International Journal of Computational Research in Science and Technology
E-ISSN: 3108-0871, Volume 2 Issue 5
Sep-Oct 2026, PP 01-8

Smart Classroom and Timetable Scheduler: A Review of Constraint-Based, Optimization, and Hybrid Scheduling Approaches

Gunjal Pritam Sandip, Prof. Abhishek Popat Bangar, Prof. S. B. Bhosale, Prof. Karad V. A.
Dept. of Computer Engineering,
Jaihind collage of Engineering, Pune, India

Abstract. Building a college timetable takes far more than filling empty slots. Classes, rooms, teachers and students all have to fit together under firm rules and local preferences, while shared space keeps shrinking and teaching moves between in-person and online modes. This paper looks at five research papers on timetabling: a systematic review of practice in higher education, a mixed-integer programming (MIP) method, a hybrid-teaching model, a comparison of solvers from the International Timetabling Competition 2019 (ITC 2019), and a web-based Smart Classroom and Timetable Scheduler. Between them they draw on constraint validation, round-robin allocation, integer programming, matheuristics, metaheuristics and explicit modelling of online and in-person delivery. One study describes a two-stage MIP matheuristic that finished second at ITC 2019 and solved 29 of the 30 instances; it did well on small and medium cases but scaled poorly to very large ones. The ITC 2019 overview reports gaps of up to 94.46% against the available lower bounds, with five instances now proven optimal. The practical scheduler reports conflict-free timetables but gives no numerical benchmarks. Overall, the papers suggest that no single method suits every institution. A practical smart scheduler should pair checked input data and strict hard-constraint enforcement with a feasibility-first, improvement-second solving strategy, optional hybrid-teaching support, open evaluation and human review. Key open problems are scalability, benchmark coverage, fair evaluation, real-time rescheduling, and the risk of accepting a generated timetable without checking it.

Keywords: smart classroom, timetable scheduling, constraint-based scheduling, round-robin, mixed-integer programming, ITC 2019, hybrid teaching, matheuristics

I. Introduction

Timetabling means placing a given set of resources into space and time so that a collection of goals is met as well as possible, while respecting a set of constraints [1]. In a college, the resources are classrooms, teachers and time slots, and what gets placed are lectures, laboratory sessions and student groups. Done by hand, the job is slow and tends to produce faculty overlaps, room clashes, lopsided subject spread and wasted space [5]. Depending on the constraints involved, the underlying problem is NP-hard or NP-complete, so working systems usually settle for a feasible answer instead of a provably optimal one [1]. Meanwhile, institutions have less shared space to work with, students expect more flexibility, and blended and online study have become common [1], [3].

A dependable scheduler therefore has to deal with several levels of requirements instead of leaning on one rule. Constraint validation enforces hard rules such as no faculty or room clash. Balancing heuristics such as round-robin spread subjects across the week. Optimization models weigh soft preferences and student clashes, and hybrid-teaching models decide who attends in

person and who joins online. To be useful in practice, the scheduler also needs a web application around it, with data entry, role-based views and export [1], [5].

This review brings together the five supplied papers and compares their methods, reported results, limitations and what they mean for a smart classroom and timetable scheduler. It stays within the supplied sources: it is a focused narrative review, not a fresh systematic search or an independent experiment.

Review approach and scope. We summarise the five publications in narrative form and do not claim a new systematic search. The sources play different roles: one broad literature review gives the overall picture of methods and practice, two studies present an optimization method and a hybrid-teaching model, one paper compares solvers on a shared benchmark, and one describes a deployable web-based scheduler. Findings are given as the source papers state them, and numbers from different studies are not treated as directly comparable.

Each study was read for what it schedules, how it represents the problem, which solution method it uses, what evidence it offers and where it falls short. We did not repeat the experiments, re-run any solver or check the original datasets. The architecture and recommendations that follow are therefore a synthesis of the supplied evidence and not a newly tested implementation.

