

Classroom Assignment: A Two-Stage Constraint Formulation

Jhojan Alexander Velasquez Erazo

Universidad del Valle

Carlos Andres Delgado Saavedra

`carlos.andres.delgado@correounivalle.edu.co`

Universidad del Valle

Research Article

Keywords: Classroom assignment, Constraint programming, MiniZinc, University course timetabling, Timetabling tools

Posted Date: October 27th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-11210083/v1>

License:   This work is licensed under a Creative Commons Attribution 4.0 International License.
[Read Full License](#)

Additional Declarations: No competing interests reported.

Classroom Assignment: A Two-Stage Constraint Formulation

Jhojan Alexander Velasquez Erazo¹ and Carlos A. Delgado S.^{1*}

¹Escuela de Ingeniería de Sistemas y Computación, Universidad del Valle, Seccional Tuluá, Tuluá, Colombia.

*Corresponding author(s). E-mail(s): carlos.andres.delgado@correounivalle.edu.co;
Contributing authors: jhojan.velasquez@correounivalle.edu.co;

Abstract

Timetabling models are known to be over-constrained, and the standard response is to soften constraints in the solver. Which constraints an institution actually breaks, and how often, is not measured. We measure it. A two-stage constraint formulation in MiniZinc places course groups into weekly slots under availability, intensity, and clash constraints, then allocates rooms by capacity and equipment. We compare three encodings of the room stage across Gecode, Chuffed, and COIN-BC. Structural constraints remove some 7 700 orders of magnitude from the search space before search begins. Measured solving time grows as about the 1.7 power of instance size. We audit eight consecutive semesters of a working campus's schedules, as published by its scheduling system, against the model's hard constraints. No semester satisfies them. Rooms are double-booked every semester, yet in three cases out of four a suitable room stood empty in the same sub-site at that hour. Scarcity therefore accounts for a minority of the violations. Such an audit, recording which hard constraints an institution breaks, how often, and with what slack, tells a modeler which to soften. It can run wherever scheduling data is published. We release the instances and the generator calibrated on them.

Keywords: Classroom assignment, Constraint programming, MiniZinc, University course timetabling, Timetabling tools

1 Introduction

Every semester, universities face the same scheduling exercise: place every course group into a time slot and a room without crossing teacher schedules, program curricula, or room capacities. The underlying problem, university course timetabling, is NP-hard, and small increases in courses, rooms, or constraints translate into large increases in solving effort (Babaei et al. 2015). Institutions that handle it manually pay the cost in staff time and in assignment quality.

Models of the problem state their requirements as hard constraints, and the field has long known

that the requirements collide. Rudová and Murray (2003) say so directly of the Purdue problem and answer it by weighting partial satisfaction, the design most deployed systems inherit. Two studies go further and confront a model with what an institution did. Murray, Müller and Rudová (2007) solve the Purdue problem and compare their solution against the timetable the university had produced. Schneider, Leuschel and Witt (2018) build a formal model of one university's rules in order to validate existing timetables rather than generate new ones, and report programs whose requirements cannot hold together, with unsatisfiable cores to say why.

What neither supplies is a measurement that a modeler can act on. Both look at a single snapshot, so a violation cannot be distinguished from an accident of one term; Schneider et al. audit curricula and exclude rooms and group sizes, which is where our violations turn out to concentrate; and neither asks whether the violations reflect scarcity or something else. Softening a constraint is a modeling decision, and it needs to know which constraints an institution breaks, how often, and why. Scheduling systems have begun to publish the evidence that would answer this.

We use one such record as an instrument. The Tuluá campus of Universidad del Valle publishes eight consecutive semesters of its assignments, 286 to 391 course groups over 107 to 153 teachers and thirty bookable spaces across three sub-sites. The campus is ordinary in the ways that matter: its room occupancy, 32% to 46%, falls inside the range reported for teaching buildings elsewhere (Li et al. 2026; Osei et al. 2025), and its scheduling is done by one person in three to four days without software (Velasquez Erazo 2022c). What makes it useful is not that it is special but that its history is visible.

Our contributions are:

1. a two-stage constraint formulation that separates timetabling from room allocation, chained so that the first stage’s output is the second stage’s input, with the conditions under which the separation is safe and the conditions under which it is not (Section 4);
2. a count of the search space the formulation actually explores, which locates the difficulty of the problem in its structure rather than in its nominal size, and a solver comparison over three encodings of the room stage (Sections 4 and 6);
3. an instrument for measuring a model against practice, needing no institutional access beyond what the scheduling system publishes: eight consecutive semesters of schedules, an audit of each against the model’s hard constraints, and a generator producing synthetic instances calibrated on the same distributions (Section 7);
4. the finding that no semester satisfies the hard constraints, that room conflicts rarely coincide with an exhausted room stock, and that the two counts the model minimizes are unequal in what they can achieve (Section 8);
5. four decisions this evidence puts within reach of an institution: which constraints are genuinely hard, which objectives are worth stating, where the next modeling effort belongs, and whether its records can support a model at all (Section 8.4); and
6. a deployed web tool built on the formulation, reported as an existence proof that the timetabling stage runs on commodity hardware within the turnaround an institution asks for, and as evidence that the room stage does not (Sections 5 and 6).

The first author’s undergraduate thesis (Velasquez Erazo 2022c) introduced the formulation, the tool, and the three room-stage encodings, and compared solvers on them; we re-run that comparison on the instances released here. Contributions 1 and 6 therefore restate prior work. They stay because the audit checks the institution’s schedules against that formulation. New in this paper are the search-space count, the audit and its findings, the decisions, and the generator.

2 Related Work

2.1 The problem, and how it has been attacked

Timetabling entered the literature as graph coloring. Welsh and Powell (1967) bounded the chromatic number and applied the result to timetables, and De Werra (1985) made explicit what the abstraction leaves out. That remainder is most of what an institution cares about. The complexity followed: Even, Itai and Shamir (1976) proved the problem NP-complete, and Cooper and Kingston (1996) showed which construction variants stay hard. Carter and Tovey (1992) asked when classroom assignment is actually hard and answered in terms of the slack between rooms and demand. We measure that slack in Section 7.

Surveys track the field through its stages: the first decade of automation (Schaerf 1999), the directions that shaped the 2000s (Burke and Petrovic 2002), the metaheuristics (Lewis 2008; Abdipoor et al. 2023), approaches classified by the constraints they can express (MirHassani and Habibi 2013), the school variant (Pillay 2014), the reference treatment of curriculum-based course timetabling that the standard benchmarks encode (Bettinelli

et al. 2015), an updated taxonomy (Babaei et al. 2015), and machine-learning formulations (Gu et al. 2025). Competitions turned scattered formulations into shared benchmarks, from the second International Timetabling Competition (McCollum et al. 2010) through ITC-2019 and its institutional datasets (Müller et al. 2024; Ceschia et al. 2023). Alongside them runs a complaint that the benchmarks drift from what universities need. McCollum named the gap between theory and practice (McCollum 2007), and a systematic review of higher-education practice found institutional requirements still poorly represented a decade later (Oude Vrielink et al. 2019).

The methods divide cleanly. Simulated annealing came first to classroom assignment (Martinez-Alfaro and Flores-Teran 1998) and remains the baseline. Genetic algorithms became the most visited, plain (Ardi Nugraha et al. 2017), hybridized with local and tabu search (Jaengchuea and Lohpetch 2015), and parallelized (Rezaeipanah et al. 2019); particle swarm extends the line to tightly constrained instances (Imran Hossain et al. 2019). Hyper-heuristics select among heuristics rather than apply one (Soria-Alcaraz et al. 2010), multi-agent formulations distribute the decision (Syskov et al. 2020), and Bayesian machinery learns the soft-constraint preferences human schedulers apply (Siame and Kunda 2018). Exact methods have not been displaced: mixed-integer formulations run at university scale (Rappos et al. 2022) and add room stability and lecturer preferences (Ozkan et al. 2025), and constraint programming handles curriculum-based high-school timetabling with half-blocks (Dubois et al. 2026), a structure close to the per-intensity day rules our campus imposes. Objectives beyond feasibility appear occasionally, and ours are of that kind: pedestrian traffic between buildings (Liu 2013), a concern our campus has and our model does not encode, and student needs ahead of administrative convenience (Almeida et al. 2015). Ferdiana et al. (2017) report a cloud-deployed timetabling application, the closest published analogue to the tool of Section 5.

