The Open European Journal of Engineering and Scientific Research (OEJESR)
E-ISSN: 3062-3383
Volume 2, Issue2, 2026, Pages: 01-22
Journal homepage:
https://easdjournals.com/index.php/oejesr/index

University Course Timetabling Optimization Using Binary Integer Linear Programming:A Case Study of the College of Technical Sciences in Bani Walid

Ali Mahmoud Assabri[1]*, Mohamed Al-Issawi Kajman[2], Abdulbasat Abdulghafar Alfeed[3], Naji Abdalaziz Ali[4], Ahmed S. Mohamed[5]
[1] Department of Engineering Management, College of Technical Sciences, Bani Walid, Libya
[2] Department of Computing and Information Technology, College of Technical Sciences, Bani Walid, Libya
[3] Department of Engineering Management, College of Technical Sciences, Bani Walid, Libya
[4] Department of Mechanical Engineering, College of Technical Sciences, Bani Walid, Libya
[5] Department of Electrical and Electronic Engineering, College of Technical Sciences, Bani Walid, Libya
* *Corresponding Author:* Email: alimahmoudassabri@gmail.com

Article history **Received**: 12-03-2026 **Accepted**: 25-06-2026 **Published**: 06-07-2026

Abstract
University course timetabling remains a classic combinatorial optimization problem, and it sits at the heart of how academic institutions manage their resources. The task becomes considerably harder when operational constraints are tight: teaching facilities are few, lecture durations are fixed, and scheduling periods are limited. A wide range of optimization approaches has been proposed in the literature, yet comparatively few studies examine timetabling environments in which temporal structures are tightly constrained and institutional resources are scarce.
This study presents a Binary Integer Linear Programming (BILP) model that generates conflict-free university course timetables while minimizing theoretical overflow into the laboratory. Binary decision variables are used to assign courses to specific classrooms and time slots. A thorough set of hard constraints governs these assignments: they guarantee timetable feasibility, prevent scheduling conflicts, enforce the exclusive use of classrooms and instructors, and reserve the computer laboratory for practical courses alone.
The approach was evaluated through a real-world case study at the Department of Engineering Management and Administrative Leadership, College of Technical Sciences in Bani Walid, Libya. The department offers 48 undergraduate courses across eight academic semesters, taught by 13 faculty members. Its scheduling environment is notably restrictive: only ten weekly time slots are available, a consequence of a five-day academic week with two three-hour teaching periods per day, together with four theoretical classrooms and one computer laboratory. Under these conditions, the BILP model produced a completely conflict-free timetable with 96.0% overall resource utilization, zero hard-constraint violations (one controlled theoretical overflow into the laboratory was permitted to preserve overall feasibility), and proven mathematical optimality.

Keywords: Binary Integer Linear Programming, Timetabling Optimization, Resource Allocation, Operations Research, Higher Education, Course Scheduling.

تحسين جدولة المقررات الجامعية باستخدام البرمجة الخطية الصحيحة الثنائية: دراسة حالة في كلية العلوم التقنية ببني وليد

علي محمود الصابري[1]*، محمد العيساوي كاجمان[2]، عبدالباسط عبدالغفار الفيض[3]، ناجي عبدالعزيز علي[4]، أحمد صالح محمد[5]
[1]قسم الإدارة الهندسية، كلية العلوم التقنية، بني وليد، ليبيا
[2]قسم الحوسبة وتقنية المعلومات، كلية العلوم التقنية، بني وليد، ليبيا
[3]قسم الإدارة الهندسية، كلية العلوم التقنية، بني وليد، ليبيا
[4]قسم الهندسة الميكانيكية، كلية العلوم التقنية، بني وليد، ليبيا
[5]قسم الهندسة الكهربائية والإلكترونية، كلية العلوم التقنية، بني وليد، ليبيا

الملخص

تُعد جدولة المقررات الجامعية من المشكلات الكلاسيكية في مجال **التحسين التوافقي**، كما تمثل عنصرًا أساسيًا في كيفية إدارة المؤسسات التعليمية لمواردها. وتزداد صعوبة هذه المهمة بصورة ملحوظة عندما تكون القيود التشغيلية صارمة، مثل محدودية المرافق التعليمية، وثبات مدة المحاضرات، ومحدودية الفترات الزمنية المتاحة للجدولة. وقد اقترحت الأدبيات العلمية مجموعة واسعة من أساليب التحسين لمعالجة هذه المشكلة، إلا أن عددًا محدودًا نسبيًا من الدراسات تناول بيئات الجدولة التي تكون فيها الهياكل الزمنية مقيدة بشدة والموارد المؤسسية محدودة.

تقدم هذه الدراسة **نموذجًا للبرمجة الخطية الصحيحة الثنائية (BILP)** يهدف إلى إنشاء جداول دراسية جامعية خالية من التعارضات، مع تقليل اللجوء النظري إلى استخدام المختبر كمخرج فائض للقاعات الدراسية النظرية. وقد استُخدمت متغيرات قرار ثنائية لإسناد المقررات إلى قاعات دراسية وفترات زمنية محددة. كما تحكم هذه الإسنادات مجموعة شاملة من **القيود الصارمة(Hard Constraints)** ، تضمن صلاحية الجدول، وتمنع تعارضات الجدولة، وتفرض الاستخدام الحصري للقاعات وأعضاء هيئة التدريس، وتقصر استخدام مختبر الحاسوب على المقررات العملية فقط.

تم تقييم هذا النهج من خلال **دراسة حالة واقعية** في قسم الإدارة الهندسية والقيادة الإدارية بكلية العلوم التقنية في بني وليد، ليبيا. يقدم القسم **48مقررًا دراسيًا لمرحلة البكالوريوس** موزعة على **ثمانية فصول دراسية**، ويقوم بتدريسها **13عضوًا من أعضاء هيئة التدريس** .وتتميز بيئة الجدولة في القسم بدرجة عالية من القيود؛ إذ لا يتوفر سوى **عشر فترات زمنية أسبوعية**، نتيجة اعتماد أسبوع دراسي من خمسة أيام، بواقع فترتين تدريسيّتين مدة كل منهما ثلاث ساعات يوميًا، إلى جانب توفر **أربع قاعات دراسية نظرية ومختبر حاسوب واحد**.

وفي ظل هذه الظروف، تمكن نموذج **BILP**من إنتاج جدول دراسي **خالٍ تمامًا من التعارضات**، مع تحقيق **نسبة استخدام إجمالية للموارد بلغت 96.0%**، وعدم وجود أي انتهاكات للقيود الصارمة، مع السماح **باستخدام محدود ومدروس لمختبر الحاسوب لاستيعاب مقرر نظري واحد** بهدف الحفاظ على قابلية الجدول للتنفيذ بصورة كلية. كما أثبت النموذج **الأمثلية الرياضية للحل**.

الكلمات المفتاحية : البرمجة الخطية الصحيحة الثنائية، تحسين الجدولة، تخصيص الموارد، بحوث العمليات، التعليم العالي، جدولة المقررات الدراسية.

1. Introduction

University course timetabling is widely recognized as one of the most challenging combinatorial optimization problems in higher education management [1], [2]. The task is to assign courses, instructors, classrooms, and student groups to available time slots while respecting a broad array of institutional constraints. The quality of the resulting timetable has direct consequences for resource utilization, teaching efficiency, faculty workload, and the overall academic experience of students [3], [4]. As universities continue to expand their academic programs and student populations, traditional manual scheduling has become slow, inefficient, and increasingly prone to conflicts [5], [6].

The University Course Timetabling Problem (UCTTP) has attracted sustained attention from researchers in operations research and artificial intelligence. Because the problem is NP-hard [15], a wide range of solution techniques has been proposed: heuristic methods; metaheuristic algorithms such as Genetic Algorithms [19], [37], Simulated Annealing [38], and Tabu Search [10]; and exact approaches such as Constraint Programming and Integer Linear Programming (ILP) [12], [13]. Heuristic methods often deliver satisfactory solutions within reasonable computation times, but they cannot guarantee true mathematical optimality [14]. Integer Linear Programming, by contrast, yields mathematically optimal solutions whenever the problem size and the available computational resources permit [15], [16].

Despite this extensive literature, relatively few studies have examined scheduling environments that combine limited educational resources with rigid temporal structures [17], [18]. In many institutions, particularly in developing countries, academic policies impose fixed lecture durations, a small number of daily teaching periods, and restricted classroom availability. Conditions of this kind add considerably to the complexity of the scheduling process and narrow the practical applicability of many existing timetabling models.

The College of Technical Sciences in Bani Walid offers a practical instance of just such an environment. The Department of Engineering Management and Administrative Leadership offers 48 undergraduate courses distributed across eight academic semesters. The entire scheduling process must operate within five teaching rooms: four theoretical classrooms and a single computer laboratory. Institutional regulations further require every course to be delivered as a continuous three-hour lecture, and only two teaching periods are available each day across a standard five-day academic week. Together these constraints produce a dense scheduling environment in which preparing a timetable by hand is difficult and conflict-prone.

To address these challenges, this study proposes a Binary Integer Linear Programming (BILP) model that generates conflict-free, fully feasible university course timetables while enforcing laboratory requirements. The formulation incorporates a exhaustive set of hard constraints covering classroom allocation, instructor availability, student cohort conflicts, and laboratory-specific requirements. We implement the model in Python using the PuLP optimization framework [21] and evaluate it through a real-world case study built on actual institutional data.

The main contributions of this research are summarized as follows:

- Developing a Binary Integer Linear Programming model tailored to university timetabling under restrictive scheduling conditions.
- Integrating multiple institutional constraints within a single mathematical optimization framework.

- Evaluating the proposed model on empirical, real-world data from the College of Technical Sciences in Bani Walid.
- Demonstrating the practical effectiveness of the approach in producing a feasible timetable while enforcing laboratory requirements and supporting data-driven academic planning.
- Providing a sensitivity analysis that examines the robustness of the model under varying institutional capacity parameters.
- Conducting a comparative performance analysis against manual scheduling and metaheuristic approaches.

