Reducing Room Usage in University Exams: A Case Study with Greedy and Exact Methods

No Author Given

No Institute Given

Abstract. Organizing university examinations involves considerable operational costs, much of which stem from the number of rooms opened during each session. Due to conservative allocation policies and course-specific constraints, room usage is often fragmented, leading to underutilized capacity. This paper examines how to reduce the number of examination rooms required for fixed exam sessions while respecting practical constraints such as room capacities, campus boundaries, and course-separation rules. We explore two scenarios: refining existing room assignments through post-processing and assigning rooms from scratch. For both, we introduce a fast greedy heuristic based on a best-fit decreasing approach, alongside an Integer Linear Programming (ILP) model that delivers optimal solutions for moderately sized instances. Using real-world data from a multi-campus university, our experiments show substantial room savings and highlight the trade-off between the scalability of heuristics and the optimality of exact methods.

1 Introduction

Examinations are a central component of university education, serving as the primary mechanism for assessing academic performance [1,7]. Beyond their pedagogical role, large-scale examination periods impose substantial operational costs, particularly in room allocation and invigilation [20]. In multi-campus institutions, these costs scale with the number of rooms opened per session and can create significant logistical and financial burdens [13]. In a real-world case study conducted at Ho Chi Minh City University of Technology (HCMUT) [15], average physical room utilization under the baseline allocation is only 39.5%, while supervision workload is estimated to exceed 17,800 staff-hours in one semester.

In practice, room assignments tend to be conservative to comply with regulations on capacity, spacing, and supervision requirements [2,26]. As a result, it is common to see fragmented room usage during the same time slot. For instance, two groups of 20 students from different courses might be placed in separate rooms that each hold 40, leaving both rooms half full only. This over-allocation drives up unnecessary costs in supervision, preparation, and facility maintenance [9]. Enhancing room utilization within fixed exam schedules is a practical way to reduce such costs without altering academic policies or timetables [18].

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 [22,16,5]. 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 [21,10,12], the problem involves additional constraints, including course separation, reduced examination capacities, and campus-level restrictions.

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.

We explore two optimization scenarios grounded in real-world practice. The first reflects current operations, where room usage is improved by merging compatible assignments after initial scheduling. The second models room allocation from scratch under the same constraints, offering a benchmark for potential efficiency gains and guiding future system design.

The remainder of the paper is structured as follows. We begin with a review of related literature in Section 2. Section 3 introduces a formal definition of the room optimization problem, incorporating key real-world features such as heterogeneous room capacities and course separation rules. In Section 4, we present two solution approaches: a fast greedy heuristic that delivers high-quality results efficiently, and an Integer Linear Programming (ILP) formulation that computes optimal solutions for moderate-sized instances, serving as a benchmark. Section 5 details our empirical evaluation using real examination data from HCMUT. Our results demonstrate significant room savings and analyzing the trade-offs between optimality and computational cost. Finally, Section 6 summarizes our findings and outlines directions for future work.

2 Related Work

Examination timetabling involves assigning exams to time slots and rooms while meeting institutional constraints. This well-established problem in operations research has been extensively studied with a focus on real-world complexity. Early work by Kahar and Kendall [16] and Bergmann et al. [5] highlights key challenges such as room capacities, exam splitting, and institutional rules. Recent studies focus on practical relevance, using real data and large-scale case studies [27,23].

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 [10,12] and set partitioning [21]. Notable examples include the work of Dammak et al. [11] and Ayob and Malik [4], which directly align with our problem formulation. Like our approach, these methods seek to maximize room utilization while satisfying all room capacity constraints.

ILP and mixed-integer programming (MIP) are popular tools for examination timetabling due to their ability to model complex constraints and provide optimal solutions. Foundational work [6] and recent case studies [17] show their effectiveness, especially for mid-sized instances. ILP has also been applied specifically to room allocation, as in the work of Lemos et al. [18], which combines MIP with local search to improve room usage. Recent advances in exact methods, including branch-and-bound and decision diagram techniques [14,3], highlight both the potential and the scalability challenges of ILP in large, real-world settings.

