1 Input

1.1 Demand Matrix

The demand matrix will have 3 dimensions:

- Time (24 rows per day, or 168 rows in total)
- Subject ($\sim$ 5-6 columns: English, Math, Accounting, CAD, Study Skills, etc.)
- Campus (6 slices: one for online and one for each of the campuses)

The entry $D_{j,k,l}$ represents the demand for time slot j, in subject k, at campus l, where each index ranges from 1 to $n_j (= 168), n_k, n_l (= 6)$ respectively.

1.2 Availability Matrix

The availability matrix will have 4 dimensions:

- Tutor
- Time
- Subject
- Campus

The entry $A_{i,j,k,l}$ represents the availability of tutor i at time slot j, in subject k, at campus l, where each index ranges from 1 to n_i, n_j, n_k, n_l respectively. Each entry is either 0 or 1.

1.3 Desired Hours

When collecting the availability information for each tutor, we also collect the maximum number of hours each tutor is willing to work per week. This is represented as a one-dimensional array; for $i = 1, \ldots, n_i$, the entry H_i represents the number of hours tutor i is willing to work each week.

1.4 Budget

For the semester, the Tutoring and Learning Centre has a budgeted number of work hours for all tutors employed during the term collectively, which we denote as B.

2 Decision Variables

The variable $x_{i,j,k,l} \in \{0,1\}$ indicates whether tutor i is assigned to time slot j for subject k at campus l.

3 Constraints

3.1 Availability

Tutors can only be booked for the time slots and subjects for which they are marked as available on the availability matrix.

For each i, j, k, l, if $A_{i,j,k,l} = 0$, then we add the constraint that $x_{i,j,k,l} = 0$.

3.2 Avoid Multiple Bookings at the Same Time

Since the variables for each subject and campus are now independent of each other, we must add the constraint that the same tutor cannot be booked for multiple subjects or at multiple campuses at the same time.

For each tutor i and time slot j, we add the constraint

$$\sum_{k=1}^{n_k} \sum_{l=1}^{n_l} x_{i,j,k,l} \leq 1$$

3.3 Avoid Consecutive Time Slots at Different Campuses

If a tutor is working at multiple campuses, each contiguous shift should take place at the same campus, so there should not be consecutive time slots for the same tutor at different campuses.

For all tutors i, time slots j, subjects k_1, k_2, and campuses l_1, l_2 such that $l_1 \neq l_2$, we add the constraint

$$x_{i,j,k_1,l_1} + x_{i,j+1,k_2,l_2} \leq 1.$$

(This disallows consecutive appointments at different campuses, but allows consecutive appointments for different subjects. If we wanted to disallow consecutive appointments for different subjects as well, we would require that $k_1 \neq k_2$ or $l_1 \neq l_2$.)

3.4 Online vs. In-person Operating Hours

The online schedule goes from 9 AM to 9 PM each day and is open on weekends, while the in-person schedule goes only from 10 AM to 5 PM each day and is only open on weekdays. Thus there should be a constraint for each in-person schedule that prevents tutors from working in non-operating hours at the in-person locations.

The time slots starting at 9:00 and 9:30 AM are the 1st and 2nd time slots of the day, and the time slots after 5:00 PM are the 17th, 18th, ..., 24th time slots of the day. Since our indexing for time slots starts at $j = 1$, the non-operating hours at in-person campuses correspond to when $((j-1) \mod 24) + 1 \in \{1, 2, 17, 18, \ldots, 24\}$, and the weekend hours correspond to when $\frac{j-1}{24} \geq 5$, i.e., $j \geq 121$.

Assuming that $l = 1$ represents the online schedule, and $j = 1, 2$ represent the time slots starting at 9:00 and 9:30 AM while $j = 17, \ldots, 24$ represent the time slots starting at 5:00, ..., 8:30 PM, we add the constraints

$$x_{i,j,k,l} = 0$$

for each tutor i, time slot j such that $((j-1) \mod 24)+1 \in \{1, 2, 17, 18 \ldots, 24\}$ or $j \geq 121$, subject k, and campus $l \geq 2$.