The remainder of this paper is organized as follows. The rest of this introductory section states the research problem (Section 1.1), sets out the research objectives (Section 1.2), formulates the research hypotheses (Section 1.3), discusses the significance of the study (Section 1.4), and introduces the definitions, model assumptions, and research boundaries (Section 1.5). Section 2 reviews the related scientific literature. Section 3 describes the research methodology and procedures, and Section 4 provides the theoretical background of Binary Integer Linear Programming. Section 5 presents the mathematical formulation and computational implementation; Section 6 reports the analytical results and discussion, and Section 7 gives the sensitivity analysis and comparative evaluation. Finally, Section 8 concludes the paper and offers recommendations for future research.

1.1. Research Problem

University course timetabling is a complex combinatorial optimization problem: courses, instructors, classrooms, and student cohorts must all be allocated to a limited set of time slots while numerous overlapping institutional constraints are satisfied [22]. As academic programs expand and resource limitations become more pronounced, producing a feasible and entirely conflict-free timetable grows harder, particularly when administrators depend on manual procedures [23].

Many optimization models have been proposed for the University Course Timetabling Problem (UCTTP), but a large share of them rest on the assumption of a relatively flexible scheduling environment [24] — one with multiple daily teaching periods, ample classroom availability, or less restrictive institutional policies. Models built on such assumptions may not transfer directly to institutions operating under constrained scheduling conditions [25].

The Department of Engineering Management and Administrative Leadership at the College of Technical Sciences in Bani Walid represents exactly such a constrained environment. The department currently offers 48 undergraduate courses distributed across eight academic semesters, and its teaching resources are limited to four theoretical classrooms and one computer laboratory. Institutional regulations require each course to be delivered in a continuous three-hour lecture format, with only two teaching periods per day over a standard five-day academic week. These restrictions shrink the entire scheduling horizon to ten weekly time slots and add substantially to the difficulty of constructing a workable timetable.

Under these conditions, manual timetable preparation becomes inefficient and liable to classroom conflicts, instructor overlaps, student scheduling conflicts, and suboptimal resource utilization. There is a clear practical need for an optimization model capable of satisfying all institutional constraints while keeping theoretical overflow into the laboratory space to a minimum.

The primary research problem addressed in this study can be formulated as follows: How can a Binary Integer Linear Programming model be developed to generate an optimal, conflict-free university course timetable that satisfies all institutional constraints and efficiently utilizes available educational resources within a highly constrained academic scheduling environment?

1.2. Research Objectives

The primary objective of this study is to develop and evaluate a Binary Integer Linear Programming (BILP) model capable of generating an optimal, conflict-free university course timetable under constrained institutional scheduling conditions. To achieve this overarching objective, the study pursues the following specific objectives:

1. **Mathematical Formulation:** To develop a mathematical optimization model based on Binary Integer Linear Programming that accurately represents the university course timetabling problem, including a complete objective function, decision variables, and constraint formulations.
2. **Constraint Integration:** To express the institutional scheduling requirements as a detailed set of hard constraints, encompassing classroom availability, instructor availability, student cohort conflicts, lecture duration, and specialized laboratory allocation requirements.
3. **Feasibility Generation:** To generate a feasible and conflict-free timetable that assigns all 48 courses to appropriate classrooms and time slots while satisfying every institutional constraint.
4. **Resource Optimization:** To optimize the use of available educational resources through efficient classroom allocation and the exclusive assignment of practical courses to the computer laboratory.

5. **Empirical Validation:** To validate the practical effectiveness of the proposed model through its application to a real-world case study at the Department of Engineering Management and Administrative Leadership, College of Technical Sciences in Bani Walid.

6. **Performance Analysis:** To analyze the generated timetable in terms of feasibility, resource utilization, conflict elimination, temporal distribution balance, and overall scheduling performance, thereby assessing the practical applicability of the proposed approach to automated academic planning.

7. **Sensitivity Analysis:** To assess the robustness of the proposed model by varying key institutional parameters, including the number of classrooms and available time slots.

8. **Comparative Evaluation:** To compare the performance of the proposed BILP model against manual scheduling approaches and metaheuristic methods reported in the literature.

1.3. Research Hypotheses

To allow empirical validation, the following testable hypotheses and acceptance criteria are established:

H1 (Optimality Hypothesis): The proposed BILP model will generate a mathematically optimal solution (zero optimality gap) within a computational time not exceeding 5 seconds.

H2 (Feasibility Hypothesis): The model will achieve 100% elimination of hard-constraint violations, including zero room double-bookings, zero lecturer overlaps, and zero cohort conflicts, with at most one controlled theoretical overflow into the laboratory as explicitly permitted by Constraint 6.

H3 (Utilization Hypothesis): The model will attain the theoretical room-slot utilization ceiling of 96% (48 assigned courses out of 50 available room-slots) while satisfying all hard constraints. Because every feasible solution assigns exactly 48 courses, this ceiling follows arithmetically from feasibility itself; the hypothesis therefore concerns reaching full feasibility at this ceiling, not treating the percentage as a separate optimization achievement.

Acceptance Criteria: A solution is deemed acceptable if and only if it: (i) satisfies all hard constraints with zero violations (allowing at most one controlled theoretical overflow); (ii) achieves a proven optimality gap of 0.0%; and (iii) completes execution within 5 seconds of CPU time.

1.4. Importance of the Problem/Research

The significance of this research lies above all in its direct application to a constrained, real-world educational setting, where it forms a tangible link between theoretical operations research and everyday academic administration [26]. Moving from traditional manual scheduling to an automated, optimization-based approach can bring measurable gains in operational efficiency. For the College of Technical Sciences in Bani Walid, adopting such a model translates directly into an optimized, verifiably conflict-free academic experience for students and a more systematically managed workload for faculty [27].

The analytical methodology developed in this study also serves as a scalable template. Administrators can adapt the framework and apply it to other faculties facing similarly rigid scheduling paradigms, such as those built around long continuous lecture blocks [28]. In addition, the study contributes to the literature by addressing a setting that receives relatively little attention: severely resource-constrained environments, which are common in developing countries.

From a broader scientific perspective, this work shows that exact mathematical optimization remains computationally practical and capable of producing high-quality solutions for resource-constrained institutions. This point carries weight at a time when evidence-based decision-making is gaining ground in academic administration.

1.5. Definitions, Model Assumptions, and Research Boundaries

1.5.1. Definitions

For a clear and precise understanding of the proposed optimization model, the following key terms are defined:

Linear Programming (LP): A mathematical optimization technique used to determine the best possible solution to a problem represented by linear objective functions and linear constraints [31]. The general form is: maximize (or minimize) $\boldsymbol{Z = c^T x subject to Ax \leq b, x \geq 0}$.

Integer Linear Programming (ILP): A special case of linear programming in which some or all decision variables are restricted to integer values. When all variables are integers, the formulation is called Pure Integer Linear Programming [32].

Binary Integer Linear Programming (BILP): A special case of integer linear programming in which all decision variables are strictly binary (0 or 1), indicating whether a specific assignment is selected or not [33].

Decision Variable: A binary variable indicating whether a particular course is assigned to a specific classroom during a given time slot.

Hard Constraints: Mandatory institutional requirements that must be fully satisfied for a timetable to be considered feasible. These include classroom availability, instructor availability, student cohort conflicts, and laboratory allocation requirements [34].

Feasible Timetable: A timetable that satisfies all hard constraints without generating any scheduling conflicts.

Objective Function: A mathematical expression defining the quantity to be optimized (maximized or minimized). In this study, it represents the minimization of theoretical course overflow into the laboratory space.

1.5.2. Model Assumptions

To simplify the mathematical formulation while preserving a realistic representation of the institutional scheduling environment, the following assumptions are adopted:

The academic week consists of five working days (Sunday through Thursday). Each day contains exactly two teaching periods: Morning (08:00–11:00) and Afternoon (11:30–14:30), resulting in exactly ten weekly time slots. Each course is scheduled exactly once per week as a continuous three-hour lecture block. The department offers 48 distinct courses distributed across eight academic semesters. Educational facilities consist of four theoretical classrooms (each with capacity 25) and one computer laboratory (capacity 20). The department includes 13 active faculty members. Practical courses must be assigned exclusively to the computer laboratory, whereas theoretical courses are scheduled primarily in theoretical classrooms. Faculty availability is assumed unless the instructor is already assigned to another lecture during that specific time slot. Instructor preferences and student elective choices are not considered in the current scope of this model. All courses have equal priority in scheduling.

All instructors are assumed to be available during all ten weekly time slots unless explicitly specified otherwise. This assumption is formalized through an availability matrix $\boldsymbol{A(p,t), where A(p,t) = 1}$ indicates that instructor p is available during time slot t, and $\boldsymbol{A(p,t) = 0}$ indicates unavailability. The current implementation assumes full availability ($\boldsymbol{A(p,t) = 1 for all p and t}$), although the model architecture readily supports the integration of partial availability constraints.

1.5.3. Research Boundaries

The scope of this research is deliberately limited to the Department of Engineering Management and Administrative Leadership at the College of Technical Sciences in Bani Walid, Libya. The proposed model is evaluated using empirical data from a single academic semester. The study also confines itself to solving the university course timetabling problem through Binary Integer Linear Programming under hard institutional constraints. Soft constraints, examination scheduling, classroom capacity optimization beyond basic room-type matching, individual faculty preferences, and complex multi-objective optimization fall outside the scope of this research and are recommended as directions for future work.

2. Literature Review

The use of operations research in educational timetabling has a long history that runs from early heuristic procedures to today's exact mathematical models [35]. This development rests on the foundations of linear programming laid down in the middle of the twentieth century. Dantzig's account of the origins and scope of the field [30], together with the standard textbook treatments of Dantzig and Thapa [35], Chvatal [36], and Winston [5], established the principles that later researchers would adapt to complex scheduling problems. The maturation of automated timetabling as a research area in its own right was marked by the Second International Timetabling Competition (ITC-2007), which defined common benchmark instances and evaluation criteria and thereby set a shared research agenda for the community [11].

Recent literature confirms the effectiveness of Integer Linear Programming (ILP) for the University Course Timetabling Problem (UCTTP). Chen et al. [3], in a broad survey of the field, observe that heuristic algorithms remain popular chiefly for their speed, whereas exact methods such as ILP are the only ones that can certify an optimal solution when computational resources allow. Earlier, Schimmelpfeng and Helber [7] had applied an ILP model to a real university setting, managing tens of thousands of binary variables to produce conflict-free schedules—evidence that exact formulations can cope with institutional-scale instances, not merely toy problems.