Due to the scalability limits of exact methods, heuristics and metaheuristics remain widely used in exam scheduling. Surveys like Siew et al. [24] highlight a range of approaches, from adaptive and hyper-heuristics [8,25] to hybrid techniques, many of which perform well on complex benchmarks. However, these methods often involve significant algorithmic complexity and parameter tuning. In institutional settings, simpler strategies are often preferred. Our greedy room-merging algorithm follows this practical line-offering a fast, easy-to-deploy alternative that pairs well with the more intensive ILP approach.

3 Problem Formalism

We address the problem of optimizing university exam room usage, where exams are organized into independent sessions, each representing a fixed time slot during which multiple courses are examined in parallel. Since exam sessions are independent, the optimization problem can be solved separately for each session.

Let $\mathcal{T}$ denote the set of examination sessions. For each session $t \in \mathcal{T}$, let $\mathcal{R}_t$ be the set of rooms available during that session. Each room $j \in \mathcal{R}_t$ has a physical capacity c_j and a reduced exam capacity $\hat{c}_j \leq c_j$, which accounts for spacing, supervision, and regulatory constraints.

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

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. For a room j, an assignment is feasible if (i) the total number of assigned students does not exceed the physical capacity and, (ii) for each course, the number of students from that course does not exceed the exam capacity. Formally,

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

A room is *open* in session t if at least one candidate group is assigned to it, and *closed* otherwise. Since opening a room incurs a fixed operational cost regardless of occupancy, 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.

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 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 that has enough remaining exam capacity and does not already 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.

In this setting, exam capacity is the binding constraint. Since the initial assignment is assumed to be feasible, physical seating limits are not re-enforced.

Algorithm 1: Greedy assignment for exam session t

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

$\mathcal{G}_t \leftarrow \text{SortDesc}(\mathcal{G}_t, s)$
if $A_0 \neq \emptyset$ **then**
    `// Case 1: room merging (post-processing)`
    $A \leftarrow \emptyset$; $\mathcal{O} \leftarrow \emptyset$; bins $\leftarrow [\,]$
    foreach $i \in \mathcal{G}_t$ **do**
        find a feasible bin b with sufficient remaining exam capacity and no course conflict
        if *no such bin exists* **then**
            open a new bin using $\text{room}(i)$ as target
            add it to $\mathcal{O}$
        assign i to the selected bin and update its load
else
    `// Case 2: direct assignment from scratch`
    $\mathcal{R}_t \leftarrow \text{SortDesc}(\mathcal{R}_t, c)$ `// Open large rooms first.`
    $A \leftarrow \emptyset$; $\mathcal{O} \leftarrow \emptyset$
    foreach $i \in \mathcal{G}_t$ **do**
        try to assign i to a feasible room in $\mathcal{O}$
        if *no feasible room exists* **then**
            open the next room from $\mathcal{R}_t$ and add it to $\mathcal{O}$
            assign i to this room
return $A, \mathcal{O}$

Case 2: Direct assignment from scratch. When no initial assignment is available, rooms are selected directly from the available set $\mathcal{R}_t$. 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 the first feasible open room. A room is feasible if: (i) it has enough remaining physical capacity, (ii) its exam capacity is not exceeded, (iii) it does not contain a group from the same course. If no open room satisfies these conditions, the next unused room from $\mathcal{R}_t$ is opened and used.

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 [10,12,16,8].

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 with the heuristic approach, the formulation distinguishes between post-processing (room merging) and full assignment from scratch.