2.2 What the field does not report

Three recent strands bear on our setting. Institutional case studies take one university’s problem rather than a benchmark’s, for an honors program (Rakotonirainy et al. 2023) and under the

room pressures hybrid teaching introduced (Davison et al. 2024). Room allocation is studied as its own problem, with multiobjective formulations that place rooms once the timetable exists (Antunes-Batista et al. 2026); we adopt the same decomposition. And formulations grow more realistic, built on datasets from working institutions (Müller et al. 2024), refined to the individual study paths students follow (Steiner et al. 2024), and applied in Latin America with a focus on professor assignment (Zunino et al. 2024), the constraint our own academic secretary named as the hardest.

Deployment is where the field is thinnest. Barnhart et al. (2022) built integer models for MIT under pandemic capacity limits, and a schedule they produced ran at the Sloan School. Chen et al. (2023) take an exact formulation into a Chinese faculty and compare the result against the timetable in use. Both report what an institution did with the output. That is the standard we try to meet.

Arratia-Martinez et al. (2021) are the nearest analogue, and the differences are instructive. They formulate course timetabling with professor assignment for one department, 24 courses in 69 sections under 29 professors, and solve to optimality with CPLEX in 3.16 seconds. Scale separates the two studies first: a single department at 69 sections against a whole campus at 286 to 391 groups, where the room stage stops being the easy half (Section 6). Validation separates them more. They display the timetable the model produces; we compare the model against the timetables the institution produced without it. Theirs asks whether the model works. Ours asks what the model would change, and answering that needs a record of what the institution currently does.

Three gaps remain once those precedents are granted. Benchmarks characterize instances by size, constraint counts, and room utilization (Müller et al. 2024), but almost none report how the institution that supplied the data was scheduling beforehand, so the hard constraints a model declares are rarely checked against what the institution treats as inviolable. Where they have been checked, the check covers one term, which cannot separate a standing practice from an accident. And no study we know of asks whether the violations it finds reflect a shortage of the resource the constraint protects.

This paper closes the three for one institution and gives a procedure for taking them anywhere a scheduling system publishes its output. The formulation is conventional; what distinguishes the work is that it is turned around and used as a measuring instrument on eight consecutive semesters the institution produced without it, and that the measurement is diagnostic rather than merely negative: it says which constraints break, how often, and whether scarcity explains them.

We chose constraint programming for two reasons. The campus rules arrive as hard feasibility statements, each mapping one-to-one onto a CP constraint, and MiniZinc (Nethercote et al. 2007) decouples the model from the solver, so one formulation runs against interchangeable back ends (Stuckey et al. 2014), a property the solver study in Section 6 exploits. Mature systems exist, and UniTime is the obvious candidate, grown from the Purdue system built on weighted partial satisfaction of soft constraints (Rudová and Murray 2003). Two things kept us from that route. The campus rules are peculiar enough that adopting a general system means encoding them anyway: a shift structure confining a program wholly to the day or wholly to the evening, three sub-sites with their own rooms, and per-intensity rules on how a course may be split across days. And the academic secretary asked for an assignment within a day, a requirement about turnaround rather than about features, so building the model directly let us test whether that requirement is reachable at all (Section 6).

3 Problem Description

The problem has two coupled decisions. We describe them as one institution poses them: the constraints that matter are the ones an institution states, and only a formulation elicited from a working process can later be tested against it. First, every course group must land on weekly hours that respect its teacher’s availability and its program’s curriculum grid. Second, every scheduled hour must receive a room whose capacity and equipment match the group.

We elicited the factors that the staff actually weigh through two instruments: a questionnaire answered by three program coordinators and a structured interview with the academic secretary, who owns the assignment. The coordinators build

the teaching offer for their programs, and all three do it alone and take several weeks over it; two of the three expect that time to grow. Change requests arrive after the work is finished for two of the three, and handling them costs another day to a week.

The secretary’s account is more precise. Each semester he compiles between 300 and 350 course groups across eight programs and roughly 1 620 students, then assigns hours and rooms by hand, with no software, in three to four days. He attributes the delay to a specific bottleneck: with around 200 hourly-paid teaching contracts, clashes in teacher availability force him to telephone teachers and ask for alternative windows. Without that, he estimates the work would take one or two days, and one day is what he considers reasonable. Between ten and twenty change requests arrive per semester, mostly because a teacher resigns and the replacement declares different availability, and a replacement takes about eight days to contract.

Asked which factor is hardest, he named teacher availability, ahead of room capacity and course hours. That answer shaped the model: availability is the one constraint the campus cannot negotiate quickly, and Section 7 shows it is also the one the records represent least reliably. He had tried no scheduling tool, said the manual method was not the best strategy, and would adopt software that returned an assignment within a day (Velasquez Erazo 2022c).

The factors carried into the model are therefore the number of rooms, their capacity and the resource each one offers, teacher availability, the subjects each teacher can teach, the resource each subject requires, and the subjects offered in a given semester. Resource here means the equipment that makes a room usable for a given subject: a computer laboratory, the food laboratory, the sports hall, or none.

4 Constraint Model

Following the observation that a monolithic model did not scale, we split the problem into two MiniZinc models executed in sequence.

4.1 How large the problem is

Before the constraints, the timetabling stage is a Boolean matrix C of $96 \times A$ entries, where A is the number of course groups. For the smallest

real semester, $A = 286$, that is $96 \times 286 = 27\,456$ variables and so $2^{27\,456}$ assignments; taking $\log_{10}$ of that, $27\,456 \log_{10} 2 \approx 8265$. The figure is not informative on its own; what matters is how much of it the formulation removes before search begins, and Fig. 1 tracks that.

Requiring each group to occupy exactly its weekly intensity cuts the count to $\prod_a \binom{96}{i_a}$, where i_a is the weekly intensity of group a . Under the campus mixture of intensities each factor averages $10^{5.58}$, so at $A = 286$ the product is 10^{1596} . Confining a group to its shift and forcing contiguous blocks under the per-intensity day rules cuts it much further. The day shift runs 06:00 to 18:00 and the evening shift 18:00 to 22:00, so a three-hour block fits the day shift in ten positions, the first starting at 06:00 and the last at 15:00, and fits the evening shift in two. A group of three weekly hours in the day shift therefore has 60 admissible placements, six days by ten positions, and the same group in the evening shift has 12. The count falls as the block lengthens and rises when the intensity splits across days: four weekly hours admit 3 684 day-shift placements, since the model allows them as one block of four or as two blocks of two on different days. Across the campus mixture of intensities and shifts the average is $10^{2.07}$ placements per group. At $A = 286$ the space the solver explores is therefore about 10^{592} .

Two readings follow. The structural constraints buy roughly 7 700 orders of magnitude, far more than any search heuristic could. That is the argument for spending effort on the formulation rather than on the solver. And what remains is still astronomically large: 10^{592} dwarfs the 10^{80} atoms usually quoted for the observable universe, so the model is not solvable by enumeration at any scale. Table 3 shows that COIN-BC nonetheless finishes in minutes. The instances are therefore far from the hard region that Carter and Tovey (1992) identify for this problem.

