Modelling and Solving Multi-objective University Course Timetabling Problems

Matthew Davison, MRes, MMath(Hons.)

Submitted for the degree of Doctor of Philosophy at Lancaster University.

August 2025

Abstract

University module timetabling involves organising hundreds of staff and thousands of students whilst respecting complex scheduling constraints. Beyond resource allocation, this process requires balancing competing stakeholder preferences: students seeking convenient schedules, staff requiring flexibility, and administrators pursuing operational efficiency. This makes university module timetabling an inherently multi-objective, multi-stakeholder optimisation problem. The current literature inadequately addresses strategic decision-making and the emerging trend of hybrid teaching, in which students participate in activities both online and in-person. This thesis develops a novel integer programming model that incorporates hybrid teaching modes and proposes solution methods to find trade-offs between objectives. We introduce a student partitioning approach based on module requests that supports bound computation for objective functions and guides neighbourhood selection in our proposed matheuristic algorithm. This algorithm finds both lexicographic solutions that respect stakeholder priorities and Pareto frontier approximations that reveal objective trade-offs. Our framework transforms timetabling from purely operational scheduling into strategic decision support, enabling universities to evaluate policy alternatives and balance competing interests. We demonstrate practical applicability using benchmark instances and real-world data from a UK university, showing how multi-objective analysis can inform timetabling decisions.

Acknowledgements

Firstly, I would like to thank my supervisors, Konstantinos and Ahmed, for their unwavering support throughout this entire process. The knowledge and experience I have gained through our interactions extend beyond this work. Thank you both for providing valuable project insights and for being infinitely patient and understanding.

Secondly, I would like to thank my parents, Judith and Paul, and my brother, Adam. Words cannot express what you mean to me, and if they could, I would not know where to start. I would also like to thank all of my friends, both near and far. Whether it was training at the gym, watching or playing football, coming second in a pub quiz, or just enjoying time together, my life would not have been the same without any of you.

Finally, I would like to thank all of the STOR-i staff. I am grateful for all of the opportunities and training that STOR-i have provided. Thank you for allowing me to push the Geordie phrase *"shy bairns get nowt"* to its limit.

Declaration

I declare that the work in this thesis has been done by myself and has not been submitted elsewhere for the award of any other degree.

This thesis is constructed as a series of papers. Chapters 2, 3, and 4 can be read as individual contributions. However, these three chapters carry an overarching narrative with Chapter 3 building on Chapter 2 and Chapter 4 building on Chapter 3.

The content of Chapter 2 has been published as Davison, M., Kheiri, A., and Zografos, K.G. (2024). Modelling and solving the university course timetabling problem with hybrid teaching considerations. *Journal of Scheduling*.

The content of Chapter 3 has been submitted for publication as Davison, M., Kheiri, A., and Zografos, K.G. (2025). Multi-objective fix-and-optimise matheuristic for university course timetabling. This paper is under review.

Please note that to ensure consistency between chapters in this thesis, some of the notation in Chapters 2, 3, and 4 may differ from the versions submitted to journals. In addition, we do not provide a stand-alone literature review chapter. Chapters 2, 3, and 4 each include a detailed literature review.

The word count for this thesis is approximately 28,500 words.

Matthew Davison

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter establishes the research context and presents the key concepts employed in this thesis. Section 1.1 outlines the motivation for the content of this thesis. Section 1.2 introduces concepts about decision-making in timetabling for a university setting. Section 1.3 introduces basic deterministic optimisation concepts. Section 1.4 introduces multi-objective optimisation concepts. Section 1.5 outlines the contributions of this thesis. Finally, Section 1.6 outlines the structure of the thesis.

1.1 Motivation

Module timetabling is critical to the operation of higher education establishments. The timetable that is produced informs both staff and students where they will be at any given time and what they will be doing. Like many critical operational processes, university timetabling can be taken for granted. Students expect that their classes will be scheduled without conflicts, staff assume their teaching preferences will be accommodated, and administrators rely on efficient resource utilisation.

Universities need to deliver hundreds of modules to thousands of students. Students expect a high-quality teaching experience, which can be severely compromised by inadequate timetabling that results in rushed transitions between classes, poorly timed sessions, or scheduling conflicts that reduce engagement. University staff have specific preferences and constraints regarding their teaching schedules, which must be balanced against their broader academic responsibilities, including research, marking, coursework development, and examination preparation. All of these stakeholder constraints and requirements must be satisfied within the limitations of finite physical resources and operational restrictions on teaching hours.

Whilst this thesis addresses several of these challenges in university module timetabling, its initial direction was inspired by observing the operational adaptations during the COVID-19 pandemic of 2020. Social distancing requirements dramatically reduced physical capacity in lecture theatres and teaching spaces, while lockdown periods necessitated a complete shift to virtual delivery. The post-pandemic period saw the emergence of "hybrid teaching", enabling activities to be conducted entirely online or in mixed formats, with students able to participate in person or virtually. The pandemic-imposed shift to remote learning revealed unexpected opportunities to enhance both institutional efficiency and student satisfaction. Rather than viewing these format changes merely as crisis responses, proper implementation of hybrid teaching could not only provide resilience against future disruptions but also offer universities a pathway to optimise resource utilisation and improve the overall educational experience. However, hybrid teaching introduces an entirely new dimension to university timetabling optimisation. The timetabling model must now determine not only whether a student attends an activity, but also how they participate.

Despite these challenges, university timetabling has historically been a manual process conducted by administrative staff. Universities are now typically so large that automated timetabling solutions are necessary. Unfortunately, many universities continue to rely on outdated software tools that inadequately capture the full scope of modern optimisation challenges, resulting in suboptimal schedules and inefficient resource utilisation. This gap between institutional needs and available technology underscores the need for optimisation-based approaches that can effectively model emerging trends in university operations, such as hybrid teaching, while also delivering efficient solutions. Moreover, these approaches must support decision-makers, enabling evidence-based policy evaluation and day-to-day scheduling optimisation that meets the evolving demands of modern higher education.

1.2 Decision-making in university timetabling

Timetabling at universities is primarily a feasibility problem. The immediate need to arrange teaching activities and allocate resources represents what is called “operational decision-making” (Lindahl et al., 2017). These are decisions that must be made to enable day-to-day university activities. While these decisions have an immediate impact on timetabling stakeholders, this impact is typically short-term due to annual changes in timetables and student populations. Nevertheless, optimising these decisions remains important, as outlined in the previous section. The objective functions that guide timetable optimisation reflect institutional priorities and values, translating high-level goals into operational criteria. This translation combines what is called “tactical decision-making” and “strategic decision-making” (Lindahl et al., 2017). These decisions are long-term and have a large impact on the university. The output from timetabling models can inform these decisions by assessing the implications of different institutional policies and resource allocation decisions (Johnes, 2015).

University timetables affect hundreds of stakeholders, each with individual opinions on how well the schedule meets their needs and preferences. This means that there will often be several iterations of the schedule as stakeholder feedback is received. A small team of administrative staff is responsible for constructing the annual timetable while balancing potentially conflicting interests to achieve an acceptable solution.

In the past, university schedules were created by hand, often based on the timetable from the previous year (Carter, 2001). This process would require a lot of time from an experienced staff member and may perpetuate existing inefficiencies. Most universities now use some form of commercial timetabling software that takes information about the university as input and can produce new timetables from scratch, allowing for efficient solutions to be found. However, these have a limited scope and may not account for complexities unique to individual institutions. This limitation forces staff to develop workarounds to accommodate their specific requirements. Another limitation of current timetabling software is that it lacks the intuitive ability of human schedulers to identify inefficiencies and contextual issues that cannot be captured in formal optimisation criteria. These issues may be difficult or impossible to represent mathematically, and while human schedulers can identify these problems intuitively, systematically evaluating trade-offs between them proves difficult (Miettinen, 1998).

Future timetabling software must serve as both a scalable scheduling tool and an evidence-based decision-support system, enabling administrators to compare alternative solutions and evaluate "what-if" scenarios. This would transform timetabling from a passive consequence of strategic decisions into an active instrument for strategic planning.

1.3 Large-scale deterministic optimisation

Thanks to the rise in computational power available to timetabling practitioners, software tools that leverage various methods from the field of mathematical optimisation are widely used to construct timetables. In many practical settings, the major assumption is that the input data is deterministic. This is not true in reality. For example, students may change their minds about what modules they would like to take (Carter, 2001). Incorporating this stochastic element would add another layer of difficulty to an already challenging problem and is outside the scope of this thesis.