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 $\mathcal{R}_0$ that are used in A_0. In this case, no new rooms may be opened. Room activation variables are fixed as:

$$y_j = 1 \quad \forall j \in \mathcal{R}_0, \qquad y_j = 0 \quad \forall j \notin \mathcal{R}_0, \tag{4}$$

  and assignments to closed rooms are forbidden:

$$x_{ij} = 0 \quad \forall i,\ \forall j \notin \mathcal{R}_0. \tag{5}$$

  This setting mirrors the greedy merging procedure, which retains a subset of the originally used rooms and merges other groups into them. Feasibility is enforced through exam-capacity and course-separation constraints.
- **Room assignment from scratch.** No initial assignment is given. All variables x_{ij} and y_j 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:

$$\min \sum_j y_j. \tag{6}$$

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

$$\sum_j x_{ij} = 1, \quad \forall i. \tag{7}$$

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**

Exam constraints. For each room and each course, the number of students from that course assigned to the room must not exceed the exam capacity:

$$\sum_{i:\, k(i)=k} s_i x_{ij} \le \hat{c}_j \, y_j, \quad \forall j,\ \forall k \in \mathcal{K}. \tag{8}$$

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

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

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

$$x_{ij} \le y_j, \quad \forall i, j. \tag{10}$$

Model properties and complexity. The ILP formulation extends classical bin packing by including room activation and course-based capacity constraints, and is therefore NP-hard. For a session with n groups, m rooms, and $|\mathcal{K}|$ courses, the model has $O(nm)$ binary variables and $O(nm + m|\mathcal{K}|)$ constraints. While solution times grow with problem size, the ILP remains tractable for moderate sessions and provides optimal benchmark solutions.

5 Case Study

We evaluate the proposed methods using a real-world case study from Ho Chi Minh City University of Technology (HCMUT) [15]. 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-organized buses, adding coordination and transport 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:

$$u_r^{\text{phys}} = \frac{n_r}{c_r} \quad \text{and} \quad \bar{u}^{\text{phys}} = \frac{1}{|\mathcal{R}^{\text{open}}|} \sum_{r \in \mathcal{R}^{\text{open}}} u_r^{\text{phys}}. \tag{11}$$

where n_r is the number of students assigned to room r, c_r is its physical capacity, and $\mathcal{R}^{\text{open}}$ 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 examination room allocation plans for each session, including candidate group sizes, course identifiers, room capacities, and campus information. 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 [19]. Each examination day, comprising up to five sessions, is solved on a per-session basis with a time limit of 15 minutes. A day is classified as a *timeout* if at least one session exceeds the time limit; in such cases, the best feasible incumbent solution at termination is retained. All experiments are conducted on a Mac mini with an Apple M4 processor (10 cores) and 16 GB of RAM. Each examination day is processed independently to reflect practical operational constraints.

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 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.

