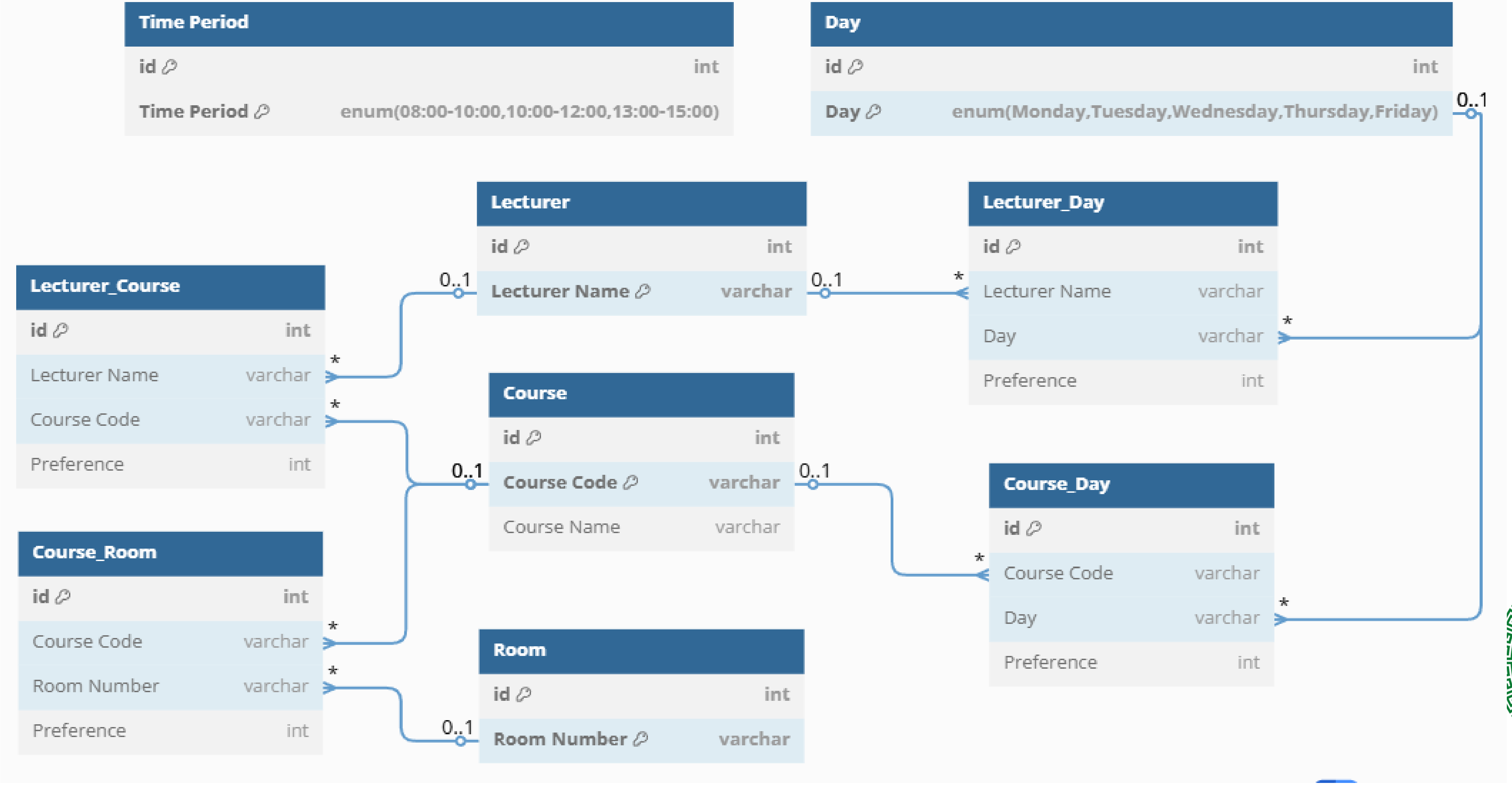
1. Define Indices, Parameters and Decision Variables
2. Define Constraints
3. Define Objective Function

Model Evaluation

1. Run the Solver
2. Display Results
3. Output Timetable Data
4. Validate Output

Base Model: ERD

Class Scheduling & Instructor Assignment Problem



Base Model: Variables

Class Scheduling & Instructor Assignment Problem

Indices

L: The number of lecturers (professors)
C: The number of courses
R: The number of rooms (classrooms)
D: The number of schooldays (days) per week
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}
r: Index for rooms where r={1,2,… ,R}
d: Index for school days d={1,2,… ,D}
t: Index for timeslots where t={1,2,…,T}

Parameters

$p1_{l,c}$: The preference of lecturer l for teaching course c.
$p2_{l,d}$: The preference of lecturer l for being invited on schoolday d.
$p3_{c,d}$: The preference of course c for being presented on schoolday d.
$a1_{l,d}$: Parameter taking value 1 if lecturer l can be invited on schoolday 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.

Decision Variables

$$Z_{l,c,d,r,t} = \begin{cases} 1 & \text{if lecturer } l \text{ teaches course } c \text{ on school day per week } d \text{ in room } r \text{ during timeslot per day } t, \\ 0 & \text{otherwise.} \end{cases}$$

$$X_{l,d} = \begin{cases} 1 & \text{if lecturer } l \text{ is scheduled to teach on school day per week } d, \\ 0 & \text{otherwise.} \end{cases}$$

Base Model: Objective Function

Class Scheduling & Instructor Assignment Problem

$$\text{Max } Z = \sum_{l=1}^{L}\sum_{d=1}^{D} X_{l,d}p2_{l,d} + \sum_{l=1}^{L}\sum_{c=1}^{C}\sum_{d=1}^{D}\sum_{r=1}^{R}\sum_{t=1}^{T} Z_{l,c,d,r,t}p1_{l,c} + \sum_{l=1}^{L}\sum_{c=1}^{C}\sum_{d=1}^{D}\sum_{r=1}^{R}\sum_{t=1}^{T} Z_{l,c,d,r,t}p3_{c,d}$$