Table 1. Scheduling layers and their complementary roles

Scheduling layer	Typical mechanism	Strength / limitation
Constraint validation	Check faculty, room and hour rules before each assignment	Produces conflict-free output; needs precise constraint data and is hard to scale [5].
Round-robin heuristic	Assign subjects one after another across slots	Simple and balanced; ignores constraints unless paired with validation [5].
Mixed-integer programming	Binary variables for time, room and student allocation	Handles students, rooms and soft penalties; model size grows very large [2].
Matheuristic	Fix most variables, then solve smaller MIPs	Practical for small and medium instances; weak with tens of thousands of students [2], [4].
Metaheuristic / hyper-heuristic	Local search, annealing, evolutionary search	Strong on benchmarks; hyper-heuristics aim at general methods [1].
Hybrid-teaching model	Online space treated as an unlimited-capacity room	Eases pressure on physical rooms; adds variables and objectives [3].

II. Reviewed Studies and Approaches

1. Systematic Review of Timetabling Practices in Higher Education

Oude Vrielink, Jansen, Hans and van Hillegersberg carried out a systematic review that sets timetabling research beside timetabling practice in higher education [1]. They follow how solution methods developed over time: graph colouring, linear and

integer programming and constraint-based techniques in the early work; tabu search, simulated annealing and genetic, memetic and ant algorithms from the mid-1990s; and more recently hyper-heuristics, which search over heuristics rather than over solutions. They also trace how software moved from early mainframe programs to tools with databases, editors, web interfaces and better graphics. The authors point to a gap between academia and industry: commercial products often still use algorithms from the late 1980s or 1990s and put their effort into the user interface, and a modern system able to timetable a whole university while measuring user satisfaction is rare [1].

This paper supplies the broad background for our review: a scheduler should join a usable application to a sound solving method instead of relying on the interface alone.

2. Mixed-Integer Programming and Two-Stage Matheuristic

Rappos and colleagues formulate the ITC 2019 problem as a mixed-integer program that is too big to solve in one go, so they solve it in two stages [2]. Binary variables stand for class time, class room, student-to-class assignment and student-to-configuration assignment. Eight basic constraints cover points such as one time and one room per class, class capacities, room unavailability and no double booking of a room. Distribution constraints are written in a similar pairwise form, and the few that cannot be written that way are checked through lazy constraints [2]. The first stage reaches a feasible timetable by adding constraints gradually while fixing a share of the variables (typically 50–80%). The second stage repeatedly solves smaller MIPs to improve the objective while staying feasible [2].

The authors report second place at ITC 2019, with 29 of the 30 instances solved before the deadline. Results were good on small and medium instances (gaps of 0–2% to the best known solution on several), but the method scaled badly to very large ones, with a 378% gap on the largest instance, mainly because student-clash constraints are generated for each student [2]. In one instance, reaching a feasible solution took more than 10 days. These scores come from the competition setting and should not be assumed to carry over to every institution.

3. Timetabling with Hybrid Teaching

Davison, Kheiri and Zografos extend the university course timetabling problem to cover hybrid teaching [3]. A class may be held in person, online, or in both modes at the same time. The online space is modelled as an always-available room of unlimited capacity, only suitably equipped rooms can host hybrid classes, students may state a preferred mode, and travel time depends on how a student attends [3]. The model is a multi-objective binary program that maximises the elective module requests met, minimises departures from mode preferences and minimises student scheduling problems, solved lexicographically in a chosen order.

Experiments on four ITC 2019 instances, using student subsets, a 75% cut in physical room capacity and the Gurobi solver, show that results depend on both the data and the order of the objectives. Students with no mode preference tend to be placed online, which relieves physical rooms [3]. The authors note that the exact method was only tested on small instances and that more efficient solution methods are still needed, so the findings should be read within this experimental setting.

4. Comparative Analysis of Solvers at ITC 2019

Müller, Rudová and Müllerová describe how ITC 2019 was organised and compare the published solvers [4]. The thirty instances come from real UniTime installations and range from roughly 400 to almost 9,000 classes, from 15 to 768 rooms and from no students to more than 38,000, combining student sectioning with time and room assignment. The comparison covers matheuristic, constraint-based metaheuristic, simulated annealing and MaxSAT solvers. Five instances are now proven optimal, while gaps to the available lower bounds reach 94.46% on the largest instance [4].