4.2 Terminology and notation

Hours are indexed over a six-day week of sixteen hours each running from 06:00 to 22:00, so $h \in 1..H$ with $H = 96$, the first twelve hours of each day forming the day shift, 06:00 to 18:00, and the last four the evening shift, 18:00 to 22:00. A group belongs to one shift and all of its hours fall inside it, so no block straddles the boundary. Nothing

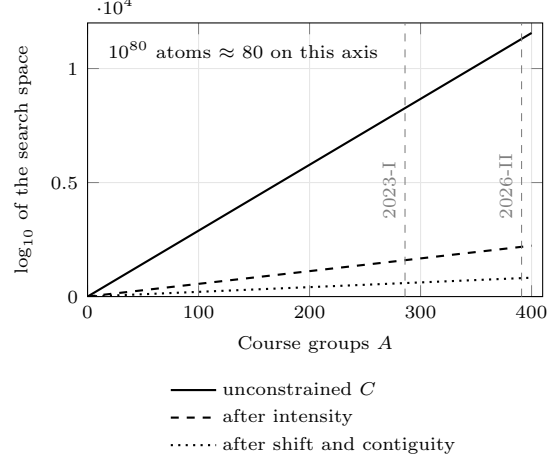


Figure 1 Size of the timetabling search space against instance size, on a logarithmic scale. Each constraint layer removes orders of magnitude before search starts. The vertical lines bracket the eight real campus semesters.

forbids teaching over the lunch band: those hours are admissible and the objective function penalizes them, as does the opening hour. Table 1 lists the symbols. The data are a tuple $A_{a,\cdot}$ per course group a giving its program, semester, course, group number, shift and weekly intensity; an availability relation $D_{h,p}$ recording that teacher p is free at hour h ; the set Da_p of groups taught by p ; the capacity Sc_s and resource kind Sr_s of each room s ; and the enrollment cap Ae_a and required resource Ar_a of each group. Two families of decision variables follow: $C_{h,a}$, true when group a is taught at hour h , for the first stage, and $SAO_{h,s}$, the group occupying room s at hour h and zero when it is free, for the second.

4.3 Timetabling model

The first stage decides C . The core constraints are:

$$\sum_{h=1}^H C_{h,a} = A_{a,6} \quad \forall a \quad (1)$$

$$C_{h,a} \rightarrow D_{h,p} \quad \forall h, \forall p, \forall a \in Da_p \quad (2)$$

$$\sum_{a \in Da_p} C_{h,a} \leq 1 \quad \forall h, \forall p \quad (3)$$

Equation (1) forces every subject to receive exactly its weekly intensity. Equation (2) confines classes to hours where the teacher is available, and Eq. (3) keeps any teacher from being in two rooms at once.

Table 1 Symbols used in both models. Indices run over hours $h \in 1..H$, course groups $a \in 1..A$, teachers $p \in 1..P$, and rooms $s \in 1..S$.

Symbol	Type	Meaning
$A_{a,\cdot}$	integer, $A \times 6$	program, semester, course, group number, shift, and weekly intensity of group a
$D_{h,p}$	Boolean, $H \times P$	teacher p is free at hour h
Da_p	set of groups	the groups taught by teacher p
Sc_s	integer	seating capacity of room s
Sr_s	resource kind	equipment room s offers
Ae_a	integer	enrollment cap of group a
Ar_a	resource kind	equipment group a requires
$C_{h,a}$	Boolean, $H \times A$	group a is taught at hour h (<i>stage one</i>)
$SAO_{h,s}$	integer $0..A$	group occupying room s at hour h , zero when free (<i>stage two</i>)

Further constraints, stated in full in the thesis (Velasquez Erazo 2022c), restrict day-shift subjects to daytime hours and evening-shift subjects to evening hours, force multi-hour subjects into contiguous blocks with per-day caps that depend on intensity, and forbid clashes between subjects of the same program and semester.

The model minimizes the number of class hours placed in the two slots the staff want empty: the 6:00–7:00 opening hour and the 12:00–14:00 lunch band.

4.4 Room-assignment model

The second model receives the timetable C and decides $SAO_{h,s}$, the subject taught in room s at hour h (zero when the room is free). With Sc_s the capacity of room s , Sr_s its resource type, Ae_a the enrollment cap of subject a , and Ar_a its required resource:

$$C_{h,a} \rightarrow \sum_{s=1}^S [SAO_{h,s} = a] = 1 \quad \forall h, \forall a \quad (4)$$

$$SAO_{h,s} = a \wedge C_{h,a} \rightarrow Sc_s \geq Ae_a \\ \wedge Ar_a = Sr_s \quad \forall h, \forall s, \forall a \quad (5)$$

Equation (4) places every scheduled subject in exactly one room; Eq. (5) matches capacity and resources. A continuity constraint keeps consecutive hours of the same subject in the same room within a day. The objective maximizes the number

of rooms left completely free over the week, so the campus keeps slack for unplanned events.

5 Tool

The models run behind a web application, split across two repositories (Velasquez Erazo 2022a,b) and sketched in Fig. 2. We describe it because the interface, not the model, is where a formulation meets the person who has to use it, and the design decisions visible in the code are the ones that decide whether a solver ever gets invoked.

The front end is a React single-page application on Firebase, which supplies email-and-password authentication, the user store, and hosting. Eight components carry the whole workflow: sign-up and login, a profile page, a menu, a landing page, user administration for the account that manages the others, a help page, and the upload form. That last one is the operational core, and it is deliberately thin: two file inputs restricted to CSV, one for teacher availability and one for the semester’s academic programming, posted as multipart form data together with the requesting user’s address. Nothing else is asked of the user, and no scheduling parameter is exposed for tuning.

Two consequences of that design deserve stating. The interface offers no way to inspect or edit a constraint, so every institutional rule is fixed in the model and a change of rule is a change of

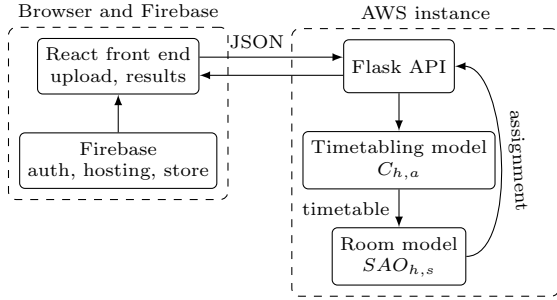


Figure 2 Deployment. The browser uploads teacher availability and the semester’s academic programming; the API runs the two MiniZinc models in sequence and returns the complete assignment.

source code. We return to that compile-time rigidity in Section 9. And because the whole request is two files, the help page carries the entire input contract: it documents the expected columns for availability and for programming, and it has to, since a malformed file fails the run outright rather than degrading.

The back end is a Flask API on an AWS virtual machine. It parses the two CSV files with pandas, writes the two MiniZinc data files, invokes the solver on the timetabling model, feeds the resulting timetable into the room model, assembles the complete assignment, and returns it both as an HTML table and as a CSV attached to an email addressed to the requesting user. That last step matters more than it looks: at the sizes reported in Tables 2 and 3, a run outlives any reasonable HTTP request, and delivering the result by mail is what makes a minutes-long or hours-long optimization usable from a browser at all.

Development followed test-driven development at the constraint level: each constraint was written against a small input with a known expected output, and regression tests over the whole model guarded against interactions between constraints. That practice is what produced the three room-stage formulations compared in Section 6, since a constraint rewritten to pass the same tests is a candidate encoding.

6 Evaluation

6.1 Model variants versus solvers

Test-driven development produced encodings that differ in form while stating the same rule, and

the original study grouped them into three variants (SALONES, SALONESCOUNT, SALONESGC). It reported Gecode as the weakest, exceeding 1800 seconds on two variants where nothing else needed more than 600, and selected SALONESCOUNT with COIN-BC. The per-cell timings behind that comparison did not survive, so we recovered the three formulations from the published backend (Velasquez Erazo 2022b), in whose source two of them survive as commented code beside the deployed one, and re-ran the comparison on the instances released here.

The three differ only in how they forbid a course group from occupying two rooms in the same hour: SALONES adds `alldifferent_except_0` over each hour’s row, SALONESCOUNT requires `count(row, a) = 1` for each scheduled group, and SALONESGC states the same through `global_cardinality`. They agreed on the objective in every run, the equivalence the original study assumed. Reconstructing them surfaced one caveat worth recording: `alldifferent_except_0` alone does not require a scheduled group to receive any room at all, so taken by itself it admits the empty assignment. It functions as a redundant constraint on top of the placement requirement, not as a replacement for it.

