Constraint Programming for Curriculum-Based High School Timetabling with Half-Blocks

Bérénice Dubois
Polytechnique Montréal, Montreal, Canada

Stephen Walsh
Dash Computer Solutions, Montreal, Canada

Quentin Cappart
UCLouvain, Louvain-la-Neuve, Belgium
Polytechnique Montréal, Montreal, Canada
CIRRELT, Montreal, Canada

Abstract

In high school timetabling, some institutions adopt a university-like structure in which students follow different curricula and multiple sections are offered for each course, without pre-assigning students to specific sections. In such settings, student sectioning and timetabling are strongly interdependent. Timetabling is usually carried out with integer linear programming or metaheuristics, before assigning students to sections on the fixed schedule. However, several Canadian high schools construct their schedules according to predefined block patterns designed to ensure a balanced distribution of instructional time. While this structure eliminates many classical timetabling constraints, it introduces a specific challenge: courses occupying half-blocks must be paired to form full blocks. This allows students to have the rest of their courses on well balanced, full blocks, thus easing student mixing between sections for improved student sectioning. The pairing of half-block courses is a critical step, as it directly impacts the balance of students across sections. Based on this context, this paper introduces a constraint programming approach to generate a feasible half-block courses schedule leading to good student balancing. The model is designed to enhance computational efficiency, accommodate additional practical constraints, and reduce the need for manual adjustments by scheduling operators. The approach is evaluated on anonymized real-world data from Canadian high schools for the 2024-2025 academic year, provided by Dash Computer Solutions, a company responsible for the annual design of many high school schedules in Quebec, and is integrated with their existing student sectioning algorithm. Our results show that our approach provides solutions of comparable quality to a human-guided local search with a hand-crafted initial solution, while reducing the designing time from 10 hours to an hour in the best case.

2012 ACM Subject Classification Computing methodologies → Artificial intelligence

Keywords and phrases High school timetabling, curriculum-based timetabling with blocks

Digital Object Identifier 10.4230/LIPIcs.CP.2026.22

Supplementary Material
Dataset: `https://github.com/corail-research/half-block-scheduling-instances`

Funding This project was funded through a Mitacs Accelerate grant (IT45197), in collaboration with IVADO and Solutions Informatiques Dash Inc.

1 Introduction

In the province of Quebec, Canada, public high schools can have a university-like structure: students can sit in any groups in accordance with their course selection and mix between sections. Each unique course-selection combination forms a *curriculum*. In secondary 3 to 5 (roughly for students between 14 to 17 years old), students are promoted by course, meaning that they may have to retake one or more individual courses from the previous year [9]. For

these grade levels, students usually have more choices of courses to take, such as option classes or advanced sciences courses. This contributes to increasing the number of curricula and the mixing of students, and therefore increasing the difficulty to design appropriate schedules.

In these schools, each course section is preassigned a specific teacher and classroom before the timetable is constructed. Students also select their courses in advance. As a result, the overall problem can be decomposed into two interrelated subproblems. The first is the *timetabling problem*, which consists of assigning course sections to periods while respecting the predefined teacher and room allocations. The second is the *student sectioning problem*, which assigns students to course sections in accordance with their course selections. The resulting timetable must satisfy all hard constraints while also supporting an effective distribution of students, ensuring that the sections of a given course have similar enrollment sizes.

Focusing on the timetabling problem, the schools considered adopt a pattern-based schedule known as a *block schedule*. In this structure, the timetable is organized as a cycle of 6 to 9 school days repeated throughout the year. The cycle is also divided into 6 to 9 blocks. A *block* corresponds to a predefined set of meeting periods distributed across the timetable cycle. Courses are scheduled by assigning them to blocks (or portions of blocks) rather than to individual periods. According to the timetabling formulations described by Ceschia et al. (2023) [4], the problem considered here is similar to a university course timetabling problem (CTT), more specifically a *curriculum-based course timetabling problem* (CB-CTT). Although scheduling occurs after student enrollment, the block structure results in constraints being defined at the course level rather than at the individual lecture level.

In most CB-CTT problems described in the literature, all hard constraints must be satisfied while a weighted sum of unsatisfied soft constraints is minimized. Hard constraints prevent infeasible situations, such as teacher schedule conflicts. Soft constraints aim to improve timetable quality, for instance by reducing gaps in students' schedules, limiting the number of daily lectures, or assigning all lectures of a course to the same room [2]. Existing approaches for solving the CB-CTT formulation mainly rely on metaheuristics [19, 12, 10], hyper-heuristics [15], mixed-integer programming [5, 1], and hybrid approaches [2] reflecting the large scale and computational complexity of these problems. However, many soft constraints typically considered in CB-CTT formulations become irrelevant when applied to this specific case of high school timetabling. Students in these schools follow a full schedule, which eliminates the possibility of free periods. Also, the predefined block structure ensures that each course section is scheduled at most once per day, thereby preventing double lectures. Furthermore, room suitability and travel distance [3] are not major concerns in this context, since preassigned rooms are only changed in the event of exceptional conflicts. Solving approaches for pattern-based timetabling methods include mixed-integer programming [17] and column generation approaches using Dantzig–Wolfe reformulation [1]. While these approaches help mitigate violations of soft constraints, they generally explore a much larger set of possible schedules than those arising in the block scheduling context considered here.

While the block structure may lead to high-quality timetables, it introduces a specific challenge: courses occupying half-blocks must be matched to form full blocks. A *half-block* is a subset of periods within a block used for most courses that require less instructional time. The motivation behind the use of half-blocks is that they increase the granularity level when building the schedule, allowing the remaining courses to be scheduled on well-balanced full blocks. Schools can therefore adjust the length of class periods so that courses requiring less instructional time are scheduled for exactly half of the periods within a block. This structure facilitates the mixing of students across sections and therefore improves the effectiveness

of the student sectioning process. However, the pairing of half-block courses is a critical step, as it directly impacts the balance of students across sections. Courses occupying half-blocks are most often elective courses, such as arts and STEM courses. These courses account for most of the variations in student curricula and are generally the most difficult to schedule. Moreover, the student sectioning problem has been shown to be NP-hard when more than one curriculum is involved [8]. Student sectioning can be performed before, during, or after the timetabling process [8, 18]. Some approaches also adopt an iterative strategy, alternating between timetabling and student sectioning stages in order to progressively refine the solution [16, 14]. Balancing course sections is important to ensure that room capacity limits are respected. It is also essential from an educational perspective, as highly imbalanced classes may disadvantage students and place additional burden on teachers. The practical importance of maintaining balanced sections, combined with the computational difficulty of the student sectioning problem, makes this problem particularly relevant to study.

