A Two-Phase Constraint Programming Model for Examination Timetabling at University College Cork

Begum Genc[1(✉)] and Barry O'Sullivan[1,2]

[1]Confirm Centre for Smart Manufacturing and
[2]Insight Centre for Data Analytics,
School of Computer Science and Information Technology,
University College Cork, Cork, Ireland
{begum.genc,b.osullivan}@cs.ucc.ie

Abstract. Examination timetabling is a widely studied NP-hard problem. An additional challenge to the complexity of the problem are many real-world requirements that can often prevent the relaxation of some constraints. We report on a project focused on automating the examination timetabling process of University College Cork (UCC) to enhance the examination schedules so that they are fairer to student, as well as being less resource intensive to generate from an administrative point of view. We work with a formulation developed in collaboration with the institution and real data that it provided to us. We propose a two-phase constraint programming approach to solving UCC's examination timetabling problem. The first phase considers the timing of examinations while the second phase considers their allocation to rooms. Both phases are modelled using bin-packing constraints and, in particular, an interesting variant in which items can be split across multiple bins. This variant is known as *bin packing with fragmentable items*. We investigate the tightly linked constraints and difficulties in decomposing the centralised model. We provide empirical results using different search strategies, and compare the quality of our solution with the existing UCC schedule. Constraint programming allows us to easily modify the model to express additional constraints or remove the pre-existing ones. Our approach generates significantly better timetables for the university, as measured using a variety of real-world quality metrics, than those prepared by their timetabling experts, and in a reasonable timeframe.

1 Introduction

Scheduling is one of the best-known problems in optimisation. It is a broad field that includes, for example, nurse scheduling in hospitals, task scheduling in smart manufacturing, and timetabling in schools and universities [26,34,38]. Examination timetabling is a particularly hard variant that has been widely studied [8]. Many approaches to the problem have been studied, such as graph-based algorithms, local search or population-based algorithms, constraint-based

H. Simonis (Ed.): CP 2020, LNCS 12333, pp. 724–742, 2020.
https://doi.org/10.1007/978-3-030-58475-7_42

techniques, decomposition techniques, etc. [31]. For example, decomposition approaches often involve first assigning exams to time-slots and then distributing these exams across a set of rooms, creating sub-problems based on some groupings. Alternatively, local search approaches start with an initial schedule which is then subsequently improved by making local changes using a heuristic search and repair method [9,13,16,20,23,29]. We refer the reader to a number of recent literature reviews [12,31].

A variety of exact methods have been reported. McCollum et al. provide an easy-to-understand, integer programming model for the automated timetabling problem [22]. Arboui et al. improve on this model, mainly by introducing some pre-processing, clique and data-dependent dual-feasible function valid inequalities, and improving the bounds of some existing constraints [3]. There exist also some Constraint Programming (CP) approaches that construct an initial solution to be improved by using heuristics [14,23].

Approaches that decompose timetabling by separately assigning exams to time-slots and assigning the exams to rooms have existed for almost three decades. For example, Lotfi and Cerveny tackle the assignment of exams to time-slots using multiple phases [20]. They model the assignment of exams to rooms as a minimum-weighted matching problem which they solve using a heuristic approach. More recently, Mirrazavi et al. apply a similar two-staged approach to course timetabling in a university department using large-scale integer goal programming [27]. They first allocate lectures to rooms, then starting times to lectures. Although the decomposition means that optimality is sacrificed, it is still preferred by some institutions due to the effort spent by administrative staff in manually creating timetables [33].

For examination timetabling, many approaches ignore the allocation of exams to rooms. This is often due to institutions not allowing the splitting of exams across multiple rooms, or not allowing multiple exams to be held in the same venue at the same time [12,22,23]. Therefore, some models discard the allocation of exams to rooms entirely [4,37]. If the splitting is allowed, the solution is often obtained using meta-heuristic approaches. Unfortunately, in many real-life cases, it is not possible to disregard the room allocation phase. In fact, in the UCC case considered here, room allocation is one of the most interesting aspects of the problem. Many real-life applications of the examination timetabling problem using hybrid approaches can be found in the literature [18,28].

In this paper, we present a CP model for University College Cork's (UCC) examination timetabling process that respects all the requirements of the institution and produces solutions quickly. Although the solutions found by the proposed CP model are not necessarily optimal, the model improves the current schedule on all metrics the university wishes to apply by a considerable margin. Exact models often cannot produce solutions to large-scale problems within reasonable time-limits [18]. We tailored the exact model in [3,22] to UCC. However, UCC has some requirements that are significantly different from the existing use-cases. When the extra constraints such as the splitting of exams are added to the model, a solution could not be obtained within reasonable time-frame. Therefore,

we developed a decomposition-based model for the problem which we present in this paper.

2 The UCC Examination Timetabling Problem

2.1 Problem Specification and Data

We summarise the main features of *University College Cork (UCC)* timetabling problem.[1] The problem comprises both hard and soft constraints. Throughout the paper, we refer to the requests given by the institution as the *requirements*, and the way we model them as *constraints*. Each hard requirement must be satisfied in each solution. On the other hand, soft requirements may be violated, but it is desirable to keep the violations to a minimum.

At the university there are 12,686 students sitting at least one exam. Of these, 855 students have some special needs that require use of a shared room (SHR), separate room (SPR), or lab (LAB). The SPR students must sit the exams alone in an SPR tagged room, but SHR (or LAB) students can share an SHR (or LAB) tagged room with other SHR (or LAB) students. We refer to the set of non-special needs students sitting exams as the *main group*, and the special needs students with their tags. We denote by $\mathbb{S}$ the set of all students. There are in total 43,002 exam seats required during the whole examination season. All examinations must be completed within a 10 day horizon. Each day has one morning time-slot and two afternoon time-slots. Morning time-slots are 180 min long, whereas the afternoon time-slots are 90 min each. We denote by $\mathbb{P}$ the set of all available time-slots. There are 717 exams, where 37 of them are 180 min, and 680 are 90 min long. We denote by $\mathbb{E}$ the set of all exams.

There are 9 *main rooms* available for the main groups with the following capacities that define the number of students that can be accommodated in a single sitting: 513, 513, 220, 171, 140, 130, 93, 91, and 56. One of the rooms with capacity 513 is called the R_{ex} room which is located on a separate campus. We denote by $\mathbb{R}$ the set of all main rooms. Additionally, there are 31 SPR-tagged auxiliary rooms, 1 SHR-tagged room with capacity 50, and 1 LAB-tagged room with capacity 60. The sets of these auxiliary rooms are denoted by $\mathbb{R}^{SPR}$, $\mathbb{R}^{SHR}$, and $\mathbb{R}^{LAB}$, respectively. We list below the set of hard and the soft requirements denoted with the prefixes 'H' and 'S', respectively.

H1 Each exam must be scheduled in exactly one time-slot where the duration of the exam is less than or equal to the duration of the time-slot.
H2 Each student must sit at most one exam at any time-slot.
H3 Each student sitting an exam must be allocated a seat.
H4 An exam can be split across multiple rooms with the exception that the main groups sitting an exam in R_{ex} cannot be split.
H5 More than one exam can be held in the same main room during the same time-slot as long as all exams have the same duration and the total capacity is not exceeded.