3.5 Work Hours Constraints

We require that no tutor works for more than 24 hours (48 time slots) per week, more than 7 hours (14 time slots) per day, or more than 5 hours (10 time slots) consecutively in the same day, summed over all campuses and subjects.

For each tutor i, we add the following constraints:

$$\sum_{j=1}^{n_j}\sum_{k=1}^{n_k}\sum_{l=1}^{n_l} x_{i,j,k,l} \leq 48$$

$$\sum_{k=1}^{n_k}\sum_{l=1}^{n_l}\sum_{t=1}^{24} x_{i,24d+t,k,l} \leq 14 \qquad \text{for each } d = 0, \ldots, 6$$

$$\sum_{k=1}^{n_k}\sum_{l=1}^{n_l}\sum_{r=0}^{10} x_{i,24d+t+r,k,l} \leq 10 \qquad \text{for each } d = 0, \ldots, 6,\ t = 1, \ldots, 24$$

3.6 Desired Hours

We require that tutors do not work for more hours than indicated in their desired hours. For each tutor i, we add the constraint

$$\sum_{j=1}^{n_j}\sum_{k=1}^{n_k}\sum_{l=1}^{n_l} x_{i,j,k,l} \leq H_i.$$

3.7 Budget Constraint

There is a total budgeted number of hours for all tutors. We require that the total working hours for all tutors, summed across all time slots, campuses, and subjects, does not exceed the budget. We add the constraint

$$\sum_{i=1}^{n_i}\sum_{j=1}^{n_j}\sum_{k=1}^{n_k}\sum_{l=1}^{n_l} x_{i,j,k,l} \leq B$$

where B is the total budgeted number of hours.

3.8 Bounds

For each i, j, k, l, we require that $0 \leq x_{i,j,k,l} \leq 1$ and $x_{i,j,k,l}$ is an integer.

Since integer programming scales exponentially in complexity, we can improve performance at the cost of precision by removing the integrality constraint. In this case, each $x_{i,j,k,l}$ is still required to be between 0 and 1, but is not required to be an integer; we must then use an algorithm to discretize the results and resolve any scheduling conflicts that arise.

4 Objective function

We want to minimize the difference between the number of tutors working at each time slot, subject, and campus, and the demand for that time slot, subject, and campus.

There are multiple ways to condense this into one objective function. For example, we could take the square difference between the number of tutors working at each tuple (j, k, l), and sum over all j, k, and l:

$$\text{minimize} \sum_{j=1}^{n_j} \sum_{k=1}^{n_k} \sum_{l=1}^{n_l} \left(D_{j,k,l} - \sum_{i=1}^{n_i} x_{i,j,k,l} \right)^2$$

Or we could take the absolute difference instead of the squared difference:

$$\text{minimize} \sum_{j=1}^{n_j} \sum_{k=1}^{n_k} \sum_{l=1}^{n_l} \left| D_{j,k,l} - \sum_{i=1}^{n_i} x_{i,j,k,l} \right|$$

Or we could separately compute the overage (i.e., by how much the output schedule exceeds demand) and underage (i.e., by how much the output schedule falls below demand) and try to minimize both, with parameters to control how much to punish overage or underage:

$$\begin{aligned} O &= \max(0, \sum_{j=1}^{n_j} \sum_{k=1}^{n_k} \sum_{l=1}^{n_l} (D_{j,k,l} - \sum_{i=1}^{n_i} x_{i,j,k,l})) \\ U &= \max(0, \sum_{j=1}^{n_j} \sum_{k=1}^{n_k} \sum_{l=1}^{n_l} (\sum_{i=1}^{n_i} x_{i,j,k,l} - D_{j,k,l})) \\ &\text{minimize } pO + qU \end{aligned}$$

where $p, q \in \mathbb{R}$ are adjustable parameters.