Contributions. Our work is motivated by this practical context. *Dash Computer Solutions* is a company responsible for the annual design of more than a hundred schedules in Quebec. Currently, these schedules are built manually, based on the expert knowledge accumulated by operators over the years. The process typically begins with a hand-crafted initial solution, which is then iteratively improved through locally designed adjustments performed by human operators. Although this approach is capable of producing high-quality schedules, it is time-consuming: generating a schedule can require more than ten person-days per school, excluding additional time spent on maintenance, data processing and other operational tasks. Our work aims to improve this process. We propose a constraint programming (CP) approach to generate feasible schedules for half-block courses that promote effective student balancing. In particular, the model incorporates student course selections in order to favor well-balanced student sectioning. The proposed model is designed to improve computational efficiency, accommodate additional practical constraints, and reduce the need for manual adjustments by scheduling operators. The approach is evaluated on four anonymized real-world datasets from Canadian high schools for the 2024-2025 academic year, provided directly by Dash Computer Solutions, our industrial partner. The results show that our approach produces solutions of comparable quality to those obtained through a human-guided local search starting from a hand-crafted initial solution, while reducing the designing time from 10 hours to an hour in the best case. Finally, the proposed approach has been integrated with their existing student sectioning algorithm and will be used starting in 2026 to generate schedules for the upcoming academic years.

Related timetabling tasks with CP. This paragraph reviews CP approaches that, while relevant, cannot be directly applied to the specific context considered in this work. Dimitsas et al. (2022) [7] address a post-enrollment timetabling problem of a scale comparable to those encountered in Quebec. Their approach relies on a mixed-integer programming model strengthened by CP, which is used to search for local improvements and to verify non-linear constraints. Nogareda and Camacho (2017) [14] propose an iterative approach in which the timetabling problem is solved using a CP model and the student sectioning problem is addressed using an ant colony metaheuristic. However, both of these hybrid approaches operate at the level of individual lectures, whereas the problem considered in this work is defined at the course-block level. Also using CP, Demirović and Stuckey (2018) [6] present a complete model with warm starts for high school timetabling, achieving good results on instances of similar size to those considered in this work. In their setting, students are

already assigned to sections, and therefore the model addresses only the timetabling problem and does not consider the student sectioning component. Finally, Valouxis and Housos [20] propose a hybrid approach that iterates between generating a timetable using a CP model and reducing the search space through local search around the obtained timetable. While this method yields good results, it has been evaluated on smaller instances and operates at the level of individual course periods.

Structure of the paper. The paper is organized as follows. Section 2 provides a detailed description of the problem. Section 3 presents the proposed CP model, and Section 4 reports and discusses the results.

2 Problem description

High schools with a sufficient number of students may offer multiple sections of the same course. In this context, a specific section of a course is referred to as a *course-group*. A course-group is identified by the course code and a section number. For example, if two sections of an art course are offered in Secondary 3, they may be labeled `ART302-01` and `ART302-02`. In high school timetabling, the periods assigned to a course-group should be well distributed throughout the timetable cycle. In addition, the total instructional time for each course must comply with the requirements set by the governing board [11]. To satisfy these constraints, many schools adopt a timetable cycle typically spanning 6 or 9 days, during which students follow a full schedule with no free periods. The cycle is then divided into predefined fixed patterns, called *blocks*, as illustrated in Table 1. In this example, there are 9 blocks (`A, B, C, D, E, F, G, H, J`). Each block contains an even number of meeting periods that are well distributed across the cycle. This configuration, with a 9-day timetable cycle and a fixed block pattern, is commonly observed in high schools in Quebec.

Table 1 Block pattern for a timetable cycle of 9 days and 4 periods per day. A course-group scheduled on block `A` will be given at period 1 on day 1, period 2 on day 3, period 3 on day 5 and period 4 on day 7.

Period	D1	D2	D3	D4	D5	D6	D7	D8	D9
1 (8:30am to 9:30am)	**A1**	E1	J1	D2	H2	C3	G3	B4	F4
2 (9:45am to 10:45am)	B1	F1	**A2**	E2	J2	D3	H3	C4	G4
3 (1:00pm to 2:15pm)	C1	G1	B2	F2	**A3**	E3	J3	D4	H4
4 (2:30pm to 3:45pm)	D1	H1	C2	G2	B3	F3	**A4**	E4	J4

Introducing half-blocks. With this structure, the timetabling task consists of assigning course-groups to blocks. Most courses require a number of periods corresponding to a regular block, which represents four periods in the example shown in Table 1. However, some courses require fewer or more meeting periods. For course-groups that require fewer periods, a half pattern will be used. Courses assigned to half-block patterns are frequently elective courses such as arts or sports. These courses are scheduled on *half-blocks* and are referred to as *half-block courses*.

These half-block courses are typically designed so that they can be combined to form a complete block. For example, in Table 1, a student could take an *arts* course-group in block periods `A1` and `A3` and a *sports* course-group in block periods `A2` and `A4`, thereby completing block `A`. This structure ensures that half-block courses remain evenly distributed across the

timetable cycle. In this example, periods A1 and A3 correspond to the first half-block of block A, denoted A13, while periods A2 and A4 correspond to the second half-block, denoted A24. The half-block A24 is the *complement* of A13, and vice versa. Half-blocks such as A13, B13, and C13 are referred to as *odd blocks*, because they contain periods with an odd number, whereas the other half-blocks (A24, B24, and C24) are referred to as *even blocks*.

With this structure, the timetable can be represented as columns of clusters of course-groups assigned to the same half-block, as illustrated in Table 2. Each student is then assigned to exactly one course-group per half-block (e.g. ART302 to half-block A13). In this example, two thirds of the students would take a combination of arts (ART302) and sports (SPO302) in block A, and history (HIS304) in block B, while the remaining third would follow the inverse schedule. Note that the specific assignment of arts sections (i.e. ART302-01, ART302-02, and ART302-03) to half-blocks does not affect student sectioning, since students can move freely between sections. Moreover, these sections are taught by the same teacher and take place in the same room. As a result, permutations of these sections correspond to *symmetrical solutions*.