The history of deterministic optimisation is rich with various problems and solution methods. To arrive at the university timetabling problem we aim to solve, we start by highlighting the links between timetabling problems and the mathematical problem of graph colouring. This problem involves assigning colours to elements of a graph, typically the vertices, so that adjacent elements do not share the same colour (Korte and Vygen, 2019). An example from Welsh and Powell (1967) involves assigning a collection of exams to different periods, where each vertex of a graph represents an exam. Edges are then drawn between pairs of vertices if the pair cannot happen in the same period. The smallest number of colours needed to colour the vertices of this graph, known as its chromatic number (Korte and Vygen, 2019), corresponds to the minimum number of periods required to deliver all the exams. Finding an upper bound on this number proves the existence of a solution using at most that number of periods. For example, a trivial upper bound on the chromatic number is the number of vertices, representing a solution where all exams occur in different periods.

Graph colouring problems are conceptually straightforward; however, they are, in general, computationally intractable, belonging to the NP-hard complexity class (Korte and Vygen, 2019). Furthermore, the basic graph colouring framework cannot adequately capture all of the complexity of real-world scheduling problems. For instance, in exam timetabling, the basic formulation from Welsh and Powell (1967) only addresses scheduling conflicts between exams, overlooking critical practical constraints such as examination hall capacity. Whilst graph colouring problems can be formulated as binary or integer programs, the reverse is not always true. Using a binary or integer programming framework allows for a mathematical formulation that extends beyond what is possible within the graph colouring framework, making the former framework a popular choice for modelling timetabling problems mathematically.

Pure binary programs have variables that lie in the set $\{0, 1\}$ and pure integer programs have variables that lie in $\mathbb{Z}$ (Wolsey and Nemhauser, 2014). These variables allow us to represent decisions relating to real-world issues. For example, we would use binary variables to model yes/no decisions and integer variables to model decisions on the quantity of problem elements. Combinations of these variables can be used to represent logical conditions (Williams, 1999), which allows the modelling of complex systems.

In general, the combinatorial optimisation problems arising from using the binary or integer programming framework for modelling timetabling problems also belong to the NP-hard complexity class (Babaei et al., 2015). All pure binary and some pure integer programs have a finite solution space; however, exhaustive enumeration of these solutions is either computationally prohibitive due to memory constraints or would require an inordinate amount of time to solve (Wolsey and Nemhauser, 2014). This contradictory behaviour has led to the development of specific methods for solving these problems.

One popular implicit enumeration approach is branch-and-bound (Wolsey and Nemhauser, 2014). In this method, variables are allowed to take any value in $\mathbb{R}$, and the relaxed problem is solved. The resulting solution is examined, and a variable that should be an integer but is taking a fractional value is chosen to be "branched on". For example, suppose a relaxed variable x takes the value 1.5. The model is copied, and the constraint $x \leq 1$ is added to one copy, and $x \geq 2$ is added to the other. This has split the solution finding into two paths or branches, forming the start of a "tree". If the optimal objective value of the relaxed problem on a branch is worse than an incumbent integer solution, then there is no use in going further down that branch. Choosing not to continue with a branch is called "pruning". The algorithm repeats this process of relaxing, branching and pruning until all avenues have been explored.

Branch-and-bound routines (and other approaches like branch-and-cut) rely on a careful selection of which variables to branch on. At worst, the selection of variables may lead to an exhaustive search of the tree (Korte and Vygen, 2019). This approach becomes computationally intractable for large-scale problems due to excessive memory requirements for tree storage and prohibitively long search times (Wolsey and Nemhauser, 2014). This motivates the use of matheuristics, a subset of heuristic approaches that leverage aspects of one or more exact methods (Jourdan et al., 2009).

The approach known as fix-and-optimise is a matheuristic that is often used for large-scale binary or integer programs. This method can be considered a large neighbourhood search (Pisinger and Røpke, 2010). The scale of the problem is reduced by constraining a significant portion of the problem to the incumbent values, and the remainder of the problem is then solved using a (typically exact) method for solving integer programs (Helber and Sahling, 2010). This process is repeated until all stopping criteria are satisfied. Performing a certain number of iterations or reaching a target objective value are examples of stopping criteria. Using an exact approach within each iteration ensures that the unfixed variables are optimal with respect to the fixed values.

1.4 Multi-objective optimisation

Section 1.2 alluded to the multi-objective nature of university timetabling decisions. Multi-objective optimisation is a branch of mathematical optimisation that addresses problems involving two or more objective functions to be optimised simultaneously (Miettinen, 1998). In single-objective optimisation, comparing two solutions is clear, as the more desirable solution is the one with the better objective. In multi-objective optimisation, this clarity is absent (Ehrgott, 2005).

Multiple objectives arise naturally in real-world problems because solutions impact several incommensurable aspects simultaneously (Miettinen, 1998). University timetabling is inherently multi-objective, requiring a balance between individual stakeholder preferences and institutional goals. The multi-stakeholder environment makes this more difficult, as staff and students often have conflicting views on what constitutes a good timetable (Mühlenthaler and Wanka, 2016).

When no single "best" solution exists, the typical approach is to generate a set of efficient solutions for decision-makers to evaluate, known as a Pareto front. In multi-objective optimisation, a solution is Pareto optimal if no objective can be improved without worsening another (Ehrgott, 2005). These Pareto optimal solutions illustrate the trade-offs between objectives and assist decision-makers in understanding the available choices. However, finding all Pareto optimal solutions is often computationally challenging, as even identifying a single such solution can be as complex as solving a single-objective problem.

Multi-objective solution approaches for university timetabling can be categorised into three categories: weighted sum, lexicographic, and Pareto-based methods. Weighted sum methods aggregate these objectives into a single function:

$$f = a_1 f_1 + \cdots + a_n f_n, \tag{1.4.1}$$

where $a_1, \cdots, a_n \in \mathbb{R}_{\geq 0}$. Whilst conceptually simple and allowing the use of single-objective optimisation techniques, this approach suffers from the difficulty of selecting appropriate values for $a_1, \cdots, a_n$, particularly when dealing with incommensurable objectives. Additional technical limitations are detailed in Marler and Arora (2010). Lexicographic methods optimise objectives sequentially according to predetermined priorities, treating each optimised objective as a constraint for subsequent optimisation steps (Ehrgott, 2005). This approach aligns well with hierarchical decision-making structures but fails when decision-makers view multiple objectives as equally important. Pareto-based methods seek to identify multiple efficient solutions distributed along the Pareto front. Unlike weighted sum or lexicographic approaches that yield single solutions, these methods present decision-makers with a range of alternative solutions, enabling explicit evaluation of objective trade-offs.

From a decision support perspective, providing multiple solution options enhances decision-making quality (Yu, 1991). Combining lexicographic approaches with different objective orderings alongside Pareto-based methods can reveal both “extreme” solutions representing strict hierarchical priorities and intermediate solutions that demonstrate objective trade-offs.

1.5 Contributions

The key research questions we are addressing are:

Q1: Hybrid teaching integration. How can the emerging mode of hybrid teaching be effectively and explicitly incorporated into a mathematical model for university timetabling, and what are the relevant objective functions and data requirements?

Q2: Student partitioning and objective bounds. How can we leverage the structure of students to partition them into near-disjoint groups and use this partition to produce meaningful bounds on objective functions?

Q3: Lexicographic optimisation methods. What method can be designed to efficiently perform steps within lexicographic optimisation across different objective orderings on large-scale instances?

Q4: Multi-objective Pareto analysis. What interactions between objectives can be identified by extending the methodology to approximate Pareto fronts for three or more objectives?

Q5: Strategic policy assessment. How can the model and solution methods be adapted to support strategic policy assessment and decision-making in university timetabling contexts?

The main contributions of this thesis are outlined in detail within each chapter and again in Chapter 5 in the context of the above research questions. However, a summary of the contributions of this thesis is provided here:

- A comprehensive mathematical model that explicitly incorporates hybrid teaching modes.
- A graph-theoretic method for partitioning students into near-disjoint groups based on module requests.
- A systematic procedure for computing bounds on objective functions by leveraging student partitions, providing theoretical limits for solution quality assessment.
- A fix-and-optimise matheuristic with partition-guided neighbourhood selection that efficiently handles lexicographic optimisation.
- An extension of the matheuristic to approximate Pareto frontiers for two or more objectives.
- A novel large-scale benchmark instance derived from Lancaster University data, formatted to International Timetabling Competition 2019 (ITC-2019) standards.
- Demonstrations of the model and methods being used for evidence-based policy evaluation.

Whilst the work done in this thesis may not be directly implemented in its current form into a specific timetabling tool, the contributions listed above lay the groundwork for future research or implementation. It aims to show what is theoretically possible through the modelling and methodology presented, providing a reference point for future research and eventual practical application.

1.6 Thesis outline

In Chapter 2, which is an adapted version of Davison et al. (2024), we look at modelling the university course timetabling problem (UCTTP) in such a way that incorporates hybrid teaching explicitly. This feature was chosen after a thorough review of models seen in the literature and observations of the shifting strategies within modern universities. We supplement the modelling of this novel feature with the modelling of constraints commonly seen in other problems. We then exactly solve the model lexicographically for three objectives across a collection of simplified benchmark instances to demonstrate the interactions between these objectives. It is then outlined how this model could be used to assist with strategic decision-making. These experiments also motivate the need for a more efficient solution-finding method.