Subject To:

$$\sum_{l=1}^{L}\sum_{d=1}^{D}\sum_{r=1}^{R}\sum_{t=1}^{T} Z_{l,c,d,r,t} = 1, \quad \forall c \quad \text{(One Schedule per Course)}$$

$$\sum_{c=1}^{C}\sum_{r=1}^{R} Z_{l,c,d,r,t} \leq X_{l,d}, \quad \forall l,d,t \quad \text{(No Lecturer Overlap)}$$

$$X_{l,d} \leq a1_{l,d}, \quad \forall l,d \quad \text{(Lecturer-Day Feasibility)}$$

$$\sum_{l=1}^{L}\sum_{c=1}^{C} Z_{l,c,d,r,t} \leq 1, \quad \forall d,r,t \quad \text{(No Room Overlap)}$$

$$\sum_{d=1}^{D}\sum_{r=1}^{R}\sum_{t=1}^{3} Z_{l,c,d,r,t} \leq a2_{l,c}, \quad \forall l,c \quad \text{(Lecturer-Course Feasibility)}$$

$$\sum_{l=1}^{L}\sum_{d=1}^{D}\sum_{t=1}^{3} Z_{l,c,d,r,t} \leq a3_{c,r}, \quad \forall c,r \quad \text{(Course-Room Fit)}$$

$$X_{l,d} \in \{0,1\}, \quad \forall l,d \quad \text{(Binary Constraint for Lecturer Assignment)}$$

$$Z_{l,c,d,r,t} \in \{0,1\}, \quad \forall l,c,d,r,t \quad \text{(Binary Constraint for Course Assignment)}$$

Base Model: Solution Analysis (1 of 4)

Class Scheduling & Instructor Assignment Problem

Key Components	Purpose	Code Snippet
$Max\ Z = \sum_{l=1}^{L}\sum_{c=1}^{C}\sum_{d=1}^{D}\sum_{r=1}^{R}\sum_{t=1}^{T} Z_{l,c,d,r,t} \cdot p1_{l,c} + \sum_{l=1}^{L}\sum_{d=1}^{D} X_{l,d} \cdot p2_{l,d} + \sum_{l=1}^{L}\sum_{c=1}^{C}\sum_{d=1}^{D}\sum_{r=1}^{R}\sum_{t=1}^{T} Z_{l,c,d,r,t} \cdot p3_{c,d}$		

$X_{l,d} \cdot p2_{l,d}$	Maximize lecturer-day preferences. **X** refers to the Lecturer Day assignment, while **p2** refers to the Day preference of Lecturer.	`model.Maximize(`
		`sum(X[l, d] * p2[l, d] for l in dfs["Lecturer"]["Lecturer Name"] for d in dfs["Day"]["Day"]) +`
		`sum(Z[l, c, d, r, t] * p1[l, c]`
		`for l in dfs["Lecturer"]["Lecturer Name"] for c in dfs["Course"]["Course Code"]`
		`for d in dfs["Day"]["Day"] for r in dfs["Room"]["Room Number"] for t in dfs["Time Period"]["Time Period"]) +`
		`sum(Z[l, c, d, r, t] * p3[c, d]`
		`for l in dfs["Lecturer"]["Lecturer Name"] for c in dfs["Course"]["Course Code"]`
		`for d in dfs["Day"]["Day"] for r in dfs["Room"]["Room Number"] for t in dfs["Time Period"]["Time Period"])`
		`)`

$Z_{l,c,d,r,t} \cdot p1_{l,c}$	Maximize lecturer-course preferences. **Z** refers to the timetable assignment, while **p1** refers to the Course preference of Lecturer.	
$Z_{l,c,d,r,t} \cdot p3_{c,d}$	Maximize course-day preferences. **Z** refers to the timetable assignment, while **p3** refers to the Day preference for the Course.	

Base Model: Solution Analysis (2 of 4)

Class Scheduling & Instructor Assignment Problem

Key Components	Purpose	Code Snippet

Constraint #1: One Schedule per Course	Each course gets exactly one slot. Ensures every course contributes to the score once	`for c in dfs["Course"]["Course Code"]:`
		`    model.AddExactlyOne(Z[l, c, d, r, t]`
		`        for l in dfs["Lecturer"]["Lecturer Name"] for d in dfs["Day"]["Day"]`
		`        for r in dfs["Room"]["Room Number"] for t in dfs["Time Period"]["Time Period"])`

$$\sum_{l=1}^{L}\sum_{d=1}^{D}\sum_{r=1}^{R}\sum_{t=1}^{T} Z_{l,c,d,r,t} = 1, \quad \forall c \quad \text{(One Schedule per Course)}$$		

Constraint #2: No Lecturer Overlap	Lecturers can't teach two courses at once. Limits **X** (Lecturer Day Schedule) and **Z** (Timetabling) to feasible schedules.	`for l in dfs["Lecturer"]["Lecturer Name"]:`
		`    for d in dfs["Day"]["Day"]:`
		`        # Link X_{l,d} to Z: X_{l,d} = 1 if any Z_{l,c,d,r,t} = 1`
		`        model.Add(sum(Z[l, c, d, r, t] for c in dfs["Course"]["Course Code"]`
		`            for r in dfs["Room"]["Room Number"] for t in dfs["Time Period"]["Time Period"]) <= X[l, d] * C)`
		`        # No overlap: At most one course per timeslot`
		`        for t in dfs["Time Period"]["Time Period"]:`
		`            model.AddAtMostOne(Z[l, c, d, r, t]`
		`                for c in dfs["Course"]["Course Code"] for r in dfs["Room"]["Room Number"])`

$$\sum_{c=1}^{C}\sum_{r=1}^{R} Z_{l,c,d,r,t} \le X_{l,d}, \quad \forall l, d, t \quad \text{(No Lecturer Overlap)}$$		