Table 2 Timetable example for one curriculum: the × mark shows the course-group assigned to the corresponding half-block. All the course-group seat 32 students.

	students	A13	A24	B13	B24	C13	...	teacher	room
ART302-01	32	×						t1	r1
ART302-02	32		×					t1	r1
ART302-03	32			×				t1	r1
SPO302-01	32	×						t2	r2
SPO302-02	32		×					t3	r2
SPO302-03	32				×			t3	r2
HIS304-01	32	×	×					t4	r3
HIS304-02	32			×	×			t4	r3
HIS304-03	32			×	×			t5	r4

Introducing multiple curricula. In this example, all students take ART302, SPO302, and HIS304, meaning that there is only one *curriculum*. However, when multiple curricula are present, the pairing of half-block courses becomes a critical step, as it directly affects the balance of students across sections. Table 3 illustrates a case with two different curricula: all students take a sports course (SPO302), but they also choose either a music course (MUS302) or a visual arts course (ART302). Students' course selections are known in advance and remain fixed. Table 3 also shows an example with a good balance in the number of students per section. In practice, grouping half-block courses into as few blocks as possible has been observed to improve the balance of students across sections. Additionally, this structure benefits the school operationally, as it makes it easier to accommodate changes in a student's course selection during the year.

Introducing conflicts and closed schedules. Unfortunately, the timetable shown in Table 3 is infeasible because there is a room conflict between the first two sports groups. Moving one of these groups to half-block A13 would not resolve the issue, since all the remaining students taking the art and music options in A13 would then be assigned to the remaining sports section in A24, because of the block-pairing constraint. This limitation arises because the students must follow a closed schedule. A *closed schedule* is a schedule for which all half-blocks courses are matched with a course on the complementing half-block.

Table 3 Example for 2 curricula: the × mark shows the course-group assigned to the corresponding half-block. Note that there is a room conflict between course-groups `SPO302-01` and `SPO302-02`.

	students	A13	A24	B13	B24	C13	...	teacher	room
ART302-01	34	×						t1	r1
ART302-02	30			×				t1	r1
ART302-03	30				×			t1	r1
MUS302-01	20	×						t2	r2
SPO302-01	27		×!					t3	r3!
SPO302-02	27		×!					t4	r3!
SPO302-03	30			×				t4	r3
SPO302-04	30				×			t4	r3

Forbidding full-block splitting. Splitting a full-block course across multiple half-blocks is theoretically possible, but it is generally discouraged. The first reason is that breaking a block increases the risk of creating poorly distributed schedules, potentially resulting in two meeting periods of the same course occurring on the same day, which is commonly forbidden. Another reason is that keeping full-block courses on their intended base pattern simplifies the management of rooms and teacher workloads, as these resources can be more easily allocated and adjusted. It also improves student sectioning, particularly when students are repeating courses from a previous grade, as sections of the same course from different grades can then be aligned more easily. Finally, doing so helps prevent conflicts with other parts of the timetable that will be assigned later. As commonly done, schools may also impose additional specific constraints. For example, they may require that two courses have at least one concurrent section in order to facilitate changes in students' course selections.

Summary. The problem considered in this paper can now be summarized as follows. The objective is to determine clusters of half-block course-groups that allow students to be distributed as evenly as possible across sections, while keeping the flexibility to potentially accommodate additional school-specific constraints. All solutions must satisfy the following hard constraints:

1. A teacher cannot teach more than one course at the same time.
2. A room cannot be assigned to more than one course-group at the same time.
3. A student cannot attend more than one course in the same half-block.
4. The number of blocks used in the timetable cannot exceed the fixed number defined by the block pattern (e.g. 9 blocks in Table 1).

3 Designing the CP model

High schools in Quebec typically involve, for grade levels secondary 3 to 5, between 140 and 1350 students, 30 to 180 different courses, 6 to 60 possible curricula, and 6 to 14 half-blocks. Considering the scale of these instances, the existing combinatorial constraints, and the need for a flexible modeling framework capable of accommodating practical constraints, constraint programming appears to be a suitable paradigm for addressing this real-world problem. This section describes the CP model we designed for this purpose.

3.1 Decision variables

Briefly, solving this problem consists of assigning a half-block to each half-block course-group. Let $\mathcal{G}$ be the set of course-groups to schedule, and let $\mathcal{B}$ be the set of available blocks. The block b from set $\mathcal{B}$ is a subset of periods. These blocks can be split in two, to form half-blocks. The half-blocks of a block b are denoted by $h \in \mathcal{H}_b$ ($|\mathcal{H}_b| = 2\ \forall b \in \mathcal{B}$). Each half-block $h \in \mathcal{H}_b$ of block b has a complementing half-block $\overline{h}$. This complementing half-block can be formally defined as $\overline{h} = b \backslash h$ for all $h \in \mathcal{H}_b$. The global set of all half-blocks $h \in \mathcal{H}$ represent the clusters to which half-block course-groups may be assigned, and cover the available blocks, i.e. $\bigcup_{h \in \mathcal{H}} h = \bigcup_{b \in \mathcal{B}} b$.

In Table 2, there are six half-block course-groups: `ART302-01`, `ART302-02`, `ART302-03`, `SPO302-01`, `SPO302-02`, and `SPO302-03`. Note that the history course (`HIS304`) occupies a full block and is therefore not part of the half-block assignment. These six half-block course-groups can be grouped into a minimum of three clusters of paired groups. However, since schedules must remain closed (i.e. half-blocks must be paired to form complete blocks), the number of clusters must be even. Consequently, the number of clusters is rounded to four, identified by two available blocks ($|\mathcal{B}| = |\texttt{A, B}| = 2$), each subdivided in two:

- $b_1 = \texttt{A} = \bigcup_{h \in \mathcal{H}_\texttt{A}} h = \texttt{A13} \cup \texttt{A24}$,
- $b_2 = \texttt{B} = \bigcup_{h \in \mathcal{H}_\texttt{B}} h = \texttt{B13} \cup \texttt{B24}$.

The number of blocks used for half-block courses $|\mathcal{B}|$ should nevertheless be kept as small as possible in order to leave as many blocks as possible available for full-block courses, which has empirically been shown to improve overall scheduling quality. In practice, the number of available half-blocks can be bounded by the maximum number of periods required by either the teacher who teaches the most, the most used room, or the longest curriculum.