In Chapter 3, we study and develop such a solution-finding approach. The work first outlines how to partition students in any given instance into smaller disjoint groups according to module requests and obtaining bounds on the objective functions outlined in Chapter 2 using this partition. This is beneficial not only for strategic decision-making but also useful for assessing the results from the fix-and-optimise inspired matheuristic proposed in the later sections of the chapter. Firstly, the method starts by finding approximate lexicographic solutions. Secondly, the method explores an approximation of the Pareto frontier. Experiments are done to demonstrate the benefit of using this method, using instances and results from Chapter 2 as a basis for comparison.

In Chapter 4, we apply the modelling and methodology developed in Chapters 2 and 3 to real-life data from Lancaster University. This chapter starts with an outline of the data we were provided with and how this was reverse-engineered to create a new benchmark instance mirroring the format of benchmark instances provided as part of the International Timetabling Competition 2019 (Müller et al., 2024). We then verify the data and the model by running subsets of the new instance exactly and comparing the solutions with the original schedule that was in operation for the academic year 2023 to 2024. The final part of this chapter lists multiple changes to policies that universities may wish to act on as part of their strategic decision-making. We show how our model and method can be used to assess the impact of these changes and therefore inform decision makers.

Finally, in Chapter 5, we conclude the thesis, summarising the academic contributions and discussing areas for future research.

Chapter 2

Modelling and Solving the University Course Timetabling Problem with Hybrid Teaching Considerations

2.1 Introduction

Timetabling for universities is a very challenging problem and one of the most interesting educational timetabling problems to research. The interest and challenges arise because the decisions that need to be made impact many different stakeholders and resources. By modelling the university timetabling problem, we hope to gain insight into some of the interactions between many stakeholders and types of resources.

One major factor increasing the difficulty of this problem is that students have a choice in the degree program and a choice in the modules that they attend. This means the output of the university course timetabling problem should be a timetable for each student. This issue is exacerbated by the fact that universities typically enrol significantly more students than other types of educational establishments. Allowing students to choose some of the modules they take means there is a stronger interdependence in these choices. Students can interact with each other and, therefore, may influence each other's choices. One possible consequence of this is inconsistent class sizes.

Teaching spaces at universities are more diverse than other educational establishments because they are primarily places of research for multiple disciplines, with each discipline requiring specialist equipment and facilities. This means there are more constraints on how classes are assigned to teaching spaces. This diversity contributes to a higher dispersion of facilities. Teaching spaces of different shapes and sizes do not fit together as neatly as teaching spaces that are uniform in size. Therefore, more space is needed for the university, which can increase travel time between classes. Dispersion of teaching facilities can also be a consequence of the university's layout. Some universities are campus universities, meaning most of the buildings are on a single site, but some universities are city universities where university-owned buildings are mixed amongst other buildings. There can be a mix of the two, with some universities spanning multiple cities and campuses. Travel time between buildings for any layout needs to be considered when timetabling. Universities also have longer working hours than other educational establishments, meaning a greater number of times that classes and other meetings can happen, increasing the scale of the timetabling problem.

An emerging factor that complicates the timetabling process is the increasing use of "hybrid teaching". This is where class can be held in a "hybrid mode" with some students attending in-person and some attending online. The inclusion of hybrid teaching means that student allocation involves deciding what classes a student attends as well as deciding how they attend these classes. Different students respond differently to each mode of teaching, so this needs to be accounted for when assigning students to classes.

If a timetable is produced that does not address these complexities, then students and staff will be unhappy with the timetable. Students often pay a lot of money to attend university, and a timetable that impacts their ability to attend classes could encourage them to discontinue their studies. For staff who research at the university, a timetable that does not allow them enough time to research around teaching or does not cater to various personal requests could cause them, at worst, to leave their position at that university, negatively impacting the teaching.

Therefore, being able to produce timetables that can consider the above complexities is imperative to the successful operation of a university. At a high level, universities have strategic goals that they want to achieve (for example, achieving a high output of novel research). This requires careful management of resources and people, which automated scheduling can help facilitate. The final timetables produced specify exactly what needs to be done at an operational level to work towards these goals.

There already exists a significant amount of literature that addresses various aspects of the university course timetabling problem. However, to the best of our knowledge, the literature lacks models that explicitly include the emerging issue of hybrid teaching. This chapter aims to close this gap by demonstrating how hybrid teaching can be incorporated explicitly into a university timetabling model. This includes specifying what information needs to be known about the university and what sort of variables and constraints need to be part of the mathematical model that produces timetables for students.

The objectives of this chapter are to achieve the following: (i) provide a timetabling model that explicitly incorporates hybrid teaching, (ii) identify the benefits of including hybrid teaching in the timetabling model, and (iii) discuss how this model gives rise to several interesting research directions and how this model could be used in a strategic decision-making context.

The rest of the chapter is structured as follows. In Section 2.2, a review of existing work in the context of university timetabling is provided. In Section 2.3, a brief description of the problem to be modelled is given. A mathematical formulation of this problem is given in Section 2.4. The method used to solve this problem is given in Section 2.5, and this method is used in Section 2.6 for various computational experiments. Section 2.7 includes a discussion about the results and extensions to the model. Finally, in Section 2.8 there is a summary of the work done in this chapter.

2.2 Related work

University timetabling problems have been studied for a long time, with one of the earliest papers in the literature presenting a method for university examination timetabling (Broder, 1964). There are three types of university timetabling problems: post-enrolment-based timetabling, curriculum-based timetabling and examination timetabling (Lewis and Thompson, 2015).

The examination timetabling problem is the problem of assigning examinations to locations and times. Considerations need to be made to ensure that students can attend all the exams they need to and to ensure that the locations have enough capacity for the exams to take place. Curriculum-based timetabling is where events (such as lectures and seminars) are grouped to form fixed curricula which are then assigned to times and locations, and post-enrolment-based timetabling is where events are assigned times and locations with respect to student demand and/or enrolment data (Lewis and Thompson, 2015).

This thesis focuses on a combination of the curriculum-based and post-enrolment-based timetabling problems known as the university course timetabling problem (UCTTP). The goal of this problem is to assign people to events and these events to times and locations subject to various constraints (Babaei et al., 2015). Events are typically grouped into "courses", hence the name UCTTP; however, this thesis uses the term "module". The reason we do this is that universities in the United Kingdom (UK) typically use the phrase "course" to describe a programme of study that is made up of modules. For example, an undergraduate mathematics course may contain modules that cover algebra.

The earliest UCTTP models were graph theoretic models (de Werra, 1985). The university course timetabling is an NP-hard problem (Cooper and Kingston, 1995) and therefore early mixed integer programming (MIP) models for timetabling problems could only be solved exactly for small instances (Badri, 1996).

As time went on and universities became bigger and more complex, the demand for sophisticated models and solution methods became greater. Lewis (2008) and Burke et al. (2012) provide reviews that primarily cover heuristic and hyper-heuristic algorithms, approaches that have dominated the field for over two decades. However, thanks to improvements in computers and MIP solvers, matheuristics are the current focus of the timetabling community (Mikkelsen and Holm, 2022).

There have been several papers reviewing research on the UCTTP. Two recent papers that review some of the state-of-the-art solution methods are Tan et al. (2021) and Chen et al. (2021). For less recent but more feature-focused reviews that are useful for understanding the field, see Babaei et al. (2015) and Aziz and Aizam (2018).

2.2.1 Key timetabling features

In this section, papers are reviewed according to general model features that are of importance when tackling the UCTTP. For each paper, a brief description of the paper is given, and what it contributes to the literature is highlighted. Model features, the modelling approach, and the data used are summarised for each paper in Tables 2.2.1-2.2.3.

Typical resource allocation

Badri (1996) proposes a binary program to model a departmental assignment problem at the United Arab Emirates University. This paper, to our knowledge, is the earliest paper to include constraints similar to the set-packing problem (Skiena, 2008), which are useful when you need to constrain choosing a certain number of items selected from a collection of options. In their model, they penalise using more rooms than available and not meeting instructor preferences. This is done using a goal programming approach where there are penalties for deviations above or below a desired level.

Di Gaspero and Schaerf (2006) use a local search method to solve the course timetabling problem involving assigning lectures for courses to periods and rooms. The main purpose of their paper is to demonstrate their local search method. By using a simple solution representation, moves between solutions in the search do not lead to infeasibility. The two hard constraints maintained between moves are ensuring no more than one lecture happens in a single room at the same time and ensuring all lectures in a module are offered. The quality of the solution is a weighted sum of violations of soft constraints. The soft constraints include features such as room capacity and instructor availability. They also include temporal constraints, spreading lectures across several days and spacing lectures within days to avoid gaps.

