Examination Room Usage Optimization Via Greedy and Exact Methods: A Vietnamese University Case Study

Minh Hieu LE [a], Surya HANJAYA [a], Trang HOANG [a] and Xuan-Bach LE [a,1]
[a] *Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology, VNU-HCM, Ho Chi Minh City, Vietnam*

Abstract. Organizing university examinations is costly, largely because many rooms must be opened for each exam session. In practice, conservative policies and course-specific constraints often fragment room usage, leaving much capacity unused. Unlike traditional examination timetabling research, which jointly optimizes time slots and room assignments, we focus on the post-timetable problem of improving room utilization once exam schedules are already fixed. This paper studies how to reduce the number of rooms required for fixed exam sessions while respecting practical constraints such as room capacities, campus boundaries, and course-separation rules. We consider two scenarios: improving existing assignments through post-processing and allocating rooms from scratch. For both, we propose a fast greedy heuristic based on a best-fit decreasing strategy, together with an Integer Linear Programming (ILP) model that produces optimal solutions for moderately sized instances. Experiments on real examination data from Ho Chi Minh City University of Technology (HCMUT), a large multi-campus public university in Vietnam, show substantial room savings and illustrate the trade-off between the scalability of heuristics and the optimality of exact methods.

Keywords. examination timetabling, room allocation, integer linear programming, greedy heuristic, room utilization optimization

1. Introduction

Examinations are a core component to assess student performance [1,2]. However, large examination periods also create significant operational costs, especially for room allocation and invigilation [3]. In multi-campus universities, these costs grow with the number of rooms opened in each session [4]. A case study at Ho Chi Minh City University of Technology (HCMUT) [5] shows that the baseline allocation uses rooms inefficiently, with only 39.5% average utilization, while supervision requires more than 17,800 staff-hours per semester.

In practice, exam room assignments are often conservative to meet capacity, spacing, and supervision regulations [6,7]. This often leads to fragmented room usage within the same time slot. For example, two groups of 20 students from different courses may

[1]Corresponding Author: Xuan-Bach Le. E-mail: lexuanbach@hcmut.edu.vn. Data and artifact: `https://github.com/hieukendu/ILP`.

be placed in separate rooms of capacity 40 because spacing rules require empty seats between students. This fragmentation increases supervision and facility costs [8]. Improving room utilization within fixed exam schedules can therefore reduce these inefficiencies without changing academic policies or timetables [9]. In the example above, we can place both groups in the same room to use the empty seats efficiently. Exam integrity is maintained because students from different courses receive different exam papers.

The problem is further complicated in multi-campus settings. In the case of HCMUT, examinations are held concurrently at two campuses located approximately 28 km apart. Students cannot be reassigned across campuses during an examination session, while invigilators frequently travel between campuses using university-organized transportation. These constraints impose strict campus-level separation and limit the applicability of naive consolidation strategies.

Most existing research on examination timetabling focuses on the integrated assignment of exams to time slots and rooms [10,11,12]. By contrast, relatively little attention has been paid to optimizing room usage once examination schedules are fixed. In practice, timetables are determined well in advance and are difficult to revise, whereas room assignments remain more flexible. This motivates the study of room-level optimization as a standalone and practically relevant problem. Although closely related to classical bin packing and set-partitioning formulations [13,14,15], the problem involves additional constraints, including course separation, reduced examination capacities, and campus-level restrictions. Thus, unlike classical exam timetabling, time slots are fixed and only room usage is optimized, making it a post-timetable operational problem.

This paper addresses the challenge of minimizing the number of examination rooms used during fixed exam sessions across multiple campuses. Exams occur at set dates and times, and courses may be split across rooms due to spacing and capacity constraints. To maintain academic integrity, students from the same course are not co-located, while groups from different courses may share rooms if space allows. Reassignments are limited to rooms within the same campus.

To investigate this operational problem, we study two practical room-optimization scenarios that capture common university practices. One improves room utilization by merging compatible assignments after the initial schedule, while the other allocates rooms from scratch under the same constraints to assess possible efficiency gains. In addition, we prove that the decision version of the problem is NP-complete and that the greedy algorithm achieves a 2-approximation under identical room capacities. Rather than proposing a fundamentally new optimization paradigm, this work contributes a deployment-oriented formulation and evaluation of post-timetable examination room optimization under realistic multi-campus constraints. The empirical results show that a simple greedy strategy can achieve near-optimal reductions at substantially lower computational cost than exact optimization, which is practically important for time-critical examination planning.