[1] UCC dataset can be found in: http://github.com/begumgenc/ucc-et.

H6 An examination can have multiple module codes. These exams are identified by their *group-ids*. If two exams have the same group-id, they must be co-scheduled.
H7 Specific requests such as an exam to be held before or during a specific time-slot, or in a specific room, must be satisfied.
H8 Requests for an exam to be scheduled after another one must be satisfied.
H9 Special needs students must be assigned to rooms that are tagged with the corresponding tags. They cannot be seated in the same rooms as the main groups.
H10 Students must not sit more than 270 min of exams in total over any two consecutive days.
H11 If a student is scheduled for multiple exams on the same day, either all of those exams must be held in R_{ex} or none of them.
H12 A number of seats in each main room must be left empty for accommodating ad-hoc students, i.e. students with unforeseen situations.

S1 Exams should be grouped within the same room as much as possible.
S2 Each student should sit at most one exam every two consecutive days.
S3 If Requirement S2 is not fully satisfied, then students should sit at most one exam a day.
S4 If Requirement S3 is not fully satisfied, then there should be at least one time-slot between each exam of each student.
S5 The use of some time-slots are more preferred to some other ones.
S6 It is desirable to schedule large exams as early as possible.
S7 It is desirable to minimise exam splits (i.e. the exam being held in more than one room) for main, SHR, and LAB groups.
S8 It is more preferred to use larger main rooms than smaller ones.

2.2 Notation and Terminology

In this section, we introduce the general notation to be followed in our model. A pair of exams, e_i, e_j, are said to be *conflicting* if there is at least one student who is enrolled for both e_i and e_j. To capture the constraints related to conflicting exams, we make use of a *conflict graph* [22]. In the conflict graph, each vertex represents a unique exam. There exists an edge between two vertices if the exams corresponding to those vertices are conflicting. The weight of each edge $a = (e_i, e_j)$, denotes the total number of students sitting both e_i and e_j. A *clique* in this graph, represents a set of exams, where all those exams are conflicting with each other, and they must all, therefore, be scheduled in different time-slots. We can find all maximal cliques in the conflict graph using the Bron-Kerbosch algorithm [6]. We describe below the main notation:

- The list $E = \langle e_0, ..., e_{|\mathbb{E}|-1} \rangle$ is an ordered list representation of exams in set $\mathbb{E}$.
- The list $P = \langle p_0, ..., p_{|\mathbb{P}|-1} \rangle$ is an ordered list representation of time-slots in set $\mathbb{P}$.

- The list $R = \langle r_0, ..., r_{|\mathbb{R}|-1} \rangle$ is an ordered list representation of main rooms in set $\mathbb{R}$. We also denote by R^{SPR}, R^{SHR}, R^{LAB} the ordered lists of special needs rooms over the sets $\mathbb{R}^{SPR}$, $\mathbb{R}^{SHR}$, $\mathbb{R}^{LAB}$, similarly.
- The list $S = \langle s_0, ..., s_{|\mathbb{S}|-1} \rangle$ is an ordered list of students in set $\mathbb{S}$. For each student s, E_s denotes the set of exams that student s is enrolled, where $E_s \subset E$.
- List D denotes the available days, in our case $D = \langle 0, .., |\mathbb{P}|/3 \rangle$. Each d in D spans exactly three time-slots $p_i, p_j, p_k \in P$ such that $\lfloor p_i/3 \rfloor = \lfloor p_j/3 \rfloor = \lfloor p_k/3 \rfloor = d$.
- Set G corresponds to a set of group-ids. For each $g \in G$, g denotes a group-id for a set of exams to be co-scheduled.
- Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denotes the conflict graph, and $\mathcal{C}^{\mathcal{G}}$ denotes the set of all maximal cliques in $\mathcal{G}$ that contain more than one vertex.
- For each edge represented as $a = [e_i, e_j] \in \mathcal{E}$, a denotes an edge having its source and target vertices corresponding to exams $e_i, e_j \in \mathcal{V}$. The weight of each a is denoted by $w(a)$, and corresponds to the number of students sitting both e_i and e_j.
- For each e in $E, size_e^E$ denotes the total number of students sitting exam e. Additionally, $size_e^N$, $size_e^{SPR}, size_e^{SHR}, size_e^{LAB}$ denote the number of students sitting e with the main group, or has SPR, SHR, and LAB tags, respectively.
- For each room r, cap_r^R denotes the number of seats available in r. For each p in P, cap_p^P, cap_p^{SPR}, cap_p^{SHR}, cap_p^{LAB} correspond to the total number of available seats for main, SPR, SHR, LAB groups during p, respectively.
- For each period p in P, dur_p^P denotes the duration of time-slot p. Similarly, dur_e^E denotes the duration of an exam e in E.

Throughout the paper, we sometimes denote the elements by using indices when dealing with multiple items from the same set, e.g. $p \in P$ or $p_i, p_j \in P$. In these cases, let $I(X)$ be a set that denotes the index set of a given set X. Then, for any element x_i in a set X, $i \in I(X)$. The list of exams E is ordered in descending order of the number of non-special needs students sitting each exam. Additionally, the time-slot list P is ordered in ascending order of the timings of the corresponding time-slots, i.e. for any $i, j \in I(P)$ with $i < j :\rightarrow p_i < p_j$, p_i is an earlier time-slot than p_j. Finally, the list of rooms (i.e. R) is presented in descending order of the capacity of each room.

3 The Constraint Programming Models

In this section we present in detail our two CP models, one for each phase of the problem. Initially, as a pre-processing step, we create a conflict graph to keep track of conflicting exams efficiently. Subsequently, we use our Phase 1 model that is based on bin-packing global constraint for determining the timing of examinations. Finally, in Phase 2, we use the solution generated by Phase 1 to assign the examinations to rooms within the given time-slot. We model Phase 2 using a variant of the bin-packing constraint.

3.1 Phase 1: Allocation of Exams into Timeslots

We find timings for each exam using bin-packing global constraints [36]. The exams correspond to items, and the time-slots correspond to bins. The capacity of each bin is the total number of available seats during the corresponding time-slot. In our case, all the bins have equal capacities. The bin-packing problem is well-known to be NP-hard [15]. We list below the main variables of our Phase 1 CP model.