Base Model: Solution Analysis (3 of 4)

Class Scheduling & Instructor Assignment Problem

Key Components	Purpose	Code Snippet

Constraint #3: Lecturer-Day Feasibility	Lecturers only work when available. Ensures **X** (Lecturer Day Schedule) aligns with **p2** (Lecturer Day Preference) boosting score.	`for l in dfs["Lecturer"]["Lecturer Name"]:`
		`    for d in dfs["Day"]["Day"]:`
		`        model.Add(X[l, d] <= a1.get((l, d), 0))`

$X_{l,d} \leq a1_{l,d}, \quad \forall l,d \quad \text{(Lecturer-Day Feasibility)}$		

Constraint #4: No Room Overlap	One course per room at a time. Ensures **Z** (Timetabling) assignments are practical.	`for r in dfs["Room"]["Room Number"]:`
		`    for d in dfs["Day"]["Day"]:`
		`        for t in dfs["Time Period"]["Time Period"]:`
		`            model.AddAtMostOne(Z[l, c, d, r, t]`
		`                                 for l in dfs["Lecturer"]["Lecturer Name"] for c in`
		`dfs["Course"]["Course Code"])`

$\sum_{l=1}^{L}\sum_{c=1}^{C} Z_{l,c,d,r,t} \leq 1, \quad \forall d,r,t \quad \text{(No Room Overlap)}$		

Constraint #5: Lecturer-Course Feasibility	Lecturers only teach courses they can. Links **Z** (Timetabling) to **p1** (Lecturer Preference) enhancing score.	`for l in dfs["Lecturer"]["Lecturer Name"]:`
		`    for c in dfs["Course"]["Course Code"]:`
		`        model.Add(sum(Z[l, c, d, r, t] for d in dfs["Day"]["Day"]`
		`                      for r in dfs["Room"]["Room Number"] for t in dfs["Time`
		`Period"]["Time Period"]) <= a2.get((l, c), 0))`

$\sum_{d=1}^{D}\sum_{r=1}^{R}\sum_{t=1}^{3} Z_{l,c,d,r,t} \leq a2_{l,c}, \quad \forall l,c \quad \text{(Lecturer-Course Feasibility)}$		

Base Model: Solution Analysis (4 of 4)

Class Scheduling & Instructor Assignment Problem

Key Components	Purpose	Code Snippet

Constraint #6: Course-Room Fit	Courses only in suitable rooms. Ensures **Z** (Timetabling) fits room needs, aiding score.	`for c in dfs["Course"]["Course Code"]:`
		`    for r in dfs["Room"]["Room Number"]:`
		`        model.Add(sum(Z[l, c, d, r, t] for l in dfs["Lecturer"]["Lecturer Name"]`
		`                    for d in dfs["Day"]["Day"] for t in dfs["Time`
		`Period"]["Time Period"]) <= a3.get((c, r), 0))`

$\sum_{l=1}^{L}\sum_{d=1}^{D}\sum_{t=1}^{3} Z_{l,c,d,r,t} \leq a3_{c,r}, \quad \forall c, r \quad \text{(Course-Room Fit)}$		