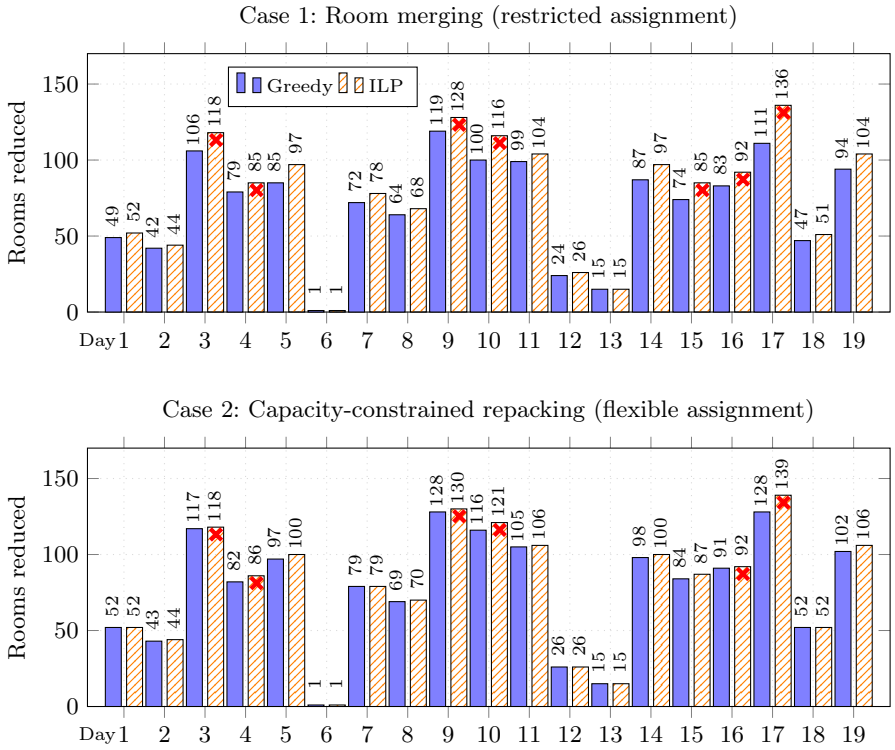

Fig. 1. Room reduction under restricted (Case 1) and flexible (Case 2) regimes. Days marked with ✖ indicate ILP timeouts (15 minutes); feasibility is still preserved.

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

	Base	**Greedy1**	**ILP1**	**Greedy2**	**ILP2**
Room usages	3392	2041	1895	1907	**1868**
Reduction (%)	–	39.8%	44.1%	44.8%	**44.9%**
Avg. phys. util.	39.54%	**63.90%**	60.31%	51.25%	56.06%

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

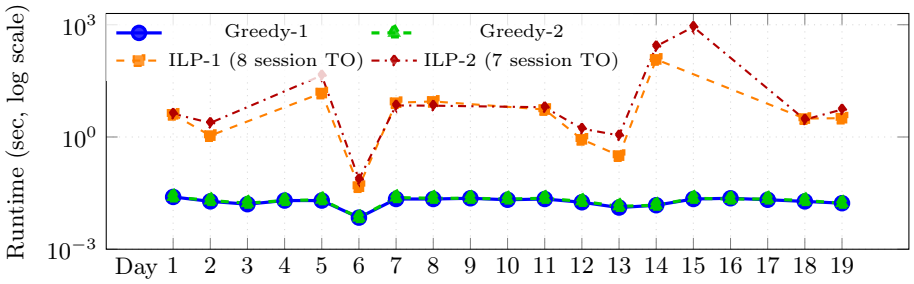

Fig. 2. Runtime comparison of Greedy and ILP (Cases 1–2) on a logarithmic y-axis. Dashed and dash-dot lines indicate Case 2 for Greedy and ILP, respectively.

physical room utilization (63.90%), representing a 1.6× increase over the baseline utilization of 39.54%.

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 further increases runtime variability due to the larger search space induced by flexible reassignment.

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 results suggest that flexible assignment largely eliminates the practical benefits of exact optimization. Future work will consider multi-objective extensions, dynamic and uncertain settings, integration into real-world examination scheduling systems.

References

1. Syariza Abdul-Rahman, Edmund K. Burke, Andrzej Bargiela, Barry McCollum, and Ender Özcan. A constructive approach to examination timetabling based on adaptive decomposition and ordering. *Annals of Operations Research*, 218:3–21, 2014.
2. Ravindra K. Ahuja, James B. Orlin, and Ashish Tiwari. A greedy genetic algorithm for the quadratic assignment problem. *Comput. Oper. Res.*, 27:917–934, 2000.
3. Edoardo Amaldi. Branch and bound method. Lecture notes, Foundations of Operations Research, Politecnico di Milano, 2019. Accessed: 2026-01-21.
4. Masri Ayob and Akram Malik. A new model for an examination-room assignment problem. *International Journal of Computer Science and Network Security*, 11(10):255–262, 2011.
5. Lisa Katharina Bergmann, Kathrin Fischer, and Sebastian Zurheide. A linear mixed-integer model for realistic examination timetabling problems. In *Proceedings of the 10th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2014)*, pages 82–101, York, United Kingdom, 2014. PATAT.
6. Natashia L. Boland, Brian D. Hughes, Lloyd T. G. Merlot, and Peter J. Stuckey. New integer linear programming approaches for course timetabling. *Computers & Operations Research*, 35(7):2209–2233, 2008.
7. E. K. Burke and S. Petrovic. Recent research directions in automated timetabling. *European Journal of Operational Research*, 140(2):266–280, 2002.
8. Edmund K. Burke, Yuri Bykov, James P. Newall, and Rong Qu. A graph-based hyper-heuristic for educational timetabling problems. *European Journal of Operational Research*, 176(1):177–192, 2007.
9. Sara Ceschia, Luca Di Gaspero, and Andrea Schaerf. Design, engineering, and experimental analysis of a simulated annealing approach to the post-enrolment course timetabling problem. *Computers and Operations Research*, 39, 04 2011.
10. Edward G. Coffman, Michael R. Garey, and David S. Johnson. Approximation algorithms for bin packing: A survey. In Dorit S. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems*, pages 46–93. PWS Publishing Co., 1996.
11. Abdelaziz Dammak, Ali Elloumi, and Hatem Kamoun. Classroom assignment for exam timetabling. *Advances in Engineering Software*, 37(10):659–666, 2006.
12. Matthieu Delorme, Matteo Iori, and Silvano Martello. Bin packing and cutting stock problems: Mathematical models and exact algorithms. *European Journal of Operational Research*, 255(1):1–20, 2016.