Overlapping timeslots and irregular weekly timetables are allowed in the problem defined by De Causmaecker et al. (2009). This was to accommodate the structure of teaching at KaHo Sint-Lieven School of Engineering. A feature not included in their problem that others, such as Badri (1996), include is the assignment of staff. It is assumed staff already know what they will teach, so constraints are included to ensure they can attend all events they need to. The solution method is very similar to Di Gaspero and Schaerf (2006), using a local search algorithm to find solutions.

Chaudhuri and De (2010) define a timetabling problem including many of the features seen in the problems discussed so far. This includes various temporal constraints, staff preferences and conflicting resource assignments. A constraint seen in this problem that has not been discussed yet is ensuring that assignments are “compatible”. For example, a chemistry class may only be held in a chemistry lab, and so the set of compatible rooms for a chemistry class is the set of all chemistry labs at the university. This notion of compatibility extends to any assignment of events or people to resources.

The problem described in Aizam and Caccetta (2014) is a binary program like in Badri (1996). In this paper, they start by describing a basic model that contains constraints that they deem necessary for every timetabling problem and then suggest extra constraints to account for additional features that could be included. This is one advantage of using a binary program formulation. One feature in this model (and the previously discussed models) that is worth pointing out is "completeness". This is where every event is assigned resources, or every student is assigned to every class they need to be.

An example of a model where completeness is not necessary is given by Méndez-Díaz et al. (2016). There is more emphasis on the post-enrolment features of the UCTTP. The objective of their model is to maximise the total weighted preference for the assignments of students to modules. Due to student demand driving the timetable, it is not necessary for all events to be assigned a location and a time. One feature that causes this uncertainty in whether events are assigned or not is the structure of modules. In Méndez-Díaz et al. (2016), modules are composed of one or several commissions, which are instances of the same module. The literature also refers to this as "configurations" (Müller et al., 2018). If it is known that all students are assigned to a single commission, then the events in other commissions do not need to be assigned.

The Integer Linear Program (ILP) described by Fonseca et al. (2017) covers many of the features described in the models seen so far. It includes some of the constraints outlined by Aizam and Caccetta (2014) in their basic model and also includes constraints described by the eXtended Markup Language for High School Timetabling (XHSTT) format (Post et al., 2014). This format is one example of an attempt to generalise a description of any high school timetabling problem.

Scheduling issues

One of the earlier discussions surrounding student scheduling issues is given in (Carter, 2001). This model operates a "demand-driven" approach where students choose modules and then a timetable is found to best match these requests. They use clustering techniques to group students with similar requests and assign these to sections to minimise expected conflicts. Once a timetable is found, the student sectioning is repeated, considering individual student conflicts.

In the model described by Schimmelpfeng and Helber (2007), room assignment and staff assignment are the most important features. Rooms should not be assigned to more than one class at a time or contain classes with an attendance that exceeds the room capacity. Staff members can not be assigned to a time when they are not available, and other staff preferences should be respected. These include a variety of teaching staff requests, such as breaks, consecutive or distributed teaching slots, and a maximum number of teaching slots. Unlike the timetabling seen in other papers, such as Carter (2001), students are an afterthought.

Gonzalez et al. (2018) create a MIP that schedules courses for the United States Air Force Academy (USAFA). The interesting scheduling issue in this problem is that students who are at the USAFA have work commitments as well as academic commitments. They utilise a goal programming approach to meet as many requirements as possible, including minimising student registration conflicts, where a student is assigned two modules that conflict. They state that in practice, it is impossible to remove every conflict.

The graph-based MIP model described by Holm et al. (2022) was constructed to solve the problem designed by the organisers of the ITC-2019 (Müller et al., 2018). In this problem, student conflicts are minimised as part of a weighted objective. In Holm et al. (2022), they discuss the importance of identifying assignments that lead to inevitable conflicts and assignments that lead to impossible conflicts. This preprocessing step helps minimise the work that needs to be performed by the solution method.

Student movement and travel

When dealing with the movement of students, one aspect Daskalaki and Birbas (2005) aim to control is the number of classroom changeovers. Minimising the number of classroom changes means that there is less noise and congestion in spaces on campus. To do this, they name a preferred classroom for each student group and try to ensure that the group stays in that classroom and has consecutive sessions in that room. This is, however, not a realistic representation of a general university, as students can not typically be grouped so easily and need to be considered as individuals.

Al-Yakoob and Sherali (2007) deal with parking and traffic congestion issues in their paper. This is achieved primarily by limiting the number of students on campus at any given time. A hard limit for the whole campus could lead to some timeslots having a few very crowded departments, whilst the others are empty. This is unfair to the busy departments, so to make this fair, they also impose a minimum and maximum attendance at the department level to distribute congestion over the entire campus.

Vermuyten et al. (2016) also try to avoid congestion, as in Daskalaki and Birbas (2005); however, their approach does not try to achieve this by fixing students in one place but by changing how many students move along various corridors at a given time. A graph that represents the faculty building is used so they can optimise the flow of students through arcs and the resulting travel times. The element they minimise overall is the maximum travel time seen in an arc. A two-stage decomposition is used, where most of the schedule is determined in the first stage, and classes and rooms are swapped around in the second to locally optimise student flow. Optimising the schedule and the flow together is computationally expensive.

Gogos et al. (2022) work with a problem that focuses on minimising the number of times in a week that students travel to university. The motivation is that students who do not live on campus do not want to spend excessive money on public transport and want to reduce the risk of catching an illness from other passengers. They approach this problem by calculating the minimum number of days a student would need to attend university and then trying to minimise the number of excess days the student is on campus. This is limited as it does not consider the time of travel (certain times are busier) or if students make multiple journeys in a single day (multiple campuses).

Scarce resources

The timetabling model in Dammak et al. (2008) includes a few of the features seen in other papers. One feature relating to the usage of resources is their aim of maximising the occupancy of classroom seats. Since their paper presents only a heuristic to produce a feasible solution, this objective is not explicitly optimised. However, in the construction of the feasible solution, they order the classrooms and student groups in a non-increasing fashion so that large student groups are placed in large classrooms.

Lindahl et al. (2017) approach the UCTTP differently. They break from the operational timetabling problem and move towards a strategic approach. Three problems are presented in this paper. The first is the "quality problem" that is similar to the other papers that produce a timetable that is high quality by some measure. The second finds the minimum number of rooms needed. The third finds the minimum number of times needed. They solve a collection of bi-objective models to create solution frontiers that can be used to analyse the gain in quality by not using the minimum amount of resources.

Barnhart et al. (2022) experience scarce resources due to the COVID-19 pandemic. This context applied to most, if not all, universities at the time. They tackle a term-planning problem and a timetabling problem within the same paper. Like most of the other papers in the literature, the timetabling problem involves working out when and where events take place. One modelling difference is that teaching spaces are bundled into blocks to create "larger" teaching spaces. This reduces the overall number of available teaching spaces. The idea at MIT was to have students "rotate" between coming onto campus and attending online. Like the work of Al-Yakoob and Sherali (2007), they have a global cap on the number of students on campus at any time to reduce the usage of unscheduled resources (toilets and shops, for example).

Hybrid teaching

The only paper, to our knowledge, that explicitly discusses the timetabling problem with hybrid teaching is Barnhart et al. (2022). In this paper, the online teaching space is modelled as a fictitious block of classrooms with zero in-person capacity. The timetabling model tries to maximise the number of modules students can attend with a preference for the in-person format. However, the limitation of this model is that classes can only be offered online or in-person, rather than potentially having some students attend physically and some attend online.

An example of where we can see multiple instructional modes, including a true hybrid approach, is in the open-source solver UniTime (UniTime, 2023). These features are implicit here and in the description of the ITC-2019 problem (Müller et al., 2018) as the ITC-2019 problem is a simplified variant of the UniTime problem. For example, the ITC-2019 problem may have two classes that should occur simultaneously, with one class not requiring a room assignment, emulating a hybrid setup. There are also cases where classes do not require rooms or where the class subscription limit is greater than the capacity of all the available rooms.

Whilst papers in the mathematical timetabling literature relating to hybrid teaching are scarce, hybrid teaching is still being used in practice. A survey conducted by the British Broadcasting Service (BBC) found that nearly a third of university courses were still combining in-person teaching with online learning in the 2022-23 academic year (Standley, 2023). Whilst hybrid teaching became prevalent during the COVID-19 pandemic, it is unlikely to make a complete disappearance. For example, there are articles published post-pandemic proposing that hybrid teaching could address shortages in teaching staff (Stone, 2024) or shortages in teaching spaces (Steed, 2024).