The remainder of the paper is organized as follows. Section 2 reviews related work. Section 3 defines the room optimization problem with heterogeneous capacities and course-separation rules. Section 4 presents two approaches: a fast greedy heuristic and an ILP model for optimal solutions on moderate instances. Section 5 reports experiments on HCMUT exam data and analyzes the trade-off between solution quality and runtime. Section 6 concludes and outlines future work.

2. Related Work

Examination timetabling assigns exams to time slots and rooms while satisfying institutional constraints. This operations research problem has been widely studied due to its real-world complexity. Early studies highlight challenges like room capacities, exam splitting, institutional rules [11,12]. More recent work focuses on practical deployments using real data and large-scale case studies [16,17].

Within this broader context, a number of works focus specifically on the *room assignment* subproblem, where exam times are predetermined and the task reduces to optimally assigning rooms. This problem is commonly modeled as a capacity-constrained optimization problem, closely related to bin packing [14,15] and set partitioning [13]. Notable examples include the work of Dammak et al. [18] and Ayob and Malik [19], which address related room-assignment problems under capacity constraints. While these studies focus on capacity-feasible room allocation, our work differs in that we study room utilization optimization after examination timetables have already been fixed, incorporating additional operational constraints such as campus separation and reduced examination capacities. Moreover, most prior studies evaluate algorithmic performance primarily from a computational perspective, whereas our work emphasizes practical deployment considerations in real multi-campus examination environments.

Integer Linear Programming (ILP) and mixed-integer programming (MIP) are widely used in examination timetabling because they can model complex constraints and produce optimal solutions. Both early studies [20] and more recent case studies [21] demonstrate their effectiveness, particularly for medium-sized problems. ILP has also been applied to room allocation. For example, Lemos et al. [9] combine MIP with local search to improve room utilization. Recent exact methods, including branch-and-bound and decision diagrams [22,23], highlight both the power and scalability limits of ILP.

Because exact methods scale poorly, exam scheduling often relies on heuristics and metaheuristics. Surveys such as Siew et al. [24] review approaches including adaptive and hyper-heuristics [25,26] and hybrid methods effective on complex benchmarks. However, these methods often require complex algorithms and careful parameter tuning. In practice, institutions prefer simpler solutions. Our greedy room-merging algorithm follows this philosophy, providing a simple yet effective heuristic that complements the ILP model and supports scalable room optimization under realistic constraints.

3. Problem Formalism

We study the problem of optimizing university exam room usage. Exams are organized into independent sessions (fixed time slots) where multiple courses run in parallel, allowing each session to be optimized separately.

Let $\mathscr{T}$ denote the set of examination sessions. For each session $t \in \mathscr{T}$, let $\mathscr{R}_t$ be the set of rooms available during that session. Each room $j \in \mathscr{R}_t$ has a physical capacity c_j and an exam capacity $\hat{c}_j \leq c_j$, which accounts for spacing and supervision constraints. At HCMUT, exam capacity is typically about half of the physical capacity ($\hat{c}_j \approx 0.5 c_j$).

Let $\mathscr{K}$ denote the set of courses. Each course is divided into one or more *candidate groups*, indexed by $i \in \mathscr{I}$. Each group represents a cohort of students that must be seated together, has size s_i, belongs to exactly one course $k(i) \in \mathscr{K}$, and participates in exactly one examination session $t(i) \in \mathscr{T}$.

Candidate groups are *indivisible*: all students in a group must be assigned to a single examination room, and any reassignment moves the group as a whole. Groups from different courses may share the same room if space allows.

An assignment to room j is *feasible* if: (i) total assigned students do not exceed the room capacity, (ii) per-course students do not exceed the exam capacity, and (iii) no two groups belong to the same course. Formally,

$$\sum_{i:\, i \to j} s_i \le c_j, \qquad \sum_{i:\, i \to j,\, k(i)=k} s_i \le \hat{c}_j, \quad \forall k \in \mathscr{K}, \qquad \sum_{i:\, i \to j,\, k(i)=k} 1 \le 1, \quad \forall k \in \mathscr{K}. \tag{1}$$

A room is *open* in session t if at least one candidate group is assigned to it, and *closed* otherwise. Since each room incurs a fixed cost, the objective for each session is to minimize the number of open rooms by packing candidate groups from different courses into shared rooms while respecting all feasibility constraints.

We consider two closely related optimization scenarios:

1. **Room merging (post-processing).** Each session begins with a feasible room assignment, typically generated by the university's existing system. The goal is to reduce the number of rooms in use by reassigning entire student groups among the already assigned rooms. No new rooms can be added, and any room left empty is closed. All reassignments must respect capacity limits and course-separation rules.
2. **Room assignment from scratch.** No initial assignment is provided. Candidate groups are assigned directly to rooms so as to minimize the number of open rooms, subject to physical capacity, exam capacity, indivisibility, and course-separation constraints.