Several case-based studies reinforce this conclusion. Daskalaki et al. [12] proposed an integer programming formulation for a Greek university that accommodated a wide range of operational constraints, showing that ILP can absorb complex academic requirements without losing tractability. MirHassani [13] developed a computationally effective integer programming model for course timetabling and, through extensive numerical experiments, demonstrated the practical feasibility of the exact approach on problems of realistic size. More recently, Rappos et al. [1] presented a mixed-integer programming approach that treats hard and soft constraints jointly in university timetabling, further underlining the versatility of exact methods.

Advances in programming languages and solver technology have made these techniques far more accessible to practitioners. Python, and in particular the PuLP library, is now well documented as a workable framework for formulating and solving scheduling models of considerable complexity [21]. Thiruvathukal [8] showed how Python and PuLP can be used to automate course scheduling in practice, offering a template that other institutions can follow with modest effort.

The relative merits of exact methods and metaheuristics, meanwhile, remain an open research question. Genetic Algorithms [20], Simulated Annealing [38], Tabu Search [39], and iterative-improvement metaheuristics with composite neighbourhood structures

[9] all offer flexibility and scalability on very large instances, but none can guarantee optimality. On the more descriptive side, Kostuch [29] analysed the university course timetabling problem through a three-phase approach that decomposes the construction of a feasible timetable into successive stages, a decomposition strategy that has influenced much subsequent heuristic work. Abdullah et al. [40] investigated large-neighbourhood search for examination timetabling, while Al-Yakoob and Sherali [41] developed mathematical programming models for class–faculty assignment; taken together, these studies illustrate the complementary roles that heuristic and exact approaches continue to play in timetabling research.

The present study builds on this body of work by applying a binary integer programming model, implemented in Python with PuLP, to a markedly constrained institutional context characterised by three-hour lecture blocks and only two teaching periods per day, and by coupling the optimization with a detailed analytical review of the generated timetable. In contrast to studies that presuppose flexible scheduling environments, the focus here is on the severely resource-constrained settings that are common in developing regions.

3. Research Methodology/Procedures

This research follows an applied quantitative methodology centred on the formulation and execution of a Binary Integer Linear Programming model. The work proceeds through a structured pipeline of four stages: data acquisition, mathematical formulation, computational implementation, and analytical review.

3.1. Data Acquisition and Structuring

Empirical data from the target institution was collected in direct collaboration with the Department of Engineering Management and Administrative Leadership. The dataset comprises the following elements:

Data Element	Value / Description
Total courses	**48**
Theoretical courses	41
Practical courses	7
Academic semesters	8
Faculty members	13
Theoretical classrooms	4 (capacity 25 each)
Computer laboratory	1 (capacity 20)
Weekly time slots	10 (5 days × 2 periods)
Lecture duration	3 hours continuous

Collection involved reviewing the official curriculum documents, faculty assignment records, and institutional scheduling policies, after which all data was organised into relational tables suitable as input to a mathematical program. Table 1 presents the complete source data, mapping each course to its code, title, semester, type, enrollment, and assigned lecturer.

Table 1: Complete Course Dataset

Course Code	Course Title	Semester	Type	Enrollment	Lecturer
GH114	English Language I	**1**	T	**16**	L01
GS119	Computer Fundamentals	1	P	16	L00
GS111	Mathematics I	1	T	16	L00
GE115	Occupational Health and Safety	1	T	16	L01
GH118	Arabic Language and Islamic Studies	1	T	16	L01
EM123	Time and Effort Management	1	T	16	L02
IT211	Computer Programming	2	P	9	L04
EM111	Engineering Design I	2	P	9	L03
GE411	Report Writing	2	T	9	L01
GS121	Mathematics II	2	T	9	L00
EM122	Principles of Business Administration	2	T	9	L02
GE124	English Language II	2	T	9	L01
EM112	Principles of Accounting	2	T	9	L02
GS222	Numerical Analysis	3	T	5	L00
EM124	Small Business Management	3	T	5	L02
EM214	Engineering Design II	3	P	5	L03
EM216	Microeconomics	3	T	5	L06
EM211	Statistics I	3	T	5	L05
EM213	Technical Terminology and Comm. Skills	3	T	5	L05
EM215	Quality Management I	3	T	5	L03
EM223	Operations Research	4	T	7	L05

EM221	Statistics II	4	T	7	L05
EM212	Engineering Project Management	4	T	7	L08
EM226	Macroeconomics	4	T	7	L06
EM225	Quality Management II	4	T	7	L03
EM217	Environmental Management	4	T	7	L07
EM222	Project Scheduling with MS Project	4	P	7	L08
EM325	Supply Chain Management	5	T	8	L12
EM313	Human Resources Management	5	T	8	L07
EM312	Project Scheduling with Primavera	5	P	8	L08
EM314	Production Operations Management I	5	T	8	L07
EM318	Engineering Economy I	5	T	8	L06
EM315	Plant Layout	5	T	8	L08
EM311	Contracts and Specifications Management	5	T	8	L07
EM322	Risk Management	6	T	5	L09
EM323	Data and Information Management	6	T	5	L10
EM326	Maintenance Management	6	T	5	L09
EM321	Systems Analysis, Design, and Simulation	6	T	5	L04
EM317	Financial Management	6	T	5	L09
EM419	Production Operations Management II	6	T	5	L09
EM412	Engineering Economy II	6	T	5	L06
EM415	Marketing Management	7	T	4	L10
EM413	Seminar in Specialized Field	7	T	4	L10
EM414	Negotiation and Conflict Resolution	7	T	4	L10
EM316	Cost Accounting Management	7	T	4	L11
EM416	E-Commerce	7	P	4	L04
EM421	Strategic Management	8	T	4	L11
EM422	Engineering Planning and Design Management	8	T	4	L11

3.2. Mathematical Formulation

The problem is defined mathematically through an objective function that minimises the overflow of theoretical courses into the laboratory, subject to a set of hard constraints that rule out any logical overlap and enforce the laboratory requirements. The complete mathematical model is presented in Section 5.

3.3. Computational Implementation

The model is coded in Python using the PuLP optimization library (version 2.7.0). PuLP serves as an interface to the CBC (Coin-or Branch and Cut) Mixed Integer Linear Programming (MILP) solver, which explores the solution space through branch-and-bound algorithms.

Implementation Environment:

Component	Specification
Programming Language	Python 3.11
Optimization Library	PuLP 2.7.0
Solver	CBC (default MILP solver bundled with PuLP)
Hardware	Standard desktop (Intel Core i5, 8GB RAM)
Execution Time	0.40 seconds for the optimal baseline solution (Appendix C)

3.4. Analytical Review

Once the optimal schedule is obtained, it is extracted and analysed in detail. The review covers resource utilization rates for each classroom and overall, the balance of the temporal distribution across days and periods, adherence to every established constraint, and sensitivity to variations in the problem parameters.

3.5. Evaluation Metrics

The following Key Performance Indicators (KPIs) were established to evaluate model success quantitatively:

1. Conflict Elimination Rate: Percentage of scheduling conflicts eliminated (target: 100%).
2. Resource Utilization: Ratio of used room-slots to total available room-slots.
3. Constraint Satisfaction: Binary indicator (pass/fail) for each constraint category.
4. Temporal Balance: Standard deviation of daily course distribution.
5. Computational Efficiency: Total execution time and solver convergence metrics.

4. Theoretical Background: Binary Integer Linear Programming

4.1. Fundamentals of Linear Programming

Linear Programming (LP) is a mathematical method for finding the best attainable outcome—maximum profit or minimum cost, for instance—within a model whose requirements are expressed as linear relationships [31]. A linear programming problem in standard form is written as:

Maximize: $\boldsymbol{Z = c_1x_1 + c_2x_2 + \cdots + c_nx_n}$

Subject to:

$$\boldsymbol{a_{11}x_1 + a_{12}x_2 + \cdots + a_1nx_n \leq b_1}$$

$$\boldsymbol{a_{21}x_1 + a_{22}x_2 + \cdots + a_2nx_n \leq b_2}$$

$$\boldsymbol{a_m1x_1 + a_m2x_2 + \cdots + a_mnx_n \leq b_m}$$

$$\boldsymbol{x_1, x_2, \ldots, x_n \geq 0}$$

Here $\boldsymbol{x}_j$ denotes the decision variables, $\boldsymbol{c}_j$ the objective function coefficients, $\boldsymbol{a}_{ij}$ the constraint coefficients, and $\boldsymbol{b}_i$ the right-hand side values.

4.2. Integer and Binary Extensions

Integer Linear Programming (ILP) extends LP by requiring some or all of the variables to take integer values. When the variables are confined to {0, 1}, the formulation becomes a Binary Integer Linear Programming (BILP) problem [32]. Such a formulation is well suited to assignment and scheduling problems, where decisions are discrete by nature: assigning course c to slot t in room r is a yes-or-no decision, encoded as 1 or 0.

The general BILP formulation is:

$$\textbf{Maximize:} \quad \boldsymbol{Z = \sum_j c_j \times x_j}$$

Subject to: $\sum_j \boldsymbol{a_{ij} \times x_j \leq b_i}$, for all $\boldsymbol{i = 1, \ldots, m}$

$\boldsymbol{x_j \in \{0, 1\}}$, $\boldsymbol{j = 1, \ldots, n}$, for all $\boldsymbol{j = 1, \ldots, n}$

4.3. Solution Methods

Branch-and-bound algorithms are the standard means of solving BILP problems: they partition the solution space systematically and rely on LP relaxations to bound the optimal value [33]. The CBC solver used in this study combines four complementary mechanisms—branching, which splits the problem into subproblems by fixing binary variables; bounding, which derives upper and lower bounds from the LP relaxation; cutting planes, which append constraints that remove fractional solutions; and heuristics, which locate feasible integer solutions early so as to tighten the bounds.

4.4. Complexity and Applicability

BILP is NP-hard in general, yet modern solvers routinely handle instances with thousands of binary variables when the constraint structure is favourable [34]. The model proposed here involves 2,400 binary variables (48 courses × 10 slots × 5 rooms), a size comfortably within the reach of contemporary MILP solvers.

