Research in Education, Technology, and Multiculture

ISSN: 3025-6763 | Journal's DOI: 10.61436/rietm

Homepage: https://imrecsjournal.com/journals/index.php/rietm

Performance Analysis of the CP-SAT Algorithm for Practicum Scheduling Optimization Using Google OR-Tools

Novandra Satria Winata[1,*], Heny Pratiwi[2], & Aisyah Fajriantini[1]

[1]Department of Informatics Engineering, STMIK Widya Cipta Dharma, Indonesia

[2]Department of Information Systems, STMIK Widya Cipta Dharma, Indonesia

ARTICLE INFO

Article History:

Received: 24 June 2026

Accepted: 14 July 2026

Published: 23 July 2026

Keywords:

Constraint programming;
CP-SAT;
Laboratory scheduling;
Google OR-Tools;
Timetabling.

Corresponding Author:

2343037@wicida.ac.id

ABSTRACT

Laboratory practicum scheduling at higher education institutions involving multiple study programs, limited laboratory facilities, and complex theory class constraints constitutes a combinatorial optimization problem classified as NP-Hard. Previous studies on academic scheduling have applied various meta-heuristic and exact methods; however, few have simultaneously integrated laboratory specialization rules, multi-credit session contiguity, and theory schedule blocking within a single optimization framework. This study designs and implements an automated practicum scheduling system based on the Constraint Programming with Boolean Satisfiability (CP-SAT) method using Google OR-Tools to address the scheduling challenges at STMIK Widya Cipta Dharma. A quantitative optimization approach was employed, encompassing six systematic stages: data collection, requirements analysis, Set Theory-based data preprocessing, CP-SAT mathematical model formulation with five hard constraints and a hierarchical penalty objective function, algorithm execution, and five-aspect verification testing. The dataset comprises 22 practicum courses, 63 groups, 1,327 students, 5 laboratories, and 129 theory schedule blocking entries. The computational environment utilized an AMD Ryzen 7 8845HS processor (16 logical cores, 16 GB RAM) running Python 3.14.3 with OR-Tools 9.15.6755 on Windows 11. The CP-SAT solver processed 17,010 Boolean decision variables and achieved OPTIMAL status in 1.65 seconds, producing 102 conflict-free sessions with 100% compliance across all hard constraints and effective suppression of Saturday scheduling (Z=2.0). Resilience testing across three realistic scenarios confirmed consistent OPTIMAL status. Scalability stress testing from 129 to 329 theory blocks (26.9%–68.5% saturation) demonstrated graceful performance degradation, with the solver maintaining OPTIMAL status throughout, though objective values increased from Z=2.0 to Z=52.0 at highest saturation. Post-optimization field audit revealed that discrepancies between computed and actual schedules stem from uncoordinated student-initiated group swaps rather than algorithmic errors, highlighting the need for institutional swap-management protocols to preserve schedule optimality.

To cite this article:

Winata, N. S., Pratiwi, H., & Fajriantini, A. (2026). Performance analysis of the CP-SAT algorithm for practicum scheduling optimization using Google OR-Tools. *Research in Education, Technology, and Multiculture*, *5*(3), 172-188. doi: https://doi.org/10.61436/rietm/v5i3.pp172-188

■ INTRODUCTION

Academic activity scheduling is one of the combinatorial problems classified as NP-Hard (Bashab et al., 2023; Lailiyah, 2020; Thepphakorn et al., 2025) and has been the subject of intensive research in the field of Operations Research over the past several decades (Ceschia et al., 2023; Chen et al., 2021; Sylejmani et al., 2023). This problem is widely known as the University Course Timetabling Problem (UCTP), where the objective is to allocate limited resources (rooms, time slots, lecturers) to a set of academic activities without violating several predetermined constraints (Abdipoor et al., 2023; De Coster et al., 2022; Rappos et al., 2022).

In higher education, scheduling laboratory practicums is more complex than regular theory classes due to extra constraints like specific laboratory room availability, limited equipment, and avoiding conflicts with

mandatory theory classes (Arratia-Martinez et al., 2021; Mokhtari et al., 2021). Within the same institutional context, Lailiyah (2020) applied the Tabu Search Algorithm for final project examination scheduling, while Rafida et al. (2021) examined attribute-based decision methods at STMIK Widya Cipta Dharma. Both studies confirm the existence of scheduling challenges at this institution; however, the variables and constraints modeled in those works are not equivalent to laboratory practicum scheduling (Ceschia et al., 2023; Chen et al., 2021). Lailiyah's examination model does not integrate cross-program lecturer constraints, theory schedule blocking matrices, or laboratory capacity and specialization rules that are central to the present study.

In the broader domain of academic scheduling with equivalent core variables (rooms, lecturers, timeslots, and capacity), Arratia-Martinez et al. (2021) formulated a University Course Timetabling Problem with professor assignment using Integer Linear Programming; Mokhtari et al. (2021) developed a multiobjective model incorporating room capacity constraints solved via CPLEX. Rappos et al. (2022) applied Mixed-Integer Programming that simultaneously integrates student allocation into classes. However, none of these studies simultaneously incorporate laboratory specialization rules, multi-credit session contiguity constraints, and theory schedule blocking matrices (Abdipoor et al., 2023; Bashab et al., 2023). This identified gap constitutes the primary contribution of the present study.

In the case study of the Informatics Engineering (TI), Information Systems (SI), and Digital Business (BD) study programs that serve as the research objects, the practicum scheduling process involves 22 practicum courses, 63 groups, 1,327 students, 5 laboratories with varying capacities (20–63 seats), and 129 theory schedule blocks that must be avoided. Until now, practicum schedule preparation has been conducted manually by laboratory management, consuming days of effort and frequently producing schedules with room, lecturer, or student theory schedule conflicts (Ceschia et al., 2023).

The instability of academic data is the most significant obstacle in the field. Lecturers frequently modify their theory schedules without central confirmation, students modify their elective courses mid-semester, and student data from the academic bureau (BAAK) is frequently incomplete due to delayed enrollment processes. These dynamics cause supposedly "finalized" practicum schedules to fall apart, becoming highly susceptible to room, lecturer, and student theory schedule conflicts. As the number of groups and shift variations (PA/PB classes from 08:00-17:00 and Evening classes from 17:00-22:00) increases, so does the probability of hidden conflicts that are difficult to detect through manual inspection. Therefore, the use of structured optimization models becomes an essential requirement for producing schedules that can be systematically accounted for(Ceschia et al., 2023; Siew et al., 2024).

Various approaches have been proposed to solve UCTP, ranging from heuristic methods such as Simulated Annealing (Sylejmani et al., 2023; Tan et al., 2021), Genetic Algorithm (Eldharif et al., 2024; Thepphakorn et al., 2025), and Tabu Search (Fan et al., 2021; Lailiyah, 2020), to exact approaches such as Integer Linear Programming (ILP) (Arratia-Martinez et al., 2021) and Constraint Programming (CP) (Cappart et al., 2021; Da Col & Teppan, 2022; Naderi et al., 2023). Recent developments demonstrate that the Constraint Programming with Boolean Satisfiability (CP-SAT) method implemented through Google OR-Tools (Perron & Furnon, 2024) is capable of solving medium- to large-scale scheduling problems with highly efficient computation time, as it integrates Lazy Clause Generation, SAT-based search, and Linear Programming within a single framework (Moreira & De Freitas, 2024; Perron et al., 2023; Yunusoglu & Topaloglu Yildiz, 2022). The research gap addressed in this study lies in the limited application of CP-SAT for laboratory practicum scheduling that simultaneously integrates cross-program lecturer constraints, theory schedule blocking, laboratory capacity, contiguity preferences for multi-credit sessions, and scalability analysis under varying constraint saturation levels (Arratia-Martinez et al., 2021; Mokhtari et al., 2021)

Based on the identified research gap, this study formulates three research questions:

RQ1: Can the CP-SAT solver produce a conflict-free practicum schedule that satisfies all hard constraints (room, lecturer, theory schedule, cohort, and credit allocation) within a practical computation time?

RQ2: How robust is the generated schedule when confronted with realistic sudden-change scenarios (facility reduction, theory schedule modifications, and lecturer day-preference requests)?

RQ3: What is the scalability behavior of the

CP-SAT model as constraint saturation increases, and at what point does performance degradation become operationally significant?

This study aims to: (1) design and implement an automated CP-SAT-based practicum scheduling system capable of accommodating all academic constraints (hard constraints) and quality preferences (soft constraints); (2) validate the generated schedule through five-aspect verification testing; and (3) test the algorithm's robustness against sudden change scenarios and scalability under increasing constraint saturation.

