IE303 Operations Research I
Final Project Report
Spring 2025

Group Members:
Asja Bašović, Sara Avdić, Hatidža Imamović, Hava Dedić
Project Title:
Optimized exam schedule room assignment

1. Problem Definition and Objective

The MIP model aims to assign students to exam rooms in a way that minimizes the total number of unused seats while respecting room capacities, exam schedules, grouping constraints, and is comfortable for the students. It is assumed that all courses require a 2-hour slot for a written paper, i.e. no course has a practical exam that requires special laboratories such as the Fabrication Lab. Additionally, the exam schedule that the room optimization is being for done is assumed to be finalized and valid, while rooms with a relevant exam capacity are available at all times during the exam period. Additional general assumptions are that room capacities are accurate and do not change, all students taking an exam must be seated simultaneously, and that there is no seating arrangement.

2. Data Collection and Estimation

Data Used:

The data used includes: rooms available for exams and room capacities for exams, code and name of courses scheduled, undergraduate midterm exam schedule for spring 2025, number of students taking each exam, buildings available (in which building is each room) and layouts (on what floor is each room). This was collected both from the IUS website and given by the professor. The data collected from the website was the midterm exam schedule with venues for spring semester of 2025. This pdf file was downloaded and converted to an excel file using an online converter. The professor provided the information about room capacities and class quotas. The files used can be found in the appendix. The data was deconstructed into appropriate lists and dictionaries using python code and used for the model.

Assumptions:

Exams for architecture students are not considered for this model as they tend to have a separate exam schedule due to the specific format of their exams. Additionally, only the courses that are currently on the exam schedule are used in this model, as they are the ones that require a slot for a written exam. However, courses that are on the exam schedule with a special note that explains the time slot is for project submission or indicates a written exam will not be held are excluded (ELIT201, CS404). This is because the students will not require a seating capacity to turn in their work. For classrooms that have no exam capacity recorded in the data provided by the professor, it is assumed that a written exam cannot take place there. It is assumed that distances between rooms are only relevant for large exams because for some courses with many students, there is only one professor. Consequently, it would be preferable for the professor to stay within the same building while visiting all rooms the exam for their course is taking place in. Otherwise, walking patterns or distances between rooms are ignored. The model does not explicitly assign weights to classrooms based on size. However, it implicitly accounts for room size through capacity constraints as each room's capacity limits the maximum number of students assigned to it. Hence, larger rooms can accommodate more students, effectively weighting the assignment by room size. The optimization then minimizes unused seats, encouraging fuller utilization of bigger rooms without the need for manual weighting.

3. Final Mathematical Model

<u>Sets:</u>

- I: Set of all exams (exam codes)

- J: Set of all available rooms
- B: Set of all buildings {A, B}

Decision Variables:

- $x_{i,j}$: Number of students from exam *i* assigned to room *j* (integer, $0 \le x_{i,j} \le c_j$)
- $w_{i,j}$: Binary variable indicating whether exam *i* uses room *j* (1 if yes, 0 if no)
- $y_{i,b}$: Binary variable indicating whether exam *i* is assigned to building *b* (1 if yes, 0 if no)

Parameters

- c_j: Capacity (maximum number of students) of room j
- s_i: Total number of students registered for exam i
- exam_day[i]: Date when exam i is scheduled
- exam_times[i]: Time slot (start_time, end_time) for exam i
- room_to_building[j]: Building where room j is located
- min_group_size(s_i): Minimum group size for exam i (0 if $s_i < 50$, otherwise 10)
- overlaps(i_1, i_2): Function returning true if exams i_1 and i_2 have overlapping schedules

Objective Function

Minimize total unused seats:

$$\min \sum\sum(c_j - x_{i,j}),\ i \in I,\ j \in J$$

Subject to: (Constraints)

1. Student Assignment Constraint:

$$\sum x_{i,j} = s_i,\ \forall i \in I\ j \in J$$

Each exam must assign all its registered students to rooms.