Table 2 gives the times, and the first row of findings corrects the deployed configuration. On the room stage COIN-BC is the weakest of the three solvers: it reaches the 240-second limit on the fifty-group instance under two of the three formulations, while Chuffed and Gecode finish the same runs in under nine seconds. The tool ships COIN-BC for both stages because it was the better choice for timetabling (Table 3); the room stage was never tuned separately. The encodings do separate, but by less than the solver does, and none dominates: SALONESGC is the only formulation COIN-BC completes at fifty groups, yet on Gecode it is the slowest at twenty-five.

Our ordering inverts the original one, but the two measure different instance families and the earlier per-cell numbers are unavailable, so they cannot be reconciled directly. The narrower claim the re-run supports is that the room stage deserves its own solver choice, and that on these instances the deployed one is the wrong choice for it.

Turning to the timetabling stage, the picture is different. Table 3 reports wall-clock times on an AMD Ryzen 5 3600 (six cores, twelve threads,

Table 2 Room-assignment stage, wall-clock seconds to a proven optimum at a 240-second limit. “limit” marks a run that reached the limit with a solution in hand; “fail” marks one that returned no parseable result at the limit, which COIN-BC does on two of the three formulations at fifty groups. Completed runs returned the same objective, five free rooms at twenty-five groups and one at fifty. One run per cell, so the seconds carry no variance estimate.

Instance	Formulation	COIN-BC	Chuffed	Gecode
syn0025	SALONES	6.4	3.6	4.1
	SALONESCOUNT	4.4	2.7	3.6
	SALONESGC	5.7	2.8	21.7
syn0050	SALONES	fail	8.4	6.5
	SALONESCOUNT	fail	5.7	6.1
	SALONESGC	14.9	4.7	241.0 (limit)

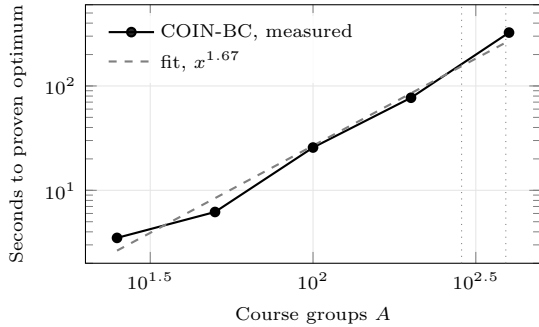


Figure 3 Measured solving time against instance size, both axes logarithmic. The fitted exponent is 1.67 (95% CI 1.28 to 2.06), below quadratic and far from the exponential the space of Fig. 1 would suggest. Dotted lines mark the real semester sizes.

31 GB) running MiniZinc 2.10.0 in a container, each instance solved once with a 120-second limit for the solver comparison and a 600-second limit for the scaling series. Gecode reaches the limit on both instances without proving optimality, while Chuffed and COIN-BC finish in seconds. COIN-BC then scales predictably: solving time grows roughly with the 1.7 power of the instance size, from 3.5 seconds at 25 course groups to 325 seconds at 400. A log-log fit over the five points gives an exponent of 1.67 (95% CI 1.28 to 2.06, $R^2 = 0.98$), so the range brackets a real campus semester at well below quadratic cost.

6.2 Scaling

On the campus instance used during development, with 215 course groups and 92 teachers over a 96-hour week, the timetabling model found a solution in 17 minutes 56 seconds. That figure sits well above the times in Table 3 for instances of comparable size, and the hardware explains most of

the gap. The tool ran on the AWS free-tier virtual machine that hosted it, a t2.micro with one core, 1 GB of RAM, and 8 GB of disk (Velasquez Erazo 2022c); Table 3 was measured on six cores and 31 GB. The comparison to draw is therefore not between two solvers but between two deployments, and it says something practical: the model is within reach of a semester’s turnaround on ordinary hardware, and the free tier was the binding constraint on the deployed tool.

Chuffed matched or beat COIN-BC on the two smallest instances and then found no feasible timetable at 100 groups or beyond within ten minutes. Gecode reached an objective of 28 at 100 groups, against the optimum of 17 that COIN-BC proved in 25.7 seconds, and found nothing at 200 or 400. At semester size the comparison is not close, and it is the MIP solver that carries the model.

The institution asked for an assignment within one day, and Table 3 answers that for the timetabling stage: at the size of a real semester it runs in minutes, not hours. The room stage does not follow. Table 2 shows COIN-BC reaching a four-minute limit at fifty groups, and extending the runs upward shows the problem is not the solver. Chuffed is the fastest of the three at fifty groups. At one hundred it exhausts a ten-minute limit, and so does every formulation at two hundred and four hundred. One hundred groups is four times smaller than a real semester.

The asymmetry is worth stating plainly, because it is the opposite of what the two stages look like. Timetabling is the stage with the astronomical space of Fig. 1 and it scales below quadratically; room assignment looks like the easy remainder and does not scale at all. The reason is structural. The timetabling model decides one Boolean per group

Table 3 Timetabling model, wall-clock seconds, one run per cell. The limit is 120 seconds at 25 and 50 groups and 600 seconds from 100 groups upward. “Limit” means the solver returned a solution without proving optimality; “none” means it returned no solution within the limit.

Instance	Groups	COIN-BC	Chuffed	Gecode
syn0025	25	3.5	2.1	120.8 (limit)
syn0050	50	6.2	5.7	121.0 (limit)
syn0100	100	25.7	none	601.6 (limit)
syn0200	200	77.1	none	none
syn0400	400	324.7	none	none

and hour, while the room model decides an integer over $A + 1$ values for every one of the $96 \times |S|$ room-hours, and its continuity constraint couples consecutive hours across the whole week. On this evidence the pipeline as built does not meet a one-day turnaround at institutional scale, and the room stage, not the timetabling stage, is where the next modeling effort belongs.

7 Characterizing the Problem from Institutional Data

This study reads two sources against each other. One is the record of what an institution did: eight consecutive semesters of schedules produced without a model, the observed schedules. The other is what the formulation produces on instances drawn from the same distributions, the simulated schedules.

Neither answers the question alone. The observed schedules show what the constraints are worth in practice but carry no counterfactual, since the institution issued one schedule per semester and no alternative. The simulated schedules supply the counterfactual, but only for instances we generated, so by themselves they say nothing about any institution. Read together they do more: where the two agree, a claim about the formulation carries to practice, and where they diverge, the divergence localizes what the model fails to represent. The comparison holds only because both are calibrated on the same distributions of intensity, shift, teaching load, and room capacity, so a difference in outcome is not a difference in the instances.

Two properties of the generated instances bound what the timings below can show. The generator seeds a complete schedule first and derives availability and room stock from it, so every generated instance is satisfiable by construction and carries a certificate to that effect. It also sizes the

room stock at 1.6 times peak concurrent demand per resource kind and gives every generated room the largest capacity observed, so the room stage faces a looser matching problem than the campus does with its fixed thirty rooms of unequal size. The timings therefore measure how the formulation behaves on feasible, loosely resourced instances of growing size. They are a lower bound on the effort a real semester demands, not an estimate of it, and the timetabling stage never runs here on an instance built faithfully from the records, because those admit no solution (Section 8.1).

We report what the instances contain, and what the institution did with them. Because the series has only eight points, we test every trend twice, with ordinary least squares and with the non-parametric Mann-Kendall test, and report a result only when the two agree. Effect sizes accompany the tests throughout. With $n = 8$ the power to detect anything short of a strong trend is low, so a non-significant result here is not evidence of stability.

7.1 Instance characteristics

Table 4 sizes each semester as an instance, in the terms a solver sees: groups to place, teachers whose availability constrains them, weekly class hours to fit, the largest number of groups running at any one hour, and the room-hour occupancy that results.

Every instance draws on the same fixed stock of thirty bookable spaces, of which four are computer rooms, three are laboratories, and one is a sports hall; capacities run from 25 to 62 seats. The thirteen institutional time bands and the six-day week are likewise constant across the series, so the only quantity that moves is demand.