The paper also raises a practical warning: computing demands differ greatly. The winning matheuristic ran on machines with up to 756 GB of RAM, the second-placed MIP solver used 32 GB, and the UniTime solver produced its results within 2 hours

using 4 GB per run and did better on the very largest instances [4]. A scheduler should therefore be judged on resource use as well as solution quality.

5. Smart Classroom and Timetable Scheduler

Renuka and colleagues present a web-based scheduler that pairs constraint-based validation with round-robin allocation [5]. Laboratory sessions go in first, in consecutive slots. A subject pool is then built from weekly hours and subjects are assigned to slots one after another. Every assignment is checked against the hard constraints (no faculty overlap, no classroom clash, fixed subject hours), and another subject is tried when a conflict appears. Subjects that cannot be placed are flagged for review. The system is built with React.js, Node.js with Express.js and MongoDB, gives administrators, faculty and students separate dashboards with role-based access control, and exports timetables to PDF and Excel. The reported time complexity is $O(D \times S \times N)$ for D days, S slots and N subjects, and the space complexity is $O(N)$ [5].

The main contribution is the practical packaging: data entry, validation, regeneration after changes and role-based views. On the other hand, the results are qualitative, with no benchmark data, runtimes or comparison figures. The authors themselves admit that the timetable may not be globally optimal, that support for several departments is limited and that real-time rescheduling is minimal [5]. The results belong to this lightweight, small-scale setting and should not be generalised to large institutions.

III. Comparative Discussion

The five studies show complementary layers of a scheduling system. The systematic review [1] gives the overview of methods and practice; the MIP approach [2] demonstrates solving in a feasibility-then-improvement sequence; the hybrid-teaching study [3] extends the model to online and in-person delivery; the ITC 2019 comparison [4] shows how varied the solvers are and how much computing they need; and the smart scheduler [5] shows how a simple method can be packaged as a usable web application.

Constraint validation with round-robin allocation is still a quick way to get a conflict-free baseline, but it does not optimise soft preferences or student clashes. Optimization-based methods can handle student sectioning, travel time and weighted penalties, whereas heuristic and metaheuristic methods scale better to very large instances. Matheuristics need careful decomposition once the number of students becomes very large. For flexible delivery, modelling the online space explicitly can reduce pressure on rooms. Across all of these, the reviewed literature supports a layered design more than it supports reliance on one universal algorithm.

The reported numbers cannot be compared directly, because the studies differ in task, datasets, hardware and metrics. For instance, the penalties and gaps in [2] and [4] are measured on ITC 2019 instances, [3] reports objective values on modified subsets of those instances, and [5] gives no numerical results. For that reason, this review does not rank the approaches by placing their scores side by side as though they measured the same thing.

The practical lesson is that feasibility and optimization should be handled separately. A first stage finds a timetable that meets every hard constraint, and a second stage raises its quality without losing feasibility [2]. A single quality score can hide whether a timetable is poor because of student clashes, room preferences or broken soft constraints, so reports should show what each component contributes to the penalty.

A second lesson concerns weights and time limits. Institutions care about different things (time preferences, soft distribution constraints or student conflicts), and the ITC 2019 weights were adjusted to reflect such needs [4]. The ordering of objectives in [3] also changes the outcome. The right setting depends on the institution and on what a poor timetable would cost it; the reviewed papers do not establish universal weights or time limits.

Table 2. Comparison of the five reviewed sources

Study / method	Evidence reported	Implication
Oude Vrielink et al. [1] Systematic review	Chronology of algorithms and applications; academia–industry gap.	Combine modern solving methods with a usable interface.
Rappos et al. [2] Two-stage MIP	Second at ITC 2019; 29 of 30 instances solved; gap up to 378% on the largest.	Staged solving works; large student counts need grouping.
Davison et al. [3] Hybrid teaching	Lexicographic results on four modified ITC 2019 instances.	Hybrid delivery can be modelled; objective order matters.
Müller et al. [4] ITC 2019 comparison	30 instances; five proven optimal; gaps up to 94.46%.	Use shared benchmarks; compare resource demands.
Renuka et al. [5] Constraint + round-robin	Qualitative results; O(D × S × N) time complexity.	Easy to deploy; no optimality guarantee or benchmark evidence.