These two settings reflect common operational practice and analytical needs. Post-processing supports low-friction improvements to legacy systems, while the from-scratch setting provides a clean baseline that reveals the theoretical limits of room consolidation under the same constraints. Together, these two scenarios allow us to compare practical improvements achievable in existing deployments with the theoretical efficiency attainable under the same institutional constraints.

4. Proposed Methods

This section outlines two approaches: a scalable greedy heuristic suitable for practical, large-scale use (Section 4.1), and an ILP model that delivers optimal solutions and serves as a benchmark for comparison (Section 4.2).

4.1. Greedy Assignment Algorithm

The greedy algorithm (Algorithm 1) operates on each exam session and aims to minimize the number of rooms by packing candidate groups from different courses into shared rooms, subject to capacity and course-separation constraints. We consider two variants, aligned with the scenarios introduced in Section 3.

Case 1: Room merging (post-processing). An initial assignment A_0 is provided, where each candidate group is assigned to a room. The objective is to keep only a subset

Algorithm 1: Greedy assignment for exam session t

Input: Candidate groups $\mathscr{G}_t$ with sizes s_i, courses $k(i)$, original rooms $\text{room}(i)$, exam capacities $\hat{c}_i$; available rooms $\mathscr{R}_t$ with physical capacities c_j and exam capacities $\hat{c}_j$; optional initial assignment A_0

Output: Assignment A and open rooms $\mathscr{O}$

$\mathscr{G}_t \leftarrow \text{SortDesc}(\mathscr{G}_t, s)$

if $A_0 \neq \emptyset$ **then**

  `// Case 1: room merging (post-processing)`

  $A \leftarrow \emptyset$; $\mathscr{O} \leftarrow \emptyset$; bins $\leftarrow [\,]$

  foreach $i \in \mathscr{G}_t$ **do**

    find a feasible bin b with sufficient remaining physical capacity, with group size $s_i \leq \hat{c}_b$, and no course conflict

    if *no such bin exists* **then**

      open a new bin using $\text{room}(i)$ as target

      add it to $\mathscr{O}$

    assign i to the selected bin and update its load

else

  `// Case 2: direct assignment from scratch`

  $\mathscr{R}_t \leftarrow \text{SortDesc}(\mathscr{R}_t, c)$ `// Open large rooms first.`

  $A \leftarrow \emptyset$; $\mathscr{O} \leftarrow \emptyset$

  foreach $i \in \mathscr{G}_t$ **do**

    find a feasible room $j^\star \in \mathscr{O}$ that minimizes the remaining capacity

    if *no such room exists* **then**

      open the next room $j \in \mathscr{R}_t$ and set $\mathscr{O} \leftarrow \mathscr{O} \cup \{j\}$

      $j^\star \leftarrow j$

    $A \leftarrow A \cup \{(i \rightarrow j^\star)\}$

return $A, \mathscr{O}$

of these rooms open by merging groups into them. No new rooms may be opened, and rooms left empty are closed.

The algorithm treats each group as indivisible and applies a best-fit decreasing strategy. Groups are sorted in descending order of size and assigned to the first feasible open room. A room is feasible for such a group if: (i) it has enough remaining physical capacity, (ii) its exam capacity is not violated, (iii) it does not contain a group from the same course. If no such room exists, the group's original room is retained. Rooms not selected as targets are implicitly closed.

Case 2: Direct assignment from scratch. When no initial assignment exists, rooms are selected directly from $\mathscr{R}_t$. Rooms are first sorted in descending order of physical capacity and opened greedily as needed. Candidate groups are processed in descending order of size. Each group is assigned to a feasible open room that minimizes the remaining capacity. If no open room is feasible, the next unused room in $\mathscr{R}_t$ is opened.

Discussion. The post-processing variant limits room usage by selecting a subset of initially assigned rooms and closing the rest, never opening new ones. In contrast, the

direct-assignment variant opens rooms from the available pool in descending order of capacity and assigns groups greedily. Both approaches are fast and scalable but do not guarantee optimal solutions, as early decisions may block better overall assignments. This trade-off between optimality and efficiency is well known in bin packing and exam timetabling literature [14,15,11,25].

4.2. Integer Linear Programming Formulation

We formulate the exam room assignment problem as an ILP, which yields optimal solutions and serves as a benchmark for comparison. As in the heuristic approach, the formulation distinguishes between post-processing (room merging) and full assignment.