Table 4 The eight semesters as instances, and their violations of the model’s hard constraints. Peak is the largest number of groups scheduled in one hour; occupancy is class hours over the thirty rooms times the 96-hour week. Violation counts are hour-level incidences.

Period	Instance					Violations			
	Grp	Tch	Hrs	Pk	Occ.%	Tch	Room	Coh	Avail
2023-I	286	107	920	22	31.9	17	17	11	26
2023-II	310	120	1030	23	35.8	3	19	7	27
2024-I	315	121	1038	25	36.0	0	22	19	15
2024-II	339	135	1133	24	39.3	3	27	29	9
2025-I	345	143	1153	25	40.0	15	41	34	38
2025-II	363	145	1219	25	42.3	9	36	15	11
2026-I	385	154	1288	25	44.7	10	38	24	17
2026-II	391	153	1312	25	45.6	0	41	4	38

7.2 Demand grows, the building does not

Demand rises monotonically on every measure. Course groups go from 286 to 391, a 36.7% increase at a Sen slope of 15 groups per semester (Mann-Kendall $p = 0.0008$; OLS $R^2 = 0.985$). Weekly class hours rise 42.6%, from 920 to 1312, and the teaching staff grows 43%, from 107 to 153. Two counts stay flat: academic programs and rooms in use (Mann-Kendall $p = 0.22$ and $p = 0.32$). The campus is teaching much more inside the same plant, so room-hour occupancy climbs from 31.9% to 45.6% ($R^2 = 0.977$).

Those occupancy figures sit inside the range reported elsewhere. A case study of one teaching building at the Harbin Institute of Technology, covering a semester of 2943 scheduled courses, reported time-utilization rates between 20% and 50% (Li et al. 2026), and a Ghanaian tertiary institution reports comparable inefficiency from a facilities-management angle (Osei et al. 2025). Our eight semesters are of similar size and land in the same band. The campus is not an outlier whose difficulties come from unusual scarcity.

The instances are not, however, reducible to the standard educational timetabling benchmarks cataloged by Ceschia et al. (2023). Three features of this campus fall outside the curriculum-based course timetabling formulation those benchmarks encode: a program is taught either wholly in the day or wholly in the evening, so a course inherits a shift and cannot cross it; rooms belong to three physically separate sub-sites; and how a course may be divided across days depends on its weekly intensity, with four-hour courses either taught in

one block or split evenly and five-hour courses split two and three. A campus instance can be translated into the standard format only by discarding those rules. We therefore release the instances in the model’s own format.

7.3 Rooms are not the binding constraint

Peak concurrent demand never exhausts the plant. The busiest hour of the week requires 22 rooms in 2023-I and 25 from 2024-I onward, against a stock of 30, so the peak occupies at most 83% of the rooms. Teaching also spreads more evenly over time rather than concentrating; the Gini coefficient of class hours per room falls steadily from 0.328 to 0.238 (Mann-Kendall $p = 0.0008$).

A direct test then bounds how much of the room conflict scarcity can explain. Across the eight semesters, 208 hours have some room hosting two groups at once, and those hours contain 241 double-booked rooms. Counting any idle room, a median of 14 of the thirty stood empty and the stock was never exhausted. That count is too generous, because the rooms are not interchangeable: five of the seven resource kinds on this campus exist as a single room, so idle general classrooms are worth nothing to a group that needs the physics laboratory. Restricting the count to rooms of the kind the clashing groups were assigned to, a median of 7 sat empty and 14.5% of the double-bookings had none free; six of those are the singleton kinds, where no substitute can exist. Adding the sub-site, since the three are physically separate and the campus reaches for the farthest only when the others fill, a median of 3 sat empty and 22.8% had

none. So scarcity of a usable room explains roughly one double-booking in four, and the other three in four happened with a suitable room standing empty nearby. Conflicts do rise with occupancy, strongly: a Poisson regression of the conflict count on the occupancy rate gives a coefficient of 0.067 (95% CI 0.036 to 0.097), a rate ratio of 1.069 per percentage point, with Spearman $\rho = 0.898$ ($p = 0.0024$). With a suitable room free in most cases, occupancy acts here mainly as a proxy for the size of the manual task rather than as a physical cause. At $n = 8$ occupancy is also collinear with every other measure of growth, so the coefficient carries a direction and not a dose. The conflicts scale with how much there is to schedule, not with how little room there is to schedule it in.

7.4 The problem keeps its shape

Instances drawn from different semesters can be pooled. The distribution of weekly intensity per course group does not differ across the eight periods ($\chi^2 = 18.5$, $df = 28$, $p = 0.91$, Cramér's $V = 0.041$), and neither does the day-evening split ($p = 0.81$, $V = 0.037$). The mix of academic programs does shift detectably ($p < 0.001$), but at $V = 0.087$ the effect is negligible: with 2734 course groups pooled the test has power to flag differences too small to change the problem. What grows is the size of the instance, not its character.

7.5 Cohorts are already scattered

A cohort, meaning one academic program in one semester, uses 1.81 buildings on average. Of the 436 cohort-periods, 78.9% have their classes split across more than one of the three sub-sites, and 27.3% of cohort-days put a cohort in more than one room on the same day. Whatever the campus optimizes when it schedules by hand, keeping a cohort in one place is not it. That bears on the room objective (Section 8).

8 Discussion: Model against Practice

8.1 The historical schedules violate the model

Auditing the eight published semesters against the model's hard constraints shows that the schedules

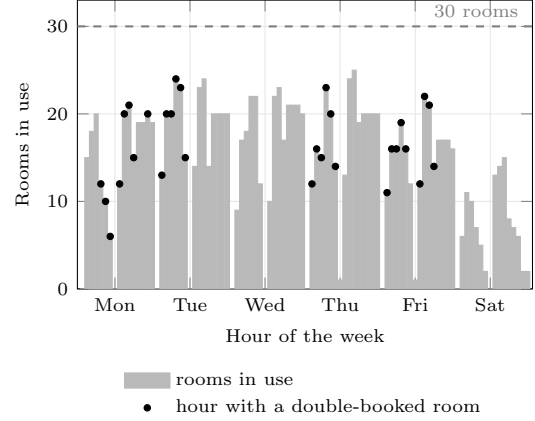


Figure 4 The schedule the institution published for 2026-II, hour by hour. Every hour in which some room hosts two groups is marked. The stock of thirty rooms is never reached: at the busiest hour twenty-five are in use, and at the marked hours between six and twenty-four rooms stood empty. The conflicts are not a consequence of a full building. Of the 241 double-booked rooms across the eight semesters, 22.8% had no room of the required resource kind free in the same sub-site.

the campus issued satisfy none of them completely: teachers appear in two rooms at once, rooms host two groups in the same hour, courses of one program and semester overlap, and classes are taught in hours the teacher never declared free. Counts range from a handful to 41 hour-level incidences per semester, and they do not move together. One of the eight had not yet begun. The institution published its 2026-II schedule before the term opened, and our snapshot predates that opening. Those conflicts are therefore in the schedule as issued, before any of the mid-term substitutions and room changes an academic secretary absorbs once a semester is running. That period ties for the most double-booked rooms of the eight, so the violations belong to the planning step rather than to the drift away from it.

Room double-bookings rise with enrollment, from 17 in 2023-I to 41 in 2026-II (Mann-Kendall $p = 0.0065$), though not in every semester. Teacher clashes show no trend at all ($p = 0.71$). That separation matters: the first count rises as a fixed stock of thirty rooms comes under pressure, while the second looks like data entry. Table 4 gives the counts beside the instance sizes, and Fig. 4 shows one semester hour by hour.

Two readings compete. Under the first, the model is stricter than the campus needs, and the constraints are correct as preferences but wrong as

necessities: rooms recorded as double-booked may host a split laboratory group, cohort courses that overlap may be electives no student takes together, and hours taught outside the declared availability may reflect a teacher who agreed after declaring. Under the second, the constraints are right and the records are inconsistent.