Base Model: Results

Class Scheduling & Instructor Assignment Problem

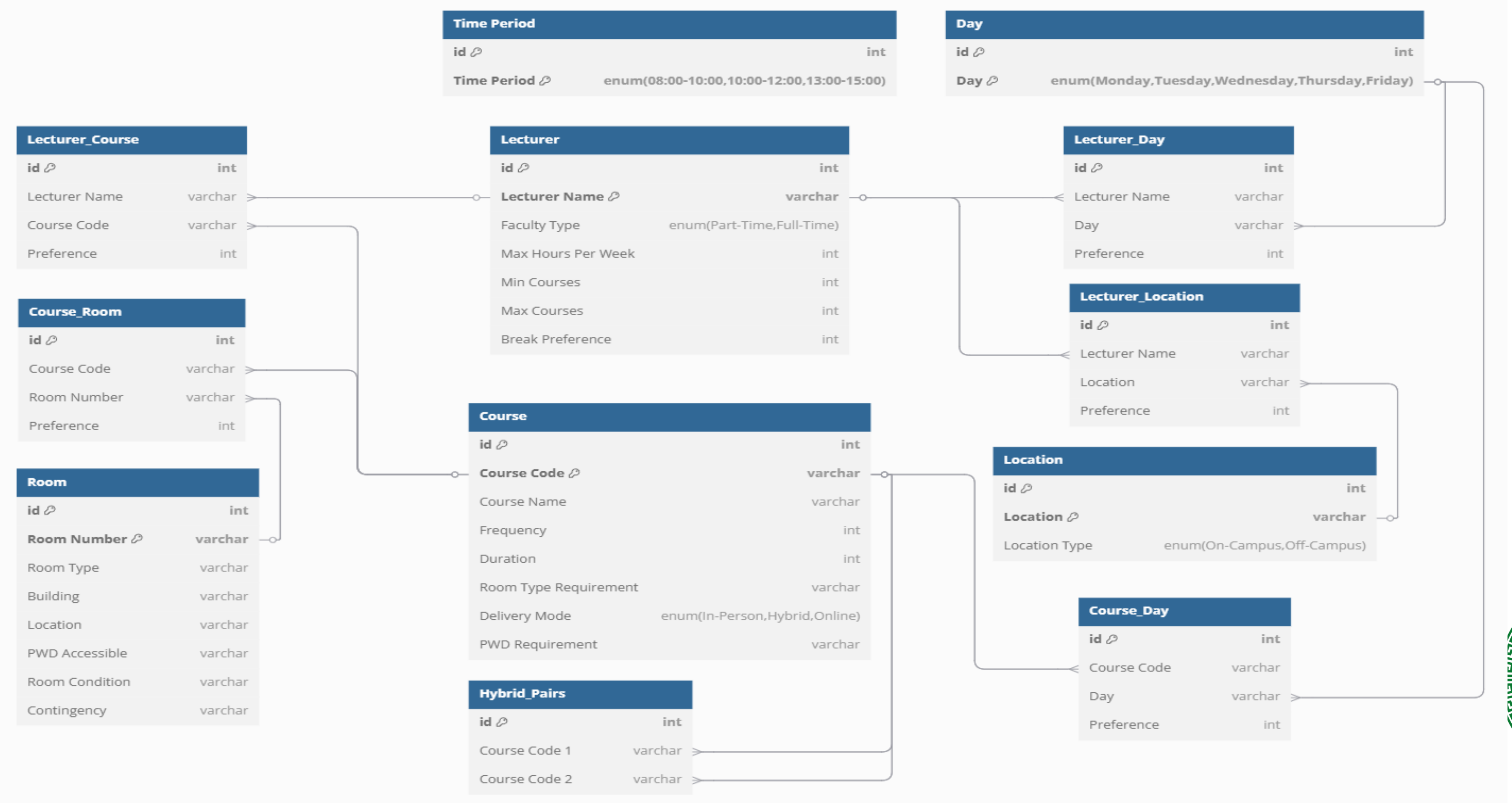
Lecturer	Course	Day	Room	Timeslot
Abbas Hamze	DAMO-500	Monday	UNF 225	13:00-15:00
Abbas Hamze	DAMO-511	Monday	UNF 215	08:00-10:00
Abbas Hamze	DAMO-600	Wednesday	UNF 235	13:00-15:00
Abbas Hamze	DAMO-610	Tuesday	UNF 235	10:00-12:00
Abbas Hamze	DAMO-611	Thursday	UNF 403	10:00-12:00
Abbas Hamze	DAMO-621	Thursday	UNF 235	13:00-15:00
Abbas Hamze	DAMO-622	Tuesday	UNF 230	13:00-15:00
Abbas Hamze	DAMO-623	Thursday	UNF 235	08:00-10:00
Abbas Hamze	CPSC-500	Friday	UNF 240	10:00-12:00
Abbas Hamze	CPSC-510	Friday	UNF 403	08:00-10:00
Abbas Hamze	CPSC-600	Wednesday	UNF 235	10:00-12:00
Abbas Hamze	CPSC-610	Friday	UNF 245	13:00-15:00
Ahmed Eltahawi	DAMO-510	Tuesday	UNF 215	08:00-10:00
Mohannad Al Mousa	DAMO-501	Wednesday	UNF 245	10:00-12:00
Zeeshan Ahmad	CPSC-620	Friday	UNF 403	13:00-15:00

Key Insights

- **Optimal Schedule:** CP-SAT maximized, scheduling 15 courses across 5 days, 7 rooms, and 3 timeslots with no conflicts.
- **Constraints Satisfied:** One slot per course, no lecturer/room overlaps, full feasibility in availability and suitability.
- **Lecturer Workload:** Abbas Hamze teaches 12 courses; others 1 each—balanced yet concentrated.
- **Room Efficiency:** Full timeslot/room use (e.g., UNF 235: 5 slots), effective resource allocation.

Extended Model: ERD

Class Scheduling & Instructor Assignment Problem



Extended Model: Variables

Class Scheduling & Instructor Assignment Problem

Indices

- ***L***: Set of lecturers.
- ***CS***: Set of course sections.
- ***R***: Set of rooms.
- ***D***: Set of days (Monday, Tuesday, Wednesday, Thursday, Friday).
- ***T***: Set of time periods (30-minute increments).
- ***P***: Set of day patterns (0 for MWF, 1 for TTh).
- ***LOC***: Set of locations (e.g., UNF NF, UNF Fort Erie).

Parameters

- ***FreqD[cs]***: Frequency of course section cs per week (1 or >1).
- ***DurT[cs]***: Duration of course section cs in time units (30-minute increments).
- ***MinC[l]***: Minimum number of courses lecturer l must teach.
- ***MaxC[l]***: Maximum number of courses lecturer l can teach.
- ***MaxT[l]***: Maximum teaching hours per week for lecturer l.
- ***BreakT[l]***: Binary indicator (0 or 1) for whether lecturer l requires a 30-minute break between all sessions (optional, separate from building transfers).
- ***p1[l, cs]***: Preference score for lecturer l teaching course section cs.
- ***p2[l, d]***: Preference score for lecturer l teaching on day d.
- ***p3[cs, d]***: Preference score for course section cs being scheduled on day d.
- ***p4[l, loc]***: Preference score for lecturer l teaching at location loc.
- ***p5[cs, r]***: Preference score for course section cs being held in room r.
- ***p6[r]***: Room condition score for room r.
- ***a1[l, cs]***: Binary indicator if lecturer l is qualified to teach course section cs.
- ***a4[cs, r]***: Binary indicator if room r meets the type requirement for course section cs.
- ***a5[cs, r]***: Binary indicator if room r meets the PWD requirement for course section cs.
- ***a6[r]***: Binary indicator if room r is a contingency room.
- ***a7[r]***: Binary indicator if room r is available (not closed for renovation).
- ***MWF_Days***: Subset of days {Monday, Wednesday, Friday}.
- ***TTh_Days***: Subset of days {Tuesday, Thursday}.
- ***Building[r]***: Building identifier for room r (e.g., UNF NF Building 1).
- ***Location[r]***: Location identifier for room r (e.g., UNF NF).
- ***ReservePct***: Percentage of total room-time capacity to reserve for contingencies (e.g., 0.1 for 10%).