2.2.2 Contributions

This review of the literature outlines not only the importance and continued relevance of the UCTTP but also outlines some of the features of the problem. These include features that are very common across models as well as features relating to specific or emerging issues. Features seen in the literature have been collected in Table 2.2.2. Table 2.2.3 provides information on how the problem was modelled and what data was used. For both tables, the columns have been ordered by year of publishing.

Table 2.2.2 shows that the choice of features included varies from model to model. This is because authors have tried to take on the timetabling issues present at the university where they work. The result is that much of the literature consists of very focused models that do not generalise well, implicitly seen in Table 2.2.3, where 76% of papers in this review use internal data to solve the problem.

Table 2.2.2 shows that the most studied features include room capacity issues, room and time preferences and staff/student conflicts. Table 2.2.3 suggests that the most popular modelling approaches include integer/binary programs. Table 2.2.2 is also useful for spotting emerging features of interest. The most notable aspect is the increasing number of models that are primarily driven by student demand or models that consider individual student requests.

It can also be seen from Table 2.2.2 that as the field has progressed over time, researchers are generally including more features in their models. This is also reflected in Table 2.2.3, which shows that researchers and practitioners in the timetabling field are starting to explore bi-objective and multi-objective approaches.

One major gap in the research is the explicit study of online or hybrid teaching.

During the COVID-19 pandemic, many universities needed to adapt to using these formats. However, the university timetabling problem with hybrid module delivery considerations has not been adequately addressed. The model that is presented in this chapter includes "traditional" features of the UCTTP and explicitly incorporates the new element of hybrid teaching. The aim of this is to introduce one approach to explicitly modelling hybrid teaching at universities using binary programming so that other researchers can include hybrid teaching in future models. Due to the rarity of this feature in existing models, there is little analysis of how this feature impacts other features of the timetabling problem that are well-studied.

Students and staff at universities are acutely aware of the pedagogical and logistical issues relating to hybrid teaching, and the starting point for resolving these issues is being able to represent hybrid teaching in a mathematical sense. With this in mind, the key contributions of this chapter are the following:

- Outline the information that needs to be collected about the activities to determine if an activity can happen in the online/hybrid mode, and what extra information about students needs to be known to best cater to their teaching preferences.
- Present a description of a generic UCTTP with hybrid elements, along with a proposed binary program formulation of the problem.
- Demonstrate the model using the most up-to-date benchmark data available to show how the hybrid elements would work in practice and illustrate some of the interactions with other problem features.
- Discuss how this model could be used to produce a timetable that achieves a particular strategic goal, as well as identify unresolved logistical problems with hybrid teaching and the research questions that these problems present.

Table 2.2.1: Indices of the features, modelling approaches and data referenced in Tables 2.2.2-2.2.3

	Index	Description
Features	1	Conflicts for teaching staff are considered
	2	Conflicts for students are considered
	3	Complete timetable
	4	Overlapping times
	5	Explicit use of online classes
	6	Explicit use of hybrid classes
	7	Mode requests
	8	Student choice in modules
	9	Students have compulsory modules
	10	Staff travel time considered
	11	Student travel time considered
	12	Room capacity
	13	Co/Prerequisite courses
	14	Individual student assignment
	15	Room assignment restrictions/preferences
	16	Time assignment restrictions/preferences
	17	Number of students on campus limited
	18	Enrolment data used in the model
	19	Physical student flows are considered
	20	Switching classes or locations
	21	Rooms and equipment have capacity and usage restrictions
	22	Compact timetable preferred
Modelling approaches	1	Mixed integer program
	2	Integer program
	3	Binary program
	4	Neighbourhoods
	5	Graph colouring
	6	No explicit objective
	7	Single objective
	8	Bi-objective
	9	Multi-objective
Data used	1	Institution (Data from the author's university)
	2	International Timetabling Competition 2019 (Müller et al., 2018)
	3	International Timetabling Competition 2011 (Post et al., 2013)
	4	International Timetabling Competition 2007 (Lewis et al., 2007)

Table 2.2.2: Summary of what features are considered by a paper. Feature numbers are given in Table 2.2.1

	Feature																					
Reference	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
Badri (1996)			✓									✓				✓						
Carter (2001)	✓	✓						✓	✓			✓		✓	✓	✓		✓			✓	
Daskalaki et al. (2004)	✓	✓	✓									✓			✓	✓			✓	✓	✓	✓
Avella and Vasil'Ev (2005)	✓	✓							✓			✓			✓	✓				✓	✓	✓
Di Gaspero and Schaerf (2006)	✓	✓	✓						✓			✓			✓	✓					✓	✓
Schimmelpfeng and Helber (2007)	✓	✓										✓	✓			✓					✓	
Al-Yakoob and Sherali (2007)			✓									✓			✓	✓	✓				✓	✓
Dammak et al. (2008)	✓	✓	✓									✓			✓	✓					✓	✓
De Causmaecker et al. (2009)	✓	✓	✓	✓								✓			✓	✓					✓	
Chaudhuri and De (2010)	✓														✓	✓			✓			✓
Santos et al. (2012)	✓	✓	✓													✓						✓
Aizam and Caccetta (2014)	✓	✓	✓						✓			✓	✓			✓					✓	✓
Méndez-Díaz et al. (2016)	✓	✓						✓	✓			✓		✓	✓	✓		✓			✓	✓
Vermuyten et al. (2016)		✓	✓						✓	✓	✓	✓			✓	✓			✓	✓	✓	✓
Fonseca et al. (2017)	✓	✓							✓			✓	✓		✓	✓					✓	✓
Lindahl et al. (2017)	✓	✓	✓									✓			✓	✓					✓	✓
Gonzalez et al. (2018)	✓	✓							✓			✓		✓	✓	✓		✓				
Barnhart et al. (2022)		✓			✓		✓	✓	✓			✓	✓	✓	✓	✓	✓	✓			✓	✓
Holm et al. (2022)	✓	✓	✓	✓				✓		✓	✓	✓	✓	✓	✓	✓		✓			✓	
Gogos et al. (2022)		✓	✓					✓	✓					✓		✓		✓		✓		✓
This paper	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓			✓	

Table 2.2.3: Summary of what modelling approach and data a paper uses. Modelling approach and data numbers are given in Table 2.2.1

	Modelling approach									Data used			
Reference	1	2	3	4	5	6	7	8	9	1	2	3	4
Badri (1996)			✓						✓	✓			
Carter (2001)					✓		✓			✓			
Daskalaki et al. (2004)		✓					✓			✓			
Avella and Vasil'Ev (2005)		✓					✓			✓			
Di Gaspero and Schaerf (2006)				✓			✓			✓			
Schimmelpfeng and Helber (2007)		✓					✓			✓			
Al-Yakoob and Sherali (2007)	✓						✓			✓			
Dammak et al. (2008)			✓			✓				✓			
De Causmaecker et al. (2009)				✓			✓			✓			✓
Chaudhuri and De (2010)						✓				✓			
Santos et al. (2012)	✓						✓						
Aizam and Caccetta (2014)		✓					✓			✓			✓
Méndez-Díaz et al. (2016)			✓				✓			✓			
Vermuyten et al. (2016)		✓					✓			✓			✓
Fonseca et al. (2017)		✓					✓					✓	
Lindahl et al. (2017)	✓							✓					✓
Gonzalez et al. (2018)	✓					✓				✓			
Barnhart et al. (2022)		✓							✓	✓			
Holm et al. (2022)	✓				✓		✓				✓		
Gogos et al. (2022)	✓	✓					✓			✓			
Our model		✓	✓						✓		✓		

2.3 Problem description

What our review has exposed is a lack of literature explicitly modelling hybrid teaching at universities. Where there are explicit instances, there is a lack of analysis regarding the benefits or drawbacks associated with the incorporation of this feature. As stated in the previous sections, the objective of this chapter is to focus on this particular feature of the UCTTP and provide some managerial insights regarding this feature. This feature is important to be studied in conjunction with other features of the timetabling problem to shed light on its impact on the resulting schedule. In this section, we introduce a multi-objective university timetabling model with hybrid teaching considerations.

The timetabling problem we are modelling is the post-enrolment approach, where students provisionally select modules they want to take and after this is done, a timetable is constructed. Timetabling practitioners at universities in the United Kingdom (UK) often take this approach. This timetable is typically constructed before the term starts, especially in the case of the first term when new students register in August and start studies in late September. To mirror this process, the timetabling problem in this chapter is also a post-enrolment timetabling problem. Part of the input to the mathematical model is a list of students and modules they are requesting to take. A module request is met if the student is assigned to an appropriate arrangement of classes (the particular arrangement varies between universities).

As we are focusing on hybrid teaching, an additional input is that each student may also provide a preference for a particular mode of module delivery (online and/or in-person). If a student expresses a preference for a particular mode, then it is assumed this preference applies to all modules they want to attend.