2. Room Capacity Constraint for Overlapping Exams:

$$\sum x_{i,j} \le c_j,\ \forall j \in J,\ \forall t \in T\ i \in E_t$$

where E_t= {set of all exams scheduled at time slot *t*}

For any room j and any time slot t, the total students assigned from all exams scheduled at that time cannot exceed the room's capacity.

3a. Building Assignment Constraint:

$$\sum y_{i,b} = 1,\ \forall i \in I\ b \in B$$

Each exam must be assigned to exactly one building.

3b. Room-Building Consistency Constraint:

$$x_{i,j} \le c_j * y_{i,b},\ \forall i \in I,\ \forall j \in J, \text{ where } b = \text{room\_to\_building}[j]$$

If exam i assigns students to room j, then exam i must be assigned to the building containing room j.

4a. Minimum Group Size Constraint:

$$x_{i,j} \ge \text{min\_group\_size}(s_i) * w_{i,j},\ \forall i \in I,\ \forall j \in J$$

If exam i uses room j, it must assign at least the minimum group size to that room.

4b. Room Usage Linking Constraint:

$$x_{i,j} \le c_j * w_{i,j},\ \forall i \in I,\ \forall j \in J$$

Students can only be assigned to room j for exam i if the exam actually uses that room.

5. Small Exam Splitting Prevention:

$$\sum w_{i,j} \le 1,\ \forall i \in I \text{ where } s_i < 50\ j \in J$$

Exams with fewer than 50 students cannot be split across multiple rooms.

6. Variable Domains:

- $x_{i,j} \in \{0, 1, 2, ..., c_j\}$, $\forall i \in I$, $\forall j \in J$
- $w_{i,j}$ is binary, $\forall i \in I$, $\forall j \in J$
- $y_{i,b}$ is binary, $\forall i \in I$, $\forall b \in B$

4. Model Implementation and Results

The optimization model was implemented using PuLP, a Python linear programming library.
The Model Solution is:

- Status: Optimal
- Unused Seats: 2635
- Optimization Objective: Minimization of total unused room capacity across all exam sessions

Hence, the model successfully found an optimal solution that satisfies all constraints while minimizing wasted seating capacity throughout the examination period. The full solution can be found in the Appendix.
Our optimization model produces a feasible examination schedule that:

- Assigns all students to appropriate rooms
- Respects room capacity limits strictly
- Ensures no exam conflicts or overbooking
- Maintains building assignment consistency
- Enforces minimum group size requirements where applicable

The key improvements over the original schedule include:
1. Enhanced Room Utilization Efficiency. Our model demonstrates a more strategic room allocation, often utilizing fewer rooms per day compared to the original schedule. This reduces operational complexity and minimizes resource requirements for exam administration, while also enhancing security as fewer rooms need to be monitored.
2. Strict Capacity Compliance. Unlike the original schedule which permits overbooking (as can be seen in the current model examination in the appendix), our model maintains strict adherence to capacity constraints. This ensures exam regulations are respected (no students are seated too close to each other), comfortable examination conditions for all students, and no last-minute room reassignments due to overcrowding.
3. Improved Student Experience. The model significantly enhances the student examination experience by minimizing exam fragmentation. Students taking the same exam are kept together unless the exam size exceeds 50 students, reducing student anxiety from being separated, confusion during exam distribution and collection, and administrative overhead in managing split exams.

However there are some trade-offs and considerations that should be mentioned. While the original schedule achieves better seat utilization through overbooking practices (519), our model prioritizes operational feasibility and regulatory compliance. The 2635 unused seats represent the cost of maintaining strict capacity limits and ensuring robust scheduling that accounts for real-world constraints. Still, it has to be acknowledged that slight overbooking can be reasonable as the numbers used are exam capacity numbers. This means that there are still seats left in each room, apart from the exam capacity mentioned. Additionally, perhaps the prioritisation of some rooms, even at the cost of overbooking, was due to the room quality as well. We assumed all rooms that have an exam capacity can be used equally for exams. However, this might not be the case, as some rooms may be more accessible, easier to clean afterwards, etc.