Equation (1) defines decision variables $\textsc{hblock}[g]$ that specify, for each course-group $g \in \mathcal{G}$, the half-block $h \in \mathcal{H}$ to which the course-group is assigned.

$$\textsc{hblock}[g] \in \mathcal{H} \qquad \forall g \in \mathcal{G} \tag{1}$$

Let $\mathcal{C}$ be the set of curricula and $\mathcal{S}$ the set of courses from different subjects (e.g. arts, sports, etc.). For each course-group $g \in \mathcal{G}$, let $s_g \in \mathcal{S}$ denote the course associated with g. Let $\mathcal{G}_s \subseteq \mathcal{G}$ denote the set of course-groups of course s. For each curriculum $c \in \mathcal{C}$, let $\mathcal{S}_c \subseteq \mathcal{S}$ denote the set of courses included in curriculum c. Note that several course-groups may correspond to the same course. In order to ensure that half-blocks can be combined within each curriculum to form complete blocks, additional decision variables are required. These variables ensure that at least one timetable is feasible per curriculum. Equation (2) defines the decision variables $\textsc{sched}[c, s]$, indicating the course-group from course $s \in \mathcal{S}_c$ in which a student from curriculum $c \in \mathcal{C}$ can sit without a conflict in their schedule.

$$\textsc{sched}[c, s] \in \mathcal{G}_s \qquad \forall c \in \mathcal{C}, \forall s \in \mathcal{S}_c \tag{2}$$

Intuitively, the sequence $(\textsc{hblock}[\textsc{sched}[c, s]])_{s \in \mathcal{S}_c}$ gives a timetable for a student following curriculum c. Table 4 provides a summary of the sets and notation used in the model. Some of these elements are introduced later in the text when they become necessary.

3.2 Constraints

This section introduces the constraints enforced in our model. We consider three types of constraints: those aimed at *preventing conflicts*, those designed for *balancing students*, and finally those used for *breaking symmetries*. Although symmetry-breaking constraints are not required for correctness, they are necessary to keep the execution time tractable.

Table 4 List of notations. All sets are defined and explained throughout the model description.

Symbol	Description
n	Number of students
$\mathcal{C}$	Set of curricula
$\mathcal{L}$	Set of grade levels
$\mathcal{B}$	Set of available blocks
$\mathcal{H}$	Set of all half-blocks in the blocks of $\mathcal{B}$
$\mathcal{H}_b$	Set of half-blocks in block b
$\mathcal{H}_l$	Subset of half-blocks on which the solution is spread for level $l \in \mathcal{L}$
$\overline{h}$	Complementing half-block of half-block h
$\mathcal{T}$	Set of teachers preassigned to the course-groups
t_g	Teacher preassigned to the course-group g
$\mathcal{R}$	Set of rooms preassigned to the course-groups
r_g	Room preassigned to the course-group g
$\mathcal{S}$	Set of courses (course titles)
$\mathcal{S}_c$	Set of courses included in curriculum $c \in \mathcal{C}$
s_g	Course to which course-group g belongs
$low[s]$	Minimum enrollment allowed for a section of course s
$cap[s]$	Maximum enrollment allowed for a section of course s
A	Enrollment matrix, $A[s, s']$ is the number of students taking both s and s'
$\mathcal{G}$	Set of course-groups to be scheduled
$\mathcal{G}_s$	Set of course-groups corresponding to course $s \in \mathcal{S}$
$\mathcal{G}_t$	Set of course-groups taught by teacher $t \in \mathcal{T}$
$\mathcal{G}_r$	Set of course-groups taught in room $r \in \mathcal{R}$

3.2.1 Constraints for preventing conflicts

The school assigns the teachers and rooms with a structure in mind that should allow for a feasible schedule. The following constraints define a feasible schedule.

Constraint 1: No teacher schedule conflict. No teacher is solicited by two course-groups at the same time. Constraint C1 specifies that all periods assigned to teacher t are different, where $\mathcal{G}_t \subseteq \mathcal{G}$ denotes the course-groups that are assigned to teacher t.

$$\texttt{allDifferent}\Big(\textsc{hblock}[g] \;\Big|\; g \in \mathcal{G}_t\Big) \qquad \forall t \in \mathcal{T} \tag{C1}$$

Constraint 2: No room conflict. No room is solicited by two course-groups at the same time. Constraint C2 specifies that all periods assigned to room r are different, where $\mathcal{G}_r \subseteq G$ denotes the course-groups that are assigned to room r.

$$\texttt{allDifferent}\Big(\textsc{hblock}[g] \;\Big|\; g \in \mathcal{G}_r\Big) \qquad \forall r \in \mathcal{R} \tag{C2}$$

Constraint 3: Minimum number of half-blocks for a curriculum. A feasible schedule must assign course-groups to distinct half-blocks. Consequently, the course-groups selected for a feasible timetable must cover at least as many half-blocks as there are courses in the schedule. For each curriculum $c \in \mathcal{C}$, let $\langle s_1^c, s_2^c, \ldots, s_{|c|}^c \rangle$ denote the courses in curriculum c,

sorted in increasing order of the number of available sections. Intuitively, courses with more sections would span over more half-blocks in the timetable. This leads to the following progressive requirement: the course-groups of course s_1^c must cover at least one half-block, the course-groups of courses $\{s_1^c, s_2^c\}$ must cover at least two half-blocks, and more generally, the course-groups of courses $\{s_1^c, s_2^c, \ldots, s_k^c\}$ must cover at least k distinct half-blocks. The course-groups associated with the set of courses $\{s_1^c, s_2^c, \ldots, s_k^c\}$ correspond to all $g \in \mathcal{G}$ such that $s_g \in \{s_1^c, s_2^c, \ldots, s_k^c\}$. Let us define the function $\texttt{nvalue}(E)$ returning the number of different values in the collection E. The constraints is as follows.

$$\texttt{nvalue}\Big(\textsc{hblock}[g] \;\Big|\; g \in \mathcal{G} \wedge s_g \in \{s_1^c, ..., s_k^c\}\Big) \geq k \qquad \forall c \in \mathcal{C}, \forall k \leq |c| \qquad \text{(C3)}$$

Constraint 4: Feasible schedules have all different half-blocks. A student cannot attend two course-groups simultaneously. Therefore, Constraint C4 enforces that the course-groups selected for a given curriculum are assigned to distinct half-blocks.

$$\texttt{allDifferent}\Big(\textsc{hblock}\big[\textsc{sched}[s, c]\big] \;\Big|\; s \in \mathcal{S}_c\Big) \qquad \forall c \in \mathcal{C} \qquad \text{(C4)}$$

Constraint 5: Feasible schedules are closed. Feasible schedules must be closed, meaning that every half-block course-group must be paired with a course-group assigned to its complementary half-block. To enforce this constraint, auxiliary variables $\mathcal{H}_c$ are introduced.

$$\mathcal{H}_c = \Big\{\textsc{hblock}\big[\ \textsc{sched}[s, c]\ \big] \;\Big|\; s \in \mathcal{S}_c\Big\} \qquad \forall c \in \mathcal{C} \qquad (3)$$

These variables represent the set of half-blocks to which the course-groups of a feasible schedule for curriculum c are assigned through the decision variable HBLOCK. For every half-block h in $\mathcal{H}_c$, the complementary half-block $\overline{h}$ must also belong to the schedule. Constraint C5 therefore ensures that half-blocks are paired to form complete blocks using $\texttt{count\_eq}(E, v, c)$. This constraint ensures that value v is found exactly c times in array E.

$$\texttt{count\_eq}(\mathcal{H}_c, \overline{h}, 1) \qquad \forall c \in C, \forall h \in \mathcal{H}_c \qquad \text{(C5)}$$

3.3 Constraints for balancing students

This second set of constraints aims to balance students across sections. For each course s, the school may specify the minimum and maximum number of students allowed in any section, denoted by the parameters $low[s]$ and $cap[s]$, respectively.