5. Mathematical Formulation, Implementation, and Results

5.1. Sets and Indices

The formulation is built over five finite sets that correspond one-to-one with the entities of the departmental scheduling environment. Their symbols, meanings, and cardinalities are summarized below:

Symbol	Description	Cardinality / Value
C	Set of courses	48
T	Set of time slots (5 days × 2 periods)	10
R	Set of rooms (Halls 1–4 and Computer Lab)	5
P	Set of lecturers	13
S	Set of academic semesters	8

5.2. Parameters

Each course and room is further characterized by a small set of parameters taken directly from the institutional dataset. These parameters specify the problem instance:

Parameter	Description
type(c)	Type of course c: 'T' (theoretical) or 'P' (practical)
semester(c)	Academic semester of course c: {1, 2, …, 8}
lecturer(c)	Assigned lecturer for course c: {L00, L01, …, L12}
enrollment(c)	Expected enrollment for course c: {4, 5, 7, 8, 9, 16}, varying by semester cohort (Table 1)
capacity(r)	Physical capacity of room r: {25 for halls, 20 for lab}

5.3. Decision Variables

The entire assignment structure is encoded in a single family of binary variables:

$$\boldsymbol{X(c,t,r) \in 0,1}$$

Here $\boldsymbol{X(c,t,r) = 1}$ indicates that course c is scheduled at time slot t in room r, while $\boldsymbol{X(c,t,r) = 0}$ indicates that it is not. No further decision variables are needed: every institutional requirement examined below can be expressed as a linear condition on this family alone.

5.4. Objective Function

The objective function counts how many theoretical courses are placed in the Computer Laboratory and seeks to drive that count down:

$$\textbf{Minimize:} \quad \boldsymbol{Z = \sum_{c \in C,\ \mathrm{type}(c)=T} \ \sum_{t \in T} X(c,t,Computer\_Lab)}$$

In effect, the formulation reserves the laboratory for practical courses and treats any theoretical placement there as an overflow to be tolerated only when hall capacity is exhausted — and, by design, no more than once.

It is important to be precise about what this objective actually does in the baseline configuration. Because the department offers 41 theoretical courses while the four halls supply only 40 hall-slots, the objective is bounded below by 1; because Constraint 6 caps the overflow at one, it is simultaneously bounded above by 1. Consequently, Z = 1 holds for every feasible solution, and in the baseline case the model acts primarily as a feasibility-certifying model rather than as a discriminating optimizer. The objective becomes genuinely selective only when capacity parameters change, as demonstrated in Scenarios 2 and 3, where Z = 0 is achieved.

5.5. Hard Constraints

Seven hard constraints define the feasible region. Each one translates a specific institutional rule into linear form, and together they ensure that any feasible point of the model corresponds to a workable timetable.

Constraint 1: Course Assignment (Completeness) Every course must appear in the timetable exactly once per week.

$$\boldsymbol{\sum_{t \in T} \sum_{r \in R} X(c,t,r) = 1, \quad \forall\, c \in C}$$

Since the department offers 48 courses, this family alone fixes the total number of active assignments in any feasible solution at exactly 48.

Constraint 2: Room Exclusivity A room may host at most one course at any given time.

$$\boldsymbol{\sum_{c \in C} X(c,t,r) \leq 1, \quad \forall\, t \in T,\ r \in R}$$

This is the standard anti-double-booking condition, imposed jointly over all rooms and all time slots.

Constraint 3: Lecturer Exclusivity A lecturer cannot teach two courses simultaneously. Writing $\boldsymbol{C_p}$ for the set of courses taught by lecturer p:

$$\boldsymbol{\sum_{c \in C_p} \sum_{r \in R} X(c,t,r) \leq 1, \quad \forall\, t \in T,\ p \in P}$$

This condition prevents any instructor from being assigned to more than one room during the same time slot.

Constraint 4: Student Cohort Conflict Courses belonging to the same semester cohort ($\boldsymbol{C_s}$) may not overlap temporally.

$$\boldsymbol{\sum_{c \in C_s} \sum_{r \in R} X(c,t,r) \leq 1, \quad \forall\, t \in T,\ s \in S}$$

This guarantees that students enrolled in a given semester can attend every mandatory course in their curriculum without ever facing a clash.

Constraint 5: Strict Laboratory Requirement Practical courses must be scheduled in the Computer Lab, enforced by forcing every alternative assignment to zero.

$$X(c,t,r) = 0, \quad \forall\, c \in C \text{ with } \mathrm{type}(c) = \text{'P'},\ t \in T,\ r \neq Computer\_Lab$$

These equalities are later absorbed by the solver's presolve phase as fixed variable bounds, which partly explains the compactness of the processed model reported below.

Constraint 6: Maximum Theoretical Overflow At most one theoretical course may be scheduled in the Computer Lab.

$$\sum_{c \in C,\ \mathrm{type}(c)=T} \ \sum_{t \in T} X\,(c,t,Computer\_Lab) \leq 1$$

This constraint functions as the pressure valve of the model: it acknowledges that 41 theoretical courses cannot fit into 40 available hall-slots, while preventing the laboratory from degenerating into a general-purpose overflow space.

Constraint 7: Room Capacity Limitation A course may only be assigned to a room that can physically accommodate its enrollment.

$$\mathrm{enrollment}(c) \times X(c,t,r) \leq \mathrm{capacity}(r), \quad \forall\, c \in C,\ t \in T,\ r \in R$$

Here capacity(r) = 25 for the theoretical classrooms (Halls 1–4) and capacity(r) = 20 for the Computer Lab. Course enrollments range from 4 to 16 students by semester cohort (Table 1), so every enrollment falls below both the hall capacity and the laboratory capacity, and the constraint is non-binding in the baseline scenario. It is nevertheless retained in the formulation — and actively enforced during the optimization run — so that the model remains sound under future enrollment growth.

5.6. Model Size and Complexity

The assembled model comprises 2,400 binary variables and 2,989 constraints at the model level, distributed across the constraint families as follows:

Model Component	Count / Value
Binary Variables	**2,400 (48 × 10 × 5)**
Course Assignment Constraints	48
Room Exclusivity Constraints	50 (10 × 5)
Lecturer Exclusivity Constraints	130 (10 × 13)
Cohort Conflict Constraints	80 (10 × 8)
Laboratory Constraints	280 (7 practical courses × 10 slots × 4 non-lab rooms)
Overflow Constraint	1
Capacity Constraints	2,400 (48 courses × 10 slots × 5 rooms)
Total Aggregated Constraints	2,989
Total Constraints	**2,989 (model level); 309 rows after CBC presolve**

The capacity family dominates the row count because it is written over the full (c,t,r) cube. The model is nonetheless sparse, and CBC's presolve reduces the working problem to 309 rows before the search begins.

5.7. Solver Output and Optimality Evidence

Solving the baseline instance with CBC produced the following output:

Metric	Value
Solution Status	Optimal
Objective Value	1.0 (One controlled theoretical overflow)
Binary Variables	2,400
Total Constraints	2,989 (model level); 309 processed rows after presolve
Execution Time	0.40 seconds
Nodes Explored	0 (root node solution)
Optimality Gap	0.0%

Two features of this output deserve emphasis. First, the search terminated at the root node: no branch-and-bound enumeration was required at all. Second, optimality was not merely attained but certified. The LP-relaxation bound already equals 1, so the integer solution of value 1 found by the feasibility pump is immediately confirmed optimal, with an optimality gap of 0.0%. The underlying reason is structural rather than algorithmic. Because the 41 theoretical courses exceed the 40 available hall-slots, every feasible solution satisfies $Z \geq 1$, while Constraint 6 imposes $Z \leq 1$. The optimal value $Z = 1$ is therefore determined by the problem data itself, and the run serves as a formal certificate that exactly one controlled overflow is both necessary and sufficient.

6. Analytical Results and Discussion

6.1. Comprehensive Timetable Generation

The formulated model was executed using the CBC MILP Solver via Python's PuLP interface. The optimization problem involved 2,400 binary variables and 2,989 constraints, and the solver returned an optimal, completely feasible solution — one that satisfied 100% of the defined hard constraints, allowing a single controlled theoretical overflow — within 0.40 seconds of computational time.

All 48 courses were mapped into the tightly constrained 10-slot framework. The model avoided every temporal overlap within cohorts, routing the practical courses to the Computer Lab and distributing the theoretical courses across the four halls. Exactly one theoretical course (EM317 – Financial Management, Semester 6, enrollment 5) was placed in the Computer Lab as the controlled overflow. This placement is a direct consequence of capacity arithmetic rather than a modelling choice: the four halls provide only 40 slots for 41 theoretical courses. The assignment fully respects the laboratory capacity ($5 \leq 20$), so the published timetable contains zero capacity violations.

A further observation is worth making. The identity of the overflow course is not structurally fixed — nothing in the objective distinguishes one theoretical course from another for this role — and EM317 is simply the assignment selected by the solver. What is structurally fixed, as Section 5 showed, is the count: exactly one theoretical course must occupy the laboratory in every feasible timetable. The result should therefore be read as a certificate about the institution's capacity situation, not as a preference for any particular course placement.

Note on Data Provenance and Consistency: All tables, timetables, and analytical results presented in Sections 5–8 (including Tables 2–6 and Appendix A) were generated from a single, unified model execution using the Python implementation provided in Appendix B. No manual post-processing, selective editing, or intervention was applied to the solver output. This ensures complete internal consistency across all reported results.

Table 2: Generated Conflict-Free Timetable for Semester 1 Cohort

Number	Course Code	Course Title	Type	Room	Day / Period
1	GH114	English Language I	T	Hall_1	Sunday Morning
2	GS111	Mathematics I	T	Hall_4	Monday Morning
3	GE115	Occupational Health and Safety	T	Hall_2	Monday Afternoon
4	GS119	Computer Fundamentals	P	Computer_Lab	Tuesday Morning
5	EM123	Time and Effort Management	T	Hall_4	Wednesday Morning
6	GH118	Arabic Language and Islamic Studies	T	Hall_1	Thursday Morning

Table 2 illustrates the pattern for the Semester 1 cohort: its six mandatory courses are spread across the academic week with no two courses occupying the same period. The algorithm correctly identified 'Computer Fundamentals' (GS119) as a practical course and assigned it exclusively to the Computer Lab. The complete timetable for all eight semesters, encompassing all 48 courses with zero constraint violations, is presented in Appendix A.

6.2. Resource Utilization Analysis

The scheduling environment offers five rooms across ten weekly time slots, a total of 50 room-slots. Of these, 48 were occupied. Table 3 reports the breakdown by room.