Decision variables. For each candidate group i and room j available in the same exam session, we introduce a binary assignment variable:

$$x_{ij} = \begin{cases} 1, & \text{if candidate group } i \text{ is assigned to room } j, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

For each room j, we define a binary activation variable:

$$y_j = \begin{cases} 1, & \text{if room } j \text{ is kept open,} \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Optimization scenarios. The ILP can be instantiated in two modes.

- **Room merging (post-processing).** An initial assignment A_0 is given, together with the set of rooms $\mathscr{R}_0$ that are used in A_0. In this mode, no new rooms may be opened and the solver may close rooms in $\mathscr{R}_0$ and reassign their groups. Formally, we restrict feasible rooms to $\mathscr{R}_0$ by enforcing:

$$y_j = 0 \; \forall j \notin \mathscr{R}_0 \quad \text{and} \quad x_{ij} = 0 \; \forall i, \; \forall j \notin \mathscr{R}_0. \tag{4}$$

  The activation variables y_j for $j \in \mathscr{R}_0$ remain decision variables, allowing the ILP to select a minimum-cardinality subset of rooms to keep open.
- **Room assignment from scratch.** No initial assignment is given. All variables $x_{ij} \in \{0,1\}$ and $y_j \in \{0,1\}$ are optimized freely, allowing rooms to be opened or closed as needed. This setting yields the minimum possible number of rooms for each session under the full set of constraints.

Objective. In both cases, we want to minimize the number of open rooms:

$$\min \sum_j y_j. \tag{5}$$

Assignment constraints. Each candidate group is assigned to exactly one room:

$$\sum_j x_{ij} = 1, \quad \forall i. \tag{6}$$

Exam constraints. Two constraints are imposed: (i) candidate groups belonging to the same course cannot share the same examination room, and (ii) each assigned group must satisfy the room's examination capacity.

$$\sum_{i:k(i)=k} x_{ij} \le y_j \quad \forall j \in \mathscr{R}, \forall k \in \mathscr{K} \quad \text{and} \quad s_i x_{ij} \le \hat{c}_j \quad \forall i \in \mathscr{G}, \forall j \in \mathscr{R}. \tag{7}$$

Physical constraints. The total number of students in a room must satisfy:

$$\sum_i s_i x_{ij} \le c_j y_j, \quad \forall j. \tag{8}$$

Room activation constraints. Groups can only be assigned to open rooms:

$$x_{ij} \le y_j, \quad \forall i, j. \tag{9}$$

For a session with n groups, m rooms, and $|\mathscr{K}|$ courses, the model has $O(nm)$ binary variables and $O(nm + m|\mathscr{K}|)$ constraints. Solution time grows with size, but the ILP remains tractable for moderate sessions and provides optimal benchmarks.

4.3. Complexity Analysis

We study the decision version of the exam room allocation problem:

Problem 4.1: Exam Room Allocation Decision Problem (ERAD)

The input consists of a set of candidate groups $\mathscr{G}$ and a set of available rooms $\mathscr{R}$. Each group i has size s_i and course label $k(i)$. Each room j has a physical capacity c_j and an examination capacity $\hat{c}_j$.
Given a bound B on the number of rooms, determine whether all groups can be assigned to at most B rooms while satisfying the physical capacity, examination capacity, and course-separation constraints.

Theorem 1. *The Exam Room Allocation Decision Problem is NP-complete.*

Proof. ERAD is in NP since a candidate assignment can be verified in polynomial time by checking the assignment and all constraints.

To prove NP-hardness, we reduce from the Bin Packing decision problem. Given items with sizes $a_1, \ldots, a_n$, bin capacity C, a limit of B bins, construct an ERAD instance as follows. For each item i, create a group g_i with size $s_i = a_i$. Create B rooms, each with capacities $c_j = \hat{c}_j = C$. Assign each group a distinct course label so that course-separation constraints are inactive.

A packing into at most B bins exists iff the ERAD instance admits an assignment using at most B rooms. Hence Bin Packing reduces to ERAD in polynomial time, implying ERAD is NP-complete. □

Theorem 2. *If all rooms have identical capacity C, the direct greedy assignment algorithm is a* 2*-approximation. That is, ALG* $\le 2\,OPT$.

Table 1. Overview of the HCMUT centralized examination period (Semester 251).