- For each e in E, an integer variable $\boldsymbol{EP[e]}$ denotes the time-slot in which e is scheduled, where $EP[e] \in P$. The search branches on this variable.
- For each e in E, an integer variable $\boldsymbol{ED[e]}$ denotes the day in which e is scheduled, where $ED[e] \in D$.
- For each p in P, integer variables $\boldsymbol{L[p]}$, $\boldsymbol{L^{SPR}[p]}$, $\boldsymbol{L^{SHR}[p]}$, $\boldsymbol{L^{LAB}[p]}$ denote the total number of students in main, SPR, SHR, and LAB groups, sitting an exam at p, resp. Note that, $L[p] \in \{0, ..., \sum_{r \in \mathbb{R}} cap_r^R\}$, $L^{SHR}[p] \in \{0, ..., \sum_{r \in \mathbb{R}^{SHR}} cap_r^R\}\}$, $L^{LAB}[p] \in \{0, ..., \sum_{r \in \mathbb{R}^{LAB}} cap_r^R\}\}$, and $L^{SPR}[p] \in \{0, ..., |\mathbb{R}^{SPR}|\}$.
- For each exam $e \in E$, a boolean variable $\boldsymbol{pl[e]}$ denotes if e is large, and scheduled late or not, where p_σ denotes the time-slot after which the exams are considered late.
- For each edge $a \in \mathcal{E}$, integer variables $\boldsymbol{V_{S2}[a]}, \boldsymbol{V_{S3}[a]}, \boldsymbol{V_{S4}[a]}$ denote for each soft requirement Requirements $S2$, $S3$ and $S4$, the number of occurrences a student is negatively affected due to the violation of the respective requirement by the exam pair a. Note that, $V_{S2}[a], V_{S3}[a], V_{S4}[a] \in \{0, w(a)\}$.
- For each $p \in P$, an integer variable $\boldsymbol{V_{S5}[p]}$ denotes the penalty incurred due to the violation of Requirement $S5$, where α_p represents a constant penalty for using p. Note that, $V_{S5}[p] \in \{0, ..., cap_p^P \times \alpha_p\}$.
- Sub-objective values $\boldsymbol{obj_2}, \boldsymbol{obj_3}, \boldsymbol{obj_4}, \boldsymbol{obj_5}, \boldsymbol{obj_6}$ denote the total penalty caused by the violation of Requirements $S2, S3, S4, S5$, and $S6$, respectively. The overall objective $\boldsymbol{obj}$ is a weighted sum of the sub-objective values.

Timeslot Capacities. Constraint 1 uses a global bin-packing constraint in the form BIN_PACKING (*items*, *item_weights*, *bin_loads*, *bin_capacities*) [36]. It ensures that each exam is assigned to a time-slot, and each time-slot has enough seats to accommodate the students sitting the assigned exams considering their corresponding tags. (see Requirements H1, H3 and H5). Subsequently, Constraint 2 ensures that each exam respects the duration of the time-slot it is assigned to (see Requirement H1) [32].

$$\begin{gathered}\text{BIN\_PACKING}(EP, size^N, L, cap^P),\\ \text{BIN\_PACKING}(EP, size^{SPR}, L^{SPR}, cap^{SPR}),\\ \text{BIN\_PACKING}(EP, size^{SHR}, L^{SHR}, cap^{SHR}),\\ \text{BIN\_PACKING}(EP, size^{LAB}, L^{LAB}, cap^{LAB}).\end{gathered} \tag{1}$$

$$\forall e \in E, \forall p \in P : \text{if } dur_p^P < dur_e^E, \text{ then } EP[e] \neq p. \tag{2}$$

One Exam at a Time. Let $\mathcal{C}^{\mathcal{G}}$ denote the set of all maximal cliques on the conflict graph $\mathcal{G}$. For each clique c, $EP_c \subset EP$ denotes a subset of decision variables that correspond to the ones for the exams in c, such that $\forall c \in \mathcal{C}^{\mathcal{G}}, \forall e \in c : EP[e] \in EP_c$. Constraint 3 ensures conflicting exams are scheduled into a different time-slot (see Req. H2).

$$\forall c \in \mathcal{C}^{\mathcal{G}} : \text{ALL\_DIFFERENT}(EP_c). \tag{3}$$

Specific Exam Times. Constraint 4 ensures that the exams with same group id are co-scheduled (see Requirement H6) [17]. We denote by EP_g a subset of the decision variables that have the same group-id g. More formally, the set of exams to be co-scheduled denoted by EP_g, is defined as $\forall g \in G, \forall e$ with group-id $g : EP[e] \in EP_g$.

$$\forall g \in G : \text{ALL\_EQUAL}(EP_g). \tag{4}$$

Specific Requests. Req. H7 and H8 can be easily modelled over the decision variables of the relevant exams using equality and inequality constraints. For instance, for Req. H8, let H_p denote a set of exam pairs that indicate priority as: $\langle e_i, e_j \rangle \in H_p$, where e_i must be scheduled before e_j. We model them as follows: $\forall \langle e_i, e_j \rangle \in H_p : EP[e_i] < EP[e_j]$.

Time-Slot Spread. There are four requirements related to time-slot spread (see Requirements S2, S3, S4, and H10). Recall that, there are exactly three time-slots in each day. Hence, Constraint 5 expresses the day on which an exam is scheduled. Consequently, we make use of some logical constraints to calculate the penalties incurred by the violation of Requirement S2 by Constraint 6, of Requirement S3 by Constraint 7, and of Requirement S4 by Constraint 8. Note that, UCC does not hold exams at the weekend. Thus, when calculating the violation of Requirement S4 by Constraint 8, we ignore the cases where two time-slots are separated by the weekend.

$$\forall e \in E : ED[e] = \lfloor EP[e]/3 \rfloor. \tag{5}$$

$$\forall a = [e_i, e_j] \in \mathcal{E} : \text{if } |ED[e_i] - ED[e_j]| < 2, \text{ then } V_{S2}[a] = w(a), \text{ else } V_{S2}[a] = 0. \tag{6}$$

$$\forall a = [e_i, e_j] \in \mathcal{E} : \text{if } ED[e_i] == ED[e_j], \text{ then } V_{S3}[a] = w(a), \text{ else } V_{S3}[a] = 0. \tag{7}$$

$$\forall a = [e_i, e_j] \in \mathcal{E} : \text{if } |EP[e_i] - EP[e_j]| < 2 \wedge \neg weekend(EP[e_i], EP[e_j]), \text{ then } V_{S4}[a] = w(a), \text{ else } V_{S4}[a] = 0. \tag{8}$$

Constraint 9 ensures that Requirement H10 is satisfied. In order to avoid repeating, we first analyse the sets of exams E_s for each student s as a pre-processing step. If there are any repeated sets, or if a set is already included in a larger set, we eliminate the repetition, or the subset. We denote by A^E the refined

set of the E_s that does not include any repeated sequences. For Constraint 9, let each set of exams in A^E be denoted by l, where for some student s, $l = E_s$. Note that, UCC only has exams of duration 90 or 180 min. For each l, for any consecutive two-day block $b_i = \langle d_i, d_{i+1} \rangle$, let $len^l_{b_i}[e]$ be an integer variable with domain $\{0, 90, 180\}$ that denotes the duration of exam e for all $e \in l$. The value of $len^l_{b_i}[e] = 0$ if the exam e is not scheduled for the days d_i or d_{i+1} in the given block b_i. Otherwise, $len^l_{b_i}[e] = dur^E_e$.

$$\begin{gathered} \forall b_i = \{d_i, d_{i+1}\} \text{ with } 0 \leq i < |D| - 1, \forall l \in A^E : \\ \forall e \in l : \text{ if } ED[e] \in b_i, \text{ then } len^l_{b_i}[e] = dur^E_e, \text{ else } len^l_{b_i}[e] = 0, \\ \textstyle\sum_{e \in l} len^l_{b_i}[e] \leq 270. \end{gathered} \tag{9}$$