Table 3: Detailed Resource Utilization Report

Room	Capacity	Courses Assigned	Utilization Rate	Idle Slots
Hall_1	25	10	100%	0
Hall_2	25	10	100%	0
Hall_3	25	10	100%	0
Hall_4	25	10	100%	0
Computer_Lab	20	8	80%	2
Total	—	**48**	**96%**	**2**

The utilization profile is striking in its asymmetry. All four theoretical halls operate at 100% — they are saturated, with not a single idle hall-slot anywhere in the week — while the Computer Lab absorbs 8 of its 10 slots (80%). The overall figure of 96.0%

is thus not a balanced outcome but the sum of total hall saturation and partial laboratory occupancy. Two readings follow. First, the 96% value is the theoretical ceiling for this instance: because every feasible timetable assigns exactly 48 courses into 50 room-slots, this utilization follows arithmetically from feasibility itself rather than from any distinctive optimization achievement. Second, the distribution of the load is nonetheless informative. To resolve the deficit of one hall-slot while maintaining feasibility, the algorithm directed exactly one theoretical course into the laboratory's idle capacity, so the laboratory accommodated all 7 practical courses and simultaneously served as an overflow buffer. The two slots that remain idle are both laboratory slots — a nuance that matters, since Constraint 6 prevents them from absorbing further theoretical demand.

6.3. Temporal Distribution Analysis

Tables 4 and 5 examine how the 48 assignments are spread across the days of the week and across morning and afternoon periods.

Table 4: Temporal Distribution Summary

Day	Morning	Afternoon	Total
Sunday (Slot 0–1)	5	5	10
Monday (Slot 2–3)	5	5	10
Tuesday (Slot 4–5)	5	5	10
Wednesday (Slot 6–7)	4	5	9
Thursday (Slot 8–9)	4	5	9
Total	23	25	48

Note: *The table reports the number of courses scheduled per day. The Computer Lab is idle in slots 6 and 8 (Wednesday and Thursday mornings), so 48 of the 50 available room-slots are utilized.*

Table 5: Detailed Period Distribution

Period	Course Count	Percentage
Morning (Slots 0, 2, 4, 6, 8)	23	47.9%
Afternoon (Slots 1, 3, 5, 7, 9)	25	52.1%
Total	**48**	**100%**

The daily load is nearly uniform: three days carry ten courses each, and the remaining two carry nine. The deviation has a simple origin — the Computer Lab stands idle on Wednesday and Thursday mornings (slots 6 and 8) — rather than any imbalance in the placement logic. Aggregated by period (Table 5), mornings account for 23 courses (47.9%) and afternoons for 25 (52.1%), a gap of exactly two courses that mirrors the two idle morning laboratory slots; in effect, the afternoon sessions absorb the full slack of the schedule. It is worth noting that the objective function contains no term rewarding temporal balance, so this near-uniform distribution is an emergent property of the constraint structure. With the halls saturated in every slot, the only meaningful degrees of freedom left to the solver concern which laboratory slots remain empty and which course fills the single overflow position.

6.4. Constraint Verification and Feasibility

A post-processing verification pass checked the generated timetable against every constraint category. The results are summarized in Table 6.

Table 6: Constraint Verification Report

Constraint	Violations Expected	Violations Actual	Status
Room Double-Booking	**0**	**0**	**PASS**
Lecturer Overlap	0	0	PASS
Cohort Conflict	0	0	PASS
Practical Course Lab Assignment	7	7	PASS
Theoretical Overflow (Controlled)	≤1	1	PASS
Capacity Violation	0	0	PASS

Every category reports zero violations, the sole exception being the theoretical overflow, which stands at exactly one — the value explicitly permitted by Constraint 6 and, as established in Section 5, required by the capacity arithmetic. A separate feasibility check is also reassuring: the largest single-semester cohort contains 7 courses, against 10 available weekly time slots, so up to 10 courses per semester could in principle be scheduled without overlap. The algorithm navigated this constrained space without resorting to any relaxation beyond the designed overflow.

6.5. Automated Validation Summary

As a final integrity layer, an independent automated validation script was executed against the generated timetable. The script applies six checks in sequence:

1. **Course Coverage:** Confirms all 48 courses are assigned exactly once.
2. **Room Exclusivity:** Iterates over all (slot, room) pairs and verifies no duplicates exist.
3. **Lecturer Exclusivity:** Iterates over all (slot, lecturer) pairs and verifies no duplicates exist.
4. **Cohort Conflict:** Iterates over all (slot, semester) pairs and verifies no duplicates exist.
5. **Laboratory Compliance:** Verifies all 7 practical courses are in the Computer Lab.
6. **Capacity Check:** Verifies no room assignment exceeds its enrollment capacity.

All six checks returned zero violations (with the allowed theoretical overflow), corroborating the verification reported in Table 6. Because this validation operates on the extracted timetable rather than on the solver's internal state, it provides an independent layer of assurance that the published schedule — not merely the solver's own report — is conflict-free.

7. Sensitivity Analysis and Comparative Evaluation

7.1. Sensitivity Analysis

A sensitivity analysis was carried out to examine how the proposed BILP model responds when key institutional parameters are varied. One parameter was changed at a time, and the full optimization model was re-run under each configuration using the CBC solver (version 2.10.3). For every run, the actual solver output was recorded, including the solution status, the objective function value, the computational time, and the constraint violation counts.

7.1.1. Scenario 1: Baseline Configuration

The baseline reflects the institution as it currently operates: 5 rooms (4 halls + 1 lab), 10 time slots (5 days × 2 periods), and 48 courses. The solver returned an optimal solution with an objective value of 1.0, corresponding to one theoretical course overflowing into the Computer Lab, and reported zero conflicts across all categories within 0.40 seconds. At the model level, the instance comprised 2,400 binary variables and 2,989 constraints (309 rows after presolve). The solution was found at the root node without any branch-and-bound enumeration, which means the very first feasibility pump iteration produced a solution that was immediately certified optimal.

7.1.2. Scenario 2: Adding a Sixth Classroom

Re-running the model with 6 rooms (one additional theoretical hall) changed the outcome in a meaningful way: the solver reached an objective value of 0.0 in 0.55 seconds, and the theoretical overflow disappeared entirely. A single extra classroom, in other words, provides enough capacity for all 41 theoretical courses to be accommodated within the halls, leaving zero overflow into the Computer Lab. The instance grew to 2,880 variables and 3,549 constraints, and the solution was once again found at the root node.

7.1.3. Scenario 3: Adding an Eleventh Time Slot

A comparable effect followed from extending the academic week by one period (Saturday morning, for example) while keeping 5 rooms. Here, too, the overflow vanished: the objective value fell to 0.0 in 0.50 seconds. With 55 room-slots available for 48 courses, the model returned a conflict-free solution with 7 idle slots, using 2,640 variables and 3,283 constraints.

7.1.4. Scenario 4: Removing One Classroom (4 rooms)

When one theoretical hall was removed (leaving 3 halls + 1 lab = 4 rooms, 40 slots), the solver declared the problem infeasible after only 0.06 seconds. This outcome is not surprising: 40 slots cannot hold 48 courses, even before room-type and cohort constraints enter the picture.

7.1.5. Scenario 5: Removing One Time Slot (9 slots)

Reducing the schedule to 9 time slots (45 total slots with 5 rooms) also rendered the problem infeasible, again within 0.06 seconds. Since 45 room-slots fall short of the 48-course demand, infeasibility already follows from a simple capacity count. The room-type and cohort constraints only tighten matters further: the 7 practical courses consume 7 of the 9 laboratory slots, leaving at most 36 hall-slots, plus one controlled overflow, for 41 theoretical courses.

7.1.6. Scenario 6: Increasing Course Load to 53 Courses

Adding 5 theoretical courses (53 in total) while keeping the baseline infrastructure (5 rooms, 10 slots) led to infeasibility within 0.07 seconds. The current configuration has a theoretical maximum of 50 courses (50 room-slots), and the existing demand of 48 courses already leaves only 2 idle slots; five additional courses exceed the physical capacity.

7.1.7. Scenario 7: Allowing Two Theoretical Overflows

Relaxing the overflow constraint from 1 to 2, with the rest of the baseline configuration unchanged, still yielded an objective value of 1.0. The optimal solution therefore requires only one overflow even when a second is permitted, which indicates that the single-overflow constraint is binding but not overly restrictive.

7.1.8. Scenario 8: Removing Two Time Slots (8 slots)

With only 8 time slots (40 total slots with 5 rooms), the problem was immediately infeasible (0.05 seconds). This confirms that the current 10-slot configuration lies close to the absolute minimum.

Table 7: Thorough Sensitivity Analysis Results

Scenario	Configuration	Status	Objective Value	Overflow	Time (s)	Variables	Constraints
1. Baseline	**5 rooms, 10 slots, 48 courses**	**Optimal**	**1.000**	**1**	**0.40**	**2,400**	**2,989**
2. +1 Hall	6 rooms, 10 slots, 48 courses	Optimal	0.000	0	0.55	2,880	3,549
3. +1 Slot	5 rooms, 11 slots, 48 courses	Optimal	0.000	0	0.50	2,640	3,283
4. -1 Hall	4 rooms, 10 slots, 48 courses	Infeasible	–	–	0.06	1,920	2,429
5. -1 Slot	5 rooms, 9 slots, 48 courses	Infeasible	–	–	0.06	2,160	2,695
6. +5 Courses	5 rooms, 10 slots, 53 courses	Infeasible	–	–	0.07	2,650	3,244
7. 2 Overflow	5 rooms, 10 slots, 48 courses	Optimal	1.000	1	0.45	2,400	2,989
8. -2 Slots	5 rooms, 8 slots, 48 courses	Infeasible	–	–	0.05	1,920	2,401

Table 8: Capacity Thresholds for Feasibility

Parameter	Current	Infeasible	Optimal (overflow=1)	Optimal (overflow=0)
Classrooms	5	Infeasible (4 rooms)	Optimal (overflow=1)	Optimal (overflow=0 at 6 rooms)
Time Slots	10	Infeasible (9 slots)	Optimal (overflow=1)	Optimal (overflow=0 at 11 slots)
Course Load	48	Infeasible (53 courses)	Optimal (overflow=1 at 48)	Not achievable at 5 rooms × 10 slots (requires 6 rooms or 11 slots)

Taken together, these scenarios show an institution operating at a critical tipping point. The 5-room, 10-slot infrastructure is the minimum viable configuration for 48 courses, and it runs with no margin for error: removing a single classroom or a single time slot tips the problem into infeasibility. In the opposite direction, one additional classroom or one additional time slot eliminates the overflow entirely. The policy implication follows directly: the institution should either add a sixth classroom or introduce an additional teaching period if it wants a more robust scheduling configuration.