One novel element here is that the travel time between two classes may be different for two students taking the same classes but in different modes. For example, a student attending online only can, in theory, switch instantly between classes, whereas a student attending in person will need to walk between rooms. If a student does not have enough time to travel between classes or is assigned classes that overlap, this is referred to as a scheduling issue.

The decisions we are making in this problem are the following: (i) when and where classes are being held, with the option for classes to be held online and in-person simultaneously and (ii) what classes students attend and the mode of study they attend the classes. These decisions are made with respect to three different objectives: (i) maximising module requests met, (ii) minimising the total number of scheduling issues, and (iii) minimising the total number of classes where a student does not attend in their preferred mode.

There are constraints on these decisions. Classes are only assigned to compatible times. What makes a time compatible for a class depends on the university; however, what we mean by compatible is that the time meets some set of criteria that allows the class to be assigned to it. Similarly, classes are only assigned to spaces that are available and compatible. In this case, available means that the space is not in use by other classes and is free to be used by a class. For example, a chemistry class may need to occur in the afternoon to allow for the setup of equipment in the morning and cannot be assigned a space without the correct equipment or space in use by people doing a different experiment.

Hybrid teaching is only done if the class is assigned the appropriate space. For a teaching space to be capable of hosting a hybrid meeting, it needs to have a particular layout and equipment. In practice, it is not usual for every room to meet these criteria. Therefore, the collection of physical spaces at the university can be partitioned into those that are capable of hybrid teaching and those that are not capable of hybrid teaching. There are two limits on class attendance. There are limits imposed on the number of students attending a class for pedagogical reasons and room capacity limits so that students can fit into the physical space assigned to the class.

The structure of modules is assumed to be the same as in the ITC-2019 competition

(Müller et al., 2018). Modules are made up of configurations, which are made up of subparts. Each subpart contains a collection of classes. For a student to attend a module, they need to attend a class from every subpart within a single configuration. For example, a module may have one configuration containing two subparts. The first subpart could contain a single class in the form of a lecture. The second subpart could contain multiple classes that are seminars. This structure is used in this problem for two reasons. Firstly, it is a good representation of most forms of university modules. Secondly, it means it is easier to utilise the ITC-2019 data sets for testing. The modelling of teachers and instructors is also done in a fashion similar to the ITC-2019 competition, where we ensure that a staff member can attend a list of classes without any scheduling issues (no overlaps/sufficient travel time). These lists of classes are another input to the model. This is adopted here for the same reasons we adopt the module structure.

Modules at UK universities are either compulsory or optional (elective). Compulsory modules are those that students are required to take, and elective modules are ones that the student can choose. Table 2.2.2 shows that several models in the literature also include this feature. In the problem described here, students are assigned to their compulsory modules as a hard constraint, and the number of elective modules that can be attended is maximised. The total number of modules that a student attends is capped at a fixed number.

To summarise, we have designed a variant of the UCTTP that includes hybrid teaching. This variant contains objectives and constraints often referred to as "essential constraints" (Sørensen and Dahms, 2014; Aziz and Aizam, 2018; Rudová et al., 2011) that have been modified to explicitly include hybrid teaching.

The model is designed this way to maintain focus on the novel aspect of modelling hybrid teaching explicitly, whilst including key elements of a typical UCTTP formulation for these novel features to interact with.

Before providing the mathematical formulation of the problem, it is important to note that there are some important university features not included in this model that are included in other models, such as the model described by the ITC-2019 competition (Müller et al., 2018). In that model description, there are "distribution constraints" that enforce rules on how classes should be distributed in the schedule (for example, ensuring certain groups of classes happen on different days or in the same room). These are not included in this model but could be modelled using the notation provided in this paper to bring the problem even closer to the real-life problem (Holm et al., 2022).

It is also important to note that there are no hard constraints on preventing classes of a particular module from being scheduled on overlapping times. If a student is attending a module, then the objective of minimising the number of scheduling issues should assist in reducing the number of overlaps that occur. Overlaps and scheduling issues are only dealt with using hard constraints for the instructors, who are often required to attend most classes of the given modules they teach.

2.4 Mathematical formulation

Before detailing the mathematical formulation, specific terminology and sets are introduced to make the presentation of the model more efficient. Firstly, the main variables are defined. Secondly, the objectives to be optimised are defined. Finally, it is explained how these objectives are constrained.

2.4.1 Terminology and notation

Timeslots and timesets

In this model, it is assumed that the university term is split into equal lengths of time called timeslots, and a timeset is defined as a subset of these timeslots. This allows for complicated arrangements that span a number of weeks to be described (with the exact number of weeks given by specific instance data). For example, an arrangement where a

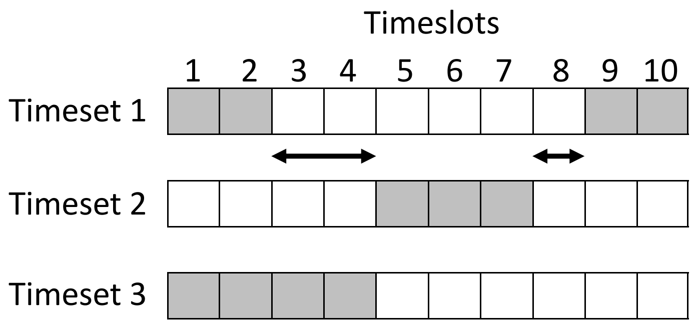


Figure 2.4.1: Illustration of three timesets where there are ten timeslots. The arrows show the "gaps" between the timesets one and two, with the minimum distance "between" the timesets being one timeslot. The intersection of timesets one and three is the set containing timeslots one and two. This means that timesets one and three overlap

class occurs every other week of a 10-week term and starts at 9:30 on Mondays can be described by a single timeset.

We say that two timesets overlap if the intersection of these sets is non-empty. We define the time between two timesets as the minimum number of timeslots between any two timeslots. Figure 2.4.1 illustrates these definitions.

Set definitions

The following list outlines the definitions of the sets that are used in the mathematical formulation of the model. In a slight abuse of notation, we use the set G as a placeholder for another set that would be a subset of some larger set. This is to streamline some of the definitions by avoiding repetition.

S: Set of students.

H: Set of teaching staff.

C: Set of classes.

K: Set of modules.

L: Set of timeslots.

T: Set of timesets.

R: Set of spaces where classes can occur.

K_s: Set of modules requested by student $s \in S$. $K_s \subseteq K$.

K_s^{core} : Set of compulsory modules student $s \in S$ is required to attend. $K_s^{\text{core}} \subseteq K$.

K_s^{elec} : Set of elective modules student $s \in S$ would like to attend. $K_s^{\text{elec}} \subseteq K$.

S_k: Set of students requesting module $k \in K$. $S_k \subseteq S$.

R_r^u: Set of timeslots when room $r \in R$ is unavailable. $R_r^u \subseteq L$.

R_c: Set of spaces that are suitable for class $c \in C$ to use. $R_c \subseteq R$.

T_c: Set of timesets that are suitable for class $c \in C$ to use. $T_c \subseteq T$.

R_G: Let $G \subseteq C$. The set R_G is defined as $R_G := \cap_{c \in G} R_c$.

C_G: Let $G \subseteq C$. The set C_G is defined as $C_G := \{(c_1, c_2) \in G \times G : c_1 \neq c_2\}$.

R_r^C: Let $r \in R$. The set R_r^C is defined as $R_r^C := \{c \in C : r \in R_c\}$.

O_l: Let $l \in L$. The set O_l is defined as $O_l := \{t \in T : l \in t\}$.

F_k: Set of configurations for module $k \in K$.

$P_{f,k}$: Set of subparts for configuration $f \in F_k$, where $k \in K$.

$C_{p,f,k}$: Set of classes for subpart $p \in P_{f,k}$, where $f \in F_k$, where $k \in K$.

C_s: Set of classes that student $s \in S$ could take if offered. In particular, $C_s = \cup_{k \in K_s} \cup_{f \in F_k} \cup_{p \in P_{f,k}} C_{p,f,k}$.

C_h: Set of classes that teaching staff member $h \in H$ must attend if offered.

Travel time

Define A as the matrix with entries that approximate the travel time between pairs of rooms. In particular, for two rooms $r_1, r_2 \in R$, the entry A_{r_1,r_2} is equal to the number of timeslots that it takes to travel from r_1 to r_2. It is assumed that A is symmetric and that $A_{r,r} = 0$ for all $r \in R$.

Hybrid teaching elements

Physical rooms have a finite capacity. For a room $r \in R$, the capacity is denoted as $cap(r)$. The online space is modelled as a room that is always available and can host multiple classes at the same time. This space will be denoted as r^*, and a class $c \in C$ can be held online if and only if $r^* \in R_c$. The capacity of this space is considered unlimited, that is, $cap(r^*) = \infty$. For pedagogical reasons, classes still have a subscription limit, denoted as $sub(c)$ for $c \in C$. This subscription limit is the maximum number of students who can take a particular class.

