Cornell University Uses Integer Programming to Optimize Final Exam Scheduling

Tinghan Ye

H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, joe.ye@gatech.edu

School of Operations Research and Information Engineering, Cornell University

Adam Jovine

School of Operations Research and Information Engineering, Cornell University, asj53@cornell.edu

Willem van Osselaer

School of Operations Research and Information Engineering, Cornell University, wpv6@cornell.edu

Qihan Zhu

Tepper School of Business, Carnegie Mellon University, qihanz@andrew.cmu.edu

School of Operations Research and Information Engineering, Cornell University

David B. Shmoys

School of Operations Research and Information Engineering, Cornell University, david.shmoys@cornell.edu

Abstract. This paper presents an integer programming-based optimization framework designed to effectively address the complex final exam scheduling challenges encountered at Cornell University. With high flexibility, the framework is specifically tailored to accommodate a variety of different constraints, including the front-loading of large courses and the exclusion of specific time slots during the exam period. By generating multiple scheduling model variants and incorporating heuristic approaches, our framework enables comprehensive comparisons of different schedules. This empowers the University Registrar to make informed decisions, considering trade-offs in terms of schedule comfort measured by different levels of exam conflicts. Our results demonstrate significant advantage over the historical lecture time-based approach, providing time and effort savings for the university administration while enhancing student and faculty satisfaction.

Key words: scheduling; integer programming; optimization; education

1. Introduction

Academic stress is a universal issue among undergraduate students. And for many, the final exam period is the most stressful time of all. With each exam often requiring hours of studying, it is not only the number of exams but also the timing of exams that decide how challenging and stressful a final exam period will be. Having three exams can be easily manageable if they are spread out over

the course of ten days, but three exams on the same day leaves students more stressed and with less time to prepare.

From the perspective of faculty, the best schedule would be one without any scheduling conflicts for students. If a student is scheduled to take two exams in the same time slot, alternative arrangements must be made for one of the exams to be taken at a different time and location. This process can be quite challenging and inconvenient for the faculty. Due to academic integrity concerns, a completely new exam may need to be created specifically for that small group of students.

Because each student takes a different set of classes, finding a final exam schedule means balancing the exam sequences of all the different students; an exam schedule that nicely spreads exams for one student might put all exams in a tight block for another student. When faced with thousands of students taking hundreds of exams, this becomes an impossible task to do by hand.

At Cornell University, faculty and students from the ORIE Department collaborated with the Office of the University Registrar to schedule final exams. Before using integer programming, final exams at Cornell University were scheduled based on a static table that maps first lecture meeting times to exam groups. Courses that started at the same hour would have their finals scheduled simultaneously. However, the number of meeting times exceeded the available exam slots. Typically, there were around 30 different meeting times but only 22 exam slots, leading to numerous scheduling conflicts. No consideration was given to the other bad events at all.

During the COVID-19 pandemic, the university sought assistance from the ORIE Department to address logistics for reopening for residential instruction in Fall 2020 (Frazier et al. 2022). Their efforts ranged from completely rescheduling all courses to maximizing in-person attendance while taking into account a 6-foot distance seating requirement (Liu et al. 2023), to determining actual classroom seat assignments (Greenberg et al. 2021). Final exam scheduling was one of the critical tasks undertaken in this collaboration.

We develop an optimization framework based on mixed-integer programming (MIP) to create final exam schedules for the entire Cornell University. The foundation of the framework comprises a multi-stage Group-then-Sequence (GtS) model and an iterative Layer-Cake heuristic algorithm. The framework has been effectively employed for the past five semesters.

1.1. Contributions

Our proposed methods advance previous research by addressing a range of final exam conflicts. In addition to direct conflicts and back-to-back exams, we consider more complex scenarios, such as having more than two exams within or beyond a 24-hour period. This approach is specifically designed to tackle the unique challenges faced by Cornell University, aiming to enhance both student and faculty comfort during finals week through trade-offs across various conflict levels.

Building upon the existing body of research, this paper presents a novel optimization framework tailored to the specific needs of students and faculty at Cornell University. The framework offers high flexibility, enabling customization to accommodate constraints such as front-loading large exams or excluding certain time slots. By generating multiple scheduling model variants with different parameter combinations and heuristic approaches, our framework allows for a comprehensive comparison of exam schedules. This comparison helps illustrate trade-offs in schedule comfort, as measured by varying levels of exam conflicts, ultimately assisting the University Registrar in selecting the most optimal final exam schedule.

Our main contribution are as follows:

1. We propose a MIP formulation that incorporates penalties for higher-order conflicts. This formulation explicitly addresses scenarios where a student is scheduled for three exams on the same day, within a 24-hour period, or across four consecutive exam slots.

2. We develop an integrated MIP-based framework that synergizes local search and heuristic techniques to efficiently generate exam schedules optimized according to multiple performance

metrics. Moreover, our framework supports hybridization strategies to further enhance solution quality.

3. We validate our framework through a comprehensive evaluation at a large American university, where it has been successfully implemented over several consecutive semesters. Additionally, comparative analyses on a benchmark dataset demonstrate that our approach is competitive with existing exam timetabling algorithms, highlighting its versatility in addressing various exam timetabling challenges.

1.2. Organization

The rest of the paper is organized as follows: First, we detail Cornell University's final exam scheduling problem in Section 2. In Section 3, we provide a brief review of relevant work. Section 4 outlines our proposed solution methods, with detailed optimization models and algorithms provided in the Appendix. In Section 5, we present the outcomes of our scheduling models and algorithms using real student enrollment data from Cornell University. Finally, Section 6 concludes our work.

2. Problem Description

Cornell University faces the complex task of scheduling final exams for a significant student body of more than 15,000 and approximately 700 courses at the beginning of every semester. These exams are concentrated within a 7 to 9-day final exam period, with three time slots available per day (9am, 2pm, and 7pm). However, certain time slots—such as Saturday nights and Sunday mornings—may have restrictions, resulting in roughly 19 to 22 available time slots. The ultimate scheduling problem is to assign around 500 exams to these time slots. The physical locations for the exams are determined separately by the University Registrar, and as such, this study focuses solely on optimizing the timing of the exams. Room assignment is not a significant concern at Cornell University for two primary reasons. First, the university has a sufficient number of classrooms with adequate seating capacities to accommodate demand. The only potential constraint is a cap

of 5,000 students taking exams per day; however, all generated schedules remain well below this limit. Second, Cornell University typically does not split exams across multiple classrooms, which further simplifies the assignment of physical locations.

The goal of an optimized final exam schedule is to enhance the experience for both students and faculty. From the faculty's perspective, student conflicts—such as overlapping or consecutive exams–may require the scheduling of make-up exams according to university policies (https://registrar.cornell.edu/calendars-exams/final-exam-policies), which imposes additional burdens. Meanwhile, students desire a well-distributed exam schedule to allow for sufficient review and study time. As a result, the optimal exam schedule should minimize various unfavorable metrics, each with its own weight. These metrics include direct conflicts, back-to-back exams, two exams in 24 hours, three exams in a row, and three exams in 4 slots.

We define the quality of a final exam schedule by the number of "bad events" it contains:

- Direct conflict: A student with two exams scheduled at the same time slot.

- Three exams in a row (triple): A student with three exams assigned to three consecutive slots.

- Back-to-back exams (B2B): A student with exams in two consecutive slots.

- Two exams in 24 hours (2-in-24hr): A student with two exams in three consecutive slots, or equivalently, within a 24-hour period.

- Three exams in 4 slots (3-in-4): A student with three exams in four consecutive slots.

We ensure no double counting of metrics; for instance, a student having a pair of exams that contribute to a triple will not be counted again as a B2B or 2-in-24hr.

The objective is to minimize the weighted sum of these "bad events", which serves as a measure of schedule quality. While the primary focus is on eliminating direct conflicts and triples to avoid make-up exams, other metrics are also considered to reduce student stress as much as possible. Additionally, each exam must be assigned to exactly one time slot.

The final exam schedule must also adhere to specific constraints to facilitate room assignments by the University Registrar. In particular, exams for large enrollment courses should be scheduled in earlier time slots to free up larger auditoriums and lecture halls for graduation events and ensure timely grading in line with university deadlines.

3. Related Literature

This section provides related works in terms of problem formulations and solution techniques.

3.1. Educational Timetabling

The scheduling problem encountered by Cornell University is situated within the broader domain of educational timetabling. A comprehensive survey by Ceschia et al. (2023) offers an up-to-date overview of this evolving field. Educational timetabling is well-supported by an array of benchmark datasets, such as those from the University of Toronto (Carter et al. 1994), the University of Nottingham (Burke et al. 1996), and the Purdue University (Müller 2016). These resources, alongside major international competitions like ITC-2007 (McCollum et al. 2007), ITC-2011 (Post et al. 2016), and ITC-2019 (Müller et al. 2024)), as well as tools like UniTime [https://www.unitime.org])), demonstrate the field's practical applications.

Educational timetabling includes three main subfields: high-school timetabling, course timetabling, examination timetabling. Many real-world problems in these areas are approached through integer programming techniques. High-school timetabling, commonly represented in the XHSTT format (Post et al. 2016), has seen the application of MIP in studies like Kristiansen et al. (2015) and Fonseca et al. (2017). Course timetabling problems, such as those at the United States Air Force Academy (Gonzalez et al. 2018), the Technion in Israel (Strichman 2017), and MIT (Barnhart et al. 2022), also employ these approaches. In examination timetabling, there are case studies from the Universidad Politécnica de Madrid (García-Sánchez et al. 2019) and the Department of Education in Vestfold County (Avella et al. 2022)

3.2. University Examination Timetabling: Problems and Formulations

Among educational timetabling problems, our work is most closely related to university examination timetabling problems (ETPs). Comprehensive surveys of ETPs can be found in the works of Qu et al. (2009), Gashgari et al. (2018), Aldeeb et al. (2019), and Siew et al. (2024).

ETPs have been extensively studied for decades, taking various forms with different restrictions based on institutional and regulatory differences. The most relevant formulation for our study is the uncapacitated examination timetabling problem, first introduced by Carter et al. (1996). A mathematical model of this formulation is provided by Bellio et al. (2021). More complex versions that incorporate room assignments and capacity constraints, such as Track 1 of ITC-2007 (McCollum et al. 2007), exist, but since room capacity is not a factor in our problem, we focus on the uncapacitated version.

In uncapacitated ETPs, the primary hard constraints include avoiding simultaneous exam conflicts and managing the total number of exams assigned to the same time slot (Aldeeb et al. 2019). Additionally, soft constraints are commonly introduced as penalty terms in the objective function. These typically involve the period spread between exams with co-enrollments and prioritizing larger exams for earlier scheduling (Aldeeb et al. 2019). In the literature, many studies focus on metrics related to pairs of exams with co-enrollments, such as back-to-backs, or more broadly speaking, two exams that are y periods apart. Nonetheless, very few of them explicitly address higher-order conflicts, such as scheduling no more than x co-enrolled exams within y consecutive periods ($x, y \geq 3$). This higher-order conflict constraint was first proposed by Carter et al. (1996) and also appears in the Purdue University formulation (Müller 2016), which includes a soft constraint for a student having three or more exams on the same day, solved using constraint programming. To date, we have not yet encountered MIP formulations that explicitly address such higher-order conflicts.