The evidence of Section 7 narrows this. Because a suitable room in the same sub-site was free in three double-bookings out of four, scarcity forced a minority of them; the rest were either a deliberate sharing the records do not distinguish from a clash, or a bookkeeping error. Both possibilities point away from the model’s capacity constraint and toward what the records mean. That is a question for the institution, not for the solver.

One of the four categories has to be read more carefully than the others, and the reason is instructive about institutional data generally. Availability looks like a declaration by each teacher, and we first treated it as one. It is not. Across the eight semesters, 98.0% of scheduled class hours fall inside the availability recorded for the same teacher and period; for 35% of teacher-periods the recorded availability is exactly the set of hours that teacher ends up teaching, hour for hour; and the median ratio of recorded availability to actual load is 1.20. A genuine declaration would be far looser than the schedule it constrains, and exact equality would be rare. These records are largely fitted to the assignments rather than collected before them.

Two things follow. The hours taught outside the recorded availability are a 2.0% residue in a table that otherwise mirrors the schedule, so they are weak evidence of teachers being scheduled against their stated wishes, and we withdraw that reading. And a model given this table as its availability constraint is told each teacher is free almost exactly when they teach. That over-constrains the instance severely, and it is part of why the faithful instances admit no solution. The remaining three categories are unaffected: they are computed from the schedule alone.

Neither reading is unusual. Rudová and Murray built the Purdue timetabling system on weighted partial satisfaction precisely because the university problem was not solvable with every constraint hard (Rudová and Murray 2003), and Thornton and Sattar treat over-constrained problems with a mandatory core as the normal case rather than

the pathological one (Thornton and Sattar 1998). What our data adds is a measurement of how far a working institution sits from its own stated rules.

8.2 Scoring the manual schedules on the model’s objectives

Because the objectives encode the campus’s own preferences, the manual schedules can be scored on them directly. Across eight semesters and 9 093 class hours, exactly four classes were placed in the 06:00–07:00 opening hour: the schedulers already avoid it, so that half of the objective is satisfied before optimization begins and the model should not claim it as a gain. The lunch band is different. Between 49 and 70 class hours land in 12:00–14:00 each semester, five to six percent of all teaching, and that is where the model has room to work.

The hour profile makes the two counts concrete. The opening hour is empty, while 12:00 still carries 53 class hours per semester on average, falling to 8 at 13:00. Those are the hours the model can move.

The room objective is the one where model and practice diverge most, and the divergence runs the other way. A free room is worth something to the institution: rooms released for the whole week absorb make-up sessions, external bookings, and the changes that arrive once the term starts, so the objective was elicited rather than inherited. Yet the manual schedules deliver between one and four free rooms out of thirty in every semester, with no trend either way (Mann-Kendall $p = 0.32$). Meanwhile the Gini coefficient falls ($p = 0.0008$): teaching spreads further across the stock each year rather than concentrating on fewer rooms. The campus wants the thing the objective maximizes and is not getting closer to it.

That is the clearest case in our data for automating the decision. Achieving it costs nothing in feasibility: occupancy sits between 32% and 46%, the peak never exceeds 83% of the stock, and so the rooms to be freed are not needed at the hours they would be freed. The manual process cannot see the whole week at once. An optimization over a global objective does nothing else. Table 2 shows the room stage returning proven optima at these sizes, so the gap is a matter of running the model rather than of solving a harder problem.

One caveat belongs beside it. Cohorts already cross sub-sites in 79% of cases, so concentrating teaching to free rooms may push cohorts further

apart, and the model represents no cost for that. Whether the campus would accept the trade is a question the objective as written cannot answer.

8.3 Consequences for the model

The model therefore needs a soft tier. Counting a violation instead of forbidding it turns an unsatisfiable instance into an optimization over how much the campus must bend. That version is solvable, and more informative than a bare refusal. The machinery exists: constraint solvers can return the subset of constraints that cannot hold together rather than only reporting that they cannot (Downing et al. 2015). The fifth item of our own future-work list asked for exactly that.

8.4 Decisions an institution has to make

The measurements above are not, on their own, advice. What they do is turn four choices institutions currently make implicitly into choices that can be made on evidence. We state them as decisions, with what our data settles and what it leaves open.

Which constraints are genuinely hard. The observed schedules violate every hard constraint in the formulation, and the room conflicts occur with rooms free. An institution therefore has to decide, constraint by constraint, whether a violation is an error to prevent or a practice to represent. The distinction is not cosmetic: a rule kept hard makes instances unsatisfiable and the tool useless, while the same rule made soft turns the instance into an optimization over how much the institution must bend. Our data identifies which constraints are at stake and how often; only the institution can say which reading is right, and until it does, no model of its process can be validated.

What the objective should be. Our formulation minimizes classes in two disfavoured bands and maximizes rooms left free. Measured against practice the two are unequal: the opening hour is already empty, so half the objective is free, while the free-room count is wanted and not achieved. An institution choosing objectives should test each candidate against its own history first, because an objective already satisfied costs nothing and buys nothing.

Where to spend the next effort. The timetabling stage searches a space of 10^{592} and finishes in minutes; the room stage does not reach institutional scale at all. An institution weighing more compute against a commercial system against reformulation should know that on this evidence the binding constraint is the room-stage formulation, and that neither more hardware nor a different off-the-shelf solver addresses it.

Whether the records can support a model at all. Availability declarations are contradicted by the schedule in every semester, and the enrollment caps the model needs are recorded nowhere. Before commissioning an optimization an institution has to decide whether to fix the data or weaken the model to match it. That decision precedes the choice of method and, in our case, determines whether any of the rest is worth doing.

Two limitations bound all four. The evidence comes from one institution, so the numbers calibrate rather than generalise (Section 8.5). And the observed schedules are the only ones that institution issued, so we can say what the model would change but not what the institution would have preferred: no alternative schedule was ever put in front of the people who use it.

8.5 What transfers, and what does not

One campus supplies the evidence here, so the question of scope has to be answered explicitly rather than left to the reader.

Three results do not depend on the institution. The search-space count of Section 4.1 is a property of the formulation: any model that fixes a weekly intensity, confines groups to a shift, and requires contiguous blocks will collapse its space by a comparable factor, and the arithmetic is reproducible from a model's own constraints. The two-stage decomposition and its failure mode are likewise structural. A timetabling stage blind to rooms can emit a timetable that no room assignment satisfies, and the trigger is peak concurrency per resource kind rather than total demand. That is a fact about the decomposition, not about this campus. And the audit procedure carries anywhere a scheduling system publishes its history: check the institution's own schedule against the model's hard constraints, then, for each violation, ask whether the resource in question was actually exhausted.

The measured quantities are local, and we do not claim otherwise. Occupancy between 32% and 46%, cohorts spread over 1.81 buildings, a lunch band carrying five percent of teaching: these describe one campus. Their value is as an existence proof and a calibration point. They show that an institution can violate every hard constraint of a reasonable model without ever running out of rooms, enough to make the general claim falsifiable. The occupancy range sits inside published figures for teaching buildings (Li et al. 2026; Osei et al. 2025). The campus is not an outlier. Whether the specific numbers recur is an empirical question the released instruments are meant to make cheap to ask.

Two features of this campus are unusual enough to flag. Programs are taught wholly in the day or wholly in the evening. That halves the effective hour grid per group, and the benchmark formulations do not commonly include it (Bettinelli et al. 2015). And 86 of 96 teachers were paid by the hour, so availability is the scarce resource. At an institution with mostly full-time staff the binding constraint would likely be rooms instead, and the conclusions about where difficulty sits would move accordingly.

9 Conclusions and Future Work

We formulated classroom assignment as two chained constraint models, counted the space the formulation searches, compared three encodings of the room stage across three solvers, and deployed the result as a working tool. The count matters more than the comparison: structural constraints remove roughly 7 700 orders of magnitude before search begins, and what remains is solved in minutes with growth below quadratic, so the difficulty of this problem in practice lies in how it is stated rather than in its nominal size.