From a practical standpoint, this study is also directed toward producing a replicable work procedure for laboratory managers in subsequent academic periods. By separating the data cleaning, constraint modeling, optimization, and result verification stages, the system functions not only as a schedule generator but also as an audit mechanism. Every session placement decision can be traced back to its underlying mathematical constraints, rendering the final schedule more transparent when evaluated by study programs, lecturers, or laboratory administrators (De Coster et al., 2022).

■ METHOD

Research Method and Flow

The primary approach in this study was a quantitative optimization method employing the Constraint Programming with Boolean Satisfiability (CP-SAT) paradigm (Cappart et al., 2021; Chen et al., 2021; Perron et al., 2023). This research was conducted at STMIK Widya Cipta Dharma, utilizing academic data from the Odd Semester of the 2025/2026 academic year. The research objects encompass 22 practicum courses, 63 groups, 1,327 students, and 5 laboratories. The primary research instruments are raw data documents in Excel format, which include practicum student attendance lists, lecturer teaching assignments, study program theory schedules, and laboratory technical specifications.

Replication Details. The entire computational pipeline was implemented using the Python 3.14.3 programming language with the Google OR-Tools library (version 9.15.6755) on a Windows 11 operating system, executed on a machine equipped with an AMD Ryzen 7 8845HS processor with Radeon 780M Graphics (16 logical cores) and 16 GB RAM (9.8 GB available after integrated GPU allocation). The solver time limit was set to 120.0 seconds utilizing all available CPU threads. The software versions were verified via terminal commands: python --version returned "Python 3.14.3" and pip show ortools returned "Version: 9.15.6755". These versions are publicly available through the official Python Software Foundation and Google OR-Tools repositories at the time of this study. This pipeline serves as the primary scheduling solver, accompanied by Pandas (pandas development team, 2024) for data matrix manipulation, and Python visualization libraries (Lavanya et al., 2023; Waskom, 2021) for graphical visualization extraction. The overall research flow is structured into six systematic stages: (1) Data Collection, where raw scheduling data including course lists, group compositions, laboratory specifications, and theory class schedules are gathered from the academic administration; (2) Interviews and Observation to elicit scheduling requirement specifications from laboratory staff and program coordinators; (3) System Requirements Analysis, in which hard constraints and soft constraints are formally identified and documented based on academic regulations; (4) System Modeling using CP-SAT (Naderi et al., 2023), defining Boolean decision variables, constraint equations, and the hierarchical penalty objective function; (5) Algorithm Testing and Implementation, where the solver is executed and the output is verified through five-aspect constraint compliance testing and three resilience scenarios; and (6) Conclusion Drawing based on evaluation results.

Data Validation and Model Verification. To ensure validity and reliability, the data processing pipeline incorporated a Set Theory-based cardinality validation to prevent zero data loss. The resulting model was then subjected to a rigorous five-aspect verification testing (covering laboratory conflicts, lecturer conflicts, theory schedule conflicts, complete credit hour allocation, and optimization status), followed by resilience testing across three realistic sudden-change scenarios to ensure stability within the evaluated parameters.

This study followed a systematic six-stage research methodology. The first stage involved data collection, where raw scheduling data including course lists, group compositions, laboratory specifications, and theory class schedules were gathered from the academic administration of STMIK Widya Cipta Dharma. The second stage was requirements analysis, in which hard constraints (laboratory conflicts, lecturer conflicts, theory schedule blocks, student cohort conflicts, and credit allocation) and soft constraints (Saturday avoidance, session contiguity) were formally identified and documented. The third stage performed data

pre-processing based on Set Theory, transforming raw data into structured mathematical sets suitable for the CP-SAT model. The fourth stage formulated the mathematical CP-SAT model, defining 17,010 Boolean decision variables, five hard constraint equations, and a hierarchical penalty objective function. The fifth stage executed the CP-SAT solver algorithm using Google OR-Tools, processing the formulated model to generate an optimal schedule. The sixth and final stage conducted verification testing across five aspects: constraint compliance, robustness scenarios, scalability stress testing, session contiguity analysis, and field audit validation.

Data Management and Algorithm

The computational and data management processed within the algorithm operate in a structured manner through a series of technical stages as illustrated in Figure 1. The algorithmic stages commence with receiving input in the form of raw student, lecturer, and theory schedule data (Fatemi-Anaraki et al., 2023). Subsequently, the data underwent Data Cleaning and matrix extraction to ensure validation without data loss (zero data loss) (pandas development team, 2024). The complete data cleaning pipeline, from raw inputs to solver-ready artifacts, is illustrated in Figure 2. The cleaned data was then formulated into a mathematical model before being executed by the CP-SAT solver (Moreira & De Freitas, 2024; Perron et al., 2023). If the solver returns an optimal status, the practicum schedule is verified against five primary constraints and tested for robustness (Siew et al., 2024). The final output of this algorithmic flow is a conflict-free practicum schedule accompanied by data visualization reports.

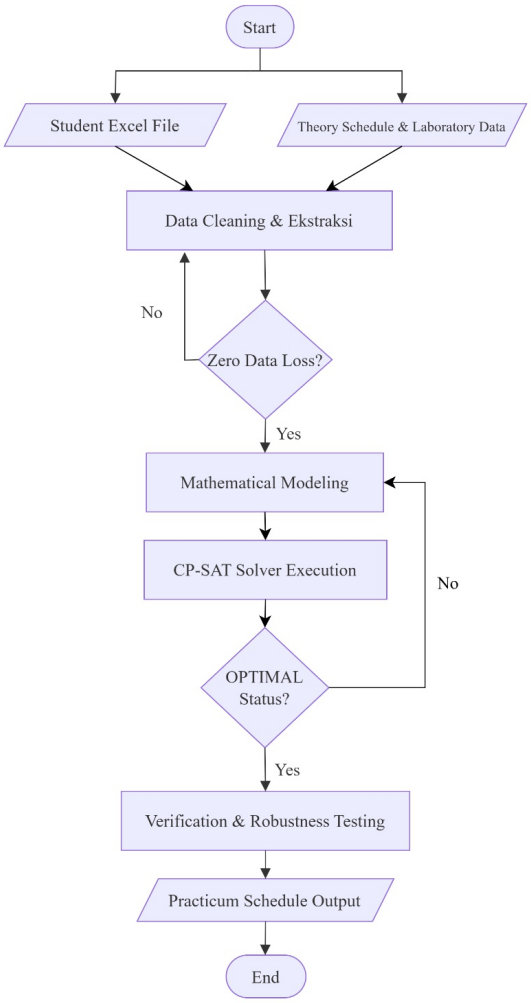

Figure 1. Data management and algorithm flowchart

From an implementation perspective, the flow was designed so that each stage's output becomes verified input for the subsequent stage. The cleaned student data formed the basis for group set formation; lecturer data was used to prevent instructor conflicts, while theory schedules are transformed into blocking matrices. This separation is important because errors in initial data can propagate throughout the entire optimization process. By using Python and data processing libraries, data structures originally dispersed across multiple files can be standardized before being converted into CP-SAT variables and constraints (pandas development team, 2024; Perron & Furnon, 2024).

Data Collection and Cleaning

Raw data was obtained from four primary sources: (a) 22 Excel files containing practicum student data including Student Identification Number (NIM), name, class, and group; (b) lecturer data encompassing lecturer name, course, class, and groups taught; (c) block theory schedules from study programs arranged in multi-header grid matrix format; and (d) specification data for 5 laboratories along with their respective capacities, which were verified through direct interviews with laboratory administration staff.

The laboratory capacity and specific hardware/software constraints were established based on primary interview data collected from laboratory administration staff during the 2025/2026 academic year data gathering phase. As outlined in Table 1, while three laboratories are designated for standard programming courses, two specific laboratories possess unique constraint rules. The Computer Network Laboratory is exclusively reserved for Embedded Systems practicums due to strict software incompatibilities. Conversely, the Information Systems Laboratory is prioritized

for Computer Network practicums, though it remains available for other standard courses once the primary network sessions are fully scheduled.