5. Validation and Sensitivity Analysis

To evaluate the robustness and practical applicability of our model, we conducted comprehensive sensitivity analysis across multiple scenarios, including capacity stress tests, infrastructure limitations, and equipment failure simulations. Firstly, we tested the model's ability to accommodate varying student enrollment levels by scaling the student population (parameter *s*) using different multipliers (α):

Scaling Factor (α)	**Model Status**	**Unused Seats**
1.00 (baseline)	Optimal	227,896
1.01 (+1% students)	Optimal	227,680
1.02 (+2% students)	Optimal	227,632
1.05 (+5% students)	Optimal	227,393

Please note that the high number of unusead seats is due to the model calculating unused seats from al rooms and not just the rooms used for exams. This could have been solved by using a binary variable in the objective function, however, the execution time then became unreasonably long. To calculate the actual unused seats, custom code was used. Regardless, for this table, using the default number of unused seats at each threshold is alright since the point is to see the amount of decrease.

Through systematic testing, we determined that the model can accommodate up to 201% of current student enrollment ($\alpha = 2.01$) while maintaining optimal solutions. Beyond this threshold, the model becomes infeasible due to insufficient total room capacity.

Secondly, when constraining the model to use only Building A, the optimization status becomes infeasible. This result validates the model's logical behavior, as building A alone lacks sufficient capacity for the complete examination schedule, the model correctly identifies when physical constraints cannot be satisfied, and no artificial or invalid solutions are generated when resources are inadequate.

Lastly, we systematically tested scenarios where individual rooms become unavailable (simulating equipment failure, maintenance issues, or emergency closures). It was found that removal of any single room still yields optimal solutions, except for the sports hall, meaning that the current IUS infrastructure can handle individual room failures without compromising examination scheduling. The sports hall, however, seems crucial during exam season. However, when the splitting constraint is changed to not allow splitting exams with fewer than 50 students into fewer than 40 students, the model remains optimal even without the sports hall. This slight change does not affect the usability of the model but it does allow for more splits as the threshold is lower. Hence, this reduction should only be applied when necessary. The final solution is still the one with threshold 50 for exam splitting.

When it comes to the practical significance of these results, it can be seen that the capacity scaling results show:

- Short-term growth (1-5%) can be easily accommodated with minimal impact on efficiency
- Current infrastructure supports up to double the current student population
- Enrollment can increase before requiring additional facilities

The infrastructure testing reveals:

- Single room failures pose no scheduling risks, except for the sports hall
- Building-level redundancy is critical - both facilities are necessary
- Maintenance windows can be planned without disrupting examination periods

Therefore, our sensitivity analysis confirms that our optimization model:

1. Behaves logically under extreme conditions (correctly identifying infeasible scenarios)
2. Maintains robustness across realistic operational variations
3. Provides practical insights for capacity planning and risk management
4. Scales effectively to accommodate substantial growth in student populations

6. Conclusions and Recommendations

The model's aim to assign students to exam rooms while minimizing the total number of unused seats and respecting room capacities, exam schedules, and grouping constraints was successful. By balancing the given constraints with practical operational requirements, it provides a foundation for enhanced academic operations while supporting institutional growth objectives. The demonstrated robustness and scalability ensure long-term value as the university continues to expand its academic programs and student population. The strict constraints (especially regarding the splitting) that helped increase student comfort (less split exams), create cleaner logistics and easier coordination are justified due to the fact that the number of students can almost double before any scheduling issues arise. However, by relaxing the exam splitting threshold constraint, the produced schedule is still very good while eliminating the dependency on the sports hall (e.g., if the sports hall is unavailable). Some recommendations to improve the model would include connecting the IUS student information system to the model. This way, the automated data synchronization can be integrated to eliminate manual input errors and reduce preparation time. Additionally, linking with the facility management system to check for room unavailability in real time would help speed up the process of optimization even more, as no reschedules would need to be performed due to unforeseen circumstances.

APPENDICES (50 points)

The link to the Google Drive where all relevant files are located.

Not included for data protection reasons. Removed before uploading to Github.