IV. Proposed Design for a Smart Classroom and Timetable Scheduler

Drawing on the reviewed evidence, a practical scheduler can be organised into the following stages:

- Data collection and validation: gather subjects, faculty, classrooms, laboratory needs and, where relevant, student demand through a web interface; check for missing values, inconsistent weekly hours and invalid assignments before scheduling [5].
- Preprocessing and reduction: remove variables whose values are forced, drop constraints that always hold, merge similar constraints and share common time patterns to keep the model small [2].
- Special-session placement: place sessions that need continuous blocks, such as laboratories, before regular classes [5].
- Feasible timetable construction: assign classes while checking hard constraints at every step, using balanced round-robin distribution, and flag anything that cannot be placed [5]; for larger instances, add constraints gradually [2].
- Improvement and optional extensions: improve the timetable with local search or fix-and-optimize while keeping it feasible [2]; where needed, add student sectioning and online or in-person mode allocation [3].
- Report, review and regeneration: show conflicts, let administrators adjust inputs and regenerate, provide role-based views and export to PDF and Excel [5].

This is a proposed synthesis of the supplied literature and not a system that has been built or experimentally validated here. The design follows the review's evidence that simple validation, staged optimization and a usable interface can complement one another [1], [2], [5].

Evaluation framework. A practical evaluation should report hard-constraint violations (which should be zero), the total weighted penalty and its components, and the gap to a best-known solution or lower bound where one exists, together with runtime and memory use. Results should also name the benchmark or dataset, the hardware, the time limit, and whether the weights match those used in other studies [2], [4].

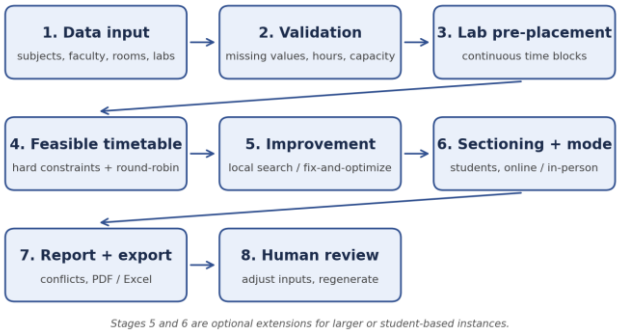


Fig. 1. Proposed layered architecture for a smart classroom and timetable scheduler (conceptual; not experimentally validated).

Recommended validation protocol. First, use shared benchmark instances where possible (ITC 2019 offers thirty, plus a solution validator [4]) and keep hardware and time limits comparable. Second, include small, medium and large instances, as well as instances with different emphasis (student clashes, time preferences, distribution constraints). Third, check hard-constraint satisfaction with an independent validator. Fourth, test regeneration after changes such as a teacher becoming unavailable or a room change. Finally, have trained administrators and faculty inspect a sample of generated timetables before deployment.

Table 3. Suggested evaluation measures for future systems

Measure	What it tells the evaluator	Caution
Hard-constraint violations	Whether the timetable is usable at all	Must be zero; report classes that could not be placed.
Weighted penalty	Overall quality under institutional weights	Compare only when weights and definitions match.
Gap to best known / lower bound	Distance from the best available solution	Lower bounds exist only for some instances [4].
Runtime and memory	Practicality on real hardware	Hardware differs widely between studies [4].
User satisfaction review	Whether administrators, faculty and students accept the result	Needs surveys or field trials; rarely reported.

Challenges and Research Gaps

First, scalability remains the central difficulty: the MIP approach performs poorly when the number of students is very large [2], and the lightweight scheduler is expected to run longer as subjects, departments and constraints grow [5]. Second, no single dataset represents every institution, and many studies rely on institution-specific data, which limits generalisation [1]. Third, evaluation should report several measures and comparable resources instead of one score. Fourth, feasibility can be hard to reach when many distribution constraints combine with many classes or very high room utilisation [2], [4], so systems should report infeasibility clearly. Fifth, a generated timetable should not be treated as final; scores and outputs need review. These concerns match the gaps and recommendations raised in the reviewed studies [1], [4].

Future work should test practical schedulers on shared benchmarks; add grouping of students for very large instances [2]; develop efficient methods for hybrid-teaching models, including fairness between students and limits on switching between

online and in-person classes [3]; combine feasibility and improvement stages in a hierarchical multi-objective formulation [2]; and report quality, resource use and user outcomes together.