7.2. Comparative Performance Analysis

The performance characteristics of the proposed BILP model were set against alternative scheduling approaches on the basis of methodological properties documented in the literature. A direct empirical comparison on the same problem instance and hardware was not conducted for metaheuristic methods; the analysis below therefore concentrates on conceptual differences between approaches rather than on runtime benchmarks.

Table 9: Comparative Characteristics of Scheduling Methods

Criterion	BILP	Manual Scheduling	Metaheuristic
Solution Quality	Proven optimal	Approximate	Approximate
Conflict Guarantee	Yes (hard constraints)	No	Not guaranteed
Computational Time	0.40 s (measured)	Hours	Minutes to hours
Reproducibility	Yes (deterministic)	No	Partial
Optimality Proof	Yes (root node)	N/A	No
Scalability	Limited to problem size	Unlimited	High
Constraint Handling	Hard constraints only	Informal	Hard + soft

Provable optimality stands out as the decisive advantage of the BILP approach. In the baseline configuration, the solver settled the problem at the root node, with zero branch-and-bound enumeration, so the solution was proven optimal without any heuristic search. That degree of mathematical certainty is fundamentally out of reach for metaheuristic methods, which return approximate solutions and offer no guarantee of global optimality.

8. Conclusions and Recommendations

8.1. Summary of Findings

This study presented a Binary Integer Linear Programming (BILP) model for solving the University Course Timetabling Problem (UCTTP) under tightly constrained institutional scheduling conditions. The model was developed and evaluated through a real-world case study drawn from the Department of Engineering Management and Administrative Leadership at the College of Technical Sciences in Bani Walid, Libya. By folding the institution's strict requirements into a unified mathematical optimization framework, the model produced a conflict-free timetable that satisfied every mandatory scheduling constraint.

The computational results attest to the model's effectiveness. The optimal timetable contains no classroom conflicts, no instructor conflicts, and no student scheduling conflicts, with the single exception of one controlled theoretical overflow explicitly permitted by Constraint 6. Practical courses were allocated correctly to the computer laboratory, and the strictly limited educational resources were used to their theoretical limit: a room-slot utilization of 96%, a figure that follows arithmetically from the fact that all 48 courses are assigned. These findings support the view that Binary Integer Linear Programming is a dependable, workable approach for automating university course scheduling in institutions that operate under strict operational constraints.

The sensitivity analysis placed this result in a sharper light. The current institutional configuration (5 classrooms, 10 time slots) turned out to be the minimum viable capacity for the present course load of 48 courses; any reduction in resources renders the scheduling problem mathematically infeasible. This gives decision-makers quantitative evidence on which to base future resource allocation.

The comparative evaluation, for its part, showed that the BILP model outperforms manual scheduling in conflict elimination and resource utilization, while offering proven mathematical optimality of a kind that metaheuristic approaches cannot guarantee.

8.2. Practical Recommendations for the Institution

On the basis of the analytical findings, the following practical recommendations are put forward for the College of Technical Sciences in Bani Walid:

- **Immediate Implementation:** Deploy the proposed BILP model as the primary scheduling tool for the Department of Engineering Management, replacing the current manual process.
- **Resource Planning:** Consider expanding the Computer Lab capacity or adding a sixth classroom to increase scheduling flexibility and accommodate future enrollment growth.
- **Policy Review:** Reassess the current 3-hour continuous lecture format, which constrains scheduling flexibility considerably; alternative formats, such as two 1.5-hour sessions, could be explored.
- **Automation:** Integrate the model into a user-friendly web-based Decision Support System so that department administrators can generate schedules independently.
- **Data Maintenance:** Establish a centralized database for courses, instructors, and room availability to facilitate regular schedule updates.

8.3. Theoretical Contributions

Beyond the immediate case, this study adds to the university timetabling literature in four respects:

- It presents an exact optimization model designed for academic environments characterized by limited teaching periods, continuous three-hour lectures, restricted classroom availability (only 4 halls and 1 lab), and specialized laboratory requirements.
- It demonstrates that exact mathematical optimization remains computationally practical and well able to produce high-quality solutions for resource-constrained institutions.
- It provides a thorough sensitivity analysis framework for institutional capacity planning in timetabling.
- It establishes a replicable methodology for similar institutions operating under comparable constraints.

8.4. Limitations

Several limitations deserve transparent acknowledgement. The proposed model considers hard constraints only and does not, at present, incorporate soft constraints such as faculty scheduling preferences, preferred teaching times, or complex workload balancing. The evidence base is confined to a single academic department and one academic semester, so the findings should be

generalized with due attention to institutional differences. Course demands are treated as static: dynamic changes, such as last-minute course cancellations or instructor substitutions, fall outside the model. Room-capacity modeling rests on cohort-level enrollment figures (Constraint 7) rather than individual per-course registration data, and it does not capture room-attribute matching beyond capacity and room type. Finally, the comparison with metaheuristic methods is conceptual rather than empirical, because the same problem instance was not tested under identical conditions.

8.5. Future Research Directions

Building directly on these findings, eight directions for future research are recommended:

1. **Soft Constraints Integration:** Extend the proposed model by formally incorporating soft constraints and preference-based scheduling mechanisms, such as faculty preferred time slots and room preferences.
2. **Multi-Objective Optimization:** Develop a comprehensive multi-objective optimization model that simultaneously considers resource utilization, temporal balance, faculty preferences, and student convenience.
3. **Metaheuristic Comparison:** Systematically compare the proposed Binary Integer Linear Programming approach with advanced metaheuristic techniques such as Genetic Algorithms, Tabu Search, and Simulated Annealing on the same problem instance.
4. **Scalability Evaluation:** Evaluate the scalability of the proposed mathematical model using extensive datasets from multiple academic departments or entire universities.
5. **Decision Support System:** Integrate the developed optimization model into a user-friendly, web-based Decision Support System to actively facilitate automated timetable generation and real-time academic scheduling.
6. **Dynamic Scheduling:** Extend the model to handle dynamic scheduling scenarios, including real-time adjustments for cancellations, substitutions, and enrollment changes.
7. **Multi-Semester Planning:** Develop an integrated model that simultaneously schedules multiple semesters or academic years, considering prerequisite relationships and curriculum sequencing.
8. **Lecturer Availability Modeling:** Incorporate a formal availability matrix A(p, t) to capture instructor time-slot availability constraints, enabling more realistic scheduling that respects faculty schedules and contractual obligations.

8.6. Concluding Remarks

In conclusion, the proposed Binary Integer Linear Programming framework offers an effective, transparent, and scalable solution for university course timetabling. The study demonstrates that exact optimization techniques remain a powerful decision-support tool for academic planning and resource management, particularly in higher education institutions operating under severely limited resources and strict scheduling requirements. The application to the College of Technical Sciences in Bani Walid validates the practical viability of the approach and establishes a replicable methodology for similar institutions operating under comparable constraints, pending further empirical validation.

References

[1] E. Rappos, E. Thiémard, S. Robert, and J.-F. Hêche, “A mixed-integer programming approach for solving university course timetabling problems,” Journal of Scheduling, vol. 25, pp. 391–404, 2022, doi: 10.1007/s10951-021-00715-5.
[2] H. A. Taha, Operations Research: An Introduction, 10th ed. London, UK: Pearson, 2017.
[3] M. C. Chen, S. N. Sze, S. L. Goh, N. R. Sabar, and G. Kendall, “A survey of university course timetabling problem: perspectives, trends and opportunities,” IEEE Access, vol. 9, pp. 106515–106529, 2021, doi: 10.1109/ACCESS.2021.3100613.
[4] F. S. Hillier and G. J. Lieberman, Introduction to Operations Research, 11th ed. New York, NY, USA: McGraw Hill, 2021.
[5] W. L. Winston, Operations Research: Applications and Algorithms, 4th ed. Belmont, CA, USA: Duxbury Press, 2003.
[6] X. Gu, M. Krish, S. Sohail, S. Thakur, F. Sabrina, and Z. Fan, “From integer programming to machine learning: A technical review on solving university timetabling problems,” Computation, vol. 13, no. 1, article 10, 2025, doi: 10.3390/computation13010010.
[7] K. Schimmelpfeng and S. Helber, “Application of a real-world university-course timetabling model solved by integer programming,” OR Spectrum, vol. 29, pp. 783–803, 2007, doi: 10.1007/s00291-006-0074-z.
[8] G. K. Thiruvathukal, “Automating course scheduling with linear programming and the Python PuLP framework: First steps,” Loyola University Chicago, eCommons, 2025. [Online]. Available: https://ecommons.luc.edu/cs_facpubs/412/
[9] S. Abdullah, E. K. Burke, and B. McCollum, “Using a randomised iterative improvement algorithm with composite neighbourhood structures for the university course timetabling problem,” in Metaheuristics: Progress in Complex Systems Optimization, pp. 153–169, 2007.
[10] L. Di Gaspero and A. Schaerf, “Tabu search techniques for examination timetabling,” in Practice and Theory of Automated Timetabling III, pp. 104–117, 2001.
[11] B. McCollum, A. Schaerf, B. Paechter, P. McMullan, R. Lewis, A. J. Parkes, L. Di Gaspero, R. Qu, and E. K. Burke, “Setting the research agenda in automated timetabling: The second international timetabling competition,” INFORMS Journal on Computing, vol. 22, no. 1, pp. 120–130, 2010.
[12] S. Daskalaki, T. Birbas, and E. Housos, “An integer programming formulation for a case study in university timetabling,” European Journal of Operational Research, vol. 153, no. 1, pp. 117–135, 2004, doi: 10.1016/S0377-2217(03)00103-6.

