Hybrid Student-level Scheduling Optimization Problem

Contents

1.1 Data Sets and Variables Definition

Set S contains all students and set C contains all classes. A_k denotes the set of students enrolled in class k, where $k \in C$. Time is represented as t, ranging from 0 (Sunday midnight) to 167 (Saturday 11 pm). Ct is the set of classes ongoing at time t. ck signifies the social distancing capacity of each class k in set C while E represents the capacity of the excess room. Weights λ and μ are assigned to Total Deviation (TD) and Surplus Simultaneous Excess (SSE), respectively, in the objective function. Below are the variables used in solving the optimization problem:

π_{ij}: Binary. 1 when a student i (for i = 1, …, N) is allocated to a group j (for j = 1, ..., M); otherwise, 0

e_{jk}: The excess in class k when group j is scheduled to attend the class in person; when there is no excess, e_{jk} is set to 0.

s^t_j: The total number of students in excess room at time t when group j is scheduled to in-person attendance.

s: The maximum number of students assigned to the excess room at any given time yet surpass the excess-room capacity E, s is set to 0 if no such surplus; i.e. $(\max_{t,j} s^t_j - E)^+$

δ_{jk}: The absolute difference between the number of students assigned to class k when group j is scheduled and the ideal number of students that would be assigned if the student assignments were evenly distributed among all groups (fractional assignment, i.e. $\frac{|A_k|}{M}$ with $|A_k|$ being the number of students in class k and M being the number of groups assigning)

1.2 Objective Function and Constraints

minimize $\sum_{k\in C}\sum_{j=1}^{M} e_{jk} + \lambda \sum_{k\in C}\sum_{j=1}^{M} \delta_{jk} + \mu s$

(1) $\sum_{k\in C}\sum_{j=1}^{M} e_{jk}$: Total Excess (TE) with the weight normalized to be 1. The summation of e_{jk} across all classes and groups is the TE.

(2) $\lambda \sum_{k\in C}\sum_{j=1}^{M} \delta_{jk}$: Total Deviation (TD) with the weight λ. The summation of δ_{jk} across all classes and groups is the TD.

(3) μs: Surplus Simultaneous Excess (SSE) with the weight μ

Constraints:

Each student must be assigned to exactly one group: $\sum_{j=1}^{M} \pi_{ij} = 1 \ \forall i \in S$

For all class *k* belonging to set C and all group *j* in 1, …, M, the difference between the number of students assigned to in-person attendance (π_{ij}) and the class capacity (c_k) is smaller than or

equal to the excess in class *k* when group *j* is scheduled to attend class k in person (e_{jk}): $\sum_{i \in A_k} \pi_{ij} - c_k \leq e_{jk} \quad 1 \leq j \leq M, \forall k \in C$

For all class *k* belonging to set C and all group *j* in 1, …, M, δ_{jk} is the absolute difference between the actual number of students assigned to class k when group j is scheduled ($\sum_{i \in A_k} \pi_{ij}$) and the expected number of students if each group had an equal number of students assigned to it (fractional allocation), with $|A_k|$ being the number of students registered in class k: $-\delta_{jk} \leq \sum_{i \in A_k} \pi_{ij} - \frac{|A_k|}{M} \leq \delta_{jk} \quad 1 \leq j \leq M, \forall k \in C$

Since $s = (\max_{t,j} s_j^t - E)^+$ as stated above, s must larger than or equal to $s_j^t - E$ for t in 0, …, T and for all group j in 1, …, M: $s \geq s_j^t - E \quad 0 \leq t \leq T, 1 \leq j \leq M$

Link variable s_j^t and variable e_{jk} as follows: $s_j^t \geq \sum_{k \in A_k} e_{jk} \quad 0 \leq t \leq T, 1 \leq j \leq M$

π_{ij} is a binary variable representing whether or not student i is assigned to group j: $\pi_{ij} \in \{0,1\} \quad 1 \leq j \leq M, \forall i \in S$

δ_{jk}, e_{jk}, and s are set to be zero for negative value: $\delta jk, ejk, s \geq 0 \quad 1 \leq j \leq M, \forall k \in C$

1.3 Assumptions

In this optimization scenario, each student is allocated to exactly one of the M groups. The classes follow a fixed weekly time slot rotation. Only one group of students out of M is designated to attend each class in person. The social distancing capacity c_k is evaluated under three protocols: "6ft", "4ft", and "50sf", indicating the spacing between students. Additionally, the excess room capacity (E) is predetermined as 140. Weights are assigned to metrics: 0.25 for Total Deviation (TD), 1 for Total Excess (TE), and 0 for Surplus Simultaneous Excess (SSE) since SE is significantly lower than E in the experiment.