Time-Slot Preference. Some time-slots are more preferred than others, as discussed in Requirement S5. Therefore, each time-slot p has an associated penalty coefficient α_p. We calculate the penalty caused by scheduling exams into less preferred time-slots p as the total number of students sitting an exam during p multiplied by α_p, as shown in Constraint 10.

$$obj_5 = \sum_{p \in P} (L[p] + L^{SPR}[p] + L^{SHR}[p] + L^{LAB}[p]) \times \alpha_p \tag{10}$$

Let F_s denote a value representing the size of an exam provided by the institution that determines if an exam is considered large or not; exams with sizes larger than or equal to F_s are said to be *large*. Similarly, let p_σ denote the time-slot after which exams are considered as being scheduled late. For each exam e, $pl[e]$ denotes if a penalty is incurred by scheduling of exam e. We capture Requirement S6 using Constraint 11. The value of obj_6 represents the overall penalty incurred by the violation of this requirement.

$$\begin{gathered} \forall e \in E \text{ with } size^E_e \geq F_s : \text{ if } EP[e] - p_\sigma > 0, \text{ then } pl[e] = 1, \text{ else } pl[e] = 0. \\ obj_6 = \sum_{e \in E \text{ with } size^E_e \geq F_s} pl[e]. \end{gathered} \tag{11}$$

Objective Function. The objective is to minimise the number of students that are affected by the soft constraint violations. The overall objective function is defined as a weighted sum of all the aggregated penalties $obj_2, obj_3, obj_4, obj_5, obj_6$ multiplied by their respective coefficients $c_{p2}, c_{p3}, c_{p4}, c_{p5}, c_{p6}$. Constraint 12 expresses the objective of Phase 1 model.

$$\begin{gathered} obj_2 = \sum_{a \in \mathcal{E}} V_{S2}[a], obj_3 = \sum_{a \in \mathcal{E}} V_{S3}[a], obj_4 = \sum_{a \in \mathcal{E}} V_{S4}[a]. \\ \textit{minimise} \quad obj = c_{p2} \times obj_2 + c_{p3} \times obj_3 + c_{p4} \times obj_4 + c_{p5} \times obj_5 + c_{p6} \times obj_6. \end{gathered} \tag{12}$$

3.2 Phase 2: Room Allocation

In Phase 2, the aim is to find a room for each exam by respecting the room-related requirements. We further decompose the model at this stage by days and find a room for each exam for each day. The problem we encounter at this stage is a version of the bin-packing problem, where the items can be fragmented (or split). In the literature, this problem is known as Bin Packing with Fragmentable Items (BPFI) and is NP-hard [21]. The objective in the BPFI problem is to either minimise the number of fragments or to minimise the number of bins required. In our case, we have both of these as objectives (see Requirements S1 and S7). The literature contains some approximation algorithms as well as MIP models for the problem [10,11,19]. Additionally, UCC has a hard requirement that mixed-duration exams cannot be timetabled in the same venue (see Requirement H5). This constraint makes the problem even more constrained, as not all the items can be placed in all bins. This version of the problem has previously been identified as Fragmentable Group and Item Bin Packing with Compatibility Preferences [2]. Although these bin-packing variants have been identified and tackled before, there are no global constraints available that express them.

- For each exam e in E, constant $\boldsymbol{EP_e} \in P$ denotes the time-slot in which exam e is held, and constant $\boldsymbol{ED_e} \in D$ denotes the day in which e is held. These are provided as input to the model by the Phase 1 solution.
- An integer value $\boldsymbol{R_{ex}}$ in R denotes the unique identifier of R_{ex} room.
- For each p in P in any room r, depending on the special needs tag of the room, one of the following integer variables: $\boldsymbol{RL[p][r]}$, $\boldsymbol{RL^{SPR}[p][r]}$, $\boldsymbol{RL^{SHR}[p][r]}$, or $\boldsymbol{RL^{LAB}[p][r]}$ is defined to denote the total number of students with the given tag sitting an exam in room r during time-slot p. Note that, the domain of these variables is $\{0, ..., cap_r^R\}$.
- For each $p \in P$ and for each room r for main groups in R, set variable $\boldsymbol{D[p][r]}$ denotes the different duration types of exams to be held in r during p. Note that, $D[p][r] \in \mathbb{P}(\{90, 180\})$.
- For each room r and $e \in E$, one of the following integer variables is defined depending on the tag of the room: $\boldsymbol{W[r][e]}, \boldsymbol{W^{SPR}[r][e]}, \boldsymbol{W^{SHR}[r][e]}$, or $\boldsymbol{W^{LAB}[r][e]}$ to denote the total number of students with the given tag sitting e in r. The domains are defined as $W[r][e] \in \{0, ..., min(cap_r^R, size_e^N)\}$, $W^{SHR}[r][e] \in \{0, ..., min(cap_r^R, size_e^{SHR})\}$, $W^{SPR}[r][e] \in \{0, 1\}$, and $W^{LAB}[r][e] \in \{0, ..., min(cap_r^R, size_e^{LAB})\}$. Search branches on these variables.
- For each $e \in E$, set variables $\boldsymbol{RS[e]}$, $\boldsymbol{RS^{SHR}[e]}$, and $\boldsymbol{RS^{LAB}[e]}$ denote the set of rooms used for examinations of the main, SHR, and LAB groups for exam e, where $RS[e] \in \mathbb{P}(R)$, $RS^{SHR}[e] \in \mathbb{P}(R^{SHR})$, and $RS^{LAB}[e] \in \mathbb{P}(R^{LAB})$.
- Sub-objectives $\boldsymbol{obj_1}, \boldsymbol{obj_2}, \boldsymbol{obj_3}$ are used to denote the total number of exam splits for main, SHR, and LAB groups, respectively. Additionally, the sub-objective $\boldsymbol{obj_4}$ denotes the total number of main rooms used during the examination process. The overall objective $\boldsymbol{obj}$ is a weighted-sum of the sub-objectives.

We model bin packing with fragmentable items using a straight-forward approach under the name BIN_PACKING_FI(X, I_W, L, C) using Constraint 13 and Constraint 14. Variable X corresponds to the weight of each item in each available bin given item set I and bin set B. Additionally, I_W denotes the weight of each item, L denotes the load of each bin, and C denotes the capacity of each bin. Constraint 13 ensures the weight of each item in each bin respects the capacity of the bin. Additionally, Constraint 14 ensures all the weight of each item is packed. Note that, BIN_PACKING_FI can be improved by using better models or heuristic approaches [7,11].

$$\forall b \in B : \sum_{i \in I} X[b][i] = L[b] \text{ and } L[b] \leq C[b]. \tag{13}$$

$$\forall i \in I : \sum_{b \in B} X[b][i] = I_W[i]. \tag{14}$$

Recall that we decompose this phase further by separately assigning exams in each day to rooms. Thus, the remaining constraints in this section are run independently for each day using the set of exams given on that day. We denote by d, the day for which the problem is being solved and by $p \in d$ the three time-slots that belong to d.