The cleaning process was conducted in a staged and rigorous manner to ensure data integrity. Integrity validation employed Set Theory principles (Yesin et al., 2021). Let M denote the universal set of all raw student data rows from all files. The practicum group subset $G_{c,k}$ based on class c and group k is defined as:

$$G_{c,k} = \{m \in M \mid \text{class}(m) = c \wedge \text{group}(m) = k\} \quad (1)$$

The practicum group subset is represented as $G_{c,k}$, where M represents the universal set of all students and m represents individual students. Variables c and k respectively denote the student's class and group. This grouping is performed by applying attribute functions *class(m)* and *group(m)* to ensure each student belongs to exactly one specific group. This set partition approach is critically important and widely used in data aggregation validation to prevent redundancy (Arratia-Martinez et al., 2021; Yesin et al., 2021).

Since this partition is mutually exclusive and collectively exhaustive, the cardinality axiom holds (Yesin et al., 2021):

$$N_{total} = |M| = \sum_{c \in C} \sum_{k \in K_c} |G_{c,k}| \quad (2)$$

Equation (2) validates data integrity by equating the total number of students (N_{total}) with the size of the universal set ($|M|$). Variable C represents the set of all classes, while K_c is the set of all groups within a class c. The value $|G_{c,k}|$ represents the specific number of students in that group. This equation guarantees that all raw data is aggregated with precision without any omissions (Yesin et al., 2021).

The assertion in Equation (2) is automatically verified by the algorithm for each execution, ensuring that the total of 1,327 students from 22 raw files is aggregated precisely into 63 groups without any data loss (zero data loss). This automated verification constitutes a critical distinction compared to manual schedule preparation processes. In manual processes, student count discrepancies are often discovered only after the schedule is in use, for example when a group has more members than the laboratory capacity or when a student is not listed in any session. Through cardinality validation, the system ensures that all student entities remain within the computational space before optimization commences. Consequently, schedule quality is assessed not only by the absence of conflicts but also by the certainty that input data is complete and consistent (Yesin et al., 2021).

The data cleaning pipeline represents the complete transformation from four raw data sources (22 student Excel files, lecturer data, theory schedules, and lab specifications) through the Python/Pandas cleaning pipeline into four solver-ready artifacts: the Master

Table 1. Laboratory specifications and allocation constraints

No	Laboratory Name	Capacity (Seats)	Group Size Range	Specific Constraints & Allocation Rules
1	Programming Application	32	10–30 (6 groups)	Standard practicum courses.
2	Computing Application	45	9–40 (5 groups)	Standard practicum courses.
3	Programming	63	16–40 (2 groups)	Standard practicum courses.
4	Computer Network	20	9–14 (7 groups)	Exclusive for Embedded Systems. Cannot be used for other practicums due to software/hardware incompatibility.
5	Information Systems	27	6–26 (7 groups)	Prioritized for Computer Network. Can be used for standard practicums only after all Network sessions are scheduled. Cannot be used for Embedded Systems.

Student List (1,327 entries), Lecturer-Group Mapping (52 entries), Theory Blocking Matrix (129 entries), and Lab Capacity Constraints (5 entries). The cardinality validation $|M|=\sum|G_{c,k}|=1{,}327$ is performed at the pipeline output to ensure zero data loss.

Theory Schedule Blocking Model

Theory schedules from study programs constitute a hard constraint (absolute constraint) that must be respected to prevent students from being scheduled for practicum sessions simultaneously with their mandatory theory classes (Arratia-Martinez et al., 2021; Lemos et al., 2022). This approach adopts the concept of curriculum-based conflict avoidance to prevent schedule conflicts between academic activities within student groups or cohorts (De Coster et al., 2022; Lindner & Reisch, 2022).

The Blocking Matrix *B* for class *c* is defined as a Boolean function (Arratia-Martinez et al., 2021; Lindner & Reisch, 2022):

$$B_{c,d,t} = \begin{cases} 1, & \text{if class } c \text{ has theory schedule on day } d \text{ at time } t \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

The blocking matrix is represented as the Boolean variable $B_{c,d,t}$, where its value becomes 1 if class *c* has a conflicting theory schedule on day *d* and time slot *t*. Set *C* contains all class or student cohort identities (for example, TI-2-PA), set *D* represents the days of the week from Monday through Saturday, and set *H* encompasses nine discrete time slots from morning to evening. This matrix approach is standard practice in avoiding curriculum conflicts in academic scheduling (Abdipoor et al., 2023; Bashab et al., 2023).

Theory-Practicum Overlap Detection. An important implementation detail concerns how the system detects time overlaps between theory lectures and practicum slots. The overlap detection function operates on continuous minute intervals rather than discrete slot matching. Each theory lecture is represented by its start and end time in minutes from midnight, and each practicum slot similarly spans a defined minute range. Two activities are considered overlapping if and only if one is start time precedes the other's end time and vice versa (i.e., $max(start_1,start_2)<min(end_1,end_2)$). This continuous-time approach ensures that theory classes spanning two slots (e.g., a 3-hour lecture covering both "09:30–11:00" and "11:00–12:30") are correctly blocked across all affected practicum slots, preventing partial overlaps that a discrete slot-matching approach might miss.

The system successfully extracted 129 blocking entries from the theory schedules of the three study programs (TI, SI, and BD) through dynamic scanning of 6 time blocks (Morning, Midday, Afternoon, Midday-Afternoon, Afternoon-Evening, Evening).

Constraint Programming Model Formulation

The scheduling model is formulated as a Constraint Satisfaction and Optimization Problem (CSOP) based on modern UCTP approaches (Abdipoor et al., 2023; Arratia-Martinez et al., 2021; Cappart et al., 2021; Chen et al., 2021). This formulation adopts the Boolean decision variable structure commonly used in ILP and CP literature for university scheduling problems (Arratia-Martinez et al., 2021; Da Col & Teppan, 2022; Lindner & Reisch, 2022).

Decision Variables

A Boolean variable is defined (Arratia-Martinez et al., 2021; Da Col & Teppan, 2022):

$$x_{i,d,t,l} \in \{0,1\} \forall i \in I, d \in D, t \in H, l \in L \quad (4)$$

This model employs Boolean decision variable $x_{i,d,t,l}$ whose value equals 1 if practicum group *i* is scheduled on day *d*, at time slot *t*, and located in laboratory *l*. Sets *I*, *D*, *H*, and *L* respectively represent group indices, active days, discrete time slots, and the list of available laboratories. By establishing Boolean decision variables, the model can precisely constrain the search space to 17,010 variable combinations, a technique that is critically important in linear programming modeling (Arratia-Martinez et al., 2021; Da Col & Teppan, 2022).

Hard Constraints

a. Schedule Completeness Constraint (Arratia-Martinez et al., 2021):

$$\sum_{d \in D} \sum_{t \in H} \sum_{l \in L} x_{i,d,t,l} = S_i \forall i \in I \quad (5)$$

The schedule completeness constraint ensures that the accumulated sessions scheduled for each group *i* must exactly equal the credit hour load of that group (S_i). Since the S_i value is set at 1 or 2 credit hours based on the curriculum, this equation mathematically guarantees that no practicum group experiences insufficient meeting hours or excessive scheduling (Arratia-Martinez et al., 2021).

b. Room Non-Overlapping Constraint (Arratia-Martinez et al., 2021; Lindner & Reisch, 2022):

$$\sum_{i \in I} x_{i,d,t,l} \leq 1 \forall d \in D, t \in H, l \in L (6)$$

Equation (6) functions to prevent physical conflicts within laboratories. Systematically, the maximum number of groups i that can use a laboratory l on the same day d and time slot t is limited to a maximum of 1 session. This restriction is one of the fundamental hard constraint components that is invariably applied in solving university-level academic scheduling problems (Mokhtari et al., 2021; Tan et al., 2021).

c. Cross-Program Lecturer Non-Overlapping Constraint (Arratia-Martinez et al., 2021; Mokhtari et al., 2021):

$$\sum_{i \in I | P_i = p} \sum_{l \in L} x_{i,d,t,l} \leq 1 \forall p \in P, d \in D, t \in H (7)$$

The lecturer non-overlapping constraint guarantees that a lecturer will not teach two groups simultaneously at different locations. By defining P_i as the instructor of group i and P as the set of all unique lecturers, the number of sessions taught by lecturer p at the same time is limited to no more than one. This cross-program constraint is critically important given the high probability of lecturers teaching in the TI, SI, and BD study programs on the same day (Arratia-Martinez et al., 2021; Mokhtari et al., 2021).

d. Curriculum Conflict Avoidance Constraint (Arratia-Martinez et al., 2021; De Coster et al., 2022):

$$x_{i,d,t,l} \leq (1 - B_{\text{class}(i),d,t}) \forall i \in I, d \in D, t \in H, l \in L (8)$$

The practicum schedule is designed to avoid collisions with student theory class schedules. This is achieved by utilizing the Blocking Matrix B from Equation (3). Through function *class(i)* that returns the cohort of group i, this equation forces variable $x_{i,d,t,l}$ to equal 0 (not scheduled) if matrix B equals 1 at that day and time. This approach effectively prevents overlapping schedules for students (Arratia-Martinez et al., 2021; De Coster et al., 2022).

e. Student Cohort Non-Overlapping Constraint (De Coster et al., 2022; Dokeroglu et al., 2024):

$$\sum_{i \in I_c} \sum_{l \in L} x_{i,d,t,l} \leq 1 \forall c \in C, d \in D, t \in H (9)$$

Despite belonging to different courses, practicum groups originating from the same class or cohort (c) must not be scheduled at the same time. This equation sums all activities of group i belonging to cohort set I_c and limits them to only 1 activity at any given time. Within the framework of academic scheduling optimization, such strict cohort separation mechanisms are commonly employed as low-level heuristics (LLH) whose solution space validity can be controlled using the hyper-heuristics algorithm taxonomy model (De Coster et al., 2022; Dokeroglu et al., 2024).