It is assumed that students can move instantly from one online class to another online class and that it is a fixed number of timeslots $d^* \in \mathbb{N}_0$ to travel from an online class to an in-person class and vice-versa. In particular, $A_{r^*,r^*} = 0$ and $A_{r^*,r} = A_{r,r^*} = d^*$ for all $r \in R \setminus \{r^*\}$.

Classes are allowed to be taught in a hybrid format if the physical room has the proper equipment. Define R^h as a subset of R containing all of the locations that allow for hybrid teaching. For a class to be a hybrid class, then not only does the room assigned to the class need to be in R^h, but the online portion of the class needs to be scheduled for the same time as the in-person class. For consistency, $r^* \in R^h$.

Different students prefer different modes of teaching, or may not have a preference. Define π_s as the preference of student $s \in S$. π_s is equal to one if the student prefers in-person teaching, negative one if the student prefers online teaching, and zero if they have no preference towards either format. It is unlikely that students have no preference towards teaching format. However, when collecting the data from students, there is a chance that some students do not reply and rather than assuming a preference, the lack of a specified preference can be modelled as having no preference.

Module restrictions

Students should not take an excessive number of modules for both financial and pedagogical reasons. We introduce a parameter k_s^{cap} that indicates the maximum number of modules student $s \in S$ is allowed to take.

Parameter arrays

To streamline the discussion of constraints in the timetabling, the notion of a parameter array is introduced. These are fully determined by the input data and record various relationships between timesets and rooms. The following list provides the definitions of these parameter arrays.

D_0: A matrix where $D_0[r, t]$ is equal to one if room r is unavailable at some point during timeset t, zero otherwise.

D_1: A matrix where $D_1[t_1, t_2]$ is equal to one if t_1 overlaps t_2, zero otherwise.

D_2: An array where $D_2[r_1, r_2, t_1, t_2]$ is equal to one if there is not enough time between t_1 and t_2 to travel between r_1 and r_2, zero otherwise.

2.4.2 Variables

The following list outlines all of the variables in the model. As can be observed from the length of this list, there are a significant number of variables in this model. Many of these are auxiliary variables and are primarily included to describe the objectives and constraints succinctly in a mathematical sense. Some pre-processing steps are applied later in this chapter and in Chapter 3 to reduce the number of variables or to relax the integrality of these variables. An efficient implementation of the model may choose to forgo using some of the variables entirely.

$x_{c,r,t}$: Binary decision variable indicating if class $c \in C$ is held in space $r \in R$ during timeset $t \in T$.

$y^R_{c,r}$: Binary decision variable indicating if class $c \in C$ is held in space $r \in R$.

$y^T_{c,t}$: Binary decision variable indicating if class $c \in C$ is held during timeset $t \in T$.

g_k: Binary decision variable indicating if module $k \in K$ is offered.

$q_{k,f}$: Binary decision variable indicating if configuration $f \in F_k$ of module $k \in K$ is offered.

$w_{k,f,p}$: Binary decision variable indicating if subpart $p \in P_{f,k}$ in configuration $f \in F_k$ of module $k \in K$ is offered.

$a_{s,k,f,p,c}$: Binary decision variable indicating if student $s \in S$ is assigned class $c \in C_{p,f,k}$, where $p \in P_{f,k}$, where $f \in F_k$, where $k \in K$.

$\alpha^{\text{onl}}_{s,c}$: Binary decision variable indicating if student $s \in S$ is assigned to the online version of class $c \in C$.

$\alpha^{\text{inp}}_{s,c}$: Binary decision variable indicating if student $s \in S$ is assigned to the in-person version of class $c \in C$.

$b_{s,k,f,p}$: Binary decision variable indicating if student $s \in S$ is assigned some class in subpart $p \in P_{f,k}$, where $f \in F_k$, where $k \in K$.

$m_{s,k,f}$: Binary decision variable indicating if student $s \in S$ is assigned to configuration $f \in F_k$, where $k \in K$.

$n_{s,k}$: Binary decision variable indicating if student $s \in S$ is assigned to module $k \in K$.

$\beta^{\text{onl}}_{s,c,t}$: Binary decision variable indicating if student $s \in S$ attends class $c \in C$ during timeset $t \in T$ in the online format.

$\beta^{\text{inp}}_{s,c,t}$: Binary decision variable indicating if student $s \in S$ attends class $c \in C$ during timeset $t \in T$ in the in-person format.

$\gamma_{s,c,r,t}$: Binary decision variable indicating if student $s \in S$ attends class $c \in C$ in room $r \in R$ during timeset $t \in T$.

$\tau_{s,c}$: Binary decision variable indicating if student $s \in S$ is not attending class $c \in C$ in their preferred mode.

h_{s,c_1,c_2}: Binary decision variable indicating if there is a scheduling issue with assigning student s to $c_1 \in C$ and $c_2 \in C$.

2.4.3 Objectives

Maximise the total number of elective module requests met

Individual students provide a list of elective modules they would like to attend. The aim is to assign students to as many of these modules as possible. This model considers the total requests:

$$\max \ z_1 = \sum_{s \in S} \sum_{k \in K^{\text{elec}}_s} n_{s,k}. \tag{2.4.1}$$

The model does not force the timetable to be complete (feature 22 in Table 2.2.2), so maximising this objective may result in classes with no time or room assignment.

Minimise total number of deviations from mode requests

Students may provide a preference for either the online format or the in-person format. The aim is to align with this preference as much as possible. This model considers the total deviation from mode requests (the amount of mode requests not met):

$$\min \ z_2 = \sum_{s\in S}\sum_{c\in C}\tau_{s,c}. \tag{2.4.2}$$

Minimise the total number of student scheduling issues

There are two scheduling issues considered in this model. The first is where a student is assigned to two classes that overlap in time. The second is where a student is assigned to two classes that are placed in space and time in such a way that it is impossible to travel between them without leaving one class early or arriving at the other late. This model considers the total number of scheduling issues in the timetable:

$$\min \ z_3 = \sum_{s\in S}\sum_{c_1\in C}\sum_{c_2\in C} h_{s,c_1,c_2}. \tag{2.4.3}$$

One advantage of having the number of scheduling issues as a soft constraint instead of a hard constraint is that the model is less likely to become infeasible. Another advantage is that this objective gives another measure of solution quality (Barnhart et al., 2022). In a decision-making context, knowing the number of issues is more informative than infeasibility (Sørensen and Dahms, 2014).

There are often so many students at a university that achieving no issues is nearly impossible. The current practice at universities is for students to meet with a staff member and discuss compromising on module choice to resolve scheduling issues.

2.4.4 Constraints

In this section, the hard constraints of the model are outlined. The main contribution of this thesis is the explicit modelling of hybrid teaching. This contribution is captured in the following key constraints: 2.4.6, 2.4.11, 2.4.12, 2.4.13 and 2.4.30. The remaining constraints are common constraints that are present in the various formulations of the UCTTP seen in the literature.

Linking constraints for resource assignment

It is convenient for the description of the model to be able to switch between the collection of $y^R_{c,r}$ and $y^T_{c,t}$ variables, and the collection of $x_{c,r,t}$ variables. The linking constraints are as follows:

$$y^R_{c,r} = \sum_{t\in T} x_{c,r,t}, \quad \forall r \in R, c \in C, \tag{2.4.4}$$

$$y^T_{c,t} \leq \sum_{r\in R} x_{c,r,t}, \quad \forall t \in T, c \in C, \tag{2.4.5}$$

$$\sum_{r\in R} x_{c,r,t} \leq 2y^T_{c,t}, \quad \forall t \in T, c \in C. \tag{2.4.6}$$

Constraints 2.4.4 state that if $y^R_{c,r}$ indicates that a class c is happening in a room r, then this is if and only if exactly one of the $x_{c,r,t}$ variables indicates the same arrangement. Constraints 2.4.5 and Constraints 2.4.6 combined achieve a similar outcome for time arrangements. Two sets of constraints are needed because the summation in Constraints 2.4.5 and Constraints 2.4.6 can be equal to two due to how hybrid teaching is modelled in this thesis.