2.1 Data Preparation for Modelling

The original data (Appendix 1) contains the term 2 timetable of six main MSBA modules, which are participated by 229 MSBA students and some students in QTEM and MMORSE. With a focus only on MSBA students, the QTEM and EXTMOR students are ignored. According to the timetable, each module has one lecture every week, with modules IB9190 and IB98D0 having 4 workshop groups, module IB9HP0 having 3 workshop groups, and the rest having 2 workshop groups each. For each module, we represented each workshop horizontally instead of vertically by assigning each workshop group to a new column and assigning a binary value of 1 or 0 to show whether the student was enrolled in the workshop group. A row of room capacities for each group was added. Appendix 2 specifies these changes.

The provided timetable isn't directly suitable for modeling purposes. Therefore, an alternative representation was created in the form of an Excel table, showing the timing of each class throughout the week (refer to Appendix 3). In this table, classes are listed as columns while the rows represent hours across the week, starting from Sunday midnight (denoted as 0) through Saturday 11 pm (denoted as 167); For classes that begin mid-hour, a new row is inserted and denoted as +0.5 hour. A 'Day in a week' column and a '24-hour clock' column was added for easier reference. Each cell in the table contains a binary value, either 1 or 0, indicating whether the corresponding class is scheduled to occur during that hour.

The lecture for Module IB98E0 spans two hours during Week 7 (9-11 am) and only one hour (9-10 am) for all other weeks. As this lecture doesn't overlap with any other classes, excluding this variation in lecture length yields the same optimization result. Therefore, we treat IB98E0 lectures as one hour every week.

2.2 Assumptions Made

The first assumption made in the student-level scheduling problem follows Warwick University's pandemic social distancing guidelines of 1.5 meters between people. Consequently, the seating capacity in classrooms decreased by 40%, as a typical 3-meter table, which would usually accommodate 5 students, could only hold 2 students under these restrictions. The hybrid scheduling model was tested at 20%, 30%, and 40% of the original room capacity to simulate different levels of distance constraint. We also assumed that the excess room would be the M1 classroom in the Teaching Center with a capacity (E) of 30. Additionally, classes were assumed to occur at the same time every week for each group, throughout the term.

The weight assigned to the three primary measurement components (TE, TD, SSE) in the objective function is 1, 0.5, and 2, respectively. This weighting reflects a lesser importance placed on TD with greater emphasis placed on the total number of students assigned to the excess room (TE) and those unable to fit into the excess room (SSE). TE causes inconvenience for students who travel to the university but cannot attend class in person, while SSE significantly impacts students' right to participate in their enrolled classes in a safe and prepared environment, which is the reason why it was considered the most important metric. Each student must be in exactly one out of the M group and the value of M was tested between 2 to 5. Lastly, for each class, only one group of students out of M groups is assigned to attend in person. The excess from the selected group is moved to the excess room, while the remaining groups will attend online.

3.1 Problem Solving Issues and Solutions

Initially, we attempted to solve the problem using the GLPK solver. However, the problem proved to be too demanding for this solver. There are several solutions for this issue: applying the optimization only on a smaller size of data, changing solver options, or changing the solver used. We opted to utilize the third solution and used the Gurobi solver due to its superior performance, particularly in handling large-scale and complex optimization problems.

3.2 Parameter Analysis

Table 1 shows the summary of the experimental results with different values of M and social-distancing capacity. If we want to maintain an SSE of 0, then M=2 is sufficient when the social-distancing capacity is 30% of the original room capacity or above; however, when the social-distancing capacity decreases to 20%, we need to have at least 3 groups to guarantee SSE=0.

Social-distancing capacity (% of the original capacity)	M	Total Excess	Simultaneous Excess	Total Deviation	Objective Value
40%	5	0	0	36.80	18.40
40%	4	0	0	32.00	16.00
40%	3	0	0	25.33	12.67
40%	2	4	0	8.00	8.00
30%	5	0	0	36.80	18.40
30%	4	0	0	32.00	16.00
30%	3	0	0	25.33	12.67
30%	2	341	28	8.00	345.00
20%	5	0	0	36.80	18.40
20%	4	16	2	32.00	32.00
20%	3	341	19	25.33	353.67
20%	2	783	57	8.00	841.00

Table 1. Summary of the results of hybrid scheduling with different values of M and different social distancing proportions.