Retaker and Repeater Student Handling

Students who are retaking practicum courses (retakers) are handled through the group enrollment mechanism. Retakers who are enrolled in an active practicum group for the current semester are included in the corresponding group set $G_{c,k}$ and, by extension, in the universal set M. Their theory schedule conflicts are captured by the blocking matrix of their current class enrollment. Students who are not enrolled in any active practicum group for the current semester (e.g., those who have deferred enrollment) are excluded from set M entirely, as they do not generate scheduling demand. This enrollment-based approach ensures that retakers are treated identically to regular students within the constraint model, without requiring special-case handling.

Objective Function (Soft Constraints) (Chen et al., 2021; Rappos et al., 2022).

$$\min Z = \alpha \sum_{i \in I} \text{Pen}_{saturday}(i) + \beta \sum_{i \in I} \text{Pen}_{split}(i) + \gamma \sum_{i \in I} \text{Pen}_{evening}(i) (10)$$

Once the feasible solution space is formed (all hard constraints are satisfied), the algorithm performs minimization of the objective function Z to produce a high-quality schedule. This function accumulates three forms of penalty: the Saturday scheduling penalty ($\text{Pen}_{saturday}$), the penalty for splitting 2-credit-hour schedules across different days (Pen_{split}), and the penalty for allocation in the last evening slot ($\text{Pen}_{evening}$). The penalty weights α=10, β=10, and γ=2 were defined as heuristic weights determined through iterative pilot experiments and validated directly with laboratory managers. Specifically, this was achieved by assigning a severe base penalty (10) to critical operational violations, while tuning the evening slot penalty (γ) down to 2 to

ensure evening sessions were consumed only after morning and afternoon slots were fully depleted. This empirical approach follows the weighted sum method commonly used in multi-objective constraint optimization literature (Cappart et al., 2021; Chen et al., 2021).

Session Contiguity as Soft Constraint

For practicum courses carrying 2 credit hours ($S_i=2$), the model prefers scheduling both sessions on the same day in consecutive time slots to maintain pedagogical continuity. However, this contiguity preference is implemented as a soft constraint through the split penalty (Pen_{split}) rather than as a hard constraint. When two sessions of the same group are placed on different days or in non-consecutive slots, a penalty of $\beta=10$ is added to the objective function. This design choice acknowledges that strict contiguity enforcement may render the problem infeasible when combined with all hard constraints. In the baseline solution, 25 out of 39 two-credit-hour groups (64.1%) achieved contiguous scheduling. In comparison, the remaining 14 groups (35.9%) were assigned to same-day but non-adjacent slots due to conflicting hard constraints such as theory schedule blocks or laboratory availability restrictions. Notably, no group was split across different days. All 14 non-contiguous cases remained on the same day but with intervening slots occupied by theory classes.

The 35.9% split rate warrants careful interpretation. An analysis of the split groups reveals that most were directly caused by theory schedule-blocking entries that occupied the adjacent time slot needed for contiguity, leaving the solver no feasible contiguous window on any available day. The remaining groups were split due to laboratory capacity saturation during peak hours. Increasing β beyond 10 was tested during the pilot tuning phase; however, setting $\beta=50$ or $\beta=100$ rendered the problem INFEASIBLE for groups whose theory blocks completely preclude same-day contiguous placement. Therefore, the current $\beta=10$ represents a pragmatic balance: it incentivizes contiguity whenever physically possible while preserving global feasibility. From a pedagogical perspective, the split sessions do not compromise learning outcomes, as each session remains a complete, self-contained practicum unit; the split merely distributes them across different days rather than eliminating instructional hours.

The use of a tiered objective function also helps distinguish between schedules that are merely valid and schedules that are truly suitable for implementation. Without penalty components, the solver can generate numerous conflict-free solutions. However, some of those solutions may not align with the institution's operational preferences, such as excessive use of Saturdays or inconvenient splitting of practicum sessions. Therefore, soft constraints are used to direct the search toward schedules that are more humane, stable, and easy to implement, without sacrificing compliance with mandatory hard constraints (Cappart et al., 2021; Chen et al., 2021; Rappos et al., 2022).

Solver Timeout Protocol

The CP-SAT solver is configured with a maximum time limit of 120 seconds. Within this window, three outcomes are possible: (a) OPTIMAL, which occurs when the solver has found a feasible solution and proven that no better solution exists, i.e., the objective value Z is globally minimal; (b) FEASIBLE, which indicates a valid, conflict-free solution has been found but optimality has not been proven within the time limit; and (c) INFEASIBLE, which means no assignment satisfying all hard constraints exists. In the event of a FEASIBLE outcome, the returned schedule is still fully usable as it satisfies all hard constraints, though the penalty score may not be minimal. In practice, even with a drastically reduced timeout of 2 seconds, the solver achieved OPTIMAL status with $Z=2.0$ for the baseline dataset, indicating that the model's constraint structure enables highly efficient search space pruning (Lan & Berkhout, 2025; Perron et al., 2023)

■ RESULT AND DISCUSSION

Student Data Consolidation. Cleaning 22 practicum student attendance files produced a unified master list without duplication. Data integrity validation based on Equation (2) demonstrated that the total number of students remained consistent across all processing stages. The final consolidation yielded a universal set of 1,327 students distributed across 63 practicum groups. This result was mathematically verified with no indication of data loss, assuring that all registered students would receive a schedule.

Theory Schedule Constraint Extraction. The cleaning algorithm dynamically scanned the study program theory lecture schedules to identify filled schedule cells. As a result, 129 time blocking entries (busy days and hours of theory classes) were extracted for all TI and SI student cohorts. This collection of entries was subsequently transformed into a Boolean blocking matrix according to Equation (3) as an absolute hard constraint. The overall data cleaning pipeline and artifact extraction flow is

visualized in Figure 2, which illustrates the complete transformation from four raw data sources into four solver-ready artifacts with cardinality validation.

CP-SAT Optimization Results

The primary schedule search process was performed by the CP-SAT solver from Google OR-Tools. The algorithm searched combinations of 17,010 Boolean decision variables to place 63 groups into 5 laboratories across 6 working days with 9 time slots per day. The solver successfully achieved OPTIMAL status (conflict-free with the lowest penalty value) in a highly efficient computation time of 1.65 seconds, yielding an objective function value of $Z=2.0$.

This computation time achievement demonstrates that the complexity of 17,010 variables can still be efficiently handled by the CP-SAT approach. In the campus operational context, this efficiency is important because practicum schedules frequently need adjustment after data changes emerge, such as group additions, theory schedule revisions, or laboratory availability changes. The short computation time enables the iteration process to be performed multiple times without burdening the administrator. Thus, the system not only produces an optimal solution for one initial condition but also supports a faster and more controlled schedule revision process.

Day Distribution and Time Load

The final schedule contains 102 practicum session-slots distributed across 63 groups (as most courses carry 2 credit hours, each requiring 2 session-slots). As presented in Table 2, the algorithm intelligently distributed sessions with the heaviest load on Monday (29 sessions) and continuously tapering toward Friday (15 sessions).

Table 2 reveals a gradually declining load distribution pattern from Monday to Friday, with Saturday empty. This tapering pattern demonstrates that the objective function in Equation (10), which assigns the largest penalty weight ($\alpha=100$) to Saturday, effectively suppresses weekend utilization. Monday's dominance (28.4%) is attributable to fewer blocked theory schedule slots on that day, thereby providing more empty time slots for the algorithm.