Constraint 15 ensures each student in each exam is assigned a room by respecting Requirements H3, H5, and H9. For each period, let W_p be a list of decision variables that correspond to a list of $W[r][e]$ variables for exams e with $EP_e = p$ in rooms $r \in R$. Similarly, let W_p^{SPR}, W_p^{SHR}, and W_p^{LAB} denote decision variables $W^{SPR}[r][e], W^{SHR}[r][e]$, and $W^{LAB}[r][e]$, for each exam e with $EP_e = p$ in room r with the respective tag. We denote by $C^N, C^{SPR}, C^{SHR}, C^{LAB}$ the list of capacities of rooms with the respective tag, where N corresponds to the main group. Constraint 15 is repeated for each time-slot p in the given day d.

$$\begin{gathered}
\forall p \in d : \text{BIN_PACKING_FI}(W_p, size^N, RL, C^N). \\
\forall p \in d : \text{BIN_PACKING_FI}(W_p^{SPR}, size^{SPR}, RL^{SPR}, C^{SPR}). \\
\forall p \in d : \text{BIN_PACKING_FI}(W_p^{SHR}, size^{SHR}, RL^{SHR}, C^{SHR}). \\
\forall p \in d : \text{BIN_PACKING_FI}(W_p^{LAB}, size^{LAB}, RL^{LAB}, C^{LAB}).
\end{gathered} \tag{15}$$

No Mixed Durations. We implement item-bin compatibility using Constraint 16. If multiple exams are held in the same room during the same time-slot, their duration must be the same (see Requirement H5). In order to achieve this, Constraint 16 uses $D[p][r]$ that keeps track of the different durations of exams being held in room r at time-slot p.

$$\begin{gathered}
\forall p \in d, \forall r \in R, \forall e \in E \text{ with } EP_e = p : r \in RS[e] \rightarrow dur_e^E \in D[p][r]. \\
\forall p \in d, \forall r \in R, \forall e \in E \text{ with } EP_e = p : |D[p][r]| \leq 1.
\end{gathered} \tag{16}$$

Room Loads and Different Campus R_{ex}. In order to implement the remote campus requirements, i.e. R_{ex}, and to find the number of rooms in use, we keep track of the set of rooms $RS[e]$ in which each exam e is held. If there are a number of students at exam e in room r, then r must be included in the set of rooms of e as detailed in Constraint 17.

$$\begin{array}{c} \forall e \in E, \forall r \in R \text{ with } ED_e = d : W[r][e] > 0 \leftrightarrow r \in RS[e]. \\ \forall e \in E, \forall r \in R^{SHR} \text{ with } ED_e = d : W^{SHR}[r][e] > 0 \leftrightarrow r \in RS^{SHR}[e]. \\ \forall e \in E, \forall r \in R^{LAB} \text{ with } ED_e = d : W^{LAB}[r][e] > 0 \leftrightarrow r \in RS^{LAB}[e]. \end{array} \quad (17)$$

Shachnai et al. describe a primitive solution to a BPFI as a feasible packing such that each bin contains at most two fragmented items, and each item is fragmented at most once [35]. They also show that there always exists an optimal solution for BPFI, that is primitive. Constraint 18 ensures each exam is split at most once.

$$\forall e \in E : RS[e] \leq 2 \text{ and } RS^{SHR}[e] \leq 2 \text{ and } RS^{LAB}[e] \leq 2. \quad (18)$$

Constraint 19 ensures for any two conflicting exams that are to be held on the same day, either both are held in R_{ex} or none of them (see Requirement H11). Additionally, Constraint 20 implements the second R_{ex} constraint, that states if an exam is held in R_{ex}, then it is not split across other rooms (see Requirement H4).

$$\forall a = [e_i, e_j] \in \mathcal{E} \text{ with } ED_{e_i} = ED_{e_j} = d : R_{ex} \in RS[e_i] \leftrightarrow R_{ex} \in RS[e_j] \quad (19)$$

$$\forall e \in E : R_{ex} \in RS[e] \rightarrow |RS[e]| = 1 \quad (20)$$

Objective Function. The sub-objectives in this model are to minimise the exam splits for main, SHR, and LAB groups (obj_1, obj_2, obj_3, resp.), and to also minimise the total number of main rooms in use (obj_4). Constraint 21 formulates these objectives. The objective function uses coefficients c_{r1}, c_{r2}, c_{r3}, and c_{r4} as the weight of each sub-objective.

$$\begin{array}{c} obj_1 = \sum_{e \in E \text{ with } EP_e = p} |RS[e]|, \; obj_2 = \sum_{e \in E \text{ with } EP_e = p} |RS^{SHR}[e]|, \\ obj_3 = \sum_{e \in E \text{ with } EP_e = p} |RS^{LAB}[e]|, \; obj_4 = \sum_{p \in d} | \bigcup_{e \in E \text{ with } EP_e = p} RS[e]|. \\ \textit{minimise } obj = c_{r1} \times obj_1 + c_{r2} \times obj_2 + c_{r3} \times obj_3 + c_{r4} \times obj_4. \end{array} \quad (21)$$

3.3 Discussion of the Decomposition Approach

One of the main drawbacks of decomposed models is that obtaining the optimal solution or any solution, in general, is not always guaranteed. The main challenge for our CP model is that a solution produced by the Phase 1 model may lead to infeasibility in Phase 2. We illustrate a scenario below to exemplify this for the reader.

An Infeasibility Scenario. Suppose that there are four exams e_1, e_2, e_3, and e_4 to be held during the same time-slot, where each exam has three students enrolled with no special needs. There are only three rooms r_1, r_2, and r_3 with equal capacities $c_1 = c_2 = c_3 = 4$. Let exam e_4 be 180-min long and the rest be 90-min. Also let room r_1 denote the R_{ex} room. Although this solution is a feasible solution for Phase 1, it is infeasible for the Phase 2 model. An illustration of two alternative exam-room allocation scenarios is given in Fig. 1.

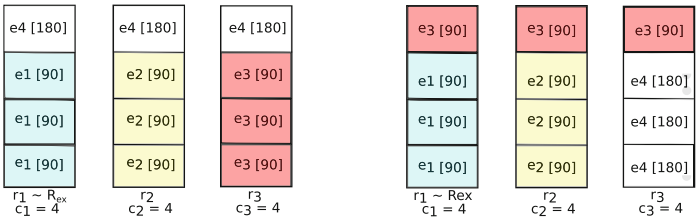

Fig. 1. Two different infeasibility scenarios where the allocation of the exams in the rooms causes the violation of Requirements H4 and H5.

Both of these cases results in infeasibility for the Phase 2 model due to the violation of Requirement H4 and H5, where an exam to be held in R_{ex} is split, and also there are mixed-duration exams in at least one room. A similar, alternative infeasibility scenario occurs when all the exams are 90-min long, but the time-slot load is filled to the capacity. In this case, one of the exams must be split to R_{ex}, violating Requirement H4.