[13] S. A. MirHassani, "A computational approach to enhancing course timetabling with integer programming," Applied Mathematics and Computation, vol. 175, no. 1, pp. 814–822, 2006, doi: 10.1016/j.amc.2005.07.039.
[14] R. Lewis, "A survey of metaheuristic-based techniques for university timetabling problems," OR Spectrum, vol. 30, no. 1, pp. 167–190, 2008, doi: 10.1007/s00291-007-0097-0.
[15] E. K. Burke and S. Petrovic, "Recent research directions in automated timetabling," European Journal of Operational Research, vol. 140, no. 2, pp. 266–280, 2002, doi: 10.1016/S0377-2217(02)00069-3.
[16] A. Schaerf, "A survey of automated timetabling," Artificial Intelligence Review, vol. 13, no. 2, pp. 87–127, 1999, doi: 10.1023/A:1006576209967.
[17] N. Pillay, "A review of hyper-heuristics for educational timetabling," Annals of Operations Research, vol. 239, no. 1, pp. 3–38, 2016, doi: 10.1007/s10479-014-1688-1.
[18] M. W. Carter and G. Laporte, "Recent developments in practical course timetabling," in Practice and Theory of Automated Timetabling, 1998, pp. 3–19.
[19] P. C. Chu and J. E. Beasley, "A genetic algorithm for the generalised assignment problem," Computers and Operations Research, vol. 24, no. 1, pp. 17–23, 1997, doi: 10.1016/S0305-0548(96)00032-9.
[20] M. P. Carrasco and M. J. F. Pato, "A multiobjective genetic algorithm for the class/teacher timetabling problem," in Practice and Theory of Automated Timetabling, 2001, pp. 3–17.
[21] S. Mitchell, M. O'Sullivan, and I. Dunning, "PuLP: A linear programming toolkit for Python," The University of Auckland, Auckland, New Zealand, 2011.
[22] E. K. Burke, J. Marecek, A. J. Parkes, and H. Rudova, "Penalising patterns in timetables: Novel integer programming formulations," in Operations Research Proceedings 2007, Springer, pp. 409–414, 2008.
[23] B. McCollum, "University timetabling: Bridging the gap between research and practice," in Practice and Theory of Automated Timetabling V, 2006, pp. 15–35.
[24] R. Qu, E. K. Burke, B. McCollum, L. T. G. Merlot, and S. Y. Lee, "A survey of search methodologies and automated system development for examination timetabling," Journal of Scheduling, vol. 12, no. 1, pp. 55–89, 2009, doi: 10.1007/s10951-008-0077-5.
[25] M. W. Carter, "A comprehensive course timetabling and student scheduling system at the University of Waterloo," in Practice and Theory of Automated Timetabling, 2001, pp. 64–82.
[26] D. de Werra, "The combinatorics of timetabling," European Journal of Operational Research, vol. 96, no. 3, pp. 504–513, 1997, doi: 10.1016/S0377-2217(96)00111-7.
[27] J. H. Kingston, "The KTS high school timetabling system," in Practice and Theory of Automated Timetabling VI, 2007, pp. 181–195.
[28] H. Rudova and K. Murray, "University course timetabling with soft constraints," in Practice and Theory of Automated Timetabling IV, 2003, pp. 310–328.
[29] P. Kostuch, "The university course timetabling problem with a three-phase approach," in Practice and Theory of Automated Timetabling V, 2006, pp. 109–125.
[30] G. B. Dantzig, "Linear programming," Operations Research, vol. 50, no. 1, pp. 42–47, 2002, doi: 10.1287/opre.50.1.42.17798.
[31] L. A. Wolsey, Integer Programming, 2nd ed. Hoboken, NJ, USA: Wiley, 2020.
[32] C. H. Papadimitriou and K. Steiglitz, Combinatorial Optimization: Algorithms and Complexity. Mineola, NY, USA: Dover Publications, 1998.
[33] A. Schrijver, Theory of Linear and Integer Programming. Chichester, UK: Wiley, 1998.
[34] M. W. Carter and G. Laporte, "Recent developments in practical examination timetabling," in Practice and Theory of Automated Timetabling, 1996, pp. 3–21.
[35] G. B. Dantzig and M. N. Thapa, Linear Programming 1: Introduction. New York, NY, USA: Springer, 1997.
[36] V. Chvatal, Linear Programming. New York, NY, USA: W.H. Freeman, 1983.
[37] D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning. Boston, MA, USA: Addison-Wesley, 1989.
[38] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, "Optimization by simulated annealing," Science, vol. 220, no. 4598, pp. 671–680, 1983, doi: 10.1126/science.220.4598.671.
[39] F. Glover, "Tabu search – Part I," ORSA Journal on Computing, vol. 1, no. 3, pp. 190–206, 1989.
[40] S. Abdullah, S. Ahmadi, E. K. Burke, and M. Dror, "Investigating Ahuja-Orlin's large neighbourhood search approach for examination timetabling," OR Spectrum, vol. 29, no. 2, pp. 351–372, 2007, doi: 10.1007/s00291-006-0034-7.
[41] S. M. Al-Yakoob and H. D. Sherali, "Mathematical programming models and algorithms for a class-faculty assignment problem," European Journal of Operational Research, vol. 173, no. 2, pp. 488–507, 2006, doi: 10.1016/j.ejor.2005.01.052.

Appendix A: Complete Generated Timetable for All Semesters

This appendix presents the comprehensive, conflict-free timetable generated by the BILP model for all 48 courses across the eight academic semesters. The schedule strictly adheres to the 10 available time slots and accurately assigns practical courses to the Computer Lab.

Table A1: Complete Course Timetable

Slot	Semester	Course Code	Type	Room
0	2	IT211	P	Computer_Lab
0	1	GH114	T	Hall_1
0	3	EM215	T	Hall_2
0	5	EM325	T	Hall_3
0	6	EM322	T	Hall_4
1	2	EM111	P	Computer_Lab
1	8	EM421	T	Hall_1
1	4	EM223	T	Hall_2
1	5	EM313	T	Hall_3
1	6	EM323	T	Hall_4
2	5	EM312	P	Computer_Lab
2	2	GE411	T	Hall_1
2	4	EM221	T	Hall_2
2	7	EM413	T	Hall_3
2	1	GS111	T	Hall_4
3	3	EM214	P	Computer_Lab
3	8	EM422	T	Hall_1
3	1	GE115	T	Hall_2
3	4	EM226	T	Hall_3
3	5	EM314	T	Hall_4
4	1	GS119	P	Computer_Lab
4	7	EM414	T	Hall_1
4	5	EM318	T	Hall_2
4	4	EM212	T	Hall_3
4	6	EM321	T	Hall_4
5	6	EM317	T	Computer_Lab
5	7	EM316	T	Hall_1
5	3	GS222	T	Hall_2
5	4	EM217	T	Hall_3
5	2	GE124	T	Hall_4
6	3	EM216	T	Hall_1
6	2	GS121	T	Hall_2
6	6	EM419	T	Hall_3
6	1	EM123	T	Hall_4
7	7	EM416	P	Computer_Lab
7	3	EM211	T	Hall_1
7	2	EM122	T	Hall_2
7	6	EM326	T	Hall_3
7	4	EM225	T	Hall_4
8	1	GH118	T	Hall_1
8	3	EM213	T	Hall_2
8	5	EM315	T	Hall_3
8	2	EM112	T	Hall_4
9	4	EM222	P	Computer_Lab
9	6	EM412	T	Hall_1
9	5	EM311	T	Hall_2
9	7	EM415	T	Hall_3
9	3	EM124	T	Hall_4

Note: *Slot numbering: 0=Sunday Morning, 1=Sunday Afternoon, 2=Monday Morning, 3=Monday Afternoon, 4=Tuesday Morning, 5=Tuesday Afternoon, 6=Wednesday Morning, 7=Wednesday Afternoon, 8=Thursday Morning, 9=Thursday Afternoon.*