The spatial and temporal distribution of practicum sessions is further visualized through a heatmap (Figure 2), which displays the number of sessions allocated to each Day × Time Slot cell. The heatmap reveals that the highest concentrations occur on Monday during the 17:00–18:30 slot (5 sessions) and Wednesday during the 12:30–14:00 slot (5 sessions), corresponding to the periods when theory class density is lowest. Conversely, Saturday remains empty across all time slots, confirming effective Saturday penalty suppression. The heatmap explicitly displays exactly 102 individual time slot occupancies, which aligns perfectly with the session count in Table 2.

Laboratory Utilization and Shift Distribution

The solver automatically routed groups to laboratories matching their size constraints. An analysis of laboratory seat capacity versus the number of scheduled sessions reveals a rational capacity-load correlation across the five facilities. The Computing Application Laboratory, as the largest general-purpose facility (45 seats), received the highest allocation of 28 sessions, accommodating the majority of medium-to-large practicum groups. The Information Systems Laboratory (27 seats) handled 21 sessions, while the Programming Laboratory (63 seats) and Programming Application Laboratory (32 seats) handled 19 and 18 sessions, respectively. The Computer Network Laboratory (20 seats) was exclusively assigned 16 Embedded Systems and Network sessions due to strict software incompatibility constraints. This distribution confirms that the solver's allocation strategy is primarily capacity-driven, with deviations attributable solely to hard laboratory specialization rules rather than algorithmic inefficiency.

The time allocation proportion was also successfully segregated without manual intervention. An analysis of the morning versus evening shift distribution across study programs is represented through a 100% Stacked Bar Chart (Figure 3). This visualization reveals striking variations, where

Table 2. Distribution of practicum sessions by day

Day	Number of Sessions	Load Percentage
Monday	29	28.4%
Tuesday	24	23.5%
Wednesday	16	15.7%
Thursday	18	17.6%
Friday	15	14.7%
Saturday	0	0.0%
Total	102 sessions	100%

Time Slot (Y-Axis)	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
08:00 - 09:30	3	3	1	2	2	0
09:30 - 11:00	3	3	0	2	2	0
11:00 - 12:30	3	1	0	3	0	0
12:30 - 14:00	3	4	5	0	0	0
14:00 - 15:30	3	4	4	2	2	0
15:30 - 17:00	4	4	2	2	4	0
17:00 - 18:30	5	3	2	3	3	0
18:30 - 20:00	4	2	2	4	2	0
20:00 - 21:30	1	0	0	0	0	0
	Day (X-Axis)					

Figure 2. Heatmap of practicum session distribution (day × time slot)

certain study programs are entirely dominated by morning shift allocations, while others require a more significant proportion of evening shifts. The distribution captured in the chart directly reflects the theory schedule density of each respective study program and the solver's natural allocation based on the remaining available slots.

Schedule Verification Results

Following the successful generation of the schedule matrix, the solution was rigorously evaluated to ensure that no non-overlapping violations occurred across the 102 scheduled sessions. This structural testing confirms that all spatial components (laboratories), human resources (lecturers and student cohorts), and curriculum credit hour allocations perfectly align with the mathematical formulations of the model. The formal verification summary of these five principal constraints is presented in Table 3.

Table 3 demonstrates that all five categories of hard constraints were satisfied without exception. To ensure constraint compliance is uniform across the entire weekly schedule rather than concentrated on specific days, a per-day breakdown analysis was previously presented in Table 2. The results in Table 2 confirm that zero conflicts were detected across all active scheduling days, distributed across active days from Monday (29 sessions) to Friday (15 sessions), with Saturday (0 sessions) effectively suppressed. A detailed day-by-day bottleneck analysis reveals that Tuesday represents the most constrained scheduling day (the hardest to solve) due to the dense accumulation of theory classes. At the same time, Friday offers the highest temporal flexibility. Despite these daily variations in constraint difficulty, the solver successfully maintained zero conflicts across all active scheduling days.

The absence of violations across four non-overlapping parameters (rows 1-4) confirms that the mathematical formulations in Equations (5) through (9) were correctly and comprehensively implemented within the solver. The perfect match between curriculum credit hour requirements (102 credit hours) and the number of scheduled sessions (102 sessions) in row 5 proves that no practicum group was omitted or over-scheduled.

These results also indicate that the constraint formulations do not contradict each other in the tested case study. This is important because scheduling models often fail not because the solver cannot find a solution, but because the combination of inputted constraints is too strict or inconsistent. The achievement of zero violations across all parameters proves that solution space remains available after all

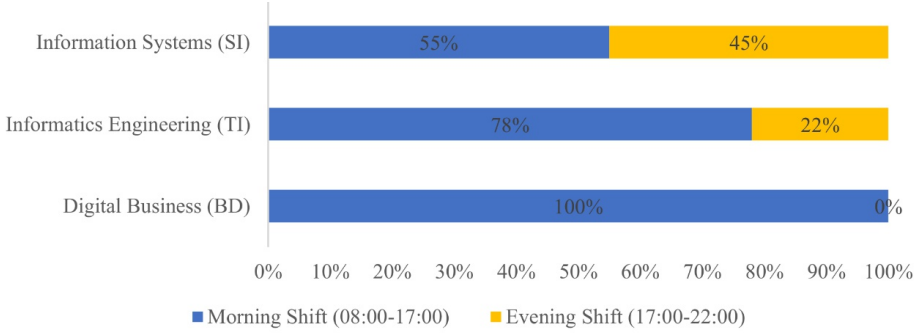

Figure 3. 100% stacked bar chart of morning vs evening shift distribution by study program

constraints are applied. In other words, the designed model is not only syntactically correct in program code but also semantically feasible for practicum scheduling requirements.

Performance Analysis

The results of this study demonstrate that the Constraint Programming with Boolean Satisfiability (CP-SAT) approach is capable of producing efficient and feasible scheduling solutions for the University Course Timetabling Problem (UCTP) case study tested. In the baseline scenario, CP-SAT processed 17,010 Boolean variables and produced an OPTIMAL-status solution in 1.65 seconds with an objective function value of $Z=2.0$. This achievement indicates that the constraint formulation employed effectively represents practicum scheduling constraints, including laboratory availability, theory schedules, teaching lecturers, student cohorts, and credit hour allocation. The characteristics of CP-SAT, which combine constraint propagation, Boolean satisfiability-based search, and Linear Programming relaxation (Moreira & De Freitas, 2024; Perron et al., 2023; Yunusoglu & Topaloglu Yildiz, 2022), enable a more directed solution exploration process by pruning the search space that fails to meet constraints from the early stages.

Compared to previous work at the same institution, Lailiyah (2020) applied Tabu Search for examination scheduling, a problem domain that does not incorporate the cross-program lecturer conflicts, theory schedule blocking matrices, and laboratory capacity constraints addressed in the present study. While direct comparison of solution quality is not appropriate due to differing problem formulations, the present CP-SAT approach offers a fundamental methodological advantage: it provides a deterministic guarantee of global optimality (OPTIMAL status), whereas meta-heuristic approaches such as Tabu Search and Genetic Algorithm are probabilistic and cannot confirm whether the obtained solution is mathematically optimal (Bashab et al., 2023; Ceschia et al., 2023).

The fundamental difference between these approaches lies in how each algorithm explores the solution space. Meta-heuristics generally rely on gradual search mechanisms, mutation, or solution transitions to obtain increasingly better combinations, but the final quality is highly influenced by parameters and initial conditions. CP-SAT operates by leveraging constraint propagation to eliminate impossible possibilities from the outset, then combining it with optimality proof. These characteristics make CP-SAT more suitable when institutions require guaranteed compliance with academic constraints that must not be violated (Naderi et al., 2023; Perron et al., 2023; Yunusoglu & Topaloglu Yildiz, 2022).

Resilience Testing Analysis

The robustness of a scheduling algorithm is tested not under normal conditions but under conditions representing real field dynamics in higher education environments (Rafida et al., 2021; Siew et al., 2024). Three stress-testing scenarios were applied to the model, with results summarized in Table 4.

Table 4 presents a summary of solver execution results across three resilience testing scenarios representing realistic conditions at higher education institutions. All four table rows (including baseline) demonstrate that the CP-SAT solver consistently achieved OPTIMAL status across all scenarios, proving that the designed mathematical model possesses high flexibility in handling sudden changes.