3.3. University Examination Timetabling: Solution Techniques

A comprehensive survey by Aldeeb et al. (2019) reviews various solution techniques for the uncapacitated ETP. Common approaches include graph-based heuristics, such as those evaluated by Carter et al. (1996), and metaheuristics like Simulated Annealing, used in Bellio et al. (2021).

Our approach to solving this scheduling problem employs MIP for solving subproblems and developing heuristic algorithms. Similar mathematical programming-based approaches have been proposed in prior works on general ETPs (Woumans et al. 2016, Güler et al. 2021, Godwin 2022, Bazari et al. 2023).

We propose a multi-stage methodology, where a large MIP is divided into smaller stages. Multi-stage approaches have also been explored in the ETP literature (Cataldo et al. 2017, Keskin et al. 2018, Al-Hawari et al. 2020). For post-processing, we apply a local search-based algorithm, which shares similarities with existing metaheuristics such as Large Neighborhood Search and fix-and-optimize strategies. These types of algorithms are commonly used in the ETP field (Elloumi et al. 2014, Müller 2016)

Additionally, we introduce a Layer-Cake heuristic algorithm, which builds the exam schedule iteratively using MIP for a subset of exams in a greedy manner. The idea of combining mathematical programming and heuristics appears in many other contexts is what is called "matheuristics" (Archetti and Speranza 2014, Sörensen 2015, Boschetti and Maniezzo 2022, Siew et al. 2024). Gogos et al. (2012) and Çimen et al. (2022) have explored applying matheuristics in general ETPs.

4. Modeling Approach

This section describes the various components of the MIP-based scheduling frameworks. We begin by introducing the multi-stage Group-then-Sequence model, explaining it stage by stage. Following this, we outline the "Layer-Cake" heuristic algorithm. Finally, we discuss approaches for creating additional variants of the scheduling models.

Figure 1 illustrates the input and output components of the scheduling framework. The primary source of input data is the student-level enrollment information, provided by the University Registrar in a spreadsheet format each semester. Each row in the spreadsheet represents a student-course pair. To facilitate the scheduling process, this data is further processed to generate three sets of course-level input data required for the subsequent models / algorithms. These include the exam sizes, the number of co-enrolled students for each pair of exams, and the number of co-enrolled students for each triplet of exams. The output of the framework is a mapping that assigns each exam to a specific time slot.
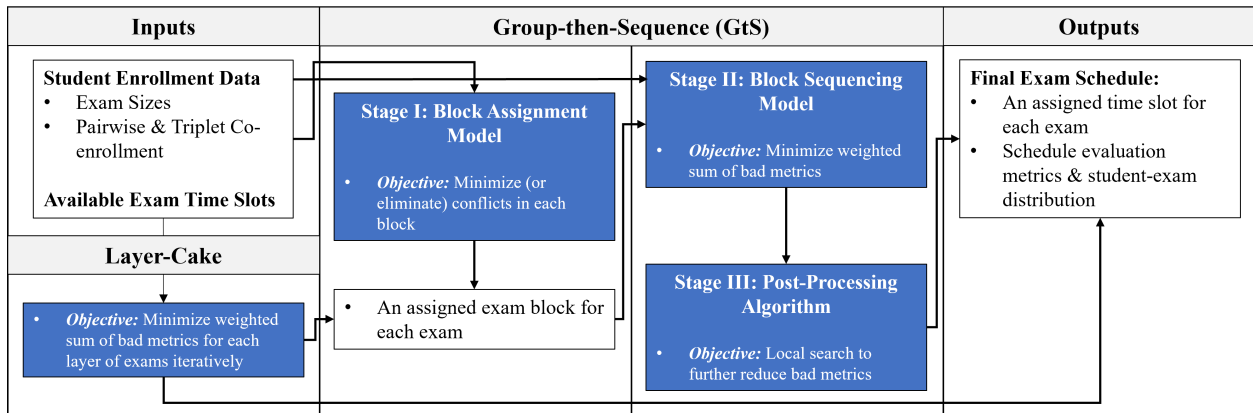

Figure 1   Workflow of the Proposed Final Exam Scheduling Framework

4.1.   Multi-Stage Group-then-Sequence (GtS) Model

The core of the GtS model consists of the Block Assignment stage and the Block Sequencing stage. A Post-Processing stage can be applied to the schedule generated by GtS. The combined three-stage model is called the GtSP (Group-then-Sequence-Post-Processing).

The idea behind the first two stages is to initially group the exams into a smaller number of exam *blocks* and then assign each block to a time slot.

4.1.1.   Block Assignment The Block Assignment stage is formulated as a MIP problem (MIN_CONFLICT_IP) that assigns each exam to one of a specified number of blocks, denoted by k.

It is assumed that $k \leq |S|$, where $S := \{0, 1, \ldots, |S| - 1\}$ is the set of available time slots. The objective of this grouping is to minimize the total number of pairwise co-enrollments (direct conflicts) among exams within the same block. Appendix A.1 provides the Min_Conflict_IP formulation. It is important to note that, at this stage, each block represents a group of exams that will be scheduled simultaneously, but is not yet associated with a specific time slot.

If it is possible to achieve zero conflicts within each block, the block assignment problem can be interpreted as a graph coloring problem. Consider a graph $G = (V, E)$, where V represents the set of exams, and an edge $(i, j) \in E$ connects exams i and j if at least one student is enrolled in both. A proper coloring of the graph G corresponds to a zero-conflict block assignment, as exams with the same color would be grouped in the same block.

We also propose a variation of the Block Assignment problem, named Zero_Conflict_IP, that imposes a strict zero-conflict constraint. We assume a fixed linear ordering of exam blocks $0, 1, \ldots, k - 1$ and declare only the consecutive pairs $(b, b+1)$, for $b = 0, \ldots, k - 2$, to be neighboring (so block 0 and block $k - 1$ are not neighbors). The model imposes a hard constraint forbidding any co-enrolled exams from occupying the same block, and it minimizes the total number of co-enrollments across all such consecutive block pairs. This objective draws an analogy to the Traveling Salesman Problem (TSP)—except that we optimize a Hamiltonian path rather than a cycle, reflecting the linear sequence of blocks. The Zero_Conflict_IP formulation is available in Appendix A.2. Figure 2 shows a three-block example together with its path-objective value. Compared to the minimum-conflict formulation, Zero_Conflict_IP has a more constrained feasible space, which increases the likelihood of infeasible solutions.

The feasibility of having a zero-conflict block assignment depends mainly on the required number of blocks k. While one way to find a lower bound on k is to enumerate all possible values and check the feasibility of the Zero_Conflict_IP problem, a smarter way is to solve for the Maximum

Clique (MAX_CLIQUE_IP) problem. Appendix B contains the MAX_CLIQUE_IP formulation. The MAX_CLIQUE_IP involves finding a subset of vertices in G that forms a complete sub-graph, where every two distinct vertices are adjacent. In our context, the size of the max clique corresponds to a lower bound on the number of blocks for which it is feasible to have no conflicts.
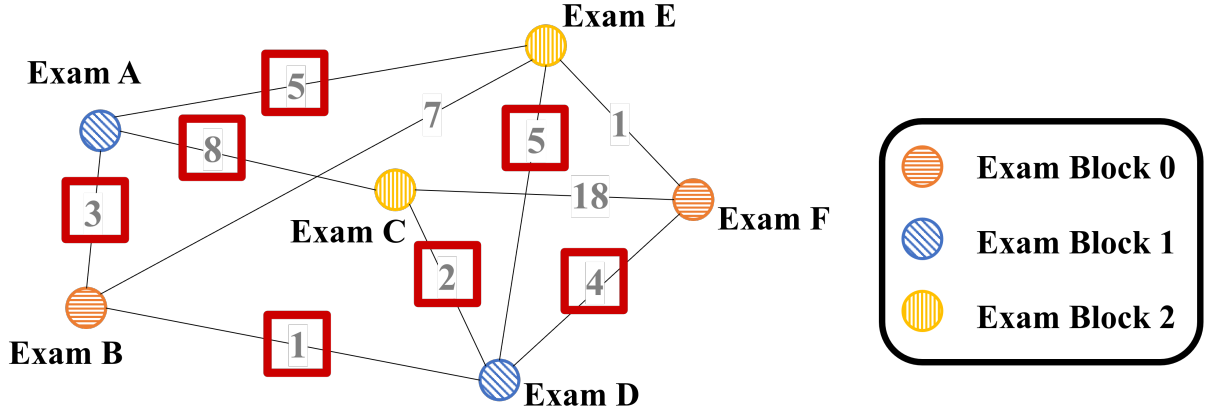
Figure 2 This is an example of a zero-conflict block assignment. The value on each edge represents the number of co-enrollments. Blocks 0 and 1, as well as Blocks 1 and 2, are considered neighboring blocks. Consequently, the path-objective value can be calculated as the sum of the highlighted values.

4.1.2. Block Sequencing After assigning exams to the designated blocks, we then allocate these blocks to time slots by *sequencing* them in Block Sequencing stage (BLOCK_SEQUENCING_IP). In the Block Assignment phase, we focus mainly on direct conflicts, but in Block Sequencing, we begin to account for other levels of undesirable metrics. Appendix C presents the BLOCK_SEQUENCING_IP formulation.

Assume for now that the number of blocks equals the number of time slots, i.e., $|S| = k$. Then, the set of exam blocks is denoted by $B := \{0, 1, \ldots, |S| - 1\}$. To maintain linearity in the objective function, we introduce the four-index variables, denoted as $x_{ij\ell s}$, which allow us to penalize metrics involving triplets of exams. $x_{ij\ell s} = 1$ when block i is assigned to time slot s, block j is assigned to time slot $(s + 1) \mod |S|$, and block ℓ is assigned to time slot $(s + 2) \mod |S|$. The one-to-one

mapping between blocks and time slots necessitates that the four-index variables adhere to a cyclic self-consistent property.

As shown in Figure 3, Block Sequencing produces a mapping of blocks to time slots that are evenly distributed and arranged in a cyclic order. Observe that if $x_{ij\ell s} = 1$, then there must be another block m assigned to time slot $(s+3)$, leading to $x_{j\ell m(s+1)} = 1$. This relationship also holds at the end of the time slot sequence due to the use of the modulo function. For instance, if $x_{npq(k-1)} = 1$, then there must be another block r assigned to time slot $(k \mod k = 0)$, making $x_{pqr0} = 1$. This observation can be generalized, leading to the formulation of the following self-consistent constraint:

$$\sum_{i \in B} x_{ij\ell s} = \sum_{m \in B} x_{j\ell mn_s}, \forall j, \ell \in B, s \in S, \tag{1}$$

where $n_s := (s+1) \mod |S|$.
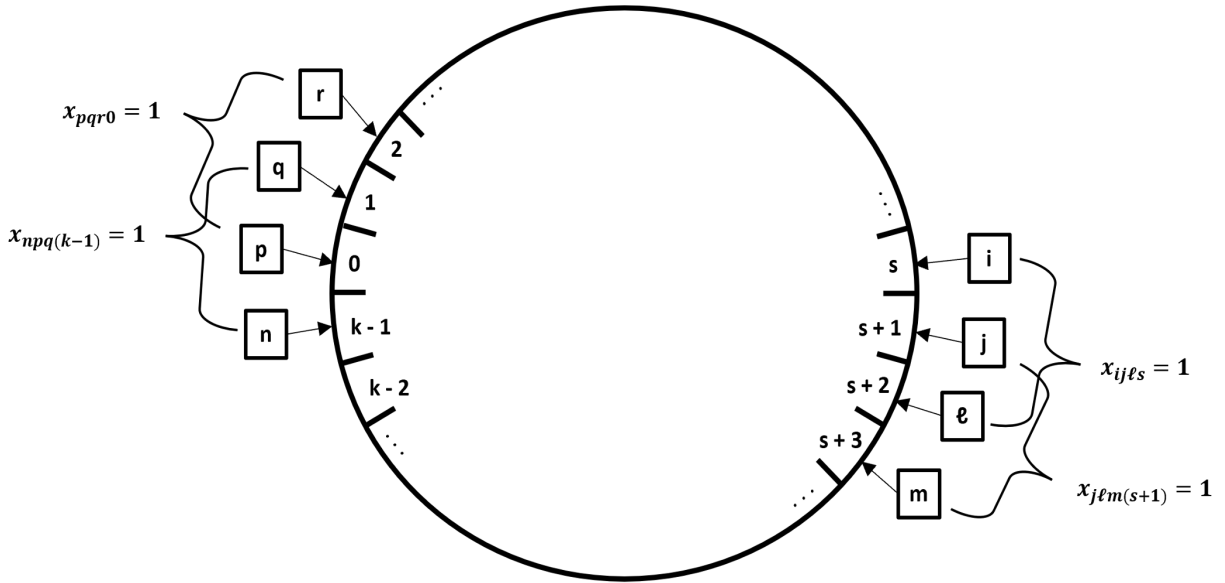
Figure 3   **Illustration of the $x_{ij\ell s}$ Variables and the Cyclic Self-Consistent Property**

To handle the situation where the number of available time slots exceeds the number of blocks, i.e., $|S| > k$, we introduce an additional layer of optimization. To match the number of blocks with the number of slots, we introduce the concept of a "virtual block" — an empty block without

any exams. By adding an appropriate number of virtual blocks, we ensure that the total number of blocks equals the number of slots. In the block-to-slot assignment generated by the model, time slots assigned to virtual blocks are considered empty and left unused. This approach helps reduce back-to-back exams and improve other metrics without the need to introduce more blocks. Additionally, virtual blocks enable the exclusion of specific time slots by enforcing constraints that assign virtual blocks to those designated time slots. In Figure 4, unshaded squares represent virtual blocks, indicating that slots s_2 and s_4 are unused in this example.
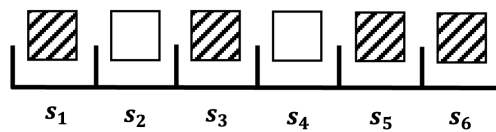
Figure 4   An Example Usage of Virtual Blocks

Additionally, the Block Sequencing model allows for the front-loading of large exams. We introduce hard constraints based on a user-defined exam size cut-off and an early time slot cut-off, ensuring that exams of courses with enrollments exceeding a certain threshold are scheduled earlier in the exam week. Note that in Block Sequencing, the front-loading applies to the entire block containing the large exams rather than scheduling individual exams earlier.

As a result of the Block Sequencing model, each block is allocated to a specific time slot, as are the exams within each block. This produces a working final exam schedule but might not be fully optimized across all metrics.

4.1.3. Post-Processing Local Search Algorithm Multiple optimal solutions with the same number of direct conflicts can be generated in the Block Assignment stage. However, the schedules derived from these assignments may vary in other metrics, as the Block Assignment model only focuses on direct conflicts. As a result, the block assignment obtained may not be optimal when considering all metrics comprehensively. To address this limitation, we develop a local search

algorithm that takes an existing exam schedule and iteratively reschedules subsets of exams with a specified size n. The full pseudocode for the Post-Processing local search algorithm is available in Appendix D.

In each iteration, the algorithm selects the top n exams to reschedule by assessing the contribution of each exam to the number of bad metrics, normalized by the number of students taking the exam. For example, an exam where 30% of its students have an exam right before or after it would be prioritized for rescheduling compared to an exam with only 5% of students in a similar situation. This normalization effectively ranks exams by the density of "bad" events rather than their absolute counts, allowing the algorithm to focus on the intensity of issues experienced per student. Empirically, this approach provides a more effective signal for guiding the local search.

Once the subset is selected, the algorithm solves a SCHEDULE_IP problem to move the exams to time slots that minimize the number of B2Bs and 2-in-24hrs, without increasing the number of direct conflicts. The process continues iteratively, with the algorithm using adaptive step sizes to adjust the starting index of the rolling window with size n. If no improvement is found after rescheduling, a larger step forward is taken to explore new solutions. Conversely, if an improvement is observed, a smaller step backward is adopted to fine-tune the schedule. This adaptive approach enables the algorithm to efficiently navigate the solution space, balancing exploration with refinement.

After applying the Post-Processing algorithm to the schedule generated by Block Sequencing, we should obtain a new schedule with significant improvements in reducing the number of bad metrics.

4.2. Layer-Cake Heuristic Algorithm

The Layer-Cake heuristic algorithm offers an alternative to the GtS model and is inspired by the Post-Processing algorithm. Unlike the GtS model, Layer-Cake addresses conflicts and other metrics simultaneously by solving the SCHEDULE_IP problem, which optimizes all metrics together. To

reduce the computational complexity of scheduling all exams at once, the problem is solved for a smaller subset of exams each time. A key difference between Layer-Cake and GtS is that Layer-Cake directly takes in a sequence of time slots, eliminating the need to specify the number of blocks, k, as an input. Appendix E provides the full algorithm pseudocode for Layer-Cake.
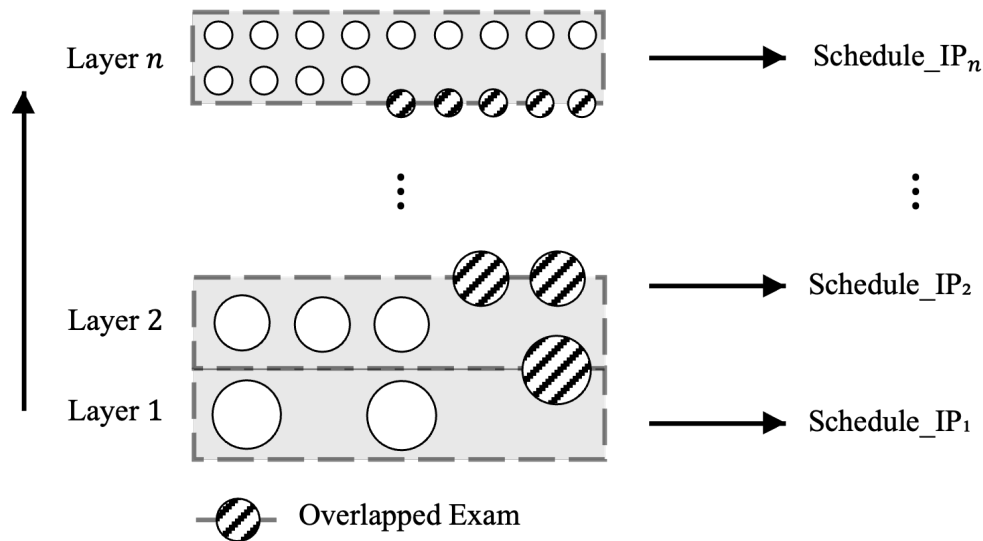
Figure 5   **Visualization of the Layer-Cake Algorithm with Partial Overlaps in Consecutive Layers**

Layer-Cake follows a layered approach, where each layer consists of a set of exams such that the total number of students per layer is similar. The layers are determined by sorting the exams in descending order of student enrollment. The initial layers focus primarily on larger exams, while subsequent layers include many smaller exams. The algorithm starts by scheduling the largest exams independently of the others. It then schedules the next subset of exams to build upon those already assigned, gradually building the schedule "layer by layer." Crucially, each new layer can partially overlap with the preceding one, as depicted in Figure 5. This ordering ensures that the largest exams are scheduled first, as the SCHEDULE_IP problem becomes more challenging when dealing with larger exams compared to smaller ones, even with a constant total number of students.

An essential feature of the Layer-Cake algorithm is the overlap between consecutive layers. In each iteration, a subset of exams that have already been scheduled are included in the IP, allowing

them to be rescheduled alongside the newly introduced exams. This overlapping of layers, serving as a warm start for the IP problems, considerably boosts the algorithm's performance.

Since each SCHEDULE_IP problem is unaware of exams in future layers and only a subset of the exams can be moved in each iteration, the Layer-Cake algorithm may not yield perfect results. However, it offers notable advantages in computational efficiency over the GtS model. By producing potentially improved schedules in much less time, Layer-Cake provides an effective balance between speed and performance. In addition, the flexibility to adjust the layer size allows for easy customization of this trade-off: larger layers tend to produce better schedules, though at the cost of increased solving time.

4.3. The Hybrid Model

Building on the strengths of both the GtS model and Layer-Cake, we also develop a new variant of an exam scheduling model. This Hybrid model is inspired by the observation that the assignment of exams to time slots in Layer-Cake, without considering their order, can be interpreted as a block assignment. This opens up the possibility of using the Layer-Cake solution as a promising feasible solution for the Block Assignment model.

Integrating the block assignment obtained from Layer-Cake into the subsequent stages of the GtS model has the potential to improve the overall schedule. Since Layer-Cake considers multiple levels of metrics beyond just direct conflicts, incorporating its block assignment into the GtS model could lead to enhanced schedules compared to the standard Min-Conflict or Zero-Conflict Block Assignment, which primarily focuses on minimizing direct conflicts.

5. Implementation Results and Impact

The proposed final exam scheduling framework has been successfully adopted by the Cornell University Registrar for five consecutive semesters, beginning in Spring 2022. The scheduling process usually starts about two weeks after the semester begins, allowing the Registrar to collect

final enrollment information during the Add/Drop period. To efficiently create the final exam schedule within a tight timeline of less than a month, a dedicated team of undergraduate students is trained in advance.

Once the team gains access to the necessary data, we conduct experiments using various parameters and models implemented in Gurobi and Python. This collaborative approach allows for comprehensive exploration and evaluation of different scheduling options. To accelerate the process, we use Google Colab to run multiple experiments in parallel. From the generated subset of "good" schedules, the University Registrar selects the best one to be published on the university website.

In the subsequent parts, we showcase the performance of our scheduling models on real student enrollment data, highlighting the framework's ability to outperform the historical baseline and its flexibility in meeting the registrar's needs. All experiments in this paper are conducted on a single machine equipped with an 8-core Intel Xeon CPU running at 2.20 GHz. All optimization models are solved using Gurobi version 11.0.3. Computation times are not reported in this paper, as they are not a primary concern in our application. All methods complete within several hours, which is acceptable given that the final exam schedule is typically prepared over the span of approximately one month each semester.

5.1. Performance of the Scheduling Models Across Multiple Semesters

In the first set of experiments, we compare the performance of different scheduling models using actual student enrollment data at Cornell University from Spring 2022 to Spring 2024. Table 1 presents descriptive statistics for student enrollment and exam timing data. Generally, more students take exams in the fall semester than in the spring. The last column shows the Max-Clique size, representing a lower bound on the number of exam blocks required for that semester. For instance, in Spring 2024, a Max-Clique of size 20 indicates that there was a subset of 20 courses with overlapping enrollments, necessitating at least 20 distinct exam blocks to avoid any direct conflicts.

Table 1   Descriptive Statistics of Input Data From Spring 2022 to Spring 2024

	Student Size	# of Unique Exams	Exam Time Range	# of Available Time Slots	Max-Clique Size
SP24	16265	544	May 11th 9 a.m. - May 18th 7 p.m.	24 (24)	20
FA23	18319	601	Dec. 8th 9 a.m. - Dec. 15th 7 .m	24 (23)	18
SP22	16261	553	May 14th 9 a.m. - May 21th 7 p.m.	24 (19)	17
FA22	18081	588	Dec. 9th 9 a.m. - Dec. 17th 9 a.m.	25 (22)	19
SP22	16429	539	May 13th 9 a.m. - May 20th 7 p.m.	24 (20)*	16

Note: *The numbers inside the parentheses correspond to the actual number of available time slots, which is constrained by the University Registrar but subject to relaxation starting from SP24.

Table 2   Performance of the Baseline and Our Scheduling Models Across 5 Semesters

Semester	Model	Conflict	Triple	B2B	2-in-24hr	3-in-4
SP24	Baseline	790	128	3029	2653	614
	GtS (Ours)	0	50	2324	2790	441
	GtSP (Ours)	0	30	1753	2474	285
	Zero-GtSP (Ours)	0	39	1763	2884	357
	Layer-Cake (Ours)	13	26	2020	2531	406
	Hybrid (Ours)	6	25	1812	2236	325
FA23	Baseline	376	143	2883	3930	736
	GtS (Ours)	0	51	2938	3534	641
	GtSP (Ours)	0	48	2215	3669	490
	Zero-GtSP (Ours)	0	33	2038	3185	551
	Layer-Cake (Ours)	0	26	2323	3088	546
	Hybrid (Ours)	0	26	1950	2775	458
SP23	Baseline	720	188	2957	2600	742
	GtS (Ours)	0	43	2332	2709	430
	GtSP (Ours)	0	40	2024	2211	281
	Zero-GtSP (Ours)	0	27	1651	2648	339
	Layer-Cake (Ours)	0	22	1784	2545	420
	Hybrid (Ours)	0	22	1566	2309	409
FA22	Baseline	614	142	3284	3930	812
	GtS (Ours)	0	31	2210	3556	667
	GtSP (Ours)	0	32	1888	2942	541
	Zero-GtSP (Ours)	0	37	2051	2625	466
	Layer-Cake (Ours)	1	25	2233	3100	427
	Hybrid (Ours)	1	34	1621	2726	299
SP22	Baseline	564	206	3050	2801	564
	GtS (Ours)	0	53	2494	2493	380
	GtSP (Ours)	0	28	1753	2309	285
	Zero-GtSP (Ours)	0	44	1712	2941	466
	Layer-Cake (Ours)	0	20	1610	2782	341
	Hybrid (Ours)	0	16	1411	2517	264
Average	Baseline	612.8	161.4	3040.6	3182.8	693.6
	GtS (Ours)	0	45.6	2459.6	3016.4	511.8
	GtSP (Ours)	0	35.6	1926.6	2721	376.4
	Zero-GtSP (Ours)	0	36	1843	2856.6	435.8
	Layer-Cake (Ours)	2.8	23.8	1994	2809.2	428
	Hybrid (Ours)	1.4	24.6	1672	2512.6	351

To ensure a fair comparison, we use the full set of time slots for all models as shown in Table 1.
For each semester, the number of blocks used remains consistent across different scheduling model
variants. For Layer-Cake-based models, the time slot configuration input matches that obtained
from the corresponding GtS models. Unless otherwise specified, we use the standard scheduling

parameters given in Appendix F throughout the paper. The Baseline model follows the historical approach described in Section 1. The current course schedule includes new meeting time patterns that are not accounted for in the lecture time-based mapping used in the baseline approach. To adapt the Baseline model for different semesters, we assigned courses with new meeting times greedily to one of the existing time slots. This approach preserved the direct mapping between meeting times and exam times.

Table 2 summarizes the metrics of the schedules created by our framework and the Baseline. The following key observations can be drawn from the table.

5.1.1. Dominance Over the Historical Approach The Baseline model, commonly used in many universities, is completely dominated by our scheduling models every semester. The limitations of relying solely on lecture times to create final exam schedules are evident, as the Baseline schedules consistently underperform across nearly all metrics, particularly in reducing direct conflicts and triples. On average, the Baseline results in 612.8 direct conflicts and 161.4 triples, leading to over 774 occurrences of exam rescheduling. In contrast, schedules generated by our framework generally have zero or minimal conflicts and fewer than 50 triples.

5.1.2. Impact of Post-Processing Comparing the GtS and GtSP models reveals that the Post-Processing stage significantly improves all metrics for the GtS model. This underscores the limitations of the draft schedule generated in the initial two stages of the GtS model, which focus on block-level metrics rather than individual exams. On average, we observe reductions of 22% in B2Bs, 22% in triples, 10% in 2-in-24hrs, and 26% in 3-in-4s from GtS to GtSP .

5.1.3. GtSP vs. Layer-Cake The Layer-Cake heuristic algorithm presents a distinct trade-off among metrics compared to the GtS-based models, primarily due to its slightly different objective. Layer-Cake generally excels in providing finer control over bad metrics as its Schedule_IP directly penalizes the occurrence of B2Bs and 2-in-24hrs. As a result, triples and 3-in-4s are indirectly

addressed within each Schedule_IP, since both of these metrics include at least one 2-in-24hr. Table 2 reveals that, on average, Layer-Cake produces 33% fewer triples than GtSP. However, this comes at the cost of 3% more B2Bs and 2-in-24hrs, as well as a 13% increase in 3-in-4s.

5.1.4. Layer-Cake vs. Hybrid We find that the Hybrid model outperforms Layer-Cake in nearly all metrics, with the exception of triples. On average, compared to Layer-Cake, the Hybrid model reduces B2Bs by 16%, 2-in-24hrs by 10%, and 3-in-4s by 18%. Additionally, the Hybrid model achieves the same or fewer direct conflicts than Layer-Cake, as it is built upon the block assignments generated by the Layer-Cake schedule. The Hybrid model also benefits from the Post-Processing stage, which can further reduce direct conflicts through rescheduling, as demonstrated in SP24. For these reasons, we typically use Layer-Cake for block assignment but rely on the Hybrid model for the final schedule.

5.1.5. Impact of Different Block Assignment Methods Table 2 also compares different approaches to Block Assignment: Min_Conflict_IP (the default GtSP), Zero_Conflict_IP (the Zero-GtSP), and using the Layer-Cake schedule for block assignment (the Hybrid).

Among the two IP methods, solving the Min_Conflict_IP to optimality is relatively easier. With $k = 22$, the Min_Conflict_IP yields a zero-conflict block assignment within an hour. However, it's important to note that the Min_Conflict_IP does not always guarantee zero conflict, particularly with smaller values of k. In contrast, the Zero_Conflict_IP model guarantees zero conflicts but is more challenging to solve, making it difficult to obtain an optimal or near-optimal solution within a reasonable time limit. When a solution is feasible, the Zero_Conflict_IP generally outperforms the Min_Conflict_IP in terms of B2Bs due to its TSP objective, as discussed earlier. Specifically, the Zero_Conflict_IP reduces B2Bs by 4% compared to the Min_Conflict_IP, but this comes with a 5% increase in 2-in-24hrs and a 15% increase in 3-in-4s.

Compared to the IP methods, the Hybrid model leverages the advantages of Layer-Cake by considering higher-level metrics during block assignment. In some semesters, the Hybrid model even outperforms GtSP and Zero-GtSP, while offering a strong alternative in others. On average, the Hybrid model results in 15% fewer B2Bs and 20% fewer triples compared to GtSP.

5.2. Flexibility of the Framework

The results presented in the following subsections highlight the flexibility of our framework. These results are derived from routine experiments conducted each semester, allowing us to explore a variety of high-quality schedules and present trade-offs to the University Registrar. Specifically, we focus on Spring 2024, the most recent semester at the time this paper is written.

5.2.1. Flexibility in Controlling the Number of Blocks

Understanding the relationship between the number of blocks and performance of the generated schedule is complex. On one hand, a higher value of k provides more time slots for exam assignments, allowing them to be spread out and potentially improving the metrics. On the other hand, the "virtual block" design enables strategic placement of empty time slots between blocks with high co-enrollment.

As shown in Table 3, the B2B, 2-in-24hr, and 3-in-4 metrics do not follow a linear trend as k increases. This experiment demonstrates that high-quality schedules can be produced with relatively few blocks, though this was achieved with complete flexibility in the placement of empty time slots.

Table 3   Performance of GtSP with Different Numbers of Blocks

# Blocks (k)	Conflict	Triple	B2B	2-in-24hr	3-in-4
20	3	17	2011	2397	396
21	0	22	1950	2574	286
22	0	43	1838	2487	358
23	0	30	2009	2684	397
24	0	60	1910	2477	283

5.2.2. Flexibility in Restricting Time Slot Availability As previously noted, the University Registrar often requests the exclusion of specific time slots to help reduce student stress and accommodate graduation events. To assess the potential impact of these exclusions, we conduct experiments by rerunning Block Sequencing on a 20-block assignment, excluding between 0 and 4 time slots using the virtual block design in Block Sequencing. The time slots considered for exclusion include a Saturday evening (slot 3), a Sunday morning (slot 4), a Friday evening (slot 21), and a Sunday evening (slot 24).

Table 4 presents the results of these exclusions. As more slots are restricted, the model's options to optimize metrics diminishes. We observed a noticeable increase in the number of 2-in-24hrs and triples, suggesting that restricting multiple time slots can result in less favorable student schedules. In response, the team has been working with the Registrar to open additional time slots, which take effects starting from SP24.

Table 4   Performance of GtSP with Different Excluded Time Slots

Excluded Times	Conflict	Triple	B2B	2-in-24hr	3-in-4
{}	3	17	2011	2397	396
{3}	3	25	2101	2297	416
{3, 4}	3	30	1881	2573	424
{3, 4, 21}	3	33	1868	2474	343
{3, 4, 21, 24}	3	41	1993	2772	397

5.2.3. Flexibility in Front-Loading Large Exams In this section, we explore the impact of different exam size and time slot cutoffs for front-loading large exams. We modify the Block Sequencing stage of the GtSP model by experimenting with a range of size thresholds for large exams and the corresponding early time slot cutoffs.

The effects of varying front-loading thresholds align with our expectations, as shown in Figure 6. Intuitively, the smaller the size cutoff and the earlier the slot cutoff, the more pronounced the front-loading effect becomes. As the size cutoff decreases from 400 to 200 and the slot cutoff moves forward from slot 24, large exams becomes more concentrated in earlier time slots, with only a

few scattered large exams remaining on the last few days. The four sub-figures provided below are visualizations we would include when sharing the final schedules with the University Registrar. These visuals clearly depict the temporal distribution of exams, helping the registrar's office to understand room demand and allocate exams accordingly.

The impact of changing the time slot cutoff on scheduling metrics is further examined and displayed in Figure 7. We observe that setting an earlier slot cutoff for large exams is associated with an increase in the number of 2-in-24hrs and triples. This outcome is anticipated, as limiting the available slots for blocks containing large exams reduces the feasible solution space for both Block Sequencing and Post-Processing.

5.3. Regression Analysis on Key Input Parameters

In the previous section, we have observed that the number of blocks, metric weights, size cutoff, and slot cutoff can crucially impact the metrics. To better understand the trade-offs among these parameters, we use the GtSP model to generate schedules for all combinations of parameter settings listed in Table 5, evaluated over the three spring semesters. In total, we produce 2430 distinct schedules (810 per semester) and computed their metrics. We then fit a Random Forest Regression model on these results and perform a SHAP Value analysis (Lundberg and Lee 2017) to evaluate the influence of each parameter on the metrics. SHAP values provide a measure of feature importance, quantifying the contribution of each parameter to the model's output. In this case, we choose the model output to be the number of students who would need rescheduled exams, as this is the most key concern. Recall that, at Cornell University, students with more than two exams in a 24-hour period are eligible for a rescheduled exam.
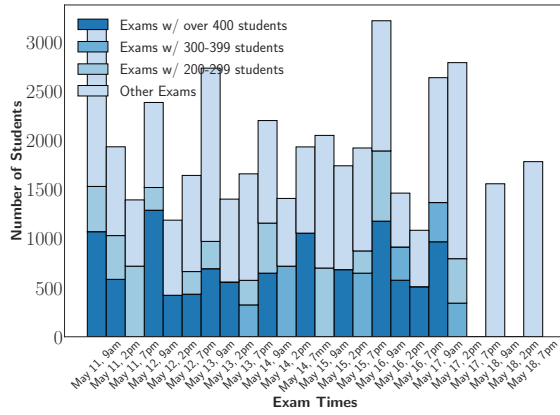
Figure 8 explores how size cutoffs, time slot cutoffs, and the number of blocks influence scheduling flexibility and the need for rescheduling. Higher size cutoffs generally correlate with fewer reschedules, while low size cutoffs often require higher slot cutoffs to maintain flexibility.
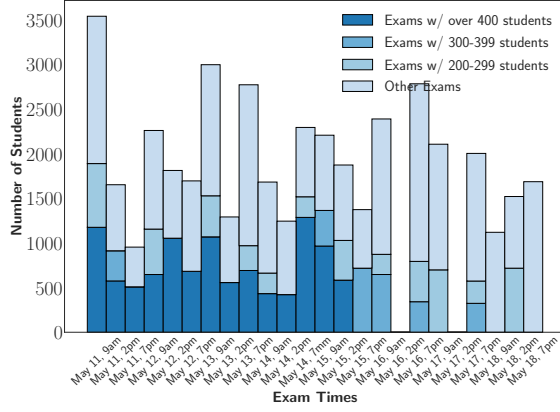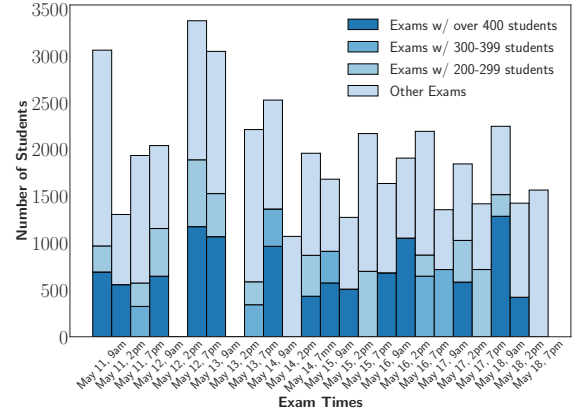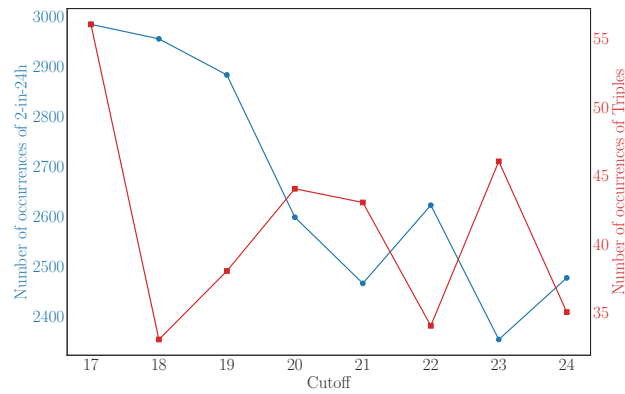
(a) Size Cutoff: 400; Slot Cutoff: 15
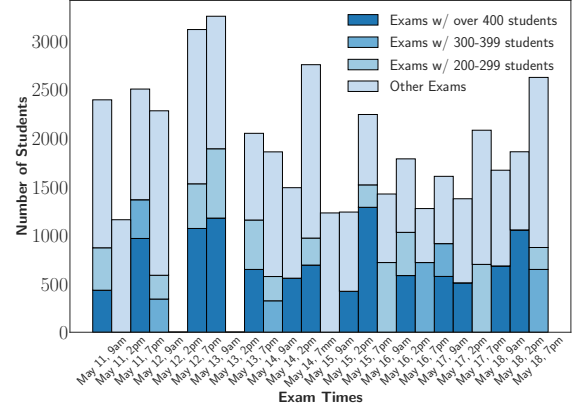
(b) Size Cutoff: 400; Slot Cutoff: 24

(c) Size Cutoff: 200; Slot Cutoff: 21

(d) Size Cutoff: 200; Slot Cutoff: 24

Figure 6   Distribution of Exams with Different Size and Slot Cutoffs

Figure 7   Impact of Different Slot Cutoffs on 2-in-24hrs and Triples

Table 5   Parameters and Their Values Used for the Regression Analysis

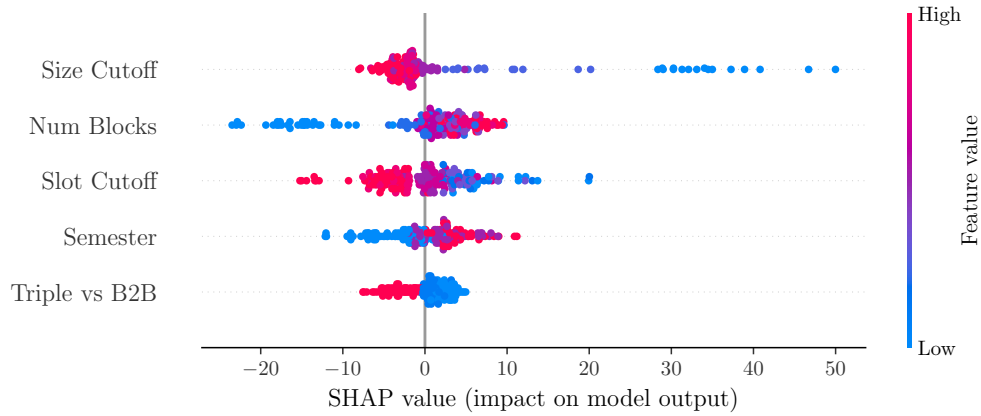| Parameter	Values
Number of Blocks	[19, 20, 21, 22, 23, 24]
Size Cutoff	[150, 200, 250, 300, 350]
Slot Cutoff*	[16, . . . , 24]
Triple vs. B2B	[1, 2, 10]

Note: Schedules were excluded from the data if using a slot cutoff proved infeasible.

Increasing the number of blocks typically improves schedule performance, though outcomes vary more widely when block numbers are low. Additionally, a higher triple vs B2B trade-off appears to slightly reduce rescheduled exams. Lastly, schedules from later semesters tend to experience more rescheduling, possibly due to higher enrollment levels.

6. Conclusions

This paper presents a MIP-based optimization framework that leverages a multi-stage Group-then-Sequence approach combined with a Layer-Cake heuristic algorithm to automate the final exam scheduling process at Cornell University. Over five consecutive semesters, we have collaborated closely with the University Registrar to schedule final exams for the entire student body, consistently generating optimized schedules that outperform historical baselines. Our results demonstrate not only the effectiveness of our method but also its capacity to adapt to a wide range of constraints and objectives. Further evidence of the framework's generalizability is provided in Appendix G, which demonstrates its competitive performance against established exam timetabling algorithms on the University of Nottingham dataset.

Furthermore, our ongoing partnerships with multiple universities underscore the inherent flexibility of our framework. By accommodating diverse institutional preferences, our approach facilitates the creation of customized exam schedules that balance improved student outcomes with other operational priorities. For instance, if an institution prefers to avoid Sunday morning exams, our

(a) Feature Importance Analysis Using SHAP Values
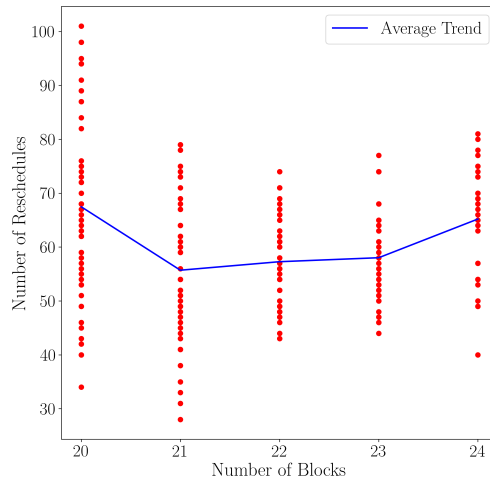
(b) Number of Blocks Vs. Reschedules
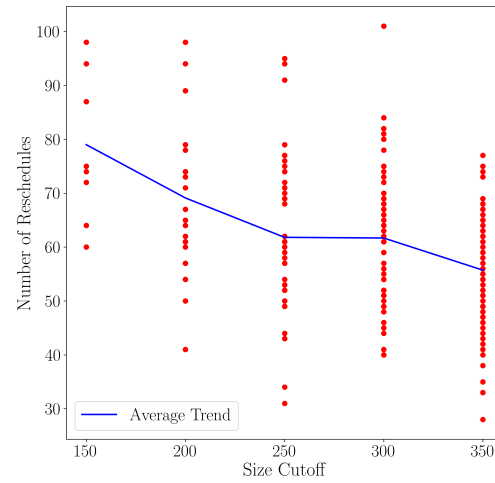
(c) Size Cutoff Vs. Reschedules

Figure 8   Impact of Different Model Parameters on the Number of Rescheduled Exams

model can be readily adjusted to enforce that constraint. Similarly, if minimizing instances of triple

exams is a priority, the objective function can be reweighted accordingly. These features highlight

the robustness and adaptability of our framework, making it a valuable tool for addressing the

complex and varied demands of exam scheduling in higher education.

Acknowledgments

References

Abdullah S, Ahmadi S, Burke EK, Dror M, McCollum B (2007) A tabu-based large neighbourhood search methodology for the capacitated examination timetabling problem. *Journal of the Operational Research Society* 58(11):1494–1502.

Al-Hawari F, Al-Ashi M, Abawi F, Alouneh S (2020) A practical three-phase ilp approach for solving the examination timetabling problem. *International Transactions in Operational Research* 27(2):924–944.

Aldeeb BA, Al-Betar MA, Abdelmajeed AO, Younes MJ, AlKenani M, Alomoush W, Alissa KA, Alqahtani MA (2019) A comprehensive review of uncapacitated university examination timetabling problem. *International Journal of Applied Engineering Research* 14(24):4524–4547.

Archetti C, Speranza MG (2014) A survey on matheuristics for routing problems. *EURO Journal on Computational Optimization* 2(4):223–246.

Avella P, Boccia M, Mannino C, Viglione S (2022) Practice summary: Solving the external candidates exam schedule in norway. *INFORMS Journal on Applied Analytics* 52(2):226–231.

Barnhart C, Bertsimas D, Delarue A, Yan J (2022) Course scheduling under sudden scarcity: Applications to pandemic planning. *Manufacturing & Service Operations Management* 24(2):727–745.

Bazari S, Pooya A, Soleimani Fard O, Roozkhosh P (2023) Modeling and solving the problem of scheduling university exams in terms of new constraints on the conflicts of professors' exams and the concurrence of exams with common questions. *Opsearch* 60(2):877–915.

Bellio R, Ceschia S, Di Gaspero L, Schaerf A (2021) Two-stage multi-neighborhood simulated annealing for uncapacitated examination timetabling. *Computers & Operations Research* 132:105300.

Boschetti MA, Maniezzo V (2022) Matheuristics: using mathematics for heuristic design. *4OR* 20(2):173–208.

Burke E, Bykov Y, Newall J, Petrovic S (2004) A time-predefined local search approach to exam timetabling problems. *Iie Transactions* 36(6):509–528.

Burke EK, Newall JP (1999) A multistage evolutionary algorithm for the timetable problem. *IEEE transactions on evolutionary computation* 3(1):63–74.

Burke EK, Newall JP (2004) Solving examination timetabling problems through adaption of heuristic orderings. *Annals of operations Research* 129:107–134.

Burke EK, Newall JP, Weare RF (1996) A memetic algorithm for university exam timetabling. *Practice and Theory of Automated Timetabling: First International Conference Edinburgh, UK, August 29–September 1, 1995 Selected Papers 1*, 241–250 (Springer).

Caramia M, Dell'Olmo P, Italiano GF (2000) New algorithms for examination timetabling. *International Workshop on Algorithm Engineering*, 230–241 (Springer).

Carter MW, Laporte G, Chinneck JW (1994) A general examination scheduling system. *Interfaces* 24(3):109–120.

Carter MW, Laporte G, Lee SY (1996) Examination timetabling: Algorithmic strategies and applications. *Journal of the operational research society* 47(3):373–383.

Cataldo A, Ferrer JC, Miranda J, Rey PA, Sauré A (2017) An integer programming approach to curriculum-based examination timetabling. *Annals of Operations Research* 258:369–393.

Ceschia S, Di Gaspero L, Schaerf A (2023) Educational timetabling: Problems, benchmarks, and state-of-the-art results. *European Journal of Operational Research* 308(1):1–18.

Çimen M, Belbağ S, Soysal M, Sel Ç (2022) Invigilators assignment in practical examination timetabling problems. *International Journal of Industrial Engineering* 29(3).

Di Gaspero L, Schaerf A (2001) Tabu search techniques for examination timetabling. *Practice and Theory of Automated Timetabling III: Third International Conference, PATAT 2000 Konstanz, Germany, August 16–18, 2000 Selected Papers 3*, 104–117 (Springer).

Elloumi A, Kamoun H, Jarboui B, Dammak A (2014) The classroom assignment problem: Complexity, size reduction and heuristics. *Applied Soft Computing* 14:677–686.

Fonseca GH, Santos HG, Carrano EG, Stidsen TJ (2017) Integer programming techniques for educational timetabling. *European Journal of Operational Research* 262(1):28–39.

Frazier PI, Cashore JM, Duan N, Henderson SG, Janmohamed A, Liu B, Shmoys DB, Wan J, Zhang Y (2022) Modeling for COVID-19 college reopening decisions: Cornell, a case study. *Proceedings of the National Academy of Sciences* 119(2):e2112532119.

García-Sánchez Á, Hernández A, Caro E, Jiménez G (2019) Universidad politécnica de Madrid uses integer programming for scheduling weekly assessment activities. *INFORMS Journal on Applied Analytics* 49(2):104–116.

Gashgari R, Alhashimi L, Obaid R, Palaniswamy T, Aljawi L, Alamoudi A (2018) A survey on exam scheduling techniques. *2018 1st International Conference on Computer Applications & Information Security (ICCAIS)*, 1–5 (IEEE).

Godwin T (2022) Obtaining quality business school examination timetable under heterogeneous elective selections through surrogacy. *Opsearch* 59(3):1055–1093.

Gogos C, Alefragis P, Housos E (2012) An improved multi-staged algorithmic process for the solution of the examination timetabling problem. *Annals of Operations Research* 194:203–221.

Gonzalez G, Richards C, Newman A (2018) Optimal course scheduling for united states air force academy cadets. *Interfaces* 48(3):217–234.

Greenberg KK, Hensel T, Zhu Q, Aarts S, Shmoys DB, Gutekunst SC (2021) An automated tool for optimal classroom seating assignment with social distancing constraints. *IIE Annual Conference. Proceedings*, 429–434 (Institute of Industrial and Systems Engineers (IISE)).

Güler MG, Geçici E, Köroğlu T, Becit E (2021) A web-based decision support system for examination timetabling. *Expert Systems with Applications* 183:115363.

Keskin ME, Döyen A, Akyer H, Güler MG (2018) Examination timetabling problem with scarce resources: a case study. *European Journal of Industrial Engineering* 12(6):855–874.

Kristiansen S, Sørensen M, Stidsen TR (2015) Integer programming for the generalized high school timetabling problem. *Journal of Scheduling* 18:377–392.

Liu B, Zhang Y, Henderson SG, Shmoys DB, Frazier PI (2023) Modeling the risk of in-person instruction during the COVID-19 pandemic. *arXiv preprint arXiv:2310.04563* .

Lundberg SM, Lee SI (2017) A unified approach to interpreting model predictions. Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R, eds., *Advances in Neural Information*

Processing Systems 30, 4765–4774 (Curran Associates, Inc.), URL http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf.

McCollum B, McMullan P, Burke EK, Parkes AJ, Qu R (2007) The second international timetabling competition: Examination timetabling track. Technical report, Technical Report QUB/IEEE/Tech/ITC2007/-Exam/v4. 0/17, Queen's University . . . .

Merlot LT, Boland N, Hughes BD, Stuckey PJ (2003) A hybrid algorithm for the examination timetabling problem. *Practice and Theory of Automated Timetabling IV: 4th International Conference, PATAT 2002, Gent, Belgium, August 21-23, 2002. Selected Revised Papers 4*, 207–231 (Springer).

Müller T (2016) Real-life examination timetabling. *Journal of Scheduling* 19:257–270.

Müller T, Rudová H, Müllerová Z (2024) Real-world university course timetabling at the international timetabling competition 2019. *Journal of Scheduling* 1–21.

Post G, Di Gaspero L, Kingston JH, McCollum B, Schaerf A (2016) The third international timetabling competition. *Annals of Operations Research* 239:69–75.

Qu R, Burke EK, McCollum B, Merlot LT, Lee SY (2009) A survey of search methodologies and automated system development for examination timetabling. *Journal of Scheduling* 12:55–89.

Siew ESK, Goh SL, Kendall G, Sabar NR, Abdullah S, et al. (2024) A survey of solution methodologies for exam timetabling problems. *IEEE Access* 12:41479–41498.

Sörensen K (2015) Metaheuristics—the metaphor exposed. *International Transactions in Operational Research* 22(1):3–18.

Strichman O (2017) Near-optimal course scheduling at the technion. *Interfaces* 47(6):537–554.

Woumans G, De Boeck L, Beliën J, Creemers S (2016) A column generation approach for solving the examination-timetabling problem. *European Journal of Operational Research* 253(1):178–194.

Appendix A: The Block Assignment Problem

Table 6   Notations for the Block Assignment Problem

Notation	Description
Sets	
B	Set of exam blocks = $\{0, \ldots, k-1\}$, indexed by b
E	Set of exams, indexed by e
Parameters	
k	Desired number of exam blocks
c_{ij}	The number of students co-enrolled in both exam i and exam j
Decision Variables	
x_{eb}	= 1 if exam e is assigned to block b; 0 otherwise
y_{ij}	= 1 if exam i and exam j are assigned to the same block; 0 otherwise
z_{ij}	= 1 if exam i and exam j are assigned to neighboring blocks; 0 otherwise

A.1.   Min-Conflict IP Formulation

The MIN_CONFLICT_IP problem is presented as follows.

$$\min \quad \sum_{i,j \in E} c_{ij} y_{ij} \tag{2}$$

$$\text{s.t.} \quad \sum_{b \in B} x_{eb} = 1, \forall \, e \in E \tag{3}$$

$$x_{ib} + x_{jb} \leq 1 + y_{ij}, \forall \, i, j \in E, \ b \in B \tag{4}$$

$$x_{eb} \in \{0, 1\}, \forall \, e \in E, b \in B \tag{5}$$

$$y_{ij} \in \{0, 1\}, \forall \, i, j \in E \tag{6}$$

Objective function (2) ensures that the total number of conflicts is minimized. Constraints (3) ensure that each exam is assigned to exactly one block. Constraints (4) ensure that $y_{ij} = 1$ when both exam i and exam j are assigned to the same block and $= 0$ otherwise. Constraints (5) and (6) are binary constraints for the variables.

A.2.   Zero-Conflict IP Formulation

The ZERO_CONFLICT_IP problem is formulated similar to the MIN_CONFLICT_IP problem but with an updated TSP objective function and additional constraints.

The objective function is updated to:

$$\min \quad \sum_{i,j \in E} c_{ij} z_{ij} \tag{7}$$

And the following constraints are added:

$$\text{s.t.} \quad \sum_{i,j \in E} y_{ij} c_{ij} = 0 \tag{8}$$

$$x_{ib} + x_{jb+1} \leq 1 + z_{ij}, \forall\, i, j \in E,\ b \in \{0, \ldots, k-2\} \tag{9}$$

$$z_{ij} \in \{0, 1\}, \forall\, i, j \in E \tag{10}$$

Objective function (7) minimizes the total number of conflicts of neighbouring blocks. Constraint (8) ensures that there is no conflict within a block. Constraints (9) and (10) define the z_{ij} variables and impose the binary constraints, respectively.

Appendix B: The Max-Clique Problem

Table 7   Notations for the Max-Clique Problem

Notation	Description
Index Sets	
NN_e	Set of non-neighbors of exam e
Parameters	
h_e	The number of non-neighbors of exam e
Decision Variables	
x_e	= 1 if exam e is included in the clique; 0 otherwise

B.1.   The Max-Clique IP Formulation

The MAX_CLIQUE_IP problem is presented as follows.

$$\max \quad \sum_{e \in E} x_e \tag{11}$$

$$\text{s.t.} \quad \sum_{e' \in NN_e} x_{e'} \leq h_e(1 - x_e), \forall e \in E \tag{12}$$

$$x_e \in \{0, 1\}, \forall e \in E \tag{13}$$

Objective function (11) minimizes the number of exams included in the clique. Constraints (12) ensure that if an exam is included in the clique, then none of its non-neighbors can be included. Constraints (13) enforce binary values for the decision variables.

Table 8   Notations for the Block Sequencing Problem

Notation	Description
Sets	
B	Set of exam blocks, $B := \{0, \ldots, |S| - 1\}$
S	Set of time slots available, $S = \{0, \ldots, |S| - 1\}$
$Tripleday$	Subset of time slots that index the start of triples on the same day, $Tripleday \subset S$
$Triple24hr$	Subset of time slots that index the start of triples in 24 hours, $Triple24hr \subset S$, with $Tripleday \cap Triple24hr = \emptyset$
$Tripleslots$	$Tripleday \cup Triple24hr$
$B2B_{evemorn}$	Subset of time slots that index the start of evening-morning back-to-backs, $B2B_{evemorn} \subset S$
$B2B_{other}$	Subset of time slots that index the start of the other types of back-to-backs, $B2B_{other} \subset S$, with $B2B_{evemorn} \cap B2B_{other} = \emptyset$
$Earlyslots$	Subset of slots earlier than a slot cutoff, $Earlyslots \subseteq S$
$LargeB$	Subset of blocks that have enrollment larger than a size cutoff, $LargeB \subseteq B$
Parameters	
n_s	Next time slot of time slot s in the cycle, $n_s := (s+1) \mod |S|$
β	Penalty weight for three exams in 24 hours
α	Penalty weight for three exams on the same day (Having one exam in the afternoon, one in the evening, and one on the next morning would count as having three exams in 24 hours, not three exams on the same day)
γ_1	Penalty weight for evening-morning back-to-back exams
γ_2	Penalty weight for the other types of back-to-back exams
δ	Penalty weight for three exams in four slots
p_{ij}	The number of students co-enrolled in both block i and block j
$t_{ij\ell}$	The number of students co-enrolled in all blocks i, j and ℓ
Decision Variables	
$x_{ij\ell s}$	$= 1$ if block i is placed in slot s, block j is placed in slot $(s+1) \mod |S|$, and block ℓ is placed in slot $(s+2) \mod |S|$; 0 otherwise
$y_{ij\ell}$	$= 1$ if block i, j, ℓ appear in a sequence where each pair of consecutive blocks is directly back-to-back; 0 otherwise
$z_{ij\ell m}$	$= 1$ if block i, j, ℓ, m appear in a sequence where each pair of consecutive blocks is directly back-to-back; 0 otherwise

Appendix C: The Block Sequencing Problem

The BLOCK_SEQUENCING_IP problem is presented as follows.

C.1.   Objective Function

$$\min \sum_{i,j,\ell \in B, s \in B2B_{evemorn}} \gamma_1 p_{ij} x_{ij\ell s} \tag{14}$$

$$+ \sum_{i,j,\ell \in B, s \in B2B_{other}} \gamma_2 p_{ij} x_{ij\ell s}$$

$$+ \sum_{i,j,\ell \in B, s \in Tripleday} \alpha t_{ij\ell} x_{ij\ell s}$$

$$+ \sum_{i,j,\ell \in B, s \in Triple24hr} \beta t_{ij\ell} x_{ij\ell s}$$

$$+ \sum_{i,j,\ell,m \in B} \delta (t_{ij\ell} + t_{i\ell m}) \cdot z_{ij\ell m}$$

Objective function (14) minimizes the weighted sum of different higher-level conflicts, including back-to-backs, three exams in a row, and three exams in four slots.

C.2. Constraints

$$\text{s.t.} \quad \sum_{j,\ell \in B, s \in S} x_{ij\ell s} = 1, \forall i \in B \tag{15}$$

$$\sum_{i,\ell \in B, s \in S} x_{ij\ell s} = 1, \forall j \in B \tag{16}$$

$$\sum_{i,j \in B, s \in S} x_{ij\ell s} = 1, \forall \ell \in B \tag{17}$$

$$\sum_{i,j,\ell \in B} x_{ij\ell s} = 1, \forall s \in S \tag{18}$$

$$x_{ii\ell s} = 0, \forall i, \ell \in B, s \in S \tag{19}$$

$$x_{ijis} = 0, \forall i, j \in B, s \in S \tag{20}$$

$$x_{ijjs} = 0, \forall i, j \in B, s \in S \tag{21}$$

$$\sum_{i \in B} x_{ij\ell s} = \sum_{m \in B} x_{j\ell mn_s}, \forall j, \ell \in B, s \in S \tag{22}$$

$$y_{ij\ell} = \sum_{s \in Tripleslots} x_{ij\ell s}, \forall i, j, \ell \in B \tag{23}$$

$$z_{ij\ell m} \geq y_{ij\ell} + y_{j\ell m} - 1, \forall i, j, \ell, m \in B \tag{24}$$

$$x_{ij\ell s} \in \{0, 1\}, \forall i, j, \ell \in B, s \in S \tag{25}$$

$$y_{ij\ell} \in \{0, 1\}, \forall i, j, \ell \in B \tag{26}$$

$$z_{ij\ell m} \in \{0, 1\}, \forall i, j, \ell, m \in B \tag{27}$$

Constraints (15) - (17) ensure that each block appears as the first, second, and third of a triple exactly once. Constraint (18) ensures that each time slot is assigned to exactly one block. Constraints (19) - (21) prevent repetition in group sequences. Constraints (22) define the triplet time slot continuity (e.g. if $x_{1,2,3,1} = 1$, then $x_{2,3,\ell,2} = 1$ for some block ℓ). Constraints (23)-(27) define the $y_{ij\ell}$ and $z_{ij\ell m}$ variables and impose the binary constraints.

Front-loading constraints (28) can be added to keep large exams out of the last couple of slots:

$$\sum_{j,\ell \in B, s \in Earlyslots} x_{ij\ell s} = 1, \forall i \in LargeB \tag{28}$$

Table 9 Notations for the Schedule_IP Problem

Notation	Description
Sets	
E'	A subset of exams, $E' \subseteq E$
S'	A subset of available time slots, $S' \subseteq S$
$B2B$	Set of time slots that index the start of back-to-backs, $B2B \subset S'$
$2i3$	Set of time slots that index the start of two-in-threes, $2i3 \subset S'$
Parameters	
λ_1	Relative weight between conflicts and back-to-backs; default is 1000
λ_2	Relative weight between two-in-threes and back-to-backs; default is 0.5
ζ_s	Front-loading penalty for large exams scheduled to slot s; default is s^{300}
$other\_conf_{es}$	Total number of conflicts between exam e and any exam $e' \in E \setminus \{e\}$ scheduled to slot s (i.e., $x_{e's} = 1$)
$other\_B2Bs_{es}$	Total number of back-to-backs between exam e and any exam $e' \in E \setminus \{e\}$ scheduled to slot s
$other\_2i3s_{es}$	Total number of two-in-threes between exam e and any exam $e' \in E \setminus \{e\}$ scheduled to slot s
Decision Variables	
x_{es}	= 1 if exam $e \in E'$ is in slot s; 0 otherwise
y_{ij}	= 1 if exams $i, j \in E'$ are in the same slot; 0 otherwise
z_{ij}	= 1 if exams $i, j \in E'$ are in consecutive slots; 0 otherwise
w_{ij}	= 1 if exams $i, j \in E'$ are two slots apart; 0 otherwise

Appendix D: The Post-Processing Algorithm

Algorithm 1 details the MIP-based local search algorithm used in the Post-Processing Stage.

The SCHEDULE_IP problem used in both the Post-Processing local search algorithm and the Layer-Cake heuristic

algorithm is presented below.

D.1. Objective Function

$$\min \quad \lambda_1 (\sum_{i,j \in E'} 0.5 y_{ij} c_{ij} + \sum_{e \in E', s \in S'} x_{es} other\_conf_{es}) + \tag{29}$$

$$\sum_{i,j \in E'} z_{ij} c_{ij} + \sum_{e \in E', s \in S'} x_{es} other\_b2bs_{es} +$$

$$\lambda_2 (\sum_{i,j \in E'} w_{ij} c_{ij} + \sum_{e \in E', s \in S'} x_{es} other\_2i3s_{es}) +$$

$$\sum_{e \in E', s \in S' \setminus Earlyslots} x_{es} \zeta_s$$

Objective function (29) minimizes the weighted sum of conflicts, back-to-backs, and two-in-threes.

D.2. Constraints

$$\text{s.t.} \quad \sum_{s \in S'} x_{es} = 1, \forall e \in E' \tag{30}$$

$$y_{ij} \geq x_{is} + x_{js} - 1, \forall i, j \in E', s \in S' \tag{31}$$

$$z_{ij} \geq x_{is} + x_{js+1} - 1, \forall i, j \in E', s \in B2B \tag{32}$$

$$w_{ij} \geq x_{is} + x_{js+2} - 1, \forall i, j \in E', s \in 2i3 \tag{33}$$

$$x_{es} \in \{0, 1\}, \forall e \in E', s \in S' \tag{34}$$

$$y_{ij} \in \{0, 1\}, \forall i, j \in E' \tag{35}$$

$$z_{ij} \in \{0, 1\}, \forall i, j \in E' \tag{36}$$

$$w_{ij} \in \{0, 1\}, \forall i, j \in E' \tag{37}$$

Constraints (30) ensures that every exam is assigned to exactly one slot. Constraints (31)-(37) define the y_{ij}, z_{ij}, and w_{ij} variables and impose the binary constraints.

Appendix E: The Layer-Cake Heuristic Algorithm

Algorithm 2 details the MIP-based Layer-Cake heuristic algorithm.

To speed up the algorithm, we construct a warm-start solution for Schedule_IP at each iteration of Layer-Cake as follows:

1. Exams that are re-entering are assigned time slots based on the results from the previous layer.

2. New exams, not present in the last layer, are first ordered by size and then assigned time slots greedily. For each exam, the time slot that results in the lowest score, as determined by Get_Score_Individual, is selected, considering the exams that have already been placed.

Appendix F: Experimental Setting

By default, we solve the Min_Conflict_IP problem for Block Assignment. For Block Sequencing, Post-Processing, and Layer-Cake, the following parameters are used unless otherwise stated. The ratios of Conflicts to B2Bs, Triples to B2Bs, 3-in-4s to B2Bs, and 2-in-24hrs to B2Bs are 1000, 10, 5, and 0.5, respectively. For front-loading parameters, the default cutoff for early time slots is 23, and the large exam size cutoff is 300. No time slot is excluded from the available set by default. In the experiments presented, we set a time limit of 1500 seconds for the Block_Sequencing_IP model,

---

Algorithm 1 A MIP-based Local Search Algorithm for Post-Processing

---

1: **Input:** $base\_sched, S$ ▷ A given complete schedule that maps the set of exams to the set of time slots S
2: $n \leftarrow 25$ ▷ Number of exams to be rescheduled
3: $f_{step}, b_{step} \leftarrow \text{ROUND}(0.4n), \text{ROUND}(0.2n)$
4: $sched \leftarrow base\_sched$
5: $i \leftarrow 0$ ▷ Starting index of exams to be rescheduled
6: **while** $i < |sched|$ **do**
7:      $best\_score \leftarrow \text{GET\_SCORE}(sched)$ *
8:      $E \leftarrow \text{SORT\_BY\_DENSITY}(sched)$ ▷ Sort the schedule by the density of bad events
9:      $new\_sched \leftarrow \text{RESCHEDULE}(sched, E[i:i+n])$ ▷ Reschedule the next n exams from index i
10:      $new\_score \leftarrow \text{GET\_SCORE}(new\_sched)$
11:      **if** $best\_score \leq new\_score$ **then**
12:          $i \leftarrow i + f_{step}$ ▷ Increase the starting index by a larger step if no improvement was found
13:      **else**
14:          $i \leftarrow max(0, i - b_{step})$ ▷ Decrease the starting index by a smaller step if a better schedule was found
15:          $sched \leftarrow new\_sched$
16:      **end if**
17: **end while**
18: **Output:** $sched$

19: **procedure** GET_SCORE_INDIVIDUAL(e, s)
20:      **return** $\lambda_1 other\_conf_{es} + other\_B2B_{es} + \lambda_2 other\_2i3_{es}$
21: **end procedure**

22: **procedure** GET_SCORE$(sched)$ ▷ Given a schedule, i.e., a mapping from exams to time slots
23:      **return** $\sum_{(e,s) \in sched} \text{GET\_SCORE\_INDIVIDUAL}(e, s)$
24: **end procedure**

25: **procedure** RESCHEDULE$(sched, E')$ ▷ Given a complete schedule and a set of exams to reschedule
26:      Solve SCHEDULE_IP$(E' = E', S' = S)$ and enforce the following additional constraints:
27:          $x_{es} = 1, \forall (e, s) \in sched : e \notin E'$
28:      **return** $\{(e, s) \mid x_{es} = 1\}$
29: **end procedure**

---

600 seconds for the Post-Processing Algorithm, and 1500 seconds for each SCHEDULE_IP in Layer-Cake. In practice, we would extend the time limits reasonably large to allow the models to solve to optimality.

Appendix G: Nottingham Dataset Benchmark

In order to compare our framework with a selection of existing exam timetabling algorithms, we evaluate an adapted version of our model using the University of Nottingham dataset introduced in Burke et al. (1996). Specifically, the Nottingham *a* variant aims to minimize instances of students having back-to-back exams on the same day, whereas the Nottingham *b* variant additionally seeks to minimize a weighted total of within-day and overnight back-to-backs. We selected this dataset not only because of the similarity in the objective functions, but also to demonstrate our framework's capability to handle capacitated examination timetabling problems. Additional constraints are detailed in Qu et al. (2009).

In adapting the Zero-GtSP model for the Nottingham dataset, the following modifications and constraints were introduced:

- **Coincident Courses Grouping:** Courses that share identical student enrollments or are intended to be scheduled concurrently are aggregated into a single exam within the model.

Algorithm 2 A MIP-based Layer-Cake Heuristic Algorithm

1: **Input:** S_2 ▷ Set of available time slots
2: $n_2 \leftarrow 15000$ ▷ The desired number of student-exam pairs per layer
3: $sched, layer \leftarrow \{\}, \emptyset$ ▷ Initializing the schedule
4: $n_3 \leftarrow \textsc{Round}(0.3n_2)$ ▷ Desired number of student-exam pairs that enter the layer for a second time
5: $E \leftarrow \textsc{Sort\_By\_Size}(E)$ ▷ Set of exams ordered by size descending
6: $layer \leftarrow \emptyset$
7: **for** $i = n_2; \ i > 0; \ i = i - \textsc{Size}(E[0])$ **do**   ▷ Construct the first layer
8:      $layer \leftarrow layer \cup E[0]$
9:      $E \leftarrow E \setminus E[0]$
10: **end for**
11: $sched \leftarrow \textsc{Layer\_Schedule}(sched, \emptyset, layer, S_2)$
12: **while** $E \neq \emptyset$ **do**
13:      $old\_layer, overlap \leftarrow layer, \emptyset$
14:      **for** $i = n_3; \ i > 0; \ i = i - \textsc{Size}(old\_layer[-1])$ **do**   ▷ Append set of exams to overlap from the previous layer
15:          $overlap \leftarrow overlap \cup old\_layer[-1]$
16:          $old\_layer \leftarrow old\_layer \setminus old\_layer[-1]$
17:      **end for**
18:      $layer \leftarrow overlap$
19:      **for** $i = n_2 - n_3; \ i > 0; \ i = i - \textsc{Size}(E[0])$ **do**   ▷ Append set of exams to the current layer from the remaining exams
20:          $layer \leftarrow layer \cup E[0]$
21:          $E \leftarrow E \setminus E[0]$
22:      **end for**
23:      $sched \leftarrow \textsc{Layer\_Schedule}(sched, overlap, layer, S_2)$
24: **end while**
25: **Output:** $sched$

26: **procedure** $\textsc{Layer\_Schedule}(sched, \ overlap, \ E', \ S')$ ▷ Given an (incomplete) schedule, a set of re-entering exams, a set of exams to schedule, and a set of available slots
27:      — **Build warm-start solution** x^0
28:      **for all** $(e, s) \in sched$ **with** $e \in overlap$ **do**
29:          set $x^0_{es} \leftarrow 1$
30:      **end for**
31:      $E_{\text{new}} \leftarrow E' \setminus overlap$
32:      sort E_{new} by descending size
33:      **for all** $e \in E_{\text{new}}$ **do**
34:          pick $s^* = \arg\min_{s \in S'} \textsc{Get\_Score\_Individual}(e, s)$
35:          set $x^0_{e\,s^*} \leftarrow 1$
36:      **end for**
37:      — **MIP solve with warm start**
38:      Solve $\textsc{Schedule\_IP}(E' = E', S' = S')$ providing x^0 as the initial solution and enforce the following additional constraints:
39:          $x_{es} = 1, \forall (e, s) \in sched : e \notin overlap$
40:      **return** $\{(e, s) \mid x_{es} = 1\}$
41: **end procedure**

- **Default Block Ordering as Time Slots:** To enforce the specific time slot and sequential constraints in all stages, we assume that the default ordering of blocks generated during the block assignment stage directly corresponds to the sequence of time slots. Consequently, the time slot set S is used interchangeably with the block set B in the subsequent constraints.

- **Capacity Constraints:** Room and seating capacities are strictly enforced for each exam slot, ensuring that no block exceeds its predefined limits. Let $C = 1550$ denote the capacity limit per time slot and let q_e represent the size of exam e. The capacity constraint is then expressed as

$$\sum_{e \in E} q_e x_{es} \leq C, \quad \forall s \in S.$$

- **Specific Time Slot Constraints:** Certain exams are restricted to specific time slots in accordance with institutional requirements (e.g., some exams must be scheduled in the morning or afternoon). Let $S_e \subset S$ be the set of permissible time slots for exam e. The corresponding constraint is given by

$$\sum_{s \in S_e} x_{es} = 1, \quad \forall\, e \in E. \tag{38}$$

- **Sequential Dependencies:** Exams that require a predefined order (i.e., one exam must precede another) are handled via sequencing constraints. Let $\mathcal{P} = \{(e_1, e_1'), (e_2, e_2'), \ldots\}$ denote the set of exam pairs with precedence requirements. Define

$$y_e = \sum_{s \in S} s\, x_{es}, \quad \forall\, e \in E, \tag{39}$$

and impose the ordering condition:

$$y_e + 1 \leq y_{e'}, \quad \forall\, (e, e') \in \mathcal{P}. \tag{40}$$

- **Non-Overlapping Constraints:** Exams identified as incompatible (e.g., due to shared instructors, equipment, or other resource limitations) are scheduled in non-overlapping time slots. Let $\mathcal{G} \subset E$ denote a group of such exams that must be assigned distinct time slots. The corresponding constraint is:

$$\sum_{e \in \mathcal{G}} x_{es} \leq 1, \quad \forall\, s \in S.$$

According to Burke et al. (1996), only the zero-conflict constraint and the slot capacity constraint are treated as hard constraints, while the remaining constraints described above may be considered soft constraints.

For this set of experiments, we used a time limit of 10 hours for block assignment, 3600 seconds for the BLOCK_SEQUENCING_IP model, and 1800 seconds for the Post-Processing Algorithm.

Table 10 compares the solution outcomes of established exam timetabling algorithms (Qu et al. 2009) with those obtained using our adapted Zero-GtSP model. Figure 9 illustrates the distribution of exam schedules generated by our approach. The strong performance of our method relative to the literature underscores the flexibility and robustness of our framework, even under stringent capacity constraints. Moreover, our framework is capable of addressing higher-order conflicts that other algorithms are unable to manage.

Table 10   **Comparison of Objective Value for Nottingham Dataset *a* and *b***

Nottingham	Burke et al. (1996)	Di Gaspero and Schaerf (2001)	Caramia et al. (2000)	Merlot et al. (2003)	Abdullah et al. (2007)	Adapted Zero-GtSP (Ours)
a (26 slots)	53	11	44	2	18	17
a (23 slots)	269	123	–	88	–	0

Nottingham	Burke and Newall (1999)	Di Gaspero and Schaerf (2001)	Merlot et al. (2003)	Burke and Newall (2004)	Burke et al. (2004)	Adapted Zero-GtSP (Ours)
b (26 slots)	–	–	–	–	–	244
b (23 slots)	519	751	401	545	384	330
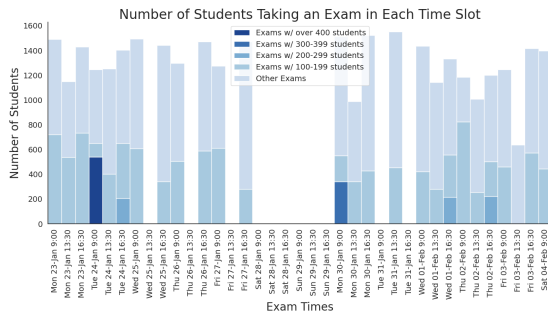
(a) Nottingham *a* with 26 slots
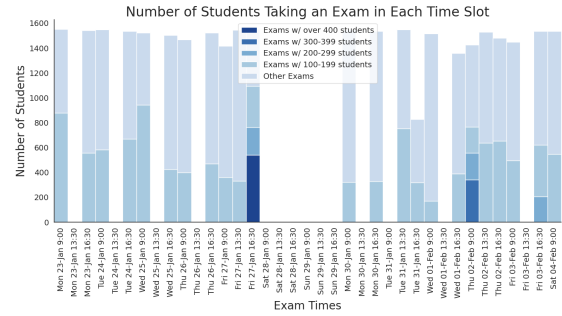
(b) Nottingham *a* with 23 slots
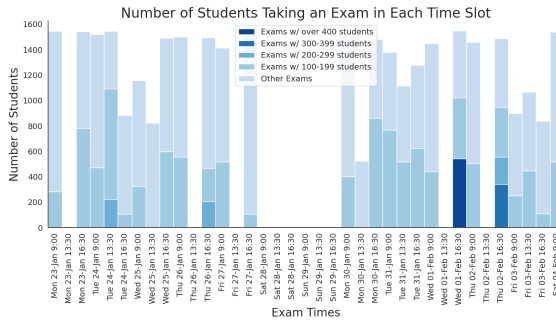
(c) Nottingham *b* with 26 slots
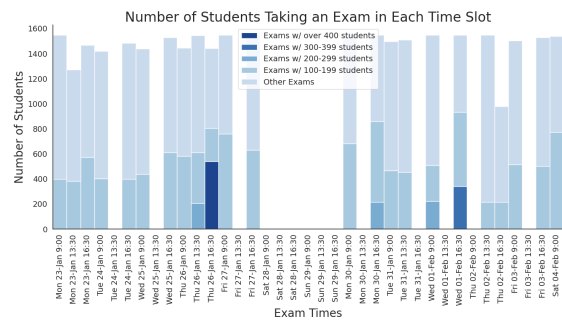
(d) Nottingham *b* with 23 slots

Figure 9   **Distribution of Exams Scheduled by Adapted Zero-GtSP on Nottingham Dataset *a* and *b***