Constraint 6: At most all students can take a half-block course at the same time. We introduce the auxiliary variable sets $\mathcal{G}_h$, regrouping all course-groups assigned to half-block h.

$$\mathcal{G}_h = \Big\{\ g \in \mathcal{G} \;\Big|\; \textsc{hblock}[g] = h\ \Big\} \qquad \forall h \in \mathcal{H} \qquad (4)$$

Constraint C6 then bounds the number of seats offered in each half-block by the total number of students (n).

$$\sum_{g \in \mathcal{G}_h} low[s_g] \leq n \qquad \forall h \in \mathcal{H} \qquad \text{(C6)}$$

Constraint 7: Complementing course-groups are linked by the student course selection. A co-enrollment matrix A, provided by the school prior to scheduling, indicates for each pair of courses (s, s') the number of students $A[s, s']$ enrolled in both courses. To obtain closed schedules, the course-groups $\mathcal{G}_h$ assigned to a half-block h must correspond to courses whose students are also enrolled in at least one different course from the course-groups $\mathcal{G}_{\overline{h}}$ assigned to the complementary half-block $\overline{h}$.

$$\sum_{g' \in \mathcal{G}_{\overline{h}} \mid s_{g'} \neq s_g} \Big(A[s_g, s_{g'}] \Big) \geq low[s_g] \qquad \forall h \in \mathcal{H}, g \in \mathcal{G}_h \qquad \text{(C7)}$$

Constraint 8: Bounds on acceptable number of students per cluster. This constraint ensures that the minimum number of students potentially assigned to one half-block never exceed the maximum number of students that can be accommodated in the complementary half-block. The minimum number of students assigned to a half-block h is obtained by summing the minimum enrollments $low[s_g]$ for the courses corresponding to the course-groups in $\mathcal{G}_h$. Similarly, the maximum number of students that can be assigned to the complementary half-block $\overline{h}$ is obtained by summing the maximum enrollments $cap[s_{g'}]$ for the courses corresponding to the course-groups in $\mathcal{G}_{\overline{h}}$.

$$\sum_{g \in \mathcal{G}_h} low[s_g] \leq \sum_{g' \in \mathcal{G}_{\overline{h}}} cap[s_{g'}] \qquad \forall h \in \mathcal{H} \qquad \text{(C8)}$$

3.4 Constraints for breaking symmetries

A specificity of this problem is that the particular half-block assigned to a course-group (e.g., `ART302-01` or `ART302-02`) does not affect the feasibility, nor the quality, of the solution when half-blocks courses are scheduled first. This creates many symmetries, since courses may have multiple sections and students have no preference regarding the specific section to which they are assigned. Consequently, identifying and breaking these symmetries can significantly reduce the search time. To break symmetries, we consider courses in increasing order of the number of sections they offer. Within each course, the corresponding course-groups are already indexed and sorted by section number (e.g. `ART302-01` before `ART302-02`).

Table 5 illustrates an example of a schedule that is progressively reordered as the symmetry-breaking constraints are applied. It can be assumed that every teacher has their own room, in which they teach all their course-groups. In this example, `MUS302` is the course with the fewest sections, followed by `ART302`, and then `SPO302`. The course-groups are considered from top to bottom. The initial clusters of course-groups in this example are: {`ART302, SPO302, SPO302`}, {`MUS302, ART302, SPO302`}, {`ART302`}, and {`SPO302`}.

Constraint 9: Identical course-groups are strictly increasing. In a school timetable, the same teacher often teaches multiple groups of the same course in the same room. From the students' perspective, these groups are identical. As a result, exchanging these groups across half-blocks produces symmetrical solutions. To eliminate such symmetries, these course-groups are assigned to strictly increasing half-block indices, as enforced by Constraint C9, where $t_g \in \mathcal{T}$ and $r_g \in \mathcal{R}$ are the teacher and room preassigned to the course-group g, respectively.

$$\texttt{strictly\_increasing}\Big(\text{HBLOCK}[g] \;\Big|\; g \in \mathcal{G}_s \wedge t_g = t \wedge r_g = r \Big) \quad \forall s, t, r \in (\mathcal{S}, \mathcal{T}, \mathcal{R}) \quad \text{(C9)}$$

Table 5 Example of symmetry breaking constraints applied progressively. Underlined marks shows difference from the schedules of the previous situation.

(a) Base example without symmetry constraints.

	A13	A24	B13	B24
MUS302-01		×		
ART302-01			×	
ART302-02		×		
ART302-03	×			
SPO302-01		×		
SPO302-02	×			
SPO302-03				×
SPO302-04	×			

(b) Adding Constraint C9 on schedule (a).

A13	A24	B13	B24	teacher
	×			t1
	×			t2
		×		t2
×				t3
×				t4
	×			t4
×				t5
			×	t5

(c) Adding Constraints C9, C10 on schedule (a).

	A13	A24	B13	B24
MUS302-01				×
ART302-01	×			
ART302-02				×
ART302-03			×	
SPO302-01			×	
SPO302-02				×
SPO302-03		×		
SPO302-04			×	

(d) Adding Constraints C9-C11 on schedule (a).

A13	A24	B13	B24	teacher
×				t1
×				t2
		×		t2
	×			t3
×				t4
	×			t4
	×			t5
			×	t5

Example in Table 5b shows the schedule after breaking this symmetry. Compared with Table 5a, the identical course-groups ART302-01 and ART302-02 have been reordered. Similarly, course-groups SPO302-01 and SPO302-02, as well as SPO302-03 and SPO302-04, are reordered.