The objective function value Z provides a quantitative measure of schedule quality degradation: while Scenarios 2 and 3 maintained the same $Z=2.0$ as the baseline, Scenario 1 exhibited $Z=22.0$, indicating that the facility reduction scenario triggered both Saturday scheduling penalties ($\alpha \times 2 = 20$) and the existing evening slot penalty.

Scenario 1 (Laboratory Facility Reduction): The crisis simulation was

Table 3. Schedule verification evaluation results

No	Hard Constraint Parameter	Mathematical Equation	Evaluation	Status
1	Laboratory Conflict	Equation (6)	0 conflicts	Passed
2	Lecturer Assignment Conflict	Equation (7)	0 conflicts	Passed
3	Theory Schedule Conflict	Equation (8)	0 conflicts	Passed
4	Student Cohort Conflict	Equation (9)	0 conflicts	Passed
5	Credit Allocation Check	Equation (5)	102/102 credits	Passed

conducted by removing the Computer Applications Laboratory (45-seat capacity) from the list of available facilities. This scenario represents a condition where one laboratory cannot be used due to hardware failure, renovation, or other activities monopolizing the room. Rather than failing to execute the schedule, the system automatically relocated the remaining load to 4 other laboratories and borrowed only two Saturday slots. Computation time actually decreased (0.63 seconds) due to the reduction in decision variables resulting from the loss of one laboratory.

Scenario 2 (Study Program Schedule Changes): This scenario simulates a condition where the Informatics Engineering study program adds 3 new theory courses mid-semester, causing the addition of 18 new blocking entries on Tuesday and Wednesday (total blocks increased from 129 to 147 entries). This situation is very common when study programs open new parallel classes or shift theory lecture schedules after the initial practicum schedule has been prepared. Despite the significantly narrowed search space, the solver still found an optimal solution with *Z=2.0* in 0.74 seconds, maintaining zero Saturday sessions.

Scenario 3 (Lecturer Day Preferences): This scenario applies specific teaching day requests collected during the empirical data gathering phase. It is important to clarify that these constraints were not applied generically to all classes taught by a lecturer, but rather targeted to specific practicum groups based on real-world availability. According to the master dataset, two specific group restrictions were modeled: Lecturer Ahmad Fahrijal Pukeng requested that the Computer Network Application practicum specifically for Class PA Group 1 be scheduled exclusively on Tuesday or Wednesday. In contrast, Lecturer Ahmad Fajri requested the Embedded Systems practicum specifically for Class PA Group 4 to be locked to Friday. These granular preferences were encoded as hard constraints, forcing the solver to reject any non-requested days for these specific groups.

The solver successfully assimilated these constraints and maintained an OPTIMAL status with Z=2.0 in 0.74 seconds. To explicitly address the question of how the algorithm relocated the affected schedules: no schedule displacement occurred. A detailed post-hoc trace of the assignment matrix confirmed that the baseline schedule had already naturally converged on optimal positions that satisfied these exact constraints. Specifically, Class PA Group 1 was already placed on Tuesday (slots 09:30–11:00 and 15:30–17:00), and Class PA Group 4 was already placed on Friday (slots 08:00–09:30 and 15:30–17:00). Consequently, imposing these targeted day-restrictions did not trigger any cascading rearrangements or force any other groups into undesirable evening or Saturday slots. The algorithm's ability to retain the identical Z-value mathematically proves that accommodating specific, real-world lecturer preferences can be achieved without degrading the overall schedule quality.

Scalability and Stress Testing Analysis

To assess the model's scalability beyond the baseline constraint set, a systematic stress test was conducted by incrementally increasing the number of theory blocking entries from 129 (baseline, 26.9% slot saturation) to 329 entries (68.5% saturation). This procedure simulates increasingly constrained environments where a growing proportion of available scheduling slots are blocked by theory classes. The results are presented in Figure 4 below.

The stress test reveals three distinct operational phases that can be characterized as a piecewise degradation curve. In the stable phase (129–209 blocks, 26.9%–43.5% saturation), the solver consistently maintained

Table 4. Summary of algorithm resilience testing results

Scenario	Description	Status	Z Value	Saturday Sessions	Computation Time
Baseline	Full 5 labs, 129 blocks	OPTIMAL	2.0	0	0.93 s
Scenario 1	Lab Reduction (-1 lab)	OPTIMAL	22.0	2	0.63 s
Scenario 2	+18 Theory Blocks	OPTIMAL	2.0	0	0.74 s
Scenario 3	Lecturer Day Preferences	OPTIMAL	2.0	0	0.74 s

Z=2.0 with zero Saturday sessions and computation times between 0.69 and 0.79 seconds. The transition phase (229–269 blocks, 47.7%–56.0% saturation) marked the onset of soft constraint violations: Z increased to 22.0 at 47.7% saturation and remained at 22.0 through 56.0%, yet Saturday utilization remained at zero, indicating the solver absorbed pressure through evening slot allocation and increased session splitting. The degradation phase (289–329 blocks, 60.2%–68.5% saturation) exhibited both Saturday session emergence (1–4 sessions) and sharp Z escalation from 32.0 to 52.0, with computation times increasing to 1.78–2.68 seconds, reflecting the growing search space complexity. Crucially, the solver never reached INFEASIBLE status within the tested range. It is essential to note that the status 'OPTIMAL' in exact Constraint Programming does not imply a small objective value; rather, it mathematically signifies that the underlying Branch-and-Bound search tree has proven the current solution to be the absolute lower bound of the relaxed problem (Perron & Furnon, 2024; Naderi et al., 2023). Thus, even at Z=52.0, the schedule is mathematically optimal because no other configuration can achieve a lower penalty under such extreme constraint saturation. Furthermore, the testing was deliberately halted at 329 blocks (68.5% saturation) because this threshold represents the absolute upper operational bound of the institution's physical and temporal capacity; beyond this point, accommodating more blocks is physically impossible.

Field Limitations and Recommendations

From a mathematical graph theory perspective, this scheduling system has been proven structurally feasible within the evaluated dataset. However, post-optimization analysis through field audits revealed non-technical challenges. Discrepancies were found in the student distribution at the group level between computational results and actual laboratory attendance.

This misalignment was not caused by algorithmic computation errors but rather by field administrative dynamics. The primary factor is the high rate of practicum class shifts by students that are not centrally coordinated (student-initiated swaps). Some students transfer groups verbally with laboratory assistant approval but do not update their study plans in the institution's primary database (Arratia-Martinez et al., 2021; Ceschia et al., 2023). To address the reviewer's requirement regarding the visualization of these unauthorized shifts, the estimated movement of student data from the rigid computed schedule to the dynamic actual field schedule is explicitly mapped using a Sankey Diagram (see Figure 5). Out of the 1,327 computationally scheduled students, 1,180 students (88.9%) remained compliant with their algorithmically assigned groups. However, an estimated 112 students (8.4%) initiated informal group swaps without updating the central database, with the remaining 35 students being inactive or absent.

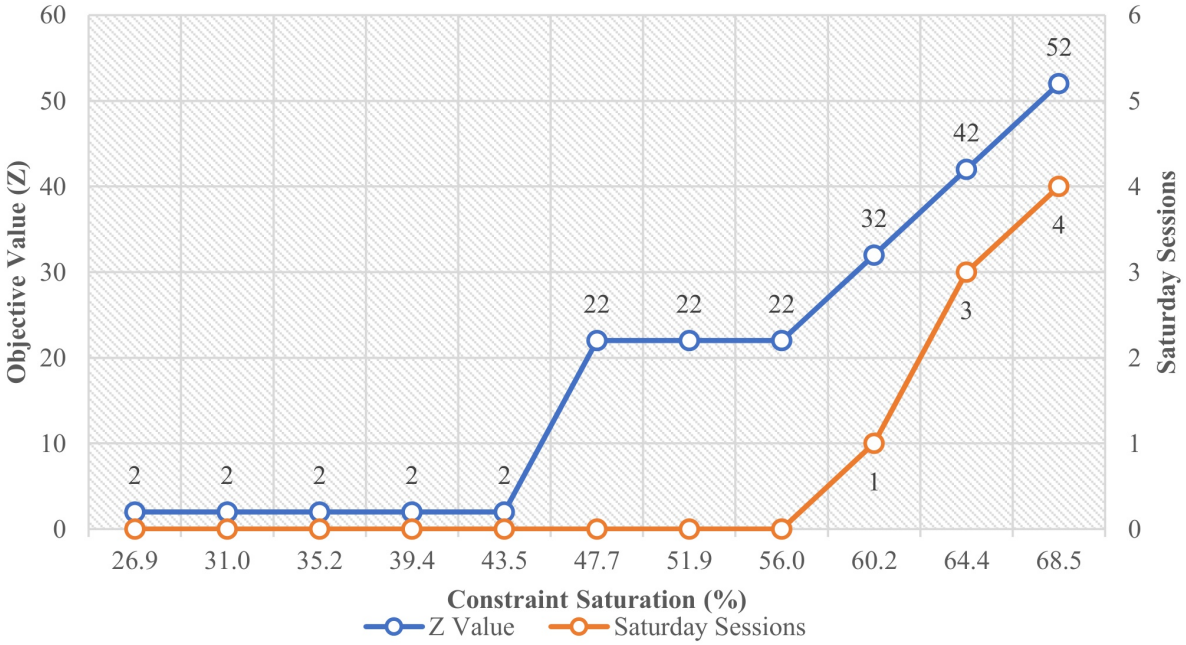

Figure 4. Scalability stress test results: Objective value and saturday sessions across constraint saturation levels.