Decision Variables

- **Z[l, cs, d, r, t]**: Binary variable indicating whether lecturer $l \in L$ teaches course section $cs \in CS$ on day $d \in D$ in room $r \in R$ at time period $t \in T$.
  - $Z[l, cs, d, r, t] = 1$ if assigned, 0 otherwise.
- **Pattern[cs, p]**: Binary variable indicating the day pattern for course section $cs \in CS$ with frequency > 1, where $p \in P$ (0 for MWF, 1 for TTh).
  - $Pattern[cs, p] = 1$ if pattern p is selected, 0 otherwise.
- **FixedTime[cs, t]**: Binary variable indicating the fixed time period for course section $cs \in CS$ with frequency > 1, where $t \in T$.
  - $FixedTime[cs, t] = 1$ if time t is selected, 0 otherwise.
- **teaches[l, cs]**: Binary variable indicating whether lecturer $l \in L$ teaches course section $cs \in CS$.
  - $teaches[l, cs] = 1$ if assigned, 0 otherwise.
- **uses_location[l, d, loc]**: Binary variable indicating whether lecturer $l \in L$ is assigned to location $loc \in LOC$ on day $d \in D$.
  - $uses_location[l, d, loc] = 1$ if assigned, 0 otherwise.

Extended Model: Objective Function (1 of 2)

Class Scheduling & Instructor Assignment Problem

$$\text{Maximize} \sum_{l\in L, cs\in CS, d\in D, r\in R, t\in T} Z[l,cs,d,r,t]\cdot(p1[l,cs]+p2[l,d]+p3[cs,d]+p4[l,\text{Location}[r]]+p5[cs,r]+p6[r]-\beta\cdot a6[r])$$

Where β is a penalty coefficient (e.g., 1).

Subject To:

$$\sum_{l\in L, d\in D, r\in R, t\in T} Z[l,cs,d,r,t] = FreqD[cs] \quad (\forall cs\in CS) \quad \text{(Course Section Frequency and Day Patterns)}$$

For $FreqD[cs] > 1$:

$$\sum_{p\in P} \text{Pattern}[cs,p] = 1 \quad (\forall cs\in CS) \quad \text{(Course Section Frequency and Day Patterns)}$$

$$\sum_{l\in L, d\in \text{MWF\_Days}, r\in R, t\in T} Z[l,cs,d,r,t] = FreqD[cs]\cdot\text{Pattern}[cs,0] \quad (\forall cs\in CS) \quad \text{(Course Section Frequency and Day Patterns)}$$

$$\sum_{l\in L, d\in \text{TTh\_Days}, r\in R, t\in T} Z[l,cs,d,r,t] = FreqD[cs]\cdot\text{Pattern}[cs,1] \quad (\forall cs\in CS) \quad \text{(Course Section Frequency and Day Patterns)}$$

$$\sum_{t\in T} \text{FixedTime}[cs,t] = 1 \quad (\forall cs\in CS) \quad \text{(Consistent Time Period)}$$

$$Z[l,cs,d,r,t] \le \text{FixedTime}[cs,t] \quad (\forall l\in L, cs\in CS, d\in D, r\in R, t\in T) \quad \text{(Consistent Time Period)}$$

$$\sum_{cs\in CS, r\in R} Z[l,cs,d,r,t] \le 1 \quad (\forall l\in L, d\in D, t\in T) \quad \text{(No Lecturer Overlap)}$$

$$\sum_{l\in L, cs\in CS} Z[l,cs,d,r,t] \le a1[l,cs]\cdot FreqD[cs] \quad (\forall l\in L, cs\in CS) \quad \text{(Lecturer-Course Feasibility)}$$

$$\sum_{cs\in CS, d\in D, r\in R, t\in T} Z[l,cs,d,r,t] \le 1 \quad (\forall r\in R, d\in D, t\in T) \quad \text{(No Room Overlap)}$$

$$\sum_{cs\in CS, d\in D, r\in R, t\in T} Z[l,cs,d,r,t]\cdot\text{DurT}[cs] \le \text{MaxT}[l] \quad (\forall l\in L) \quad \text{(Lecturer Availability)}$$

Extended Model: Objective Function (2 of 2)

Class Scheduling & Instructor Assignment Problem

$$\text{Maximize} \sum_{l\in L, cs\in CS, d\in D, r\in R, t\in T} Z[l,cs,d,r,t]\cdot(p1[l,cs]+p2[l,d]+p3[cs,d]+p4[l,\text{Location}[r]]+p5[cs,r]+p6[r]-\beta\cdot a6[r])$$

Where β is a penalty coefficient (e.g., 1).

Subject To: (continued)

$$\sum_{l\in L, d\in D, t\in T} Z[l,cs,d,r,t] \le a4[cs,r]\cdot a5[cs,r]\cdot a7[r] \quad (\forall cs\in CS, r\in R) \quad \text{(Course-Room Fit)}$$

$$Z[l,cs_1,d,r_1,t_1]+Z[l,cs_2,d,r_2,t_2]\le 1 \quad (\forall l\in L \text{ where BreakT}[l]=1, cs_1,cs_2\in CS, d\in D, r_1,r_2\in R, t_1,t_2\in T) \quad \text{(Lecturer Breaks)}$$