Statistic	Overall	Campus 1	Campus 2
Total candidate demand	115,641	39,245	76,396
Total room usages	3,392	1,377	2,015
Estimated supervision workload (hours)	17,808	7,807	10,001
Number of examination rooms	202	98	104
Average exam capacity per room	42	31	53
Average physical capacity per room	80	58	100
Average students per room	34	29	38
Avg physical utilization (%)	**39.5**	**42.2**	**37.7**

Proof. Let ALG be the number of rooms opened by the greedy algorithm, $L_1, \ldots, L_{ALG}$ their loads, and $S = \sum_i s_i$ the total number of candidates. Since each room has capacity C, any solution using OPT rooms satisfies:

$$S \leq OPT \cdot C.$$

In the greedy process, a new room is opened only if the current group of size s cannot fit into any existing room. Hence for every open room R_ℓ:

$$L_\ell + s > C.$$

In particular, for the most recently opened room R_t, $L_t + s > C$. After opening a new room R_{t+1} and assigning the group to it, we get: $L_t + L_{t+1} > C$. Thus every pair of consecutive rooms has combined load greater than C, so:

$$S = \sum_{j=1}^{ALG} L_j > \left\lfloor \frac{ALG}{2} \right\rfloor C.$$

Combining this with the inequality $S \leq OPT \cdot C$ yields:

$$ALG \leq 2\,OPT.$$

Thus the greedy algorithm is a 2-approximation. □

The NP-completeness result shows that optimal solutions are hard to compute in general, which motivates the use of heuristics. The approximation theorem guarantees that the greedy algorithm uses at most twice the minimum number of rooms under identical capacities. This guarantee is restricted to the identical-capacity direct-assignment setting and should be interpreted as a theoretical baseline rather than a direct bound for the heterogeneous room capacities considered in our case study.

5. Case Study

We evaluate the proposed methods using a real-world case study from Ho Chi Minh City University of Technology (HCMUT) [5]. In semester 251 (August–December 2025),

HCMUT held a centralized 19-day examination period with up to five sessions per day, running in parallel across two campuses 28 km apart.

Each course was usually offered at both campuses, with students taking the same exam simultaneously. Examination logistics involve significant costs, as each room requires two invigilators per session, plus a 5% reserve for contingencies. Invigilators often travel between campuses by university buses, adding coordination overhead.

Table 1 summarizes key details of the exam period. A total of 81 session slots were scheduled, resulting in 3,392 room usages under the baseline allocation. This corresponds to an estimated supervision workload of about 17,808 staff-hours, nearly two years of labor. Despite the scale, room utilization remains low. Rooms vary in size, and due to spacing and supervision constraints, exam capacity is typically limited to around 50% of each room's physical capacity.

The physical utilization of a room r and the average physical utilization over the examination period are defined as:

$$u_r^{\text{phys}} = \frac{n_r}{c_r} \quad \text{and} \quad \bar{u}^{\text{phys}} = \frac{1}{|\mathscr{R}^{\text{open}}|} \sum_{r \in \mathscr{R}^{\text{open}}} u_r^{\text{phys}}. \tag{10}$$

where n_r is the number of students assigned to room r, c_r is its physical capacity, and $\mathscr{R}^{\text{open}}$ the set of all rooms that are open during the examination period.

Under the baseline allocation, average physical room utilization is only 39.5%, and even lower at Campus 2 (37.7%). This low utilization leaves substantial room for cost reduction by co-locating candidate groups from different courses within the same room, while respecting capacity and supervision constraints.

5.1. Experimental Setup

The input data consist of official room allocation plans for each examination session, including group sizes, course identifiers, room capacities, and campus information. Examinations are handled separately by campus: within each session, rooms are assigned only to candidate groups from the same campus, and no room is shared across campuses.

Both methods are evaluated relative to the baseline allocation produced by the university scheduling system. We consider three evaluation metrics: (i) room reduction with respect to the baseline, (ii) average physical room utilization after merging, and (iii) computational running time.

For the ILP approach, we use the CBC solver via the PuLP interface [27]. Each examination day (up to five sessions) is solved per session with a 15-minute time limit. A day is marked as a *timeout* if any session exceeds the limit, in which case the best incumbent solution is retained. Experiments run on a Mac mini (Apple M4, 10 cores, 16 GB RAM). Days are processed independently to reflect operational constraints. To improve reproducibility, the full exam-room instance dataset used in this study is provided as supplementary material. The source code, implementation scripts, and usage instructions are publicly available in the project repository: `https://github.com/hieukendu/ILP`. The released dataset contains the fields required by the optimization model, including examination session identifiers, campus identifiers, course identifiers, room identifiers, group sizes, physical room capacities, and examination capacities. Student-level records and personally identifiable student information are not used by the model and are not included in the released dataset.