13. Luca Di Gaspero and Andrea Schaerf. Tabu search techniques for examination timetabling. In *Selected Papers from the Third International Conference on Practice and Theory of Automated Timetabling III*, PATAT '00, page 104–117, Berlin, Heidelberg, 2000. Springer-Verlag.
14. Jaime E. González, André Augusto Ciré, Andrea Lodi, and Louis-Martin Rousseau. Integrated integer programming and decision diagram search tree with an application to the maximum independent set problem. *Constraints*, 25(1):23–46, 2020.
15. Ho Chi Minh City University of Technology. Ho chi minh city university of technology (hcmut). https://www.hcmut.edu.vn, 2025. Accessed: 2025-01.
16. Mohamad N. Mohamad Kahar and Graham Kendall. The examination timetabling problem at universiti malaysia pahang: Comparison of a constructive heuristic with an existing software solution. *European Journal of Operational Research*, 207(2):557–565, 2010.
17. Teeradech Laisupannawong and Supphakorn Sumetthapiwat. An integer linear programming model for the examination timetabling problem: A case study of a university in thailand. *Advances in Operations Research*, 2024:1–17, 2024.
18. Alexandre Lemos, Francisco S. Melo, Pedro T. Monteiro, and Inês Lynce. Room usage optimization in timetabling: A case study at universidade de lisboa. *Operations Research Perspectives*, 6:100092, 2019.
19. Stuart Mitchell, Michael O'Sullivan, and Iain Dunning. PuLP: A linear programming toolkit for Python. *The University of Auckland, Auckland, New Zealand*, 2011.
20. Rong Qu, Edmund K. Burke, Barry McCollum, Louis T. G. Merlot, and San Y. Lee. A survey of search methodologies and automated system development for examination timetabling. *Computers & Operations Research*, 37(3):397–409, 2009.
21. David M. Ryan and Brian A. Foster. An integer programming approach to scheduling. In *Computer Scheduling of Public Transport*, pages 269–280. North-Holland, 1981.
22. A. Schaerf. A survey of automated timetabling. *Artificial Intelligence Review*, 13:87–127, 1999.
23. Riccardo Schininà. An integer programming model for timetabling at the university of groningen. Bachelor's thesis, University of Groningen, 2024.
24. E. S. K. Siew, Graham Kendall, Edmund K. Burke, et al. A survey of solution methodologies for exam timetabling. *Journal/Preprint*, 2024. Update journal, volume, pages when final bibliographic data are known.
25. Amr Soghier and Rong Qu. Adaptive selection of heuristics for assigning time slots and rooms in exam timetables. *Applied Intelligence*, 39(2):438–450, 2013.
26. Ahmed Wasfy and Fadi A. Aloul. Solving the university class scheduling problem using advanced ilp techniques. In *GCC Conference*, 2007.
27. Ásgeir Örn Sigurpálsson. Examination timetable modelling at the university of iceland, 2017. Accessed: 2026-01-29.