Constraint 10: Symmetries between blocks. This constraint indicates that no distinction should be made between blocks. We formalize it using the MiniZinc predicate value_precede_chain(c, X), which ensures that the value $c[i]$ appears before $c[i+1]$ in the array of variables X. Constraint C10 applies this predicate to an array of increasing odd half-blocks. It enforces that the first occurrence of each odd half-block appears in increasing order within the HBLOCK variable array. The HBLOCK array itself is ordered according to increasing course-group identification numbers.

$$\texttt{value\_precede\_chain}\Big([h \in \mathcal{H} \mid h \text{ is an odd half block}], \text{HBLOCK}\Big) \tag{C10}$$

We note that blocks with even identifiers cannot be considered for this constraint, since the uniqueness of the solution lies in the pairing between each odd block and its complementary even block. The example in Table 5b shows that the odd half-block A13 has its first course-group, ART302-03, appearing after the first course-group in odd half-block B13, namely ART302-02. Therefore, these blocks are exchanged, together with their complementary half-blocks, to satisfy Constraint C10, yielding the configuration shown in Table 5c. In addition, to satisfy Constraint C9, some course-groups must also now exchange clusters: ART302-01 with ART302-02, and SPO302-03 with SPO302-04.

Constraint 11: Symmetries between complementing half-blocks. Similarly, Constraint C11 ensures that paired half-blocks appear in a canonical order in the solutions. Specifically, for each pair of complementary half-blocks $(h, \overline{h})$, the odd half-block h must contain the course-group with the smallest identification number compared to those assigned to the even half-block $\overline{h}$. Lexicographic constraint `value_precede`(s, t, E) ensures that value s precedes value t in collection E.

$$\texttt{value\_precede}\Big(h, \overline{h}, \text{HBLOCK}\Big) \qquad \forall h \in \mathcal{H} | h \text{ is an odd half block} \tag{C11}$$

In Table 5c, this constraint is satisfied for block `A`: the odd half-block `A13` contains the course-group `ART302-01`, which precedes `SPO302-03` assigned to the complementary half-block `A24` in the course-group ordering. Therefore, the relative order between these clusters remains unchanged in Table 5d. However, the symmetry must be broken for half-blocks `B13` and `B24`. Their first course-groups appear in reversed order, which violates the constraint. As a result, the course-groups in these two columns are exchanged. Furthermore, to satisfy Constraint C9, the odd half-blocks `A13` and `B13` must also be reordered. Finally, the sections of identical courses are reordered accordingly.

Constraint 12: Fixing the first half-block variable. The first course-group is assigned to the first half-block. In Table 5d, `MUS302-01` is therefore placed on the first half-block.

$$\text{HBLOCK}[1] = \texttt{A13} \tag{C12}$$

3.5 Objectives

We recall that the objective is to obtain a well-balanced schedule. In an ideal solution, each section of a course would contain the same number of students. However, student sectioning is not performed within the CP model, as including it would make the problem too large to solve within an acceptable time (i.e., it would introduce an additional decision variable for each student) and may require school-specific constraints. Instead, an alternative metric is used to evaluate the solutions.

Objective 1: Theoretical balance. Solving the model described above yields an assignment of course-groups to half-blocks. Assuming an ideal student sectioning, all sections of a course would contain the same number of students. In a suitable timetable, the total ideal number of students per group on a half-block should be close to the average number of seats offered per half-block for each grade level considered. Maintaining this balance facilitates the subsequent student sectioning process. We refer to this metric as the *theoretical balance*.

$$\text{THEORETICALBALANCE} = \sum_{l \in \mathcal{L}} \sum_{h \in \mathcal{H}} \left| \underbrace{\left(\sum_{\substack{g \in \mathcal{G}_l \\ \text{HBLOCK}[g]=h}} \frac{A[s_g, s_g]}{|\mathcal{G}_{s_g}|} \right)}_{\text{Ideal number of students per group}} - \underbrace{\left(\frac{\sum_{s \in \mathcal{S}_l} A[s,s]}{|\mathcal{H}_l|} \right)}_{\text{Number of seats offered}} \right| \tag{5}$$

For each level $l \in \mathcal{L}$ and each block $h \in \mathcal{H}$, the model computes the difference between the ideal number of students expected in the groups assigned to that half-block and the number of seats that should ideally be offered in that half-block. These differences are then aggregated over all levels and blocks. The ideal number of students per group corresponds to the average enrollment of a course assuming that students are evenly distributed across all

its sections. The number of seats that should be offered in a block is obtained by summing the available seats for all courses at level l and dividing this total by $|\mathcal{H}_l|$, the number of half-blocks used for that level. This metric is also used by our industrial partner to build a first solution prior the student sectioning phase.

Objective 2: Actual spread after student sectioning. The solution produced by the CP model is then passed to the student sectioning algorithm of the industrial partner. This algorithm iteratively assigns students using a greedy heuristic that has historically produced excellent results within seconds. After student sectioning, the *spread* metric is computed. The spread of a course is defined as the difference between the number of students in its largest and smallest sections, as formalized in Equation 6, where $N_{g,s}$ denotes the number of students enrolled in group g for course s. This value corresponds to the actual imbalance observed in practice once all students have been assigned to sections. Consequently, it can only be computed *a posteriori* after student sectioning. The objective is to minimize the spread, thereby improving the balance of students among course sections.

$$\text{SPREAD} = \sum_{s \in S} \Big(\max(N_{g,s} \mid g \in \mathcal{G}_s) - \min(N_{g,s} \mid g \in \mathcal{G}_s) \Big) \tag{6}$$

4 Experimental results

The goal of the experiments is to evaluate the performance of the proposed CP model in generating suitable schedules. The solutions are assessed using both the theoretical balance, which is the metric minimized by the model, and the actual spread observed after student sectioning. The student course selections and course-group information are provided by our industrial partner. The CP model is implemented in MiniZinc [13] and solved using Gecode 6.3.0. The solver configuration is kept to its default settings and heuristics. The input data for the CP model are preprocessed using a Python script, which then calls the MiniZinc model through their Python API. The output produced by the CP model is subsequently passed to the student sectioning algorithm of the industrial partner to complete the timetable construction. Finally, all experiments were conducted on a laptop equipped with an Intel i7-1065G7 CPU (1.30 GHz) and 8 GB of RAM.