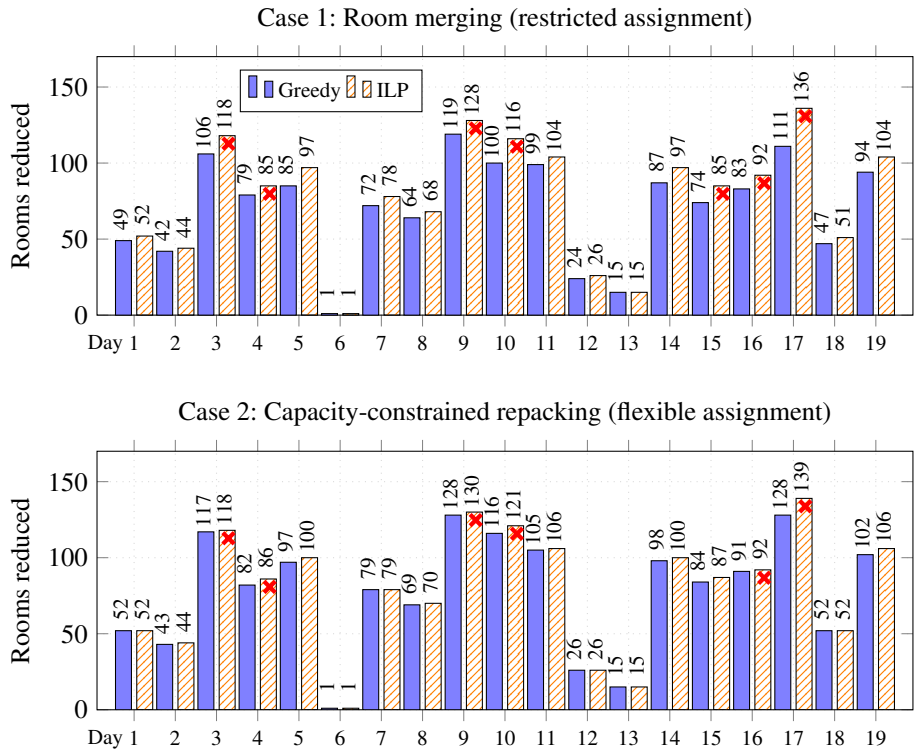

Figure 1. Room reduction under restricted (Case 1) and flexible (Case 2) regimes. Days marked with * indicate ILP timeouts (15 minutes); feasibility is still preserved.

The ILP models were implemented in Python 3.12.7 using PuLP 3.3.0 and solved with CBC 2.10.3 through the `PULP_CBC_CMD` interface. Each exam session was solved independently with a 900-second time limit. CBC solver output was enabled during execution. The relative MIP gap, absolute MIP gap, thread count, presolve, and cutting-plane parameters were not explicitly modified and were therefore left at the default CBC/PuLP settings. When CBC returned an optimal solution, the corresponding objective value and assignment were recorded. When the time limit was reached, the best feasible incumbent solution, if available, was retained and the session was marked as a timeout. In the post-processing setting, the baseline allocation is assumed feasible; therefore, if a group cannot be merged into any feasible target room, it remains assigned to its original room. In the from-scratch setting, the greedy algorithm opens additional rooms from the same session and campus as needed. If no feasible room remains, the instance is flagged as infeasible and the original operational allocation is retained. No cross-campus reassignment is allowed in any setting.

5.2. Evaluation Results

This section evaluates the effectiveness and computational performance of the proposed greedy heuristic and ILP-based optimization under two assignment regimes. Case 1 restricts assignments to room merging, while Case 2 allows flexible, capacity-constrained

Table 2. Room usage and room utilization comparison (best results in **bold**).

	Base	Greedy1	ILP1	Greedy2	ILP2
Room usages	3,392	2,041	1,895	1,907	**1,868**
Reduction (%)	–	39.8%	44.1%	44.8%	**44.9%**
Avg. phys. util.	39.5%	**63.9%**	60.3%	51.2%	56.1%

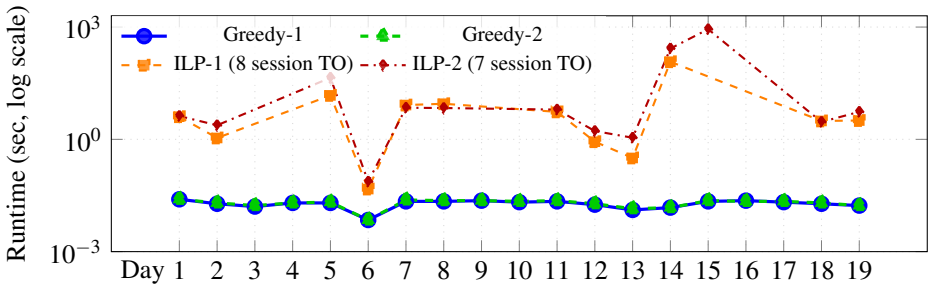

Figure 2. Runtime comparison between the Greedy and ILP approaches (Cases 1–2) on a logarithmic y-axis; days with session timeouts are omitted.

reassignment. We examine three aspects: room reduction relative to the baseline, average physical room utilization, and runtime performance. Aggregate results for the first two metrics are summarized in Table 2, with day-level trends shown in Figure 1.