There are different strategies to tackle this problem. One approach is to develop a Benders decomposition of the problem. However, this approach is challenging as Phase 2 is a hard problem in itself, but imposes an interesting research direction. An alternative approach is to relax some of the hard constraints in the Phase 2 model. As discussed above, when the load of a time-slot is large, it becomes more difficult to find a solution for Phase 2 due to the hard "no mixed duration" requirement (see Requirement H5) and R_{ex} requirements (see Requirements H4 and H11). Hence, if the Phase 1 solution has a large load for some time-slots, then the "no mixed duration" constraint (see Constraint 16) and/or R_{ex} constraints (see Constraints 19 and 20) may be relaxed and their penalties can be calculated and added to the objective for minimisation.

Alternatively, we can keep the load of each period at an 'acceptable level', where this level is defined with respect to the dataset in use. If the time-slot loads are kept as small as possible, then some rooms can be left unused. A basic solution that emanates from this observation is to keep the load of each time-slot to a value that avoids the use of R_{ex} room, and also accommodate some empty-seats to help with the "no mixed duration" requirement. Hence, we add a sub-objective to minimise the maximum excess load for time-slots, where the excess load is calculated over the main groups as the difference between the load and the mean value of seats required for each period. It can then be possible

to find schedules without relaxing any of the hard constraints. Note that, by minimising the maximum excess load, the model can leave some rooms empty. Therefore, this sub-objective also helps with minimising the total number of used rooms in Phase 2. Recall that in Phase 1, for each time-slot p, $L[p]$ denotes the load of p. Minimising the excess load can be achieved by adding Constraint 22 to the Phase 1 model, and also adding sub-objective obj_1 to obj by using a coefficient c_{p1}.

$$mean = \sum_{r \in R} cap_r^R / |P|, \text{ and } obj_1 = \text{MAX}(L) - mean. \tag{22}$$

4 Empirical Results

We used the Choco 4.10.2 constraint solver to perform our experiments [30]. We used a library of graph theory data structures and algorithms called JGraphT for the creation of conflict graph, and for finding all maximal cliques [24]. All experiments were performed on a Dell i7-8700 machine with 3.20 GHz processors under Linux.

The model we developed was given 60-min in total to produce a solution, spending 45 min on the assignment of examinations to time-slots, and the remaining 15 min on the examination-room assignment where each day is limited to produce a solution in 1.5 min. Note that, none of the solutions we report in this section was proven to be optimal by the solver. Therefore, it may be possible to obtain better solutions, i.e. solutions with lower objective values, if more time is allowed for the models or some filtering, symmetry breaking, and redundant constraints, are added to the model.

We tested the Phase 1 model using four different built-in search strategies selected by observation and considering the nature of the problem: activity-based search ($\boldsymbol{s_{act}}$), dominated weighted degree ($\boldsymbol{s_{wdeg}}$), assigning the first non-instantiated variable to its lower bound ($\boldsymbol{s_{ilb}}$), and assigning the non-instantiated variable of smallest domain size to its lower bound ($\boldsymbol{s_{mdlb}}$) [5,25]. In addition to each search strategy, we used a large-neighbourhood search based on randomly creating neighbours using a restart strategy based on fails. The search also makes use of a last conflict heuristic and geometrical restarts. All search strategies used are available in the Choco library.

In the UCC data, the mean of total seats required per time-slot is reasonably low: it is approximately 1341 for main, 9 for SPR, 28 for LAB, and 54 for SHR groups. The total number of available seats for the main groups per time-slot is 1900, and there are 31 seats for SPR, 57 for LAB students, 47 for SHR students, where 3 seats from each room are left empty for ad-hoc students (see Requirement H12). It is important to note that the model with the specific venue for SHR students is not enough and renders the model infeasible. Therefore, we manually add an additional venue for SHR students.

In Table 1, we present the performance of the four search strategies mentioned above on the Phase 1 model. The coefficients for the objectives are used

as follows: $c_{p1} = 25$, $c_{p2} = 3$, $c_{p3} = 40$, $c_{p4} = 10$, $c_{p5} = 2$, and $c_{p6} = 60$. We use $F_s = 100$ to decide if an exam is large or not. In UCC, afternoon time-slots in Days 7 and 8, and also all time-slots on Days 9 and 10, are less preferred to earlier ones. It is preferred to not schedule any examinations during the last time-slot of Days 9 and 10. So, we penalise the preference of time-slots as: $\alpha_{20} = \alpha_{23} = \alpha_{24} = \alpha_{25} = \alpha_{27} = \alpha_{28} = 1$, and $\alpha_{26} = \alpha_{29} = 3$. We use large coefficients to penalise the sub-objectives obj_1 and obj_3 in Phase 1 for an increased chance of avoiding the infeasibility scenario discussed in Sect. 3.3. The penalty values were selected based on the observation that reflects the requirements and priorities of the institution. Although the importance of obj_3 is less than obj_2 in the given requirements, penalising obj_3 is essential for providing a better resolution between the phases for the problem. Additionally, one can observe that if obj_1 has a larger value, then the time-slot preferences (obj_5) are violated less. The CP model for Phase 1 contains 527,831 variables and 367,303 constraints in total.

Table 1. Performance comparison of different search strategies for Phase 1 model on UCC dataset.

	s_{act}	s_{wdeg}	s_{ilb}	s_{mdlb}
Best-solution objective ($obj - b$)	60,203	64,254	60,888	62,514
Load balancing penalty (obj_1)	292	135	218	145
Maximum 1 exam every 2 days penalty (obj_2)	7,897	9,079	7,294	8,587
Maximum 1 exam per day penalty (obj_3)	290	297	325	283
Back-to-back exams penalty (obj_4)	152	178	252	139
Time-slot preference penalty (obj_5)	6,756	8,611	7,788	8,649
Front load penalty (obj_6)	43	46	41	52
Best-solution time ($time_b$ in sec)	1,308.4	1,335.4	2,313.7	1,158.3
Ratio: feasible/total solutions (cnt_{feas}/cnt_{sol})	5/5	11/11	5/5	10/10
First-solution objective ($objf$)	64,997	71,024	61,994	69,978
First-solution time ($time_f$ in sec)	7	10	8.8	8.8

We observe from Table 1 that the s_{ilb} strategy has a good starting point but s_{act} is able to find a better solution in the end. Recall that we branch on EP variables that allocate exams to time-slots starting from the largest exam to the smallest one. Thus, assigning the first non-instantiated variable to its lowest bound corresponds to scheduling a large exam to an earlier time-slot. Therefore, we observe a clear advantage for s_{ilb} in obj_6, that corresponds to the front load penalty. In each case, the solver can find an initial solution for the Phase 1 model within 10 s . Also note that in this table we report how many feasible solutions there are (cnt_{feas}) among all solutions found by each strategy (cnt_{sol}).