In solving the problem, we noticed that increasing the value of M leads to longer run times. Implementing the hybrid scheduling project with a large M would also pose management challenges. Conversely, a small M would result in a higher TE if the social-distancing capacity is limited. Refer to Appendix 4 for the visualized relationship between M and TE.

Conclusion

The research provides insights for the university during pandemics, aiding policy planners in ensuring even student participation in classes. One group will be assigned to attend classes in person and this particular group should vary day-to-day; if M is co-prime with the number of class days, each student can have an equal chance to attend each of their classes in person.

Appendix

Appendix 1

First 5 rows of the original data.

ID	Course	IB9190 - W	IB98D0 - B	IB9HP0 - W	IB98E0 - B (Elective)	IB9MJ0 - W (Elective)	IB9BS0 - S (Elective)
1	MSBA	1	1	2		1	
2	MSBA	2	2	1			2
3	MSBA	3	3	3		2	
4	MSBA	4	4	2		1	
5	MSBA	1	1	3		2	

Appendix 2

First 5 rows and first 5 columns for the adjusted data. Notice that module IB9190-W is split into IB9190-Lecture, IB9190 - W1, IB9190 - W2, IB9190 - W3, and IB9190 - W4.

ID	Course	IB9190-Lecture	IB9190 - W1	IB9190 - W2	IB9190 - W3	IB9190 - W4
	room capacity	250	73	73	73	73
1	MSBA	1	1	0	0	0
2	MSBA	1	0	1	0	0
3	MSBA	1	0	0	1	0
4	MSBA	1	0	0	0	1
5	MSBA	1	1	0	0	0

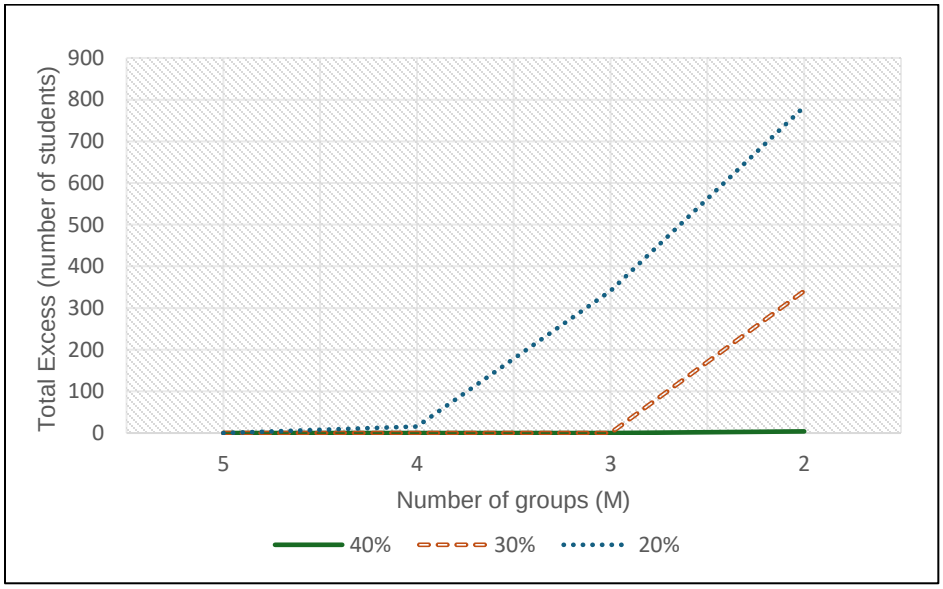
Appendix 3

Part of the schedule table. As shown in this table, IB9190 - W1 takes place on Tuesday 14-16, IB9190 - W2 on Tuesday 16-18, etc.

Hour in a week	Day in a week	24-hour clock	IB9190 - Lecture	IB9190 - W1	IB9190 - W2	IB9190 - W3	IB9190 - W4
62	Tuesday	14	0	1	0	0	0
63	Tuesday	1500-1530	0	1	0	0	0
63.5	Tuesday	1530-1600	0	1	0	0	0
64	Tuesday	16	0	0	1	0	0
65	Tuesday	1700-1730	0	0	1	0	0
65.5	Tuesday	1730-1800	0	0	1	0	0
66	Tuesday	18	0	0	0	0	0

Appendix 4

The relationship between number of groups (M) and total excess (TE) under different level of social-distancing constraints