Room reduction and utilization. Figure 1 shows the number of rooms reduced per exam day relative to the baseline. Across all 19 sessions, both methods achieve substantial reductions, confirming over-provisioning in the original schedules.

In Case 1, the greedy heuristic removes 1,351 rooms (39.8%), while the ILP formulation achieves a larger reduction of 1,497 rooms (44.1%), as summarized in Table 2. This corresponds to an absolute improvement of 4.3 percentage points in favor of ILP, reflecting the benefit of global optimization over local packing. Among all methods, the Case 1 greedy heuristic attains the highest average physical room utilization (63.9%), representing a 1.6 times increase over the baseline utilization of 39.5%, as its aggressive local packing strategy prioritizes filling rooms as tightly as possible.

In Case 2, both methods benefit from the added flexibility of capacity-constrained repacking. The greedy heuristic reduces 1,485 rooms (44.8%), while ILP achieves 1,524 rooms (44.9%). The 0.1% gap (Table 2) shows that with flexible reassignment the greedy approach is nearly as effective as ILP.

Runtime performance and scalability. Figure 2 compares runtimes on a logarithmic scale. The greedy heuristic is consistently fast and stable across all sessions and both regimes, with a total runtime of about 0.37 seconds over 19 days in Case 1 and a comparable time in Case 2, demonstrating strong scalability.

In contrast, ILP runtimes are highly variable and sensitive to both instance structure and assignment regime. While many sessions solve within seconds, others exhibit sharp runtime spikes, with several exceeding hundreds of seconds and reaching the 15-minute limit. Overall, ILP incurs 8 session timeouts in Case 1 and 7 in Case 2. Although the best feasible solutions before timeout remain valid, these results highlight the limited

predictability of exact optimization in large, tightly constrained settings. Case 2 increases runtime variability due to the larger reassignment search space.

Comparative analysis and implications. Overall, the results highlight a clear trade-off between solution quality and efficiency. ILP provides a modest advantage under restricted assignment, improving room reduction by 4.3% in Case 1, but this shrinks to just 0.1% in Case 2. By contrast, the greedy heuristic is orders of magnitude faster and incurs no timeouts in either regime, making it more practical for large-scale deployment.

These findings show that flexible assignment shifts the balance toward the greedy heuristic. With repacking allowed, the greedy approach achieves near-optimal room reduction at minimal cost, making it well suited for large-scale and time-critical planning. ILP remains useful for benchmarking and offline analysis, but its small gains are unlikely to justify the higher and less predictable computational cost in practice.

6. Conclusion and Future Work

We studied post-hoc examination room optimization under restricted and flexible assignment regimes, comparing a greedy heuristic with an ILP formulation. While ILP yields a modest advantage under restricted assignment (4.3%), this gap shrinks to 0.1% with flexible reassignment, whereas the greedy approach remains orders of magnitude faster and incurs no timeouts. These findings suggest that flexible assignment largely eliminates the practical benefits of exact optimization. Future work will explore multi-objective extensions, dynamic and uncertain settings, and integration into real-world examination scheduling systems, as well as evaluate the proposed methods on examination datasets from multiple institutions to assess robustness and generalizability across different room inventories, scheduling policies, and campus structures.

Acknowledgements

This research is funded by Murata Science and Education Foundation under grant number 26VH05. We acknowledge the support of time and facilities from Ho Chi Minh City University of Technology (HCMUT), VNU-HCM for this study.

References

[1] Abdul-Rahman S, Burke EK, Bargiela A, McCollum B, Özcan E. A constructive approach to examination timetabling based on adaptive decomposition and ordering. Annals of Operations Research. 2014;218:3-21.

[2] Burke EK, Petrovic S. Recent research directions in automated timetabling. European Journal of Operational Research. 2002;140(2):266-80.

[3] Qu R, Burke EK, McCollum B, Merlot LTG, Lee SY. A survey of search methodologies and automated system development for examination timetabling. Comput Oper Res. 2009;37(3):397-409.