Classes can only be assigned compatible teaching spaces and timesets

For each $c \in C$ add the following constraints:

$$\sum_{r\in R} x_{c,r,t} = 0, \quad \forall t \in T \setminus T_c, \tag{2.4.7}$$

$$\sum_{t\in T} x_{c,r,t} = 0, \quad \forall r \in R \setminus R_c. \tag{2.4.8}$$

Classes should not happen in a teaching space when it is not available

$$\sum_{t\in T}\sum_{r\in R} D_0[r,t]x_{c,r,t} = 0, \quad \forall c \in C. \tag{2.4.9}$$

Classes can only be assigned at most one timeset

$$\sum_{t\in T} y^T_{c,t} \leq 1, \quad \forall c \in C. \tag{2.4.10}$$

Classes can only be assigned a maximum of two teaching spaces

$$\sum_{r\in R\setminus\{r^*\}} y^R_{c,r} \leq 1, \quad \forall c \in C, \tag{2.4.11}$$

$$\sum_{r\in R} y^R_{c,r} \leq 2, \quad \forall c \in C. \tag{2.4.12}$$

Constraints 2.4.11 ensure that a class can only be held in at most one in-person teaching space. Constraints 2.4.11 and Constraints 2.4.12 combined then ensure that if there are two teaching spaces assigned, exactly one will be held in person and the other will be held online.

Classes can happen online and in-person if the physical room is appropriate

$$y^R_{c,r^*} \leq 1 - \sum_{r\in R\setminus R^h} y^R_{c,r}, \quad \forall c \in C. \tag{2.4.13}$$

Constraints 2.4.13 ensure that when a class is assigned an in-person teaching space not capable of hybrid teaching, then it is impossible for the class to also be assigned the online teaching space and vice versa.

In-person classes should not use the same teaching space at the same time

$$\sum_{c\in R^c_r} \sum_{t\in O_l} x_{c,r,t} \leq 1, \quad \forall r \in R \setminus \{r^*\}, l \in L. \tag{2.4.14}$$

Module is offered if at least one configuration is offered

$$g_k|F_k| \geq \sum_{f\in F_k} q_{k,f}, \quad \forall k \in K, \tag{2.4.15}$$

$$g_k \leq \sum_{f\in F_k} q_{k,f}, \quad \forall k \in K. \tag{2.4.16}$$

Configuration is offered if and only if every subpart is offered

$$q_{k,f}|P_{f,k}| = \sum_{p \in P_{f,k}} w_{k,f,p}, \quad \forall f \in F_k, k \in K. \tag{2.4.17}$$

Subpart is offered if at least one class in the subpart is offered

$$w_{k,f,p}|C_{p,f,k}||R||T| \geq \sum_{c \in C_{p,f,k}} \sum_{r \in R} \sum_{t \in T} x_{c,r,t}, \quad \forall p \in P_{f,k}, f \in F_k, k \in K, \tag{2.4.18}$$

$$w_{k,f,p} \leq \sum_{c \in C_{p,f,k}} \sum_{r \in R} \sum_{t \in T} x_{c,r,t}, \quad \forall p \in P_{f,k}, f \in F_k, k \in K. \tag{2.4.19}$$

Staff must be able to attend classes they can teach

For each staff member $h \in H$, let $G = C_h$. For each $(c_1, c_2) \in C_G$ add the following constraints:

$$D_2[r_1, r_2, t_1, t_2](x_{c_1,r_1,t_1} + x_{c_2,r_2,t_2}) \leq 1, \quad \forall t_1 \in T_{c_1}, t_2 \in T_{c_2}, r_1 \in R_{c_1}, r_2 \in R_{c_2}. \tag{2.4.20}$$

Student does not attend a module they do not request

$$n_{s,k} \leq 0, \quad \forall k \in K \setminus K_s, s \in S. \tag{2.4.21}$$

Student does not attend a module that is not offered

$$n_{s,k} \leq g_k, \quad \forall k \in K, s \in S. \tag{2.4.22}$$

Student must attend all compulsory modules

$$n_{s,k} = 1, \quad \forall k \in K_s^{\text{core}}, s \in S. \tag{2.4.23}$$

Student does not attend too many modules

$$\sum_{k \in K} n_{s,k} \leq k_s^{\text{cap}}, \quad \forall s \in S. \tag{2.4.24}$$

Student does not attend a class that is not offered

$$\alpha^{\text{inp}}_{s,c} \leq \sum_{t \in T} \sum_{r \in R \setminus \{r^*\}} x_{c,r,t}, \quad \forall c \in C, s \in S, \tag{2.4.25}$$

$$\alpha^{\text{onl}}_{s,c} \leq \sum_{t \in T} x_{c,r^*,t}, \quad \forall c \in C, s \in S. \tag{2.4.26}$$

Student attends a module if they attend a configuration for that module

$$\sum_{f \in F_k} m_{s,k,f} = n_{s,k}, \quad \forall k \in K, s \in S. \tag{2.4.27}$$

Student assigned configuration if they attend a class from each subpart

$$\sum_{p \in P_{f,k}} b_{s,k,f,p} = |P_{f,k}| m_{s,k,f}, \quad \forall f \in F_k, k \in K, s \in S. \tag{2.4.28}$$

Student has at most one class from a subpart

$$\sum_{c \in C_{p,f,k}} a_{s,k,f,p,c} = b_{s,k,f,p}, \quad \forall p \in P_{f,k}, f \in F_k, k \in K, s \in S. \tag{2.4.29}$$

Student attends either the online session or the in-person session

$$a_{s,k,f,p,c} = \alpha^{\text{onl}}_{s,c} + \alpha^{\text{inp}}_{s,c}, \quad \forall c \in C_{p,f,k}, p \in P_{f,k}, f \in F_k, k \in K, s \in S. \tag{2.4.30}$$

Physical room capacities cannot be exceeded

$$\sum_{s \in S_k} \alpha^{\text{inp}}_{s,c} \leq \sum_{r \in R_c \setminus \{r^*\}} cap(r) y^R_{c,r}, \quad \forall c \in C_{p,f,k}, p \in P_{f,k}, f \in F_k, k \in K. \tag{2.4.31}$$

Class subscription capacities cannot be exceeded

$$\sum_{s \in S_k} (\alpha^{\text{inp}}_{s,c} + \alpha^{\text{onl}}_{s,c}) \leq sub(c), \quad \forall c \in C_{p,f,k}, p \in P_{f,k}, f \in F_k, k \in K. \tag{2.4.32}$$

Parent-child classes

It is often the case that some classes are prerequisites for other classes. For example, to attend a workshop in a module, the student should also attend the lecture for that module. Given a student $s \in S$, for every parent/child class pair (with the child class denoted as c_{ch} and the parent denoted as c_{par}), add the following constraint:

$$\alpha^{\text{inp}}_{s,c_{ch}} + \alpha^{\text{onl}}_{s,c_{ch}} \leq \alpha^{\text{inp}}_{s,c_{par}} + \alpha^{\text{onl}}_{s,c_{par}}. \tag{2.4.33}$$

Mode requests

The τ variables need to be linked to the allocation of students.

$$\tau_{s,c} \geq \pi_s \left(\alpha^{\text{onl}}_{s,c} - \alpha^{\text{inp}}_{s,c}\right), \quad \forall s \in S, c \in C, \tag{2.4.34}$$

$$\tau_{s,c} \leq \alpha^{\text{onl}}_{s,c} + \alpha^{\text{inp}}_{s,c}, \quad \forall s \in S, c \in C. \tag{2.4.35}$$

Constraints 2.4.34 force the value to one if the preference is not met, and Constraints 2.4.35 force the value to zero if the class is not attended by that student.

Detect if a student has an overlapping class

For the objective z_3 described in Equation 2.4.3 to correctly detect overlaps, the h_{s,c_1,c_2} variables need to be linked to the assignment of student $s \in S$. This is done by first adding the following constraints:

$$\alpha^{\text{inp}}_{s,c} \times y^{T}_{c,t} = \beta^{\text{inp}}_{s,c,t}, \quad \forall s \in S, c \in C, t \in T, \tag{2.4.36}$$

$$\alpha^{\text{onl}}_{s,c} \times y^{T}_{c,t} = \beta^{\text{onl}}_{s,c,t}, \quad \forall s \in S, c \in C, t \in T. \tag{2.4.37}$$

Next, for every student $s \in S$ and each $(c_1, c_2) \in C_{C_s}$, where C_{C_s} is the pair-wise combinations of classes that the student can take, the following constraints are added:

$$D_1[t_1, t_2](\beta^{\text{onl}}_{s,c_1,t_1} + \beta^{\text{inp}}_{s,c_1,t_1} + \beta^{\text{onl}}_{s,c_2,t_2} + \beta^{\text{inp}}_{s,c_2,t_2}) \leq 1 + h_{s,c_1,c_2}, \quad \forall t_1 \in T_{c_1}, t_2 \in T_{c_2}. \tag{2.4.38}$$

Constraints 2.4.36 to 2.4.38 ensure that h_{s,c_1,c_2} is equal to one if c_1 and c_2 overlap in time and student s is assigned to both of the classes. Constraints 2.4.36 and 2.4.37 are non-linear, but this can be resolved by replacing them with the following constraints:

$$\beta^{\text{inp}}_{s,c,t} \leq \alpha^{\text{inp}}_{s,