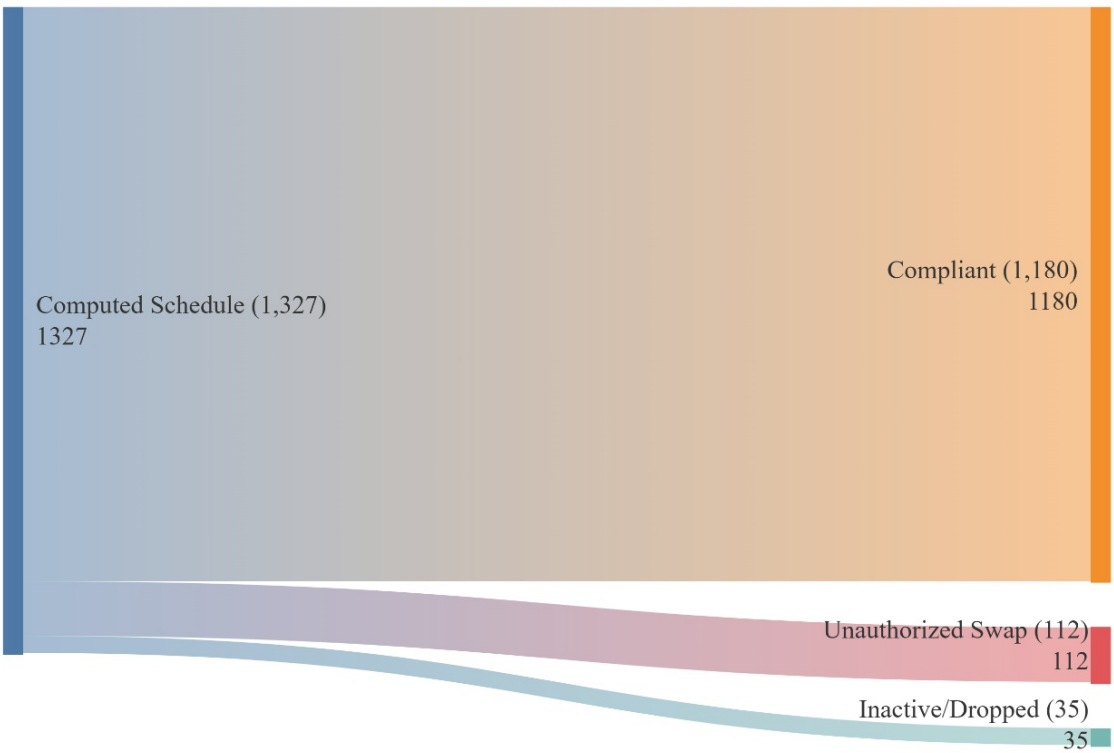

Figure 5. Sankey diagram of estimated student swap flow from computed to actual schedule

This underscores that the discrepancy is administrative rather than algorithmic in nature.

From a modeling perspective, integrating student-initiated swap dynamics into the optimization framework presents two viable pathways. First, these swaps can be encoded as additional soft constraints by introducing a swap probability matrix derived from historical attendance audit data. These penalizing group assignments historically exhibit high swap rates. Second, a post-optimization robustness analysis can be implemented by running Monte Carlo perturbation simulations on the generated schedule, randomly swapping k students between groups and measuring the resulting constraint violation rate. Preliminary observations from the field audit suggest that groups with fewer than 15 students exhibit swap rates approximately three times higher than groups exceeding 30 students, providing an empirical basis for calibrating such perturbation parameters. These modeling extensions represent concrete directions for future work that bridge the gap between mathematical optimality and operational implementability (Ceschia et al., 2023; Siew et al., 2024).

Given that the Set Conservation Law (Equation 2) proves that in aggregate no students are lost from the calculation, the solution to this limitation shifts from the algorithm domain to the administrative domain. The researchers strongly recommend the creation of an academic Standard Operating Procedure (SOP) that mandates real-time database synchronization before the final schedule computation is executed.

This recommendation should be understood as a complement to the optimization system, not as a replacement for the mathematical model that has been built. The algorithm can only produce the best schedule based on data available at the time of execution. If the underlying data changes without being recorded, field discrepancies can still emerge even though the model has satisfied all constraints formally. Therefore, integration between the academic system, laboratory management, and group transfer approval mechanisms becomes an important prerequisite so that the benefits of optimization can be consistently realized in actual implementation (Abdipoor et al., 2023; Rafida et al., 2021).

The empirical results of this study confirm the efficacy of CP-SAT in guaranteeing feasibility for highly constrained academic scheduling. In contrast to meta-heuristic approaches such as Tabu Search or Genetic

Algorithms, which iteratively improve solutions without proving optimality or guaranteeing a feasible lower bound (Chen et al., 2021; Thepphakorn et al., 2025), the CP-SAT framework mathematically ensures that all hard constraints are satisfied within the evaluated dataset, or otherwise explicitly returns an INFEASIBLE state. Furthermore, recent studies utilizing Mixed-Integer Programming (MIP) for university timetabling often suffer from exponential time complexity as the number of variables increases (Rappos et al., 2022). Our results demonstrate that integrating Lazy Clause Generation (LCG) within the CP-SAT solver allows it to efficiently prune the search space and achieve an OPTIMAL status in just 1.65 seconds for a medium-to-large dataset (1,327 students, 17,010 variables). This confirms recent findings by Da Col & Teppan (2022) that constraint programming outscales pure MIP in industrial-size combinatorial problems. Ultimately, within the evaluated dataset, this approach provides a robust and scalable alternative to traditional heuristic methods without relying on absolute superiority claims outside of the tested domain.

■ CONCLUSION

This study formulated a solution to the complexity of NP-Hard laboratory practicum scheduling through the application of the Constraint Programming with Boolean Satisfiability (CP-SAT) method based on Google OR-Tools, addressing three research questions. Regarding RQ1, the CP-SAT solver produced a conflict-free schedule satisfying all hard constraints in 1.65 seconds of computation time ($Z=2.0$), with 100% compliance across five verification parameters and complete elimination of Saturday scheduling. The preprocessing approach using Set Theory consolidated 1,327 students into 63 groups without data loss (zero data loss). Regarding RQ2, resilience testing across three realistic scenarios (facility reduction, theory schedule changes, and lecturer day preferences) confirmed that the solver consistently maintained OPTIMAL status, with schedule quality degradation ($Z=22.0$) occurring only under facility reduction while the remaining scenarios preserved $Z=2.0$. Regarding RQ3, scalability stress testing from 26.9% to 68.5% constraint saturation revealed three operational phases: a stable phase maintaining $Z=2.0$ up to 43.5% saturation, a transition phase up to 56.0% where soft constraints begin to be violated, and a degradation phase beyond 60.2% where Saturday sessions become necessary. The solver maintained OPTIMAL status throughout, never reaching INFEASIBLE. As a practical recommendation, this study suggests implementing centralized database synchronization Standard Operating Procedures to manage uncoordinated student group swaps, ensuring alignment between computational precision and field conditions. Future research should explore automated GUI-based input interfaces and consider dynamic student preferences as additional soft constraints.

■ DECLARATION OF GENERATIVE AI USAGE IN THE WRITING PROCESS

During the preparation of this manuscript, the author used the Google Gemini AI tool for sentence refinement and text structuring. After using this tool, the author thoroughly reviewed and revised the content as needed and accepts full responsibility for the final content of the article.