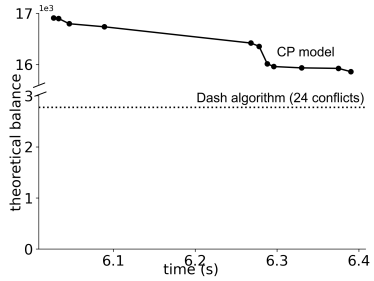
Description of the instances and baselines. Our approach is evaluated on four real-world instances from high schools in the province of Quebec for the 2024–2025 academic year. Their characteristics are summarized in Table 6. The results are compared with the schedules that were accepted and used by the schools during that year, which we refer to as the *historical solutions*. It is difficult to provide an exact estimate of the time that has been required to produce these schedules, as it depends on the operator guiding the search as well as on the number of iterations (or restarts) needed following consultations with the school. As a rough estimate, one iteration typically takes between 3 and 12 hours, and at least 2 iterations are required. For example, instance `school-2` took over 5 hours per iteration to schedule and took two iterations, for a total of a day and a half of work for the operator. Moreover, these solutions may violate some constraints due to late requirements introduced by the schools and may therefore not be reachable by our CP model.

Our method is also compared with the hand-crafted approach used by the industrial partner to find a first feasible solution, which we refer to as the *Dash algorithm*. This algorithm iteratively assigns course-groups to blocks by greedily minimizing the theoretical balance while ensuring that no teacher or room conflicts occur. However, this approach may generate conflicts at the student level (i.e. conflicts on Constraints C3, C4, C6, C7 and C8).

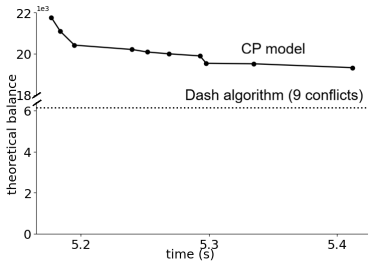
Table 6 Test instances features: numbers of students (n), courses ($|\mathcal{S}|$), total course-groups ($|\mathcal{G}|$), curricula ($|\mathcal{C}|$), half-blocks ($|\mathcal{H}|$), teachers ($|\mathcal{T}|$), rooms ($|\mathcal{R}|$), groups per curriculum ($|\mathcal{G}|/|\mathcal{C}|$), and course-groups per curriculum ($size(c)$).

Instance	n	$\|\mathcal{S}\|$	$\|\mathcal{G}\|$	$\|\mathcal{C}\|$	$\|\mathcal{H}\|$	$\|\mathcal{T}\|$	$\|\mathcal{R}\|$	$\|\mathcal{G}\|/\|\mathcal{C}\|$	$size(c)$
`School-1`	553	20	108	38	10	32	27	2.84	2 to 6
`School-2`	714	19	97	34	12	28	23	2.85	4
`School-3`	781	31	112	97	12	27	29	1.15	2 to 6
`School-4`	399	18	46	25	10	18	17	1.84	2 to 6

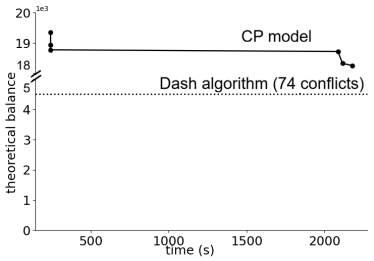
Result: Analysis of the theoretical balance. Figure 1 shows the evolution of the theoretical balance, used as the objective function, during the search for each instance. Instances `School-1` and `School-2` reach their best found solution quickly (in roughly 6 seconds). Instances `School-3` and `School-4` continue to produce improved solutions until approximately 37 and 10 minutes, respectively. These two instances are more challenging, mainly because they contain fewer groups per curriculum (lower ratio $|\mathcal{G}|/|\mathcal{C}|$). Consequently, students have fewer groups available for each course, which reduces the number of feasible scheduling options. Optimality has not been proven for any of the instances. Dash algorithm provides a solution almost instantly, with a significantly better theoretical balance. However, it generates many conflicts, i.e., students without a feasible schedule (74 for `School-3` and up to 221 for `School-4`), whereas no conflicts are produced by our CP model.



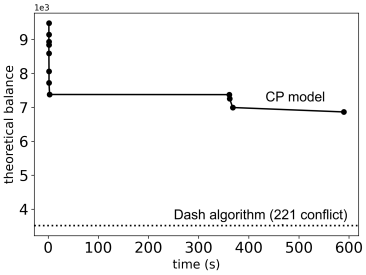
(a) Results for `School-1`.



(b) Results for `School-2`.



(c) Results for `School-3`.

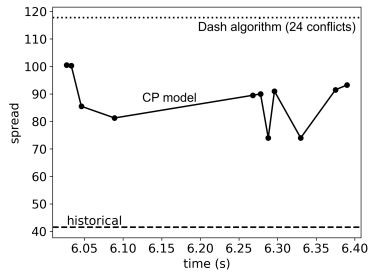


(d) Results for `School-4`.

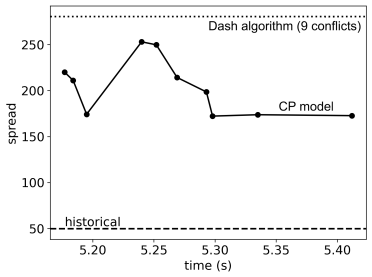
Figure 1 Theoretical balance in 60 min of search time. No improvements were observed after: (a) 6.39 s, (b) 5.33 s, (c) 2179 s, and (d) 590 s. Figures are truncated accordingly.

Result: Analysis of the actual spread. Figure 2 presents the actual *spread* obtained after student sectioning. As baselines, we report both the historical solution and the results of *Dash algorithm*. The results support the hypothesis that lower values of the theoretical

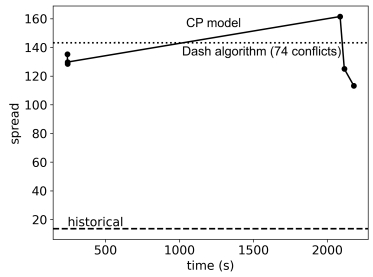
balance tend to yield a lower spread. The CP model achieves a better total spread than the Dash algorithm on all instances, notably because there is no conflict that must be manually handled. The timetable obtained for `School-4` closely match the historical result and could be finalized with only a few minor manual adjustments during the student sectioning process.



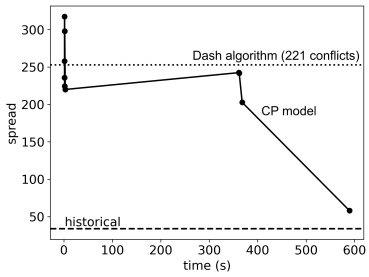
(a) Results for `School-1`.



(b) Results for `School-2`.



(c) Results for `School-3`.



(d) Results for `School-4`.

Figure 2 Actual spread in 60 min of search time. Thresholds indicate historical spread and spread with the Dash algorithm.