Tested against eight semesters of one institution’s published schedules, the model failed in a specific and informative way. None of the eight satisfies its hard constraints, and few of the room conflicts coincide with an exhausted stock of the room kind required. That timetabling problems are over-constrained is not news, and softening constraints is the standard answer (Rudová and Murray 2003). What has been missing is the input that answer needs. A soft tier has to be told which constraints to relax and by how much.

Eight semesters of an institution’s own schedules say exactly that. The room constraint breaks in every term, and mostly without scarcity. The teacher constraint breaks without trend and looks like bookkeeping. And the two counts the model minimizes are unequal in what they can achieve. Measuring practice turns the choice of a soft tier from a judgment into a reading.

Three limitations set the next steps. Input domains are fixed at compile time, so a growing offering means editing the model rather than the data. When an instance admits no solution the tool reports only that fact, leaving the user to find the conflict, though the machinery for returning the conflicting subset exists (Downing et al. 2015). And the two-stage decomposition needs a feasibility check between stages, since the first stage cannot see the room capacity the second depends on. Whether large language models can generate such formulations, or explain their infeasibility, is the subject of a companion study.

Data and code availability. The replication package holds the eight institutional instances, the synthetic generator calibrated on them, the three room-stage formulations, and the scripts behind every table and figure in this paper. It is at <https://github.com/papers-data/classroom-assignment-replication> and archived at <https://doi.org/10.5281/zenodo.21852370>. The tool’s own source is kept separately, at <https://github.com/JAlexanderVelasquez/BackEndAsignacionSalones>. We release the instances in the MiniZinc data format the models read.

The class schedules analyzed here are published by the institution and readable without credentials. The room catalog and the availability records sit behind authentication; an author who teaches on the campus obtained them with institutional credentials. Rights in the underlying records belong to Universidad del Valle, which is also the data controller, and the campus studied here is the authors’ own. Treatment of the data follows the campus policy on personal data.

We audited the released extract before publication against the test Colombian Law 1581 of 2012 sets, which asks whether a natural person is determined *or determinable* rather than whether a name appears. The first pass failed. Dropping the name fields left the source system’s internal teacher

identifier in place, and that system’s own unauthenticated endpoint returns the identifier beside the name, as do the surrogate keys of the assignment and availability rows. Teachers therefore carry labels local to the extract and the row-level keys are gone; the instances needed no change, since they number teachers $1..n$ internally. Rerunning the analysis on the relabeled extract reproduces every figure above.

The extract is pseudonymized rather than anonymous, because a schedule is close to an identifier itself: a row’s day, hour, room, subject and group can be matched against the institution’s public schedule page. Closing that channel would destroy the analytical content, and the residual risk is tolerable only because the institution had already published those schedules. No personal datum was estimated or inferred, and no student-level data was collected. The only approximations are the enrollment cap and required resource of a course group, both properties of a course, each flagged in the instance headers. The package documents the audit in full.

Statements and Declarations

- **Funding.** The authors did not receive support from any organization for the submitted work.
- **Competing interests.** Both authors are affiliated with Universidad del Valle, the institution whose schedules this paper audits. They have no other competing interests to declare.
- **Ethics approval and consent to participate.** In 2022, three program coordinators and the academic secretary answered questionnaires about the scheduling process; the questions did not concern the respondents themselves. Participation was voluntary, and each respondent agreed to take part after the purpose of the study was explained. We did not seek review by an ethics committee.
- **Consent for publication.** [TODO: obtain the academic secretary’s written consent to publish his account in Section 3, then replace this note with the consent statement]
- **Data availability.** See *Data and code availability* above.
- **Materials availability.** Not applicable.
- **Code availability.** See *Data and code availability* above.
- **Author contributions.** J. A. Velasquez Erazo: Conceptualization, Methodology (formulation), Software, Investigation, Formal analysis, Writing – original draft. C. A. Delgado S.: Conceptualization, Methodology (audit), Investigation, Formal analysis, Data curation, Writing – original draft, Writing – review & editing, Supervision.

References

- Abdipoor S, Yaakob R, Goh SL, et al (2023) Meta-heuristic approaches for the university course timetabling problem. *Intelligent Systems with Applications* <https://doi.org/10.1016/j.iswa.2023.200253>
- Almeida MWS, Medeiros JPS, Oliveira PR (2015) Solving the academic timetable problem thinking on student needs. In: 2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA). IEEE, <https://doi.org/10.1109/ICMLA.2015.184>
- Antunes-Batista B, Atta S, Basto-Fernandes V, et al (2026) Multiobjective optimization approaches for room allocation in university course timetabling. *Procedia Computer Science* 278:663–670. <https://doi.org/10.1016/j.procs.2026.03.037>
- Ardi Nugraha T, Trinanda Putra K, Hayati N (2017) University course timetabling with genetic algorithm: A case study. *Journal of Electrical Technology UMY* 1(2). <https://doi.org/10.18196/jet.1213>
- Arratia-Martinez NM, Maya-Padron C, Avila-Torres PA (2021) University course timetabling problem with professor assignment. *Mathematical Problems in Engineering* 2021:1–9. <https://doi.org/10.1155/2021/6617177>
- Babaei H, Karimpour J, Hadidi A (2015) A survey of approaches for university course timetabling problem. *Computers & Industrial Engineering* 86:43–59. <https://doi.org/10.1016/j.cie.2014.11.010>
- Barnhart C, Bertsimas D, Delarue A, et al (2022) Course scheduling under sudden scarcity: Applications to pandemic planning. *Manufacturing &*

- Service Operations Management 24(2):727–745. <https://doi.org/10.1287/msom.2021.0996>
- Bettinelli A, Cacchiani V, Roberti R, et al (2015) An overview of curriculum-based course timetabling. TOP 23(2):313–349. <https://doi.org/10.1007/s11750-015-0366-z>
- Burke EK, Petrovic S (2002) Recent research directions in automated timetabling. European Journal of Operational Research 140(2):266–280. [https://doi.org/10.1016/S0377-2217\(02\)00069-3](https://doi.org/10.1016/S0377-2217(02)00069-3)
- Carter MW, Tovey CA (1992) When is the classroom assignment problem hard? Operations Research 40(1-supplement-1):S28–S39. <https://doi.org/10.1287/opre.40.1.s28>
- 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, Werner F, Shokouhifar M (2023) Mathematical modeling and exact optimizing of university course scheduling considering preferences of professors. Axioms 12(5):498. <https://doi.org/10.3390/axioms12050498>
- Cooper TB, Kingston JH (1996) The complexity of timetable construction problems. In: Practice and Theory of Automated Timetabling. Springer Berlin Heidelberg, https://doi.org/10.1007/3-540-61794-9_66
- Davison M, Kheiri A, Zografos KG (2024) Modelling and solving the university course timetabling problem with hybrid teaching considerations. Journal of Scheduling <https://doi.org/10.1007/s10951-024-00817-w>
- Downing N, Feydy T, Stuckey PJ (2015) Unsatisfiable cores and lower bounding for constraint programming. <https://doi.org/10.48550/arXiv.1508.06096>, arXiv:1508.06096
- Dubois B, Walsh S, Cappart Q (2026) Constraint programming for curriculum-based high school timetabling with half-blocks. In: 30th International Conference on Principles and Practice of Constraint Programming (CP 2026), <https://doi.org/10.4230/LIPIcs.CP.2026.22>
- Even S, Itai A, Shamir A (1976) On the complexity of timetable and multicommodity flow problems. SIAM Journal on Computing 5(4):691–703. <https://doi.org/10.1137/0205048>
- Ferdiana R, Ridwan N, Hidayat NF (2017) A pragmatic algorithm approach to develop course timetable web application based on cloud technology. In: 2017 7th International Annual Engineering Seminar (InAES). IEEE, <https://doi.org/10.1109/INAES.2017.8068540>
- Gu X, Krish M, Sohail S, et al (2025) From integer programming to machine learning: A technical review on solving university timetabling problems. Computation 13(1):10. <https://doi.org/10.3390/computation13010010>
- Imran Hossain S, Akhand MAH, Shuvo MIR, et al (2019) Optimization of university course scheduling problem using particle swarm optimization with selective search. Expert Systems with Applications 127:9–24. <https://doi.org/10.1016/j.eswa.2019.02.026>
- Jaengchuea S, Lohpetch D (2015) A hybrid genetic algorithm with local search and tabu search approaches for solving the post enrolment based course timetabling problem. In: 2015 7th International Conference on Information Technology and Electrical Engineering (ICITEE). IEEE, <https://doi.org/10.1109/ICITEED.2015.7408907>
- Lewis R (2008) A survey of metaheuristic-based techniques for university timetabling problems. OR Spectrum 30(1):167–190. <https://doi.org/10.1007/s00291-007-0097-0>
- Li J, Zhao W, Xue M (2026) An empirical study on assessing classroom space utilization efficiency in higher education institutions: Indicators, methodological advances, and a comprehensive analytical framework. a case study of the zhengxin building at harbin institute of technology. Buildings 16(10):1929. <https://doi.org/10.3390/buildings16101929>