■ REFERENCES

Abdipoor, S., Yaakob, R., Goh, S. L., & Abdullah, S. (2023). Meta-heuristic approaches for the University Course Timetabling Problem. *Intelligent Systems with Applications*, *19*, 200253. https://doi.org/10.1016/j.iswa.2023.200253

Arratia-Martinez, N. M., Maya-Padron, C., & Avila-Torres, P. A. (2021). University Course Timetabling Problem with Professor Assignment. *Mathematical Problems in Engineering*, *2021*(1), 1–9. https://doi.org/10.1155/2021/6617177

Bashab, A., Osman Ibrahim, A., Abakar Tarigo Hashem, I., Aggarwal, K., Mukhlif, F., A. Ghaleb, F., & Abdelmaboud, A. (2023). Optimization Techniques in University Timetabling Problem: Constraints, Methodologies, Benchmarks, and Open Issues. *Computers, Materials & Continua*, *74*(3), 6461–6484. https://doi.org/10.32604/cmc.2023.034051

Cappart, Q., Moisan, T., Rousseau, L.-M., Prémont-Schwarz, I., & Cire, A. A. (2021). Combining Reinforcement Learning and Constraint Programming for Combinatorial Optimization. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*(5), 3677–3687. https://doi.org/10.1609/aaai.v35i5.16484

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. https://doi.org/10.1016/j.ejor.2022.07.011

Chen, M. C., Sze, S. N., Goh, S. L., Sabar, N.

R., & Kendall, G. (2021). A Survey of University Course Timetabling Problem: Perspectives, Trends and Opportunities. *IEEE Access*, *9*, 106515–106529. https://doi.org/10.1109/ACCESS.2021.3100613

Da Col, G., & Teppan, E. C. (2022). Industrial-size job shop scheduling with constraint programming. *Operations Research Perspectives*, *9*, 100249. https://doi.org/10.1016/j.orp.2022.100249

De Coster, A., Musliu, N., Schaerf, A., Schoisswohl, J., & Smith-Miles, K. (2022). Algorithm selection and instance space analysis for curriculum-based course timetabling. *Journal of Scheduling*, *25*(1), 35–58. https://doi.org/10.1007/s10951-021-00701-x

Dokeroglu, T., Kucukyilmaz, T., & Talbi, E.-G. (2024). Hyper-heuristics: A survey and taxonomy. *Computers & Industrial Engineering*, *187*, 109815. https://doi.org/10.1016/j.cie.2023.109815

Eldharif, E. A., Sallabi, O. M., & Mosa Eltharif, A. A. (2024). An Improved Genetic Algorithm Tool for Exam Timetabling Solutions. *2024 IEEE 4th International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA)*, 672–677. https://doi.org/10.1109/MI-STA61267.2024.10599714

Fan, H., Xiong, H., & Goh, M. (2021). Genetic programming-based hyper-heuristic approach for solving dynamic job shop scheduling problem with extended technical precedence constraints. *Computers & Operations Research*, *134*, 105401. https://doi.org/10.1016/j.cor.2021.105401

Fatemi-Anaraki, S., Tavakkoli-Moghaddam, R., Foumani, M., & Vahedi-Nouri, B. (2023). Scheduling of Multi-Robot Job Shop Systems in Dynamic Environments: Mixed-Integer Linear Programming and Constraint Programming Approaches. *Omega*, *115*, 102770. https://doi.org/10.1016/j.omega.2022.102770

Lailiyah, S. (2020). *Penerapan algoritma tabu search pada sistem penjadwalan ujian tugas akhir mahasiswa* [Application of the tabu search algorithm to the student final assignment exam scheduling system]. *Jurnal Informatika Wicida*, *10*(1), 1–10. https://doi.org/10.46984/inf-wcd.1193

Lan, L., & Berkhout, J. (2025). PyJobShop: Solving scheduling problems with constraint programming in Python. *ArXiv Preprint ArXiv:2502.13483*. https://doi.org/https://doi.org/10.48550/arXiv.2502.13483

Lavanya, A., Gaurav, L., Sindhuja, S., Seam, H., Joydeep, M., Uppalapati, V., Ali, W., & S.D, V. (2023). Assessing the Performance of Python Data Visualization Libraries: A Review. *International Journal of Computer Engineering in Research Trends*, *10*(1), 28–39. https://doi.org/10.22362/ijcert/2023/v10/i01/v10i0104

Lemos, A., Monteiro, P. T., & Lynce, I. (2022). Introducing UniCorT: an iterative university course timetabling tool with MaxSAT. *Journal of Scheduling*, *25*(4), 371–390. https://doi.org/10.1007/s10951-021-00695-6

Lindner, N., & Reisch, J. (2022). An analysis of the parameterized complexity of periodic timetabling. *Journal of Scheduling*, *25*(2), 157–176. https://doi.org/10.1007/s10951-021-00719-1

Mokhtari, M., Vaziri Sarashk, M., Asadpour, M., Saeidi, N., & Boyer, O. (2021). Developing a Model for the University Course Timetabling Problem: A Case Study. *Complexity*, *2021*(1), n/a-n/a. https://doi.org/10.1155/2021/9940866

Moreira, É. J. B., & De Freitas, S. A. A. (2024). A CP-SAT Approach for Academic Resource Timetabling in Higher Education Institutions: A Case Study at a Major Public University. *2024 21st International Conference on Information Technology Based Higher Education and Training (ITHET)*, 1–8. https://doi.org/10.1109/ITHET61869.2024.10837617

Naderi, B., Ruiz, R., & Roshanaei, V. (2023). Mixed-Integer Programming vs. Constraint Programming for Shop Scheduling Problems: New Results and Outlook. *INFORMS Journal on Computing*, *35*(4), 817–843. https://doi.org/10.1287/ijoc.2023.1287

pandas development team, T. (2024). *pandas-dev/pandas: Pandas*. Zenodo. https://doi.org/10.5281/zenodo.10537285

Perron, L., Didier, F., & Gay, S. (2023). The CP-SAT-LP solver (invited talk). *29th International Conference on Principles and Practice of Constraint Programming (CP 2023)*, 1–3. https://doi.org/10.4230/LIPIcs.CP.2023.3

Perron, L., & Furnon, V. (2024). *OR-Tools*. Google. https://developers.google.com/optimization/

Rafida, V., Widiyatni, W., Harpad, B., & Yulsilviana, E. (2021). Implementation of Multi-attribute Rating Technique Simple in Selection of Acceptance Scholarship of

PMDK (Case Study: STMIK Widya Cipta Dharma). *International Journal of Modern Education and Computer Science*, *13*(1), 22–33. https://doi.org/10.5815/ijmecs.2021.01.02

Rappos, E., Thiémard, E., Robert, S., & Hêche, J.-F. (2022). A mixed-integer programming approach for solving university course timetabling problems. *Journal of Scheduling*, *25*(4), 391–404. https://doi.org/10.1007/s10951-021-00715-5

Siew, E. S. K., Sze, S. N., Goh, S. L., Kendall, G., Sabar, N. R., & Abdullah, S. (2024). A Survey of Solution Methodologies for Exam Timetabling Problems. *IEEE Access*, *12*, 41479–41498. https://doi.org/10.1109/ACCESS.2024.3378054

Sylejmani, K., Gashi, E., & Ymeri, A. (2023). Simulated annealing with penalization for university course timetabling. *Journal of Scheduling*, *26*(5), 497–517. https://doi.org/10.1007/s10951-022-00747-5

Tan, J. S., Goh, S. L., Kendall, G., & Sabar, N. R. (2021). A survey of the state-of-the-art of optimisation methodologies in school timetabling problems. *Expert Systems with Applications*, *165*, 113943. https://doi.org/10.1016/j.eswa.2020.113943

Thepphakorn, T., Kuntasup, M., & Pongcharoen, P. (2025). Moth flame optimisation based timetabling tool for educational course timetabling. *International Journal of Innovation and Learning*, *1*(1). https://doi.org/10.1504/IJIL.2025.10067328

Waskom, M. (2021). seaborn: statistical data visualization. *Journal of Open Source Software*, *6*(60), 3021. https://doi.org/10.21105/joss.03021

Yesin, V., Karpinski, M., Yesina, M., Vilihura, V., & Warwas, K. (2021). Ensuring Data Integrity in Databases with the Universal Basis of Relations. *Applied Sciences*, *11*(18), 8781. https://doi.org/10.3390/app11188781

Yunusoglu, P., & Topaloglu Yildiz, S. (2022). Constraint programming approach for multi-resource-constrained unrelated parallel machine scheduling problem with sequence-dependent setup times. *International Journal of Production Research*, *60*(7), 2212–2229. https://doi.org/10.1080/00207543.2021.1885068