We measure feasibility by running the Phase 2 algorithm for each solution found in Phase 1. We immediately stop the search when an initial solution is

found by the Phase 2 model. The time required by this feasibility check using the s_{iub} strategy in Phase 2 for a solution obtained from Phase 1 took on the average 1.2 s, including building the model and search, with the worst time being 12.2 s; details are discussed below. We allow this feasibility check to run for a maximum of 1.5 min per day. If no solutions are found within the given time limit or if the model is proven as infeasible, we say that the Phase 1 solution is infeasible, and increment cnt_{feas}. In our experiments, all the solutions found in Phase 1 were feasible. For comparison, we evaluated the original UCC timetable for Phase 1 by ignoring obj_1 as it is not an original constraint given by the institution, which minimises the excess load. When we evaluated the corresponding $obj_2, obj_3, obj_4, obj_5$, and obj_6 values with the coefficients mentioned above for Phase 1, the original timetable is found to have an objective value $obj = 117,342$. This value is nearly twice worse than any of the initial solutions found by CP in less than 10 s.

Our Phase 2 model per day has on the average 10,592 variables and 5,139 constraints. We allow Phase 2 to run for 15 min in general, which results in each day reporting a solution in 1.5 min. For a search strategy comparison for Phase 2, we use the best solution found by Phase 1 model, i.e. the one with the lowest obj value, and use it as input for the Phase 2 model. We again test four different search strategies on this model: activity based search ($\boldsymbol{s_{act}}$), dominated weighted degree ($\boldsymbol{s_{wdeg}}$), assigning the first non-instantiated variable to its upper bound ($\boldsymbol{s_{iub}}$), and assigning the non-instantiated variable of smallest domain size to its upper bound ($\boldsymbol{s_{mdub}}$). Note that, we do not use the strategies that assign a variable to its lower bound, but we use the assignment to upper bound in Phase 2. Considering that we branch on the decision variables that denote the number of students sitting an exam in each room, an upper bound strategy assigns the maximum possible number of students into the next available room in a Next-Fit manner. A split of the main groups are heavily penalised in UCC, so we use the following coefficients for the penalisation of sub-objective values in Phase 2: $c_{r1} = 4, c_{r2} = c_{r3} = c_{r4} = 1$. The search strategy s_{act} was unable to find a solution to 1/10 days, s_{wdeg} to 6/10 days, and s_{mdub} to 8/10 days, within the given time-limit. The strategy s_{iub} has a clear advantage over the other strategies as it finds a very quick solution to each day by assigning as many students as possible from the same group into the first available room. The aggregated solution found by s_{iub} has $obj_1 = 8$, $obj_2 = 0$, $obj_3 = 0$, $obj_4 = 156$, and all the hard constraints are satisfied.

Table 2 summarises the improvements over the UCC's original schedule by the final schedule found by our model. In this table, $\mathcal{M}$ represents a solution found by our CP model using the combination of s_{act} for Phase 1 and s_{iub} for Phase 2. Note that, if desired, using different penalty values for the soft constraints, it is possible to adjust the model further to obtain improvement on a specific value. The model we propose effectively reduces the violation of soft requirements related to student preferences. In model $\mathcal{M}$, the average preparation time for students between their consecutive exams is increased. Additionally, it reduces the total number of rooms in use, which helps the institution lower costs, and

Table 2. A comparison of the expert-generate schedule at UCC with our proposed schedule ($\mathcal{M}$).

Number of ...	UCC	$\mathcal{M}$
Students that sit more than one exam in the same time-slot	7	0
Conflicting exam pairs and held during the same time-slot	5	0
<Main, SPR, SHR, LAB> rooms used in total	<197, 269, 107, 64>	<165,270,50,27>
Time-slots used during the whole examination process	28	28
Average number of time-slots a student has between two exams	6.02	6.46
Students that sit more than 270-min exam in any 2-day block	34	0
Large exams scheduled after the fifth day	42	43
Students that sit an exam during the less preferred time-slots	8,253	6,756
Splits of <Main, SHR, LAB> groups per exam	<29, 88, 34>	<8, 0, 0>
Exams that are held in different campuses	1	0
Students that sit exams on the same day in different campuses	0	0
Cases where the room capacity is exceeded	4	0
Main rooms in which mixed-duration exams are held	1	0
Violation of Requirement S2: at most one exam every two days		
Affected conflicting exam pairs	1,090	943
Affected students (may include one student more than once)	13,056	7,897
Affected students (a student is counted only once)	6,811	4,955
Violation of Requirement S3: at most one exam every day		
Affected conflicting exam pairs	196	104
Affected students (may include one student more than once)	939	290
Affected students (a student is counted only once)	872	284
Violation of Requirement S4: no back-to-back exams		
Affected conflicting exam pairs	228	85
Affected students (may include one student more than once)	1,856	152
Affected students (a student is counted only once)	1,687	148

also reduces invigilator resource requirements. The exam splits are significantly reduced. It is important to note that, in the evaluation phase of the proposed solution, the expert in UCC expressed that the reason to a large number of exam splits in UCC may be due to accommodating the ad-hoc students. However, UCC data does not let us identify these students.

A final remark is that we also implemented a preliminary version of this two-phase model on a black-box local-search solver, namely LocalSolver [1]. We observed that LocalSolver is producing quick solutions when compared to our CP model if the model is mono-objective. However, when the objective is represented as a weighted sum as in our case, LocalSolver does not have a clear advantage over the CP model. Therefore, considering that both our phases are multi-objective, we did not implement the final version on LocalSolver. It will be an interesting direction to implement this model on different solvers and present a performance comparison.

5 Conclusion and Future Work

In this work, we provide a model for the examination timetabling problem at University College Cork (UCC). Currently, examination scheduling in UCC is done manually by expert staff but it takes weeks. The proposed automated model is shown to improve the current timetable in terms of fairness, satisfying hard requirements, and the allocation of resources. We use the Constraint Programming framework to model our two-phase approach, mainly based on bin-packing constraints. The proposed timetable has been evaluated by the administrative staff, and integration work is in progress.

An advantage of the proposed model is its adaptability to real-world situations, such as the crisis caused by the COVID-19 pandemic. The pandemic-related constraints include mandatory exam splits due to social distancing and the capacity of venues being reduced. It is possible to tailor this model for the needs of an institute and find the number of rooms required to be able to hold physical exams. The performance of the proposed model can be improved further by using more sophisticated filtering, and symmetry-breaking techniques. We plan to develop heuristic approaches and compare their performance with the proposed CP model. Considering the combinatorial and difficult nature of bin-packing with fragmentable items, it is interesting to focus on developing global constraints with efficient filtering techniques that reduce the search space.

The proposed framework is not only applicable to UCC, but by adjusting the constraints for local institutional preferences, it should be straightforward to adapt to other institutions.

Acknowledgement. Special thanks to Margo Hill, the Examinations Administrator, and Siobhán Cusack, Head of Student Records and Examinations Office at University College Cork. We thank Léa Blaise from the LocalSolver team for their efforts, comments, and suggestions for improving our model, as well as the anonymous reviewers for their valuable feedback. This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grants 16/RC/3918

and 12/RC/2289-P2 which are co-funded under the European Regional Development Fund.