5 Conclusion

This paper addresses a specific variant of the curriculum-based timetabling problem arising in institutions that rely on predefined block patterns. We formulate and solve the half-block course-group clustering problem using constraint programming. The model was evaluated on real instances from high schools, involving between 46 and 112 course-groups to schedule. On Instance 4, it produced a timetable of comparable quality to the historically approved solution for the 2024–2025 academic year roughly six time faster than the current manual process. Overall, the proposed approach generates solutions of a lower quality compared to those obtained historically through an expensive human-guided local search, but does so with significantly reduced computation time, while still being acceptable in practice. This makes the model useful for validating teacher and room preassignments, as well as the number of sections offered for each course. The approach has been integrated in the student sectioning algorithm of our industrial partner and will be used starting in 2026 to generate schedules for the upcoming academic years. As future work, alternative metrics derived from the co-enrollment matrix could be investigated to guide the CP search toward solutions that provide greater scheduling flexibility for students and further improve the quality of the resulting sectioning.

References

1 Niels-Christian Fink Bagger, Guy Desaulniers, and Jacques Desrosiers. Daily course pattern formulation and valid inequalities for the curriculum-based course timetabling problem. *Journal of Scheduling*, 22(2):155–172, 2019. doi:10.1007/S10951-018-0582-0.

2 Andrea Bettinelli, Valentina Cacchiani, Roberto Roberti, and Paolo Toth. An overview of curriculum-based course timetabling. *Top*, 23(2):313–349, 2015.

3 Alex Bonutti, Fabio De Cesco, Luca Di Gaspero, and Andrea Schaerf. Benchmarking curriculum-based course timetabling: formulations, data formats, instances, validation, visualization, and results. *Annals of Operations Research*, 194(1):59–70, 2012. doi:10.1007/S10479-010-0707-0.

4 Sara Ceschia, Luca Di Gaspero, and Andrea Schaerf. Educational timetabling: Problems, benchmarks, and state-of-the-art results. *European Journal of Operational Research*, 308(1):1–18, 2023. doi:10.1016/J.EJOR.2022.07.011.

5 Gabriella Colajanni and Patrizia Daniele. A new model for curriculum-based university course timetabling. *Optimization Letters*, 15(5):1601–1616, 2021. doi:10.1007/S11590-020-01588-X.

6 Emir Demirović and Peter J Stuckey. Constraint programming for high school timetabling: A scheduling-based model with hot starts. In *International conference on the integration of constraint programming, artificial intelligence, and operations research*, pages 135–152. Springer, 2018.

7 Angelos Dimitsas, Vasileios Nastos, Christos Gogos, and Christos Valouxis. An exact based approach for the post enrollment course timetabling problem. In *Proceedings of the 26th Pan-Hellenic Conference on Informatics*, pages 77–82, 2022. doi:10.1145/3575879.3575970.

8 Maria Dostert, A Politz, and Heinz Schmitz. A complexity analysis and an algorithmic approach to student sectioning in existing timetables. *Journal of Scheduling*, 19(3):285–293, 2016. doi:10.1007/S10951-015-0424-2.

9 Gouvernement du Québec. I-13.3, r. 8 – Basic school regulation for preschool, elementary and secondary education, art. 28, 2025. Article 28: Evaluation and promotion of students, updated 1 July 2025. URL: https://www.legisquebec.gouv.qc.ca/en/version/cr/I-13.3%2C%20r.%208%20?code=se%3A28.

10 Jade Diane Fernandes Targino Filgueira, Hugo Valadares Siqueira, and Carmelo José Albanez Bastos Filho. A survey of the state of the art of educational timetabling problems. In Frederico Guimar aes, Danton Ferreira, and Guilherme Barreto, editors, *Anais do XVII Congresso Brasileiro de Inteligência Computacional (CBIC 2025)*, pages 1–7, Belo Horizonte, MG, 2025. SBIC. doi:10.21528/CBIC2025-1171435.

11 Gouvernement du Québec. Loi sur l'instruction publique. https://www.legisquebec.gouv.qc.ca/fr/document/lc/i-13.3, 1988. RLRQ, c. I-13.3.

12 Marco Antonio Aguirre Lam, Fausto Balderas, Nelson Rangel-Valdez, Jose A Martinez-Flores, José A Pérez-Vázquez, et al. University timetabling problem: An analysis of the state of the art. *International Journal of Combinatorial Optimization Problems and Informatics*, 16(3):489, 2025.

13 Nicholas Nethercote, Peter J Stuckey, Ralph Becket, Sebastian Brand, Gregory J Duck, and Guido Tack. MiniZinc: Towards a standard CP modelling language. In *International conference on principles and practice of constraint programming*, pages 529–543. Springer, 2007. doi:10.1007/978-3-540-74970-7_38.

14 Ana-Maria Nogareda and David Camacho. Optimizing satisfaction in a multi-courses allocation problem combined with a timetabling problem. *Soft Computing*, 21(17):4873–4882, 2017. doi:10.1007/S00500-016-2375-8.

15 I Gusti Agung Premananda, Aris Tjahyanto, and Ahmad Muklason. Timetabling problems and the effort toward generic algorithms: A comprehensive survey. *IEEE Access*, 12:143854–143868, 2024. doi:10.1109/ACCESS.2024.3463721.

16 Vincent Robert and Alain Hertz. How to decompose constrained course scheduling problems into easier assignment type subproblems. In *International Conference on the Practice and Theory of Automated Timetabling*, pages 364–373. Springer, 1995. doi:10.1007/3-540-61794-9_71.

17 Landir Saviniec, Maristela O Santos, Alysson M Costa, and Lana MR dos Santos. Pattern-based models and a cooperative parallel metaheuristic for high school timetabling problems. *European Journal of Operational Research*, 280(3):1064–1081, 2020. doi:10.1016/J.EJOR.2019.08.001.

18 Elmar Steiner, Ulrich Pferschy, and Andrea Schaerf. Curriculum-based university course timetabling considering individual course of studies. *Central European Journal of Operations Research*, 33(1):277–314, 2025. doi:10.1007/S10100-024-00923-2.

19 Joo Siang Tan, Say Leng Goh, Graham Kendall, and Nasser R Sabar. A survey of the state-of-the-art of optimisation methodologies in school timetabling problems. *Expert Systems with Applications*, 165:113943, 2021. doi:10.1016/J.ESWA.2020.113943.

20 Christos Valouxis and Efthymios Housos. Constraint programming approach for school timetabling. *Computers & Operations Research*, 30(10):1555–1572, 2003. doi:10.1016/S0305-0548(02)00083-7.