Operational and governance considerations. Institutions should decide who may edit constraints, who may approve a timetable, and how changes are recorded. The scheduler should keep an audit trail of input data, constraint settings, weights and generated versions. A generated timetable should be presented as a proposal awaiting approval and not as an automatic decision, and faculty and students should be able to view their schedules and report problems.

Hybrid and flexible delivery. Hybrid and blended delivery changes what a timetable has to handle: online classes are not tied to room capacity or travel time the way in-person classes are, and only suitably equipped rooms can host hybrid sessions [3]. Systems that support blended learning should record room capabilities and student mode preferences explicitly and report mode-related results separately from room and time results.

Data and constraint handling. A scheduler is only as good as its data. If faculty assignments, room capacities or subject hours are wrong, even a strong algorithm will produce a poor timetable. Data should be stored in a structured form, shared time patterns should be reused, pre-calculated information such as room travel times should be kept, and constraints should be configurable to local rules [2], [4], [5].

Preprocessing and input validation. Input validation should check for missing values, subject hours, faculty assignments and classroom capacities before scheduling [5]. Reductions such as fixing forced variables and removing constraints that always hold can shrink the problem considerably [2], but they should be tested to make sure they do not discard valid solutions.

Interpretable timetable reports. A useful report should identify unplaced subjects, show each conflict and its reason, indicate which soft constraints were violated, and separate the time, room, distribution and student components. Where several objectives are used, the interface should explain them one by one instead of showing a single opaque score [3], [4].

Human-centred workflow. The system should support administrative judgement, not replace it. An administrator can adjust inputs, relax constraints, regenerate a timetable and approve it. Where the consequences are serious, a second review is appropriate. Such safeguards fit the observation in [1] that user satisfaction is rarely measured in large systems.

Deployment and maintenance. After deployment, the scheduler should be monitored for changes in courses, rooms, curricula and delivery modes. Changes to constraints, weights or algorithms should be versioned and tested before release, and evaluation should be repeated after major updates, because a result on one benchmark or at one institution does not guarantee the same performance elsewhere [1], [4].

V. Conclusion

The reviewed research shows that smart timetable scheduling works best as a combination of methods and not as a single algorithm. Constraint validation with balanced allocation offers a simple, usable and conflict-free baseline for institutions with modest needs; staged MIP and fix-and-optimize methods deal with student sectioning, travel time and soft preferences on small and medium instances; heuristic methods scale better to very large ones; and hybrid-teaching models can ease pressure on physical rooms. The studies report encouraging results, but differences in tasks, hardware and evaluation conditions limit direct comparison, and the practical scheduler reports no benchmark figures. A dependable scheduler should combine validated data, complementary solving stages and open evaluation on representative benchmarks, and it should leave administrators in charge of consequential decisions.

References

1. R. A. Oude Vrielink, E. A. Jansen, E. W. Hans, and J. van Hillegersberg, “Practices in timetabling in higher education institutions: a systematic review,” Annals of Operations Research, vol. 275, pp. 145–160, 2019. doi: 10.1007/s10479-017-2688-8.
2. E. Rappos, E. Thiémard, S. Robert, and J.-F. Hêche, “A mixed-integer programming approach for solving university course timetabling problems,” Journal of Scheduling, vol. 25, pp. 391–404, 2022. doi: 10.1007/s10951-021-00715-5.
3. M. Davison, A. Kheiri, and K. G. Zografos, “Modelling and solving the university course timetabling problem with hybrid teaching considerations,” Journal of Scheduling, vol. 28, pp. 195–215, 2025. doi: 10.1007/s10951-024-00817-w.
4. T. Müller, H. Rudová, and Z. Müllerová, “Real-world university course timetabling at the International Timetabling Competition 2019,” Journal of Scheduling, vol. 28, pp. 247–267, 2025. doi: 10.1007/s10951-023-00801-w.
5. D. Renuka, B. Madhava Rao, R. Ajay, and R. Enosh Anshul Benjamin, “Smart Classroom and Timetable Scheduler,” International Journal of Data Science and IoT Management System, vol. 5, no. 2(3), pp. 357–364, 2026.