References

1. Localsolver 9.0 (2019). http://www.localsolver.com/
2. Adeyemo, A.: Fragmentable group and item bin packing with compatibility preferences. In: Proceedings of the 2015 International Conference on Industrial Engineering and Operations Management (2015)
3. Arbaoui, T., Boufflet, J.P., Moukrim, A.: Preprocessing and an improved MIP model for examination timetabling. Ann. Oper. Res. **229**(1), 19–40 (2015)
4. Asmuni, H., Burke, E.K., Garibaldi, J.M., McCollum, B.: Fuzzy multiple heuristic orderings for examination timetabling. In: Burke, E., Trick, M. (eds.) PATAT 2004. LNCS, vol. 3616, pp. 334–353. Springer, Heidelberg (2005). https://doi.org/10.1007/11593577_19
5. Boussemart, F., Hemery, F., Lecoutre, C., Sais, L.: Boosting systematic search by weighting constraints. In: In European Conference on Artificial Intelligence (ECAI 2004) (2004)
6. Bron, C., Kerbosch, J.: Algorithm 457: finding all cliques of an undirected graph. Commun. ACM **16**(9), 575–577 (1973)
7. Byholm, B., Porres, I.: Fast algorithms for fragmentable items bin packing. J. Heuristics **24**(5), 697–723 (2018)
8. Carter, M.W., Laporte, G.: Recent developments in practical examination timetabling. In: Burke, E., Ross, P. (eds.) PATAT 1995. LNCS, vol. 1153, pp. 1–21. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-61794-9_49
9. Carter, M.W., Laporte, G., Chinneck, J.W.: A general examination scheduling system. Interfaces **24**(3), 109–120 (1994)
10. Casazza, M., Ceselli, A.: Mathematical programming algorithms for bin packing problems with item fragmentation. Comput. Oper. Res. **46**, 1–11 (2014)
11. Casazza, M., Ceselli, A.: Exactly solving packing problems with fragmentation. Comput. Oper. Res. **75**, 202–213 (2016)
12. Cataldo, A., Ferrer, J.C., Miranda, J., Rey, P.A., Sauré, A.: An integer programming approach to curriculum-based examination timetabling. Ann. Oper. Res. **258**(2), 369–393 (2017)
13. De Haan, P., Landman, R., Post, G., Ruizenaar, H.: A four-phase approach to a timetabling problem in secondary schools. Pract. Theory Autom. Timetabling VI **3867**, 423–425 (2006)
14. Duong, T.A., Lam, K.H.: Combining constraint programming and simulated annealing on university exam timetabling. In: RIVF, pp. 205–210. Citeseer (2004)
15. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York (1979)
16. Gogos, C., Alefragis, P., Housos, E.: An improved multi-staged algorithmic process for the solution of the examination timetabling problem. Ann. Oper. Res. **194**(1), 203–221 (2012)
17. Hebrard, E., O'Sullivan, B., Razgon, I.: A soft constraint of equality: complexity and approximability. In: Stuckey, P.J. (ed.) CP 2008. LNCS, vol. 5202, pp. 358–371. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85958-1_24
18. Kasm, O.A., Mohandes, B., Diabat, A., El Khatib, S.: Exam timetabling with allowable conflicts within a time window. Comput. Indu. Eng. **127**, 263–273 (2019)

19. LeCun, B., Mautor, T., Quessette, F., Weisser, M.A.: Bin packing with fragmentable items: presentation and approximations. Theor. Comput. Sci. **602**, 50–59 (2015)
20. Lotfi, V., Cerveny, R.: A final-exam-scheduling package. J. Oper. Res. Soc. **42**(3), 205–216 (1991)
21. Mandal, C.A., Chakrabarti, P.P., Ghose, S.: Complexity of fragmentable object bin packing and an application. Comput. Math. Appl. **35**(11), 91–97 (1998)
22. McCollum, B., McMullan, P., Parkes, A.J., Burke, E.K., Qu, R.: A new model for automated examination timetabling. Ann. Oper. Res. **194**(1), 291–315 (2012)
23. Merlot, L.T.G., Boland, N., Hughes, B.D., Stuckey, P.J.: A hybrid algorithm for the examination timetabling problem. In: Burke, E., De Causmaecker, P. (eds.) PATAT 2002. LNCS, vol. 2740, pp. 207–231. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45157-0_14
24. Michail, D., Kinable, J., Naveh, B., Sichi, J.V.: JGrapht-A Java library for graph data structures and algorithms. arXiv preprint arXiv:1904.08355 (2019)
25. Michel, L., Van Hentenryck, P.: Activity-based search for black-box constraint programming solvers. In: Beldiceanu, N., Jussien, N., Pinson, É. (eds.) CPAIOR 2012. LNCS, vol. 7298, pp. 228–243. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29828-8_15
26. Miller, H.E., Pierskalla, W.P., Rath, G.J.: Nurse scheduling using mathematical programming. Oper. Res. **24**(5), 857–870 (1976)
27. Mirrazavi, S.K., Mardle, S.J., Tamiz, M.: A two-phase multiple objective approach to university timetabling utilising optimisation and evolutionary solution methodologies. J. Oper. Res. Soc. **54**(11), 1155–1166 (2003)
28. Müller, T.: Real-life examination timetabling. J. Sched. **19**(3), 257–270 (2016)
29. Pillay, N., Banzhaf, W.: An informed genetic algorithm for the examination timetabling problem. Appl. Soft Comput. **10**(2), 457–467 (2010). https://doi.org/10.1016/j.asoc.2009.08.011
30. Prud'homme, C., Fages, J.G., Lorca, X.: Choco Documentation. TASC - LS2N CNRS UMR 6241, COSLING S.A.S. (2017). http://www.choco-solver.org
31. Qu, R., Burke, E.K., McCollum, B., Merlot, L.T., Lee, S.Y.: A survey of search methodologies and automated system development for examination timetabling. J. Sched. **12**(1), 55–89 (2009)
32. Régin, J.C.: A filtering algorithm for constraints of difference in CSPs. AAAI. **94**, 362–367 (1994)
33. Ribić, S., Konjicija, S.: A two phase integer linear programming approach to solving the school timetable problem. In: Proceedings of the ITI 2010, 32nd International Conference on Information Technology Interfaces, pp. 651–656. IEEE (2010)
34. Schaerf, A.: A survey of automated timetabling. Artif. Intell. Rev. **13**(2), 87–127 (1999)
35. Shachnai, H., Tamir, T., Yehezkely, O.: Approximation schemes for packing with item fragmentation. Theory Comput. Syst. **43**(1), 81–98 (2008)
36. Shaw, P.: A constraint for bin packing. In: Wallace, M. (ed.) CP 2004. LNCS, vol. 3258, pp. 648–662. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30201-8_47
37. Yasari, P., Ranjbar, M., Jamili, N., Shaelaie, M.H.: A two-stage stochastic programming approach for a multi-objective course timetabling problem with courses cancelation risk. Comput. Ind. Eng. **130**, 650–660 (2019)
38. Yin, L., Luo, J., Luo, H.: Tasks scheduling and resource allocation in fog computing based on containers for smart manufacturing. IEEE Trans. Indu. Inf. **14**(10), 4712–4721 (2018)