[4] Gaspero LD, Schaerf A. Tabu Search Techniques for Examination Timetabling. In: Selected Papers from the Third International Conference on Practice and Theory of Automated Timetabling III. PATAT '00. Berlin, Heidelberg: Springer-Verlag; 2000. p. 104—117.

[5] Ho Chi Minh City University of Technology. Ho Chi Minh City University of Technology (HCMUT); 2025. Accessed: 2025-01. https://www.hcmut.edu.vn.

[6] Ahuja RK, Orlin JB, Tiwari A. A greedy genetic algorithm for the quadratic assignment problem. Comput Oper Res. 2000;27:917-34.

[7] Wasfy A, Aloul FA. Solving the University Class Scheduling Problem Using Advanced ILP Techniques. In: GCC Conference; 2007. .
[8] Ceschia S, Di Gaspero L, Schaerf A. Design, Engineering, and Experimental Analysis of a Simulated Annealing Approach to the Post-Enrolment Course Timetabling Problem. Comput Oper Res. 2011 04;39.
[9] Lemos A, Melo FS, Monteiro PT, Lynce I. Room Usage Optimization in Timetabling: A Case Study at Universidade de Lisboa. Operations Research Perspectives. 2019;6:100092.
[10] Schaerf A. A survey of automated timetabling. Artificial Intelligence Review. 1999;13:87-127.
[11] Kahar MNM, Kendall G. The Examination Timetabling Problem at Universiti Malaysia Pahang: Comparison of a Constructive Heuristic with an Existing Software Solution. European Journal of Operational Research. 2010;207(2):557-65.
[12] Bergmann LK, Fischer K, Zurheide S. A Linear Mixed-Integer Model for Realistic Examination Timetabling Problems. In: PATAT; 2014. p. 82-101.
[13] Ryan DM, Foster BA. An Integer Programming Approach to Scheduling. In: Computer Scheduling of Public Transport. North-Holland; 1981. p. 269-80.
[14] Coffman EG, Garey MR, Johnson DS. Approximation Algorithms for Bin Packing: A Survey. In: Hochbaum DS, editor. Approximation Algorithms for NP-Hard Problems; 1996. p. 46-93.
[15] Delorme M, Iori M, Martello S. Bin Packing and Cutting Stock Problems: Mathematical Models and Exact Algorithms. European Journal of Operational Research. 2016;255(1):1-20.
[16] Sigurpálsson ÁÖ. Examination timetable modelling at the University of Iceland [Master's thesis]. Reykjavík, Iceland: University of Iceland; 2017.
[17] Schininà R. An Integer Programming Model for Timetabling at the University of Groningen [Bachelor's thesis]. University of Groningen; 2024.
[18] Dammak A, Elloumi A, Kamoun H. Classroom Assignment for Exam Timetabling. Advances in Engineering Software. 2006;37(10):659-66.
[19] Ayob M, Malik A. A New Model for an Examination-Room Assignment Problem. International Journal of Computer Science and Network Security. 2011;11(10):255-62.
[20] Boland NL, Hughes BD, Merlot LTG, Stuckey PJ. New integer linear programming approaches for course timetabling. Computers & Operations Research. 2008;35(7):2209-33.
[21] Laisupannawong T, Sumetthapiwat S. An ILP Model for the Examination Timetabling Problem: A Case Study of a University in Thailand. Advances in Operations Research. 2024;2024:1-17.
[22] González JE, Ciré AA, Lodi A, Rousseau LM. Integrated IP and Decision Diagram Search Tree with an Application to the Maximum Independent Set Problem. Constraints. 2020;25(1):23-46.
[23] Amaldi E. Branch and Bound Method; 2019. Accessed: 2026-01-21. Lecture notes, Foundations of Operations Research, Politecnico di Milano.
[24] Siew ESK, Sze SN, Goh SL, Kendall G, Sabar NR, Abdullah S. A Survey of Solution Methodologies for Exam Timetabling Problems. IEEE Access. 2024;12:41479-98.
[25] Burke EK, Bykov Y, Newall JP, Qu R. A Graph-based Hyper-heuristic for Educational Timetabling Problems. European Journal of Operational Research. 2007;176(1):177-92.
[26] Soghier A, Qu R. Adaptive Selection of Heuristics for Assigning Time Slots and Rooms in Exam Timetables. Applied Intelligence. 2013;39(2):438-50.
[27] Mitchell S, O'Sullivan M, Dunning I. PuLP: A Linear Programming Toolkit for Python; 2011. Available from: `https://coin-or.github.io/pulp/`.