Appendix 5

Complete python codes of the optimization problem:

```
import pyomo.environ as pyo
from pyomo.opt import SolverStatus, TerminationCondition
from pyomo.environ import *
import pandas as pd
import numpy as np

file_name = "MSBA-Term---Workshop-and-Seminar-Allocations-1.xlsx"

df = pd.read_excel(file_name, "Sheet2")
df2 = pd.read_excel(file_name, "Sheet3")

students = df.iloc[1:, 0] #focus on MSBA student only
classes = df.iloc[0, 2:] #view different workshop slot as independent classes
time = df2.iloc[0:, 0]
```

```
#set a M number of groups
M = 3

df_np = df.to_numpy()
df2_np = df2.to_numpy()

# data preparation

## Ak set: containing students enrolled in class k
participation = df_np[1:, 2:]

## Ct set: containing classes taking place on time t
seminar_time = df2_np[0:, 3:]

## room capacity data
room_cap = df_np[0, 2:]



## ck: social distancing limitatino for class k (assume half of current room cap)
ck = np.floor(room_cap*0.2)

# E: escess room capacity (assumption)
E = 30

# model building
model = ConcreteModel()

# decision variable
model.pi = Var(range(len(students)), range(M), domain = Binary)

model.e = Var(range(M), range(len(classes)), domain = NonNegativeReals)
model.stj = Var(range(len(time)), range(M), domain = Reals)
model.delta = Var(range(M), range(len(classes)), domain = NonNegativeReals)
```

```
model.s = Var(domain = NonNegativeReals)

# objective function
model.obj = Objective(expr = sum(model.e[j,k] for j in range(M) for k in range(len(classes))) +\
                        0.5*sum(model.delta[j,k] for j in range(M) for k in range(len(classes))) +\
                           2*model.s, sense = minimize)
#constraints
# constraint(7)
def rule1(model, i):
    return sum(model.pi[i,j] for j in range(M)) == 1
model.constr1 = Constraint(range(len(students)), rule = rule1)

# constraint(8)
def rule2(model, j, k):
    return sum(participation[i,k]*model.pi[i,j] for i in range(len(students)))-ck[k] <= model.e[j,k]
model.constr2 = Constraint(range(M), range(len(classes)), rule = rule2)

# constraint(9-1)
def rule3(model,j,k):
    return -model.delta[j,k] <= sum(
        participation[i,k]*model.pi[i,j] for i in range(len(students))) - \
        sum(participation[i,k] for i in range(len(students)))/M
model.constr3 = Constraint(range(M), range(len(classes)), rule = rule3)

# constraint(9-2)
def rule4(model,j,k):
    return model.delta[j,k] >= sum(
        participation[i,k]*model.pi[i,j] for i in range(len(students))) - \
        sum(participation[i,k] for i in range(len(students)))/M
model.constr4 = Constraint(range(M), range(len(classes)), rule = rule4)
# constraint(10)
def rule5(model, t, j):
    return model.s >= model.stj[t, j] - E
```

```
model.constr5 = Constraint(range(len(time)), range(M), rule=rule5)

# constraint(11)
def rule6(model, t, j):
    return model.stj[t,j] == sum(model.e[j,k]*seminar_time[t,k] for k in range(len(classes)))
model.constr6 = Constraint(range(len(time)), range(M), rule=rule6)

# specify the solver we are using and solve the problem
solver = SolverFactory('gurobi')
results = solver.solve(model, tee = True)

# print the objective value and the value of TD, TE, SSE
if(results.solver.status == SolverStatus.ok) and (results.solver.termination_condition ==
TerminationCondition.optimal):
    print("Objective value=", round(model.obj(),2),'\n'
          "TE =", sum(model.e[j,k] for j in range(M) for k in range(len(classes)))(),'\n'
          "TD =", round(sum(model.delta[j,k] for j in range(M) for k in range(len(classes)))(),2),'\n'
          "SSE =", model.s())
else:
    print("Solve Failed")

# print the delta value for each group-class combination
for k in range(len(classes)):
    for j in range(M):
        print("For group", j+1, "in class", k+1, "delta =", round(model.delta[j,k](),2))

# print a list showing which group each student is in
for i in range(len(students)):
    for j in range(M):
        if model.pi[i,j]() > 0.5:
            print("Student", i+1, "is in group", j+1)

# print the stj value for each time-class combination
for t in range(len(time)):
```

```
    for j in range(M):
        print('at time', time[t], model.stj[t,j]())

# print the ejk value (excess for class k) for each group-class combination
for k in range(len(classes)):
    for j in range(M):
        print("If group", j+1, "is assigned to attend class", k+1, "in person,", "excess =", model.e[j,k]())

# print the value of SE (max stj)
max_value = 0
for t in range(len(time)):
    for j in range(M):
        max_value = max(max_value, model.stj[t, j]())
print("SE =", max_value)
```