Midsem Scheduling

Software Defined Networks

Shashank P
200010048

A. Aim

This assignment aims to schedule mid-semester examinations for students optimally. The solution is to take care of multiple courses, venues and slots and ensure all students attend the exam without conflicts.

B.   Setup

The primary variable for optimisation is a binary matrix of the following size:

X – slots * courses * venues * students.

The inputs are given in the following fashion:
1. **Slots:** List of integers (even). Morning and Afternoon for every day
2. **Campuses:** List of integers representing campuses
3. **Courses:** Dictionary of course mapped to campus and is_lab
4. **Students:** Dictionary of students mapped to a set of courses
5. **Venues:** Dictionary of venue mapped to campus and capacity.

For simplicity, I will denote **slot = t, course = c, venue = v** and **student = s**

C. Constraints

To ensure consistency and validity, the following constraints are introduced.

1. Each student must have precisely one exam for each course enrolled in, and 0 if not enrolled.

$$\forall c \ \forall s \ \sum_t \sum_v X_{t,\,c,\,v,\,s} = \begin{cases} 1 & \text{if student took the course} \\ 0 & \text{otherwise} \end{cases}$$

2. Theory courses are scheduled in a single slot.

For this constraint, I took a slightly different approach. I created another variable defined below

slot_course (SC) – slots * courses

With the following constraints:

    a. Each value in SC now stores one if the course (c) is scheduled at least once in slot (t)

$$\forall c \ \forall t \quad SC_{c,t} = \max_{\substack{v \in V, \\ s \in S}} X_{t,c,v,s}$$

    b. For every course (c), find the total number of schedules across all slots (T). This value should be equal to the sum of all schedules of course (c) in slot (t) if at least one schedule is in slot (t) and 0 otherwise. Assume here **c = theory course**

$$\forall c \ \forall t$$
$$(SC_{c,t}) * \sum_v \sum_t \sum_s X_{t,\,c,\,v,\,s} = (SC_{c,t}) * \sum_v \sum_s X_{t,\,c,\,v,\,s}$$

3. Total strength should not exceed venue capacity

$$\forall t \ \forall v \ \sum_{c} \sum_{s} X_{t, c, v, s} \leq capacity(v)$$

4. Each course must be scheduled at the designated campus.

Assume here **v** to be all the venues not in course **c**

$$\forall c \ \forall v \sum_{t} \sum_{s} X_{t, c, v, s} = 0$$

5. Each student has at most one exam per day.

To achieve this, I go through every student and every slot (**te**) and add a constraint that the total number of schedules in the morning and evening slots are at most **1.**

$$\forall s \ \forall te \sum_{c} (X_{te, c, v, s} + X_{te+1, c, v, s}) \leq 1$$

6. In one slot and one venue, there can be multiple courses scheduled

To achieve this, I declared another variable called. **Slot_venue_course** as described below

SVC - slots * venues * courses

Similar to **Constraint 2,** I added the following constraints.

$$\forall \ t, v, c : \ SCV_{t, v, c} = \max_{s} X_{t, c, v, s}$$

$$\forall \ t, v : \sum_c SCV_{t, \, v, \, c} \leq 1$$

D. Objective

The objective that I chose was to minimise the total number of slots that were used. For this, I defined a new binary variable to track whether a slot has an exam:

S – slots.

The following constraint is then added to the model:

$$\forall t \ \ S_t = \max_c (SC_{t, \, c})$$

The objective function becomes:

$$\min \sum_t S_t$$

E. Experimentation

The variables and constraints were declared in Python using the **gurobipy** module, and the corresponding outputs were saved to a file as mentioned in the **README.md** file. The following input ranges were given.

1. **Students:** 30
2. **Campuses:** 2
3. **Courses:** 8
4. **Venues:** 10 rooms with an Average 30 capacity

5. **Slots:** 14 (7 days with two slots each)

The hardware used for optimisation is shown below:

CPU model: Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz
Thread count: 6 cores, 12 logical processors, using up to 12 threads

Since the optimisation took a long time, I terminated the optimisation after **5 minutes** from the beginning.