- Liu Z (2013) Optimal model for university classroom assignment to reduce traffic volume. In: Third International Conference on Innovative Computing Technology (INTECH 2013). IEEE, <https://doi.org/10.1109/INTECH.2013.6653705>
- Martinez-Alfaro H, Flores-Teran G (1998) Solving the classroom assignment problem with simulated annealing. In: SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics. IEEE, <https://doi.org/10.1109/ICSMC.1998.726655>
- McCollum B (2007) A perspective on bridging the gap between theory and practice in university timetabling. In: Practice and Theory of Automated Timetabling VI. Springer Berlin Heidelberg, pp 3–23, https://doi.org/10.1007/978-3-540-77345-0_1
- McCollum B, Schaerf A, Paechter B, et al (2010) Setting the research agenda in automated timetabling: The second international timetabling competition. *INFORMS Journal on Computing* 22(1):120–130. <https://doi.org/10.1287/ijoc.1090.0320>
- MirHassani SA, Habibi F (2013) Solution approaches to the course timetabling problem. *Artificial Intelligence Review* 39(2):133–149. <https://doi.org/10.1007/s10462-011-9262-6>
- Müller T, Rudová H, Müllerová Z (2024) Real-world university course timetabling at the international timetabling competition 2019. *Journal of Scheduling* <https://doi.org/10.1007/s10951-023-00801-w>
- Murray K, Müller T, Rudová H (2007) Modeling and solution of a complex university course timetabling problem. In: Practice and Theory of Automated Timetabling VI, Lecture Notes in Computer Science, vol 3867. Springer, pp 189–209, https://doi.org/10.1007/978-3-540-77345-0_13
- Nethercote N, Stuckey PJ, Becket R, et al (2007) MiniZinc: Towards a standard CP modelling language. In: Principles and Practice of Constraint Programming – CP 2007. Springer Berlin Heidelberg, pp 529–543, https://doi.org/10.1007/978-3-540-74970-7_38
- Osei S, Quansah JYD, Mensah MA (2025) Assessing and improving teaching space utilization for operational efficiency: A study in a Ghanaian tertiary institution. *European Journal of Contemporary Education and E-Learning* 3(4):79–102. [https://doi.org/10.59324/ejceel.2025.3\(4\).06](https://doi.org/10.59324/ejceel.2025.3(4).06)
- Oude Vrielink RA, Jansen EA, Hans EW, et al (2019) Practices in timetabling in higher education institutions: a systematic review. *Annals of Operations Research* 275(1):145–160. <https://doi.org/10.1007/s10479-017-2688-8>
- Ozkan A, Ulucan A, Dirik C, et al (2025) University course timetabling with multi-section courses, room stability and lecturer preferences: an application in a business school. *Computational Management Science* 22(1). <https://doi.org/10.1007/s10287-025-00529-2>
- Pillay N (2014) A survey of school timetabling research. *Annals of Operations Research* 218(1):261–293. <https://doi.org/10.1007/s10479-013-1321-8>
- Rakotonirainy RG, Thompson A, Ghoor W, et al (2023) Honours course timetabling and classroom assignment. *ORiON* 39(2):175–193. <https://doi.org/10.5784/39-2-769>
- Rappos E, Thiémarc E, Robert S, et al (2022) A mixed-integer programming approach for solving university course timetabling problems. *Journal of Scheduling* <https://doi.org/10.1007/s10951-021-00715-5>
- Rezaeipana A, Abshirini Z, Zade MB (2019) Solving university course timetabling problem using parallel genetic algorithm. *International Journal of Scientific Research in Computer Science and Engineering* 7(5):5–13. <https://doi.org/10.26438/ijsrcse/v7i5.513>
- Rudová H, Murray K (2003) University course timetabling with soft constraints. In: Practice and Theory of Automated Timetabling IV. Springer Berlin Heidelberg, pp 310–328, https://doi.org/10.1007/978-3-540-45157-0_21

- Schaerf A (1999) A survey of automated timetabling. *Artificial Intelligence Review* 13(2):87–127. <https://doi.org/10.1023/A:1006576209967>
- Schneider D, Leuschel M, Witt T (2018) Model-based problem solving for university timetable validation and improvement. *Formal Aspects of Computing* 30(5):545–569. <https://doi.org/10.1007/s00165-018-0461-7>
- Siame A, Kunda D (2018) University course timetabling using bayesian based optimization algorithm. *International Journal of Recent Contributions from Engineering, Science & IT (iJES)* 6(2):14. <https://doi.org/10.3991/ijes.v6i2.8990>
- Soria-Alcaraz JA, Carpio M, Terashima-Marin H (2010) Several strategies to improve the performance of hyperheuristics for academic timetabling design problem. In: 2010 IEEE Electronics, Robotics and Automotive Mechanics Conference. IEEE, <https://doi.org/10.1109/CERMA.2010.23>
- Steiner E, Pferschy U, Schaerf A (2024) Curriculum-based university course timetabling considering individual course of studies. *Central European Journal of Operations Research* <https://doi.org/10.1007/s10100-024-00923-2>
- Stuckey PJ, Feydy T, Schutt A, et al (2014) The MiniZinc challenge 2008–2013. *AI Magazine* 35(2):55–60. <https://doi.org/10.1609/aimag.v35i2.2539>
- Syskov A, Rabovskaya M, Makhneva V (2020) Timetable model for individualization of education based on multi-agent approach. In: 2020 Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBREIT). IEEE, <https://doi.org/10.1109/USBREIT48449.2020.9117621>
- Thornton J, Sattar A (1998) Dynamic constraint weighting for over-constrained problems. In: *PRICAI'98: Topics in Artificial Intelligence*. Springer Berlin Heidelberg, pp 377–388, <https://doi.org/10.1007/BFb0095285>
- Velasquez Erazo JA (2022a) Asignacion-de-salones: React front end for constraint-based classroom assignment. <https://github.com/JAlexanderVelasquez/Asignacion-de-Salones>
- Velasquez Erazo JA (2022b) Backendasignacionsalones: constraint-based classroom assignment backend (MiniZinc + Flask). <https://github.com/JAlexanderVelasquez/BackEndAsignacionSalones>
- Velasquez Erazo JA (2022c) Herramienta para la asignación de salones en la universidad del valle sede tuluá. Bachelor's thesis, Universidad del Valle, Tuluá, Colombia
- Welsh DJA, Powell MB (1967) An upper bound for the chromatic number of a graph and its application to timetabling problems. *The Computer Journal* 10(1):85–86. <https://doi.org/10.1093/comjnl/10.1.85>
- de Werra D (1985) An introduction to timetabling. *European Journal of Operational Research* 19(2):151–162. [https://doi.org/10.1016/0377-2217\(85\)90167-5](https://doi.org/10.1016/0377-2217(85)90167-5)
- Zunino M, del Valle SV, Gatti L (2024) An automated approach to university course timetabling focused on professor assignment. In: 2024 L Latin American Computer Conference (CLEI). IEEE, pp 1–4, <https://doi.org/10.1109/CLEI64178.2024.10700361>