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

$$\text{MinC}[l]\le\sum_{cs\in CS}\text{teaches}[l,cs]\le\text{MaxC}[l] \quad (\forall l\in L) \quad \text{(Workload Limits)}$$

$$\text{teaches}[l,cs]\ge\sum_{d\in D, r\in R, t\in T} Z[l,cs,d,r,t]/FreqD[cs] \quad (\forall l\in L, cs\in CS) \quad \text{(Workload Limits)}$$

$$\text{teaches}[l,cs]\le\sum_{d\in D, r\in R, t\in T} Z[l,cs,d,r,t] \quad (\forall l\in L, cs\in CS) \quad \text{(Workload Limits)}$$

$$\sum_{loc\in LOC}\text{uses\_location}[l,d,loc]=1 \quad (\forall l\in L, d\in D) \quad \text{(Lecturer Location per Day)}$$

$$\text{loc\_usage}[l,d,loc]=\sum_{r\in R_{loc}, cs\in CS, t\in T} Z[l,cs,d,r,t] \quad (\forall l\in L, d\in D, loc\in LOC) \quad \text{(Lecturer Location per Day)}$$

$$\text{uses\_location}[l,d,loc]\cdot(|CS|\cdot|T|)\ge\text{loc\_usage}[l,d,loc] \quad (\forall l\in L, d\in D, loc\in LOC) \quad \text{(Lecturer Location per Day)}$$

$$\sum_{l\in L, cs\in CS, d\in D, r\in R, t\in T} Z[l,cs,d,r,t]\le(1-\text{ReservePct})\cdot|R|\cdot|D|\cdot|T| \quad \text{(Contingency Reservation)}$$

$$Z[l,cs_1,d,r_1,t_1]+Z[l,cs_2,d,r_2,t_2]\le 1 \quad (\forall l\in L, cs_1,cs_2\in CS, d\in D, r_1,r_2\in R, t_1,t_2\in T) \quad \text{(Building Transfer Break)}$$

Where:

- $\text{Location}[r_1]=\text{Location}[r_2]$ (same location),
- $\text{Building}[r_1]\ne\text{Building}[r_2]$ (different buildings),
- $0<|t_2-t_1|<2$ (30-minute gap).

Extended Model: Solution Analysis (1 of 6)

Class Scheduling & Instructor Assignment Problem

Key Components	Purpose	Code Snippet
$$\textit{Maximize } Z = \sum_{l\in L, cs\in CS, d\in D, r\in R, t\in T} Z_{l,cs,d,r,t} \cdot (p1_{l,cs} + p2_{l,d} + p3_{cs,d} + p4_{l,loc} + p5_{cs,r} + p6_r) - \beta \cdot a6_r$$		

$Z_{l,cs,d,r,t}$	**Z** refers to the timetable assignment.	`objective = solver.Sum(`
		`    Z[l, cs, d, r, t] * (`
		`        p1.get((l, cs.split('-')[0]), 1) +`
		`        p2.get((l, d), 1) +`
		`        p3.get((cs.split('-')[0], d), 1) +`
		`        p4.get((l, room_location[r]), 1) +`
		`        p5.get((cs.split('-')[0], r), 1) +`
		`        p6.get(r, 1)`
		`    ) for l, cs, d, r, t in Z.keys()`
		`)`
		`solver.Maximize(objective)`

$p1_{l,cs}$	Maximize lecturer's course preferences. Assigning lecturers to course sections they prefer to teach.	
$p2_{l,d}$	Maximize lecturer's teaching day preferences. Scheduling lecturers on their preferred teaching days.	
$p3_{cs,d}$	Maximize course day preferences. Scheduling course sections on the most suitable days.	
$p4_{l,loc}$	Maximize lecturer's location preferences. Assigning lecturers to their preferred campus.	
$p5_{cs,r}$	Maximize course room preferences. Ensuring courses are assigned to preferred or better-equipped rooms.	
$p6_r$	Maximize room condition score. Giving priority to well-maintained rooms over contingency or under-renovation rooms.	
$a6_r$	Minimizing contingency room use. Avoiding the use of backup (contingency) rooms, which are penalized in the function.	

Extended Model: Solution Analysis (2 of 6)

Class Scheduling & Instructor Assignment Problem

Key Components	Purpose	Code Snippet

Constraint #1: Course Section Frequency and Day Patterns	Ensures each course section is scheduled according to its frequency (`FreqD[cs]`). Multi-session courses (`FreqD[cs] > 1`) follow MWF or TTh patterns.	`for cs in course_sections:`
		`    solver.Add(sum(Z.get((l, cs, d, r, t), 0) for l in lecturers for d in days`
		`                for r in rooms for t in time_periods) == freq_d[cs])`
		`    if freq_d[cs] > 1:`
		`        solver.Add(Pattern[cs, 0] + Pattern[cs, 1] == 1)`
		`        solver.Add(sum(Z.get((l, cs, d, r, t), 0) for l in lecturers for d in mwf_days`
		`                for r in rooms for t in time_periods) == freq_d[cs] * Pattern[cs, 0])`
		`        solver.Add(sum(Z.get((l, cs, d, r, t), 0) for l in lecturers for d in tth_days`
		`                for r in rooms for t in time_periods) == freq_d[cs] * Pattern[cs, 1])`

Constraint #2: Consistent Time Period	Multi-session courses (`FreqD[cs] > 1`) must occur at the same time period across all sessions.	`for cs in course_sections:`
		`    if freq_d[cs] > 1:`
		`        solver.Add(sum(FixedTime[cs, t] for t in time_periods) == 1)`
		`        for l in lecturers:`
		`            for d in days:`
		`                for r in rooms:`
		`                    for t in time_periods:`
		`                        solver.Add(Z.get((l, cs, d, r, t), 0) <= FixedTime[cs, t])`

Constraint #3: No Lecturer Overlap	Prevents a lecturer from being scheduled for multiple sessions at the same time.	`for l in lecturers:`
		`    for d in days:`
		`        for t in time_periods:`
		`            solver.Add(sum(Z.get((l, cs, d, r, t), 0) for cs in course_sections for r in rooms) <= 1)`

Extended Model: Solution Analysis (3 of 6)

Class Scheduling & Instructor Assignment Problem

Key Components	Purpose	Code Snippet

Constraint #4: Lecturer-Course Feasibility	Ensures lecturers only teach course sections they are qualified for (`a1[l, cs]`). Implicitly enforced by creating `Z` variables only where `a1[l, cs] = 1` (assumed 1 for all in code).	`Z = {}`
		`for l in lecturers:`
		`for cs in course_sections:`
		`if a1[l, cs]:`
		`for d in days:`
		`for r in rooms:`
		`if a4[cs, r] and a5[cs, r] and a7[r]:`
		`for t in time_periods:`
		`Z[l, cs, d, r, t] =`
		`solver.BoolVar(f'Z[{l},{cs},{d},{r},{t}]')`

Constraint #5: No Room Overlap	Ensures a room is not booked for multiple sessions at the same time.	`for r in rooms:`
		`for d in days:`
		`for t in time_periods:`
		`solver.Add(sum(Z.get((l, cs, d, r, t), 0) for l in lecturers for cs`
		`in course_sections) <= 1)`

Constraint #6: Lecturer Availability	Limits total teaching hours per week for each lecturer to `MaxT[l]`.	`for l in lecturers:`
		`solver.Add(sum(Z.get((l, cs, d, r, t), 0) * dur_t[cs] for cs in`
		`course_sections`
		`for d in days for r in rooms for t in time_periods) <=`
		`max_t[l])`

Extended Model: Solution Analysis (4 of 6)

Class Scheduling & Instructor Assignment Problem

<table>
<tr><th>Key Components</th><th>Purpose</th><th>Code Snippet</th></tr>
<tr><td>Constraint #7: Course-Room Fit</td><td>Ensures rooms meet course requirements (type: `a4[cs, r]`, PWD: `a5[cs, r]`, availability: `a7[r]`). Implicitly enforced by creating `Z` variables only where `a4`, `a5`, and `a7` are satisfied (all 1 in this case).</td><td>

```
Z = {}
for l in lecturers:
    for cs in course_sections:
        if a1[l, cs]:
            for d in days:
                for r in rooms:
                    if a4[cs, r] and a5[cs, r] and a7[r]:
                        for t in time_periods:
                            Z[l, cs, d, r, t] =
solver.BoolVar(f'Z[{l},{cs},{d},{r},{t}]')
```

</td></tr>
<tr><td>Constraint #8: Lecturer Breaks</td><td>Ensures a 30-minute break between sessions for lecturers with `BreakT[l] = 1`, separate from building transfers.</td><td>

```
for l in lecturers:
    if break_t[l] == 1:
        for d in days:
            for t_idx in range(len(time_periods) - 1):
                t1, t2 = time_periods[t_idx], time_periods[t_idx + 1]
                solver.Add(sum(Z.get((l, cs1, d, r1, t1), 0) for cs1 in
course_sections for r1 in rooms) +
                           sum(Z.get((l, cs2, d, r2, t2), 0) for cs2 in
course_sections for r2 in rooms) <= 1)
```

</td></tr>
<tr><td>Constraint #9: Workload Limits</td><td>Ensures number of courses per lecturer is between `MinC[l]` and `MaxC[l]`.</td><td>

```
for l in lecturers:
    teaches = solver.NumVar(0, solver.infinity(), f'teaches[{l}]')
    for cs in course_sections:
        total_assign = sum(Z.get((l, cs, d, r, t), 0) for d in days for r in
rooms for t in time_periods)
        solver.Add(teaches >= total_assign / freq_d[cs])
    solver.Add(teaches >= min_c[l])
    solver.Add(teaches <= max_c[l])
```

</td></tr>
</table>

Extended Model: Solution Analysis (5 of 6)

Class Scheduling & Instructor Assignment Problem

Key Components	Purpose	Code Snippet

Constraint #10: Lecturer Location per Day	Ensures each lecturer is assigned to only one location per day.	`for l in lectures:`
		`    for d in days:`
		`        solver.Add(sum(uses_location[l, d, loc] for loc in locations) == 1)`
		`        for loc in locations:`
		`            loc_usage = sum(Z.get((l, cs, d, r, t), 0) for cs in course_sections`
		`                            for r in rooms if room_location[r] == loc for t in time_periods)`
		`            solver.Add(loc_usage <= uses_location[l, d, loc] * len(course_sections) * len(time_periods))`

Constraint #11: Contingency Reservation	Reserves a percentage (`ReservePct`) of total room-time capacity for contingencies.	`total_capacity = len(rooms) * len(days) * len(time_periods)`
		`solver.Add(sum(Z.get((l, cs, d, r, t), 0) for l in lecturers for cs in course_sections`
		`               for d in days for r in rooms for t in time_periods) <= (1 - reserve_pct) * total_capacity)`

Constraint #12: Building Transfer Break	Ensures a 30-minute break between sessions in different buildings within the same location for all lecturers.	`for l in lecturers:`
		`    for d in days:`
		`        for t_idx in range(len(time_periods) - 1):`
		`            t1, t2 = time_periods[t_idx], time_periods[t_idx + 1]`
		`            for r1, r2 in [(r1, r2) for r1 in rooms for r2 in rooms`
		`                           if room_location[r1] == room_location[r2] and room_building[r1] != room_building[r2]]:`
		`                solver.Add(sum(Z.get((l, cs1, d, r1, t1), 0) for cs1 in course_sections) +`
		`                           sum(Z.get((l, cs2, d, r2, t2), 0) for cs2 in course_sections) <= 1)`

Extended Model: Solution Analysis (6 of 6)

Class Scheduling & Instructor Assignment Problem

<table>
<tr><th>Key Components</th><th>Purpose</th><th>Code Snippet</th></tr>
<tr><td>Constraint #13: Single Lecturer per Course Section</td><td>Ensures each course section is assigned to exactly one lecturer.</td><td><pre>
for cs in course_sections:
    solver.Add(sum(sum(Z.get((l, cs, d, r, t), 0) for d in days for r in rooms for t in time_periods)
                   for l in lecturers) == freq_d[cs]) # Total assignments match frequency
    for l in lecturers:
        assign = solver.BoolVar(f'assign[{l},{cs}]')
        total_assign = sum(Z.get((l, cs, d, r, t), 0) for d in days for r in rooms for t in time_periods)
        solver.Add(total_assign <= freq_d[cs] * assign) # If assigned, must match frequency
        solver.Add(total_assign >= assign) # If any assignment, activate assign
    solver.Add(sum(solver.BoolVar(f'assign[{l},{cs}]') for l in lecturers) <= 1) # At most one lecturer
</pre></td></tr>
<tr><td>Constraint #14: One Session per Day</td><td>Ensures multi-session courses are scheduled only once per day.</td><td><pre>
for cs in course_sections:
    for d in days:
        solver.Add(sum(Z.get((l, cs, d, r, t), 0) for l in lecturers for r in rooms for t in time_periods) <= 1)
</pre></td></tr>
</table>

Extended Model: Results

Class Scheduling & Instructor Assignment Problem

Course	Course Section	Lecturer	Room	Building	Location	Day	Time Period
CPSC	CPSC-500-1	Marin Vratonjic	UNF 402	UNF NF Building 2	UNF NF	Friday	11:00-11:30
CPSC	CPSC-500-1	Marin Vratonjic	UNF 215	UNF NF Building 1	UNF NF	Monday	11:00-11:30
CPSC	CPSC-500-1	Zeeshan Ahmad	UNF 215	UNF NF Building 1	UNF NF	Wednesday	11:00-11:30
CPSC	CPSC-500-2	Hany Osman	UNF 215	UNF NF Building 1	UNF NF	Friday	16:30-17:00
CPSC	CPSC-500-2	Hany Osman	UNF 215	UNF NF Building 1	UNF NF	Monday	16:30-17:00
CPSC	CPSC-500-2	Touraj Banirostam	UNF 402	UNF NF Building 2	UNF NF	Wednesday	16:30-17:00
CPSC	CPSC-500-3	Hassan Baz Chamas	UNF 402	UNF NF Building 2	UNF NF	Friday	16:30-17:00
CPSC	CPSC-500-3	Ahmed Eltahawi	UNF 402	UNF NF Building 2	UNF NF	Monday	16:30-17:00
CPSC	CPSC-500-3	Ahmed Eltahawi	UNF 215	UNF NF Building 1	UNF NF	Wednesday	16:30-17:00
CPSC	CPSC-510-1	Ali El-Sharif	UNF 402	UNF NF Building 2	UNF NF	Friday	12:30-13:00
CPSC	CPSC-600-1	Zeeshan Ahmad	UNF 215	UNF NF Building 1	UNF NF	Thursday	14:00-14:30
CPSC	CPSC-610-1	Hany Osman	UNF 215	UNF NF Building 1	UNF NF	Friday	10:30-11:00
CPSC	CPSC-620-1	Sundus Shanef	UNF 402	UNF NF Building 2	UNF NF	Tuesday	15:30-16:00

Key Insights

- **Optimal Schedule:** MIP was used instead of CP-SAT due to constraints for **Course Section Frequency and Day Patterns**, **Workload Limits**, and **Lecturer Location per Day**. All 7 course sections assigned across MWF/TTh patterns using UNF 215 and UNF 402.
- **Constraints Satisfied:** No lecturer/room overlaps; workload limits (1-5 courses) met; single location (UNF NF) per day.
- **Preference Maximization:** Consistent room/time assignments reflect lecturer preferences.
- **Observation:** Unused rooms (e.g., UNF 220, UNF 225) indicate potential data or capacity gaps.

Model Comparison (1 of 2)

Class Scheduling & Instructor Assignment Problem

Feature	Base Model	Extended Model
Input Data	• Handles ***9*** Tables: Lecturer, Course, Room, Day, Time Period, Lecturer_Course, Course_Room, Lecturer_Day, and Course_Day	• New Table: Lecturer_Location • Enhanced ***3*** of ***9*** Tables: Lecturer, Course, and Room
One Schedule per Course	**Implemented**	**Replaced**
No Lecturer Overlap	**Implemented**	**Enhanced**
Lecturer-Day Feasibility	**Implemented**	**Implemented**
No Room Overlap	**Implemented**	**Enhanced**
Lecturer-Course Feasibility	**Implemented**	**Implemented**
Course-Room Fit	**Implemented**	**Enhanced**
Course Section Frequency and Day Patterns	**-**	**Implemented**
Consistent Time Period	**-**	**Implemented**
Lecturer Availability	**-**	**Implemented**
Lecturer Breaks	-	**Implemented**
Workload Limits	-	**Implemented**
Lecturer Location per Day	-	**Implemented**
Contingency Reservation	-	**Implemented**

Model Comparison (2 of 2)

Class Scheduling & Instructor Assignment Problem

Feature	Base Model	Extended Model
Building Transfer Break	-	Implemented
Single Lecturer per Course Section	-	Implemented
One Session per Day	-	Implemented
Output Data	Timetable	• Enhanced Timetable • Course Allocation Report • Lecturer Allocation Report • List of Unallocated Slots

Simulation

University of Notable Future (UNF) Web App

Conclusion

Summary

- Expanded the Naderi (2016) baseline model from ***6*** features to a total of ***14*** enhanced features, incorporating both new additions and improvements to existing ones.
- Integrated preference-based constraints to enhance schedule quality while maintaining feasibility.
- Applied a mixed constraint satisfaction and optimization approach to ensure both compliance and scheduling efficiency.
- Delivered a scalable, data-driven solution for effective university course scheduling.

Recommendations for Future Work

- **Advanced Preference Weighting:** Refine the objective function with customizable weights for preferences (e.g., room quality vs. time-of-day).
- **Hybrid Course Pairing:** Maximize room utilization for hybrid courses that alternate in-person and online classes.
- **UI Enhancement:** Improve current user interface for easier data input, scenario testing, and result visualization.
- **Real-World Pilot Testing:** Test the model with actual data from the University of Niagara Falls to validate and refine assumptions.

Q&A