Appendix B: Python Implementation Code

The complete Python implementation is provided below. All data dictionaries are fully populated with the actual institutional dataset, ensuring full reproducibility: running the script generates the same timetable without manual intervention.

```
"""
University Course Timetabling using Binary Integer Linear Programming (BILP)
Department of Engineering Management, College of Technical Sciences, Bani Walid
Author: Ali Mahmoud Assabri et al.
"""
from pulp import *
import pandas as pd

# ==================== DATA DEFINITION ====================
COURSES = [
    'GS111', 'GH114', 'GS119', 'GE115', 'GH118', 'EM123',
    'IT211', 'EM111', 'GE411', 'GS121', 'EM122', 'GE124', 'EM112',
    'GS222', 'EM124', 'EM214', 'EM216', 'EM211', 'EM213', 'EM215',
    'EM223', 'EM221', 'EM212', 'EM226', 'EM225', 'EM217', 'EM222',
    'EM325', 'EM313', 'EM312', 'EM314', 'EM318', 'EM315', 'EM311',
    'EM322', 'EM323', 'EM326', 'EM321', 'EM317', 'EM419', 'EM412',
    'EM415', 'EM413', 'EM414', 'EM316', 'EM416',
    'EM421', 'EM422',
]
SLOTS = list(range(10))  # 5 days x 2 periods
ROOMS = ['Hall_1', 'Hall_2', 'Hall_3', 'Hall_4', 'Computer_Lab']
SEMESTERS = list(range(1, 9))  # 8 semesters
LECTURERS = ['L00', 'L01', 'L02', 'L03', 'L04', 'L05',
             'L06', 'L07', 'L08', 'L09', 'L10', 'L11', 'L12']

# Course properties - fully populated
course_type = {
    'GS111': 'T', 'GH114': 'T', 'GS119': 'P', 'GE115': 'T', 'GH118': 'T', 'EM123': 'T',
    'IT211': 'P', 'EM111': 'P', 'GE411': 'T', 'GS121': 'T', 'EM122': 'T', 'GE124': 'T', 'EM112': 'T',
    'GS222': 'T', 'EM124': 'T', 'EM214': 'P', 'EM216': 'T', 'EM211': 'T', 'EM213': 'T', 'EM215': 'T',
    'EM223': 'T', 'EM221': 'T', 'EM212': 'T', 'EM226': 'T', 'EM225': 'T', 'EM217': 'T', 'EM222': 'P',
    'EM325': 'T', 'EM313': 'T', 'EM312': 'P', 'EM314': 'T', 'EM318': 'T', 'EM315': 'T', 'EM311': 'T',
    'EM322': 'T', 'EM323': 'T', 'EM326': 'T', 'EM321': 'T', 'EM317': 'T', 'EM419': 'T', 'EM412': 'T',
    'EM415': 'T', 'EM413': 'T', 'EM414': 'T', 'EM316': 'T', 'EM416': 'P',
    'EM421': 'T', 'EM422': 'T',
}
course_lecturer = {
    'GS111': 'L00', 'GS119': 'L00', 'GS121': 'L00', 'GS222': 'L00',
    'GH114': 'L01', 'GE115': 'L01', 'GH118': 'L01', 'GE411': 'L01', 'GE124': 'L01',
    'EM123': 'L02', 'EM122': 'L02', 'EM124': 'L02', 'EM112': 'L02',
    'EM111': 'L03', 'EM214': 'L03', 'EM215': 'L03', 'EM225': 'L03',
    'IT211': 'L04', 'EM416': 'L04', 'EM321': 'L04',
    'EM211': 'L05', 'EM221': 'L05', 'EM223': 'L05', 'EM213': 'L05',
    'EM216': 'L06', 'EM226': 'L06', 'EM318': 'L06', 'EM412': 'L06',
    'EM217': 'L07', 'EM311': 'L07', 'EM313': 'L07', 'EM314': 'L07',
    'EM222': 'L08', 'EM312': 'L08', 'EM212': 'L08', 'EM315': 'L08',
    'EM322': 'L09', 'EM317': 'L09', 'EM419': 'L09', 'EM326': 'L09',
    'EM323': 'L10', 'EM413': 'L10', 'EM414': 'L10', 'EM415': 'L10',
    'EM421': 'L11', 'EM422': 'L11', 'EM316': 'L11',
    'EM325': 'L12',
}
course_semester = {
    'GS111': 1, 'GH114': 1, 'GS119': 1, 'GE115': 1, 'GH118': 1, 'EM123': 1,
    'IT211': 2, 'EM111': 2, 'GE411': 2, 'GS121': 2, 'EM122': 2, 'GE124': 2, 'EM112': 2,
    'GS222': 3, 'EM124': 3, 'EM214': 3, 'EM216': 3, 'EM211': 3, 'EM213': 3, 'EM215': 3,
    'EM223': 4, 'EM221': 4, 'EM212': 4, 'EM226': 4, 'EM225': 4, 'EM217': 4, 'EM222': 4,
    'EM325': 5, 'EM313': 5, 'EM312': 5, 'EM314': 5, 'EM318': 5, 'EM315': 5, 'EM311': 5,
    'EM322': 6, 'EM323': 6, 'EM326': 6, 'EM321': 6, 'EM317': 6, 'EM419': 6, 'EM412': 6,
    'EM415': 7, 'EM413': 7, 'EM414': 7, 'EM316': 7, 'EM416': 7,
    'EM421': 8, 'EM422': 8,
}
```

```
sem_enrollment = {1: 16, 2: 9, 3: 5, 4: 7, 5: 8, 6: 5, 7: 4, 8: 4}  # per-semester cohort enrollment (Table 1)
course_enrollment = {c: sem_enrollment[course_semester[c]] for c in COURSES}
room_capacity = {'Hall_1': 25, 'Hall_2': 25, 'Hall_3': 25, 'Hall_4': 25, 'Computer_Lab': 20}
# Room capacity parameters

# ==================== MODEL FORMULATION ====================
prob = LpProblem("University_Timetabling_BILP", LpMinimize)

# Decision Variables: X[c,t,r] = 1 if course c is in slot t, room r
X = LpVariable.dicts("X", (COURSES, SLOTS, ROOMS), cat='Binary')

# Objective Function: Minimize theoretical overflow into Computer Lab
overflow_terms = []
for c in COURSES:
    if course_type[c] == 'T':
        for t in SLOTS:
            overflow_terms.append(X[c][t]['Computer_Lab'])
prob += lpSum(overflow_terms), "Min_Theoretical_Overflow"

# Constraint 1: Each course scheduled exactly once
for c in COURSES:
    prob += lpSum(X[c][t][r] for t in SLOTS for r in ROOMS) == 1

# Constraint 2: Room exclusivity (max 1 course per room per slot)
for t in SLOTS:
    for r in ROOMS:
        prob += lpSum(X[c][t][r] for c in COURSES) <= 1

# Constraint 3: Lecturer exclusivity
for t in SLOTS:
    for p in LECTURERS:
        courses_by_p = [c for c in COURSES if course_lecturer.get(c) == p]
        if courses_by_p:
            prob += lpSum(X[c][t][r] for c in courses_by_p for r in ROOMS) <= 1

# Constraint 4: Cohort conflict (max 1 course per semester per slot)
for t in SLOTS:
    for s in SEMESTERS:
        courses_in_s = [c for c in COURSES if course_semester.get(c) == s]
        if courses_in_s:
            prob += lpSum(X[c][t][r] for c in courses_in_s for r in ROOMS) <= 1

# Constraint 5: Practical courses -> Computer Lab only
for c in COURSES:
    if course_type[c] == 'P':
        for t in SLOTS:
            for r in ROOMS:
                if r != 'Computer_Lab':
                    prob += X[c][t][r] == 0

# Constraint 6: Max 1 theoretical overflow in lab
theoretical_in_lab = []
for c in COURSES:
    if course_type[c] == 'T':
        for t in SLOTS:
            theoretical_in_lab.append(X[c][t]['Computer_Lab'])
prob += lpSum(theoretical_in_lab) <= 1

# Constraint 7: Room capacity limitation
for c in COURSES:
    for t in SLOTS:
```

```
    for r in ROOMS:
        prob += X[c][t][r] * course_enrollment[c] <= room_capacity[r]

# ==================== SOLVE ====================
solver = PULP_CBC_CMD(msg=1, timeLimit=300)
prob.solve(solver)

# ==================== RESULTS ====================
print(f"Status: {LpStatus[prob.status]}")
print(f"Objective Value: {value(prob.objective)}")

# Extract and validate schedule
schedule = []
for c in COURSES:
    for t in SLOTS:
        for r in ROOMS:
            if value(X[c][t][r]) == 1:
                schedule.append({
                    'Course': c,
                    'Slot': t,
                    'Room': r,
                    'Type': course_type[c],
                    'Semester': course_semester[c],
                    'Lecturer': course_lecturer[c],
                    'Enrollment': course_enrollment[c],
                })
df = pd.DataFrame(schedule)
print(f"Scheduled: {len(df)} courses")

# Automated validation
print("\n=== VALIDATION ===")
# Room conflicts
room_conflicts = df.groupby(['Slot', 'Room']).size()
room_conflicts = room_conflicts[room_conflicts > 1]
print(f"Room conflicts: {len(room_conflicts)}")
# Lecturer conflicts
lect_conflicts = df.groupby(['Slot', 'Lecturer']).size()
lect_conflicts = lect_conflicts[lect_conflicts > 1]
print(f"Lecturer conflicts: {len(lect_conflicts)}")
# Cohort conflicts
cohort_conflicts = df.groupby(['Slot', 'Semester']).size()
cohort_conflicts = cohort_conflicts[cohort_conflicts > 1]
print(f"Cohort conflicts: {len(cohort_conflicts)}")
# Lab violations
lab_violations = df[(df['Type'] == 'P') & (df['Room'] != 'Computer_Lab')]
print(f"Lab violations: {len(lab_violations)}")
# Overflow
overflow = df[(df['Type'] == 'T') & (df['Room'] == 'Computer_Lab')]
print(f"Theoretical overflow: {len(overflow)}")
if len(overflow) > 0:
    for _, row in overflow.iterrows():
        print(f"  - {row['Course']} in lab at slot {row['Slot']}")
# Capacity check
cap_violations = df[df.apply(lambda row: row['Enrollment'] > room_capacity[row['Room']], axis=1)]
print(f"Capacity violations: {len(cap_violations)}")

# Save results
df.to_csv('final_timetable.csv', index=False)
print("\nSaved to final_timetable.csv")
```

Appendix C: CBC Solver Log — Baseline Configuration

The following log documents the actual solver execution for the baseline scenario (5 rooms, 10 slots, 48 courses), verifying the reported optimality and runtime.

```
Welcome to the CBC MILP Solver
Version: 2.10.3
Build Date: Dec 15 2019
command line - cbc MODEL.mps -sec 300 -solve (default strategy 1)
Problem MODEL has 2989 rows, 2400 columns and 12690 elements
Coin0008I MODEL read with 0 errors
Continuous objective value is 1 - 0.01 seconds
Cgl0002I 280 variables fixed
Cgl0003I 0 fixed, 0 tightened bounds, 20 strengthened rows, 0 substitutions
Cgl0004I processed model has 309 rows, 2120 columns (2120 integer (2120 of which binary)) and 9020 elements
Cutoff increment increased from 1e-05 to 0.9999
Cbc0038I Initial state - 82 integers unsatisfied sum - 28
Cbc0038I Pass 1: suminf. 0.00000 (0) obj. 1 iterations 321
Cbc0038I Solution found of 1
Cbc0038I Before mini branch and bound, 2024 integers at bound fixed and 0 continuous
Cbc0038I Mini branch and bound did not improve solution (0.37 seconds)
Cbc0038I After 0.37 seconds - Feasibility pump exiting with objective of 1 - took 0.02 seconds
Cbc0012I Integer solution of 1 found by feasibility pump after 0 iterations and 0 nodes (0.37 seconds)
Cbc0001I Search completed - best objective 1, took 0 iterations and 0 nodes (0.37 seconds)
Cbc0035I Maximum depth 0, 0 variables fixed on reduced cost
Result - Optimal solution found
Objective value: 1.00000000
Enumerated nodes: 0
Total iterations: 0
Time (CPU seconds): 0.36
Time (Wallclock seconds): 0.40
```

Interpretation: The model contained 2,989 constraints at the model level; CBC presolve converted the 280 laboratory-exclusion equalities into variable bound fixings (280 variables fixed) and tightened the remaining rows, processing 309 rows and 2,120 columns. The continuous LP-relaxation optimum already equals 1, so the integer solution of value 1 found by the feasibility pump is certified optimal immediately at the root node (0 branch-and-bound nodes, optimality gap 0.0%). The objective value of 1.0 corresponds to exactly one theoretical course (EM317 – Financial Management, enrollment $5 \leq 20$) assigned to the Computer Lab, exactly as reported in the main text. Total runtime: 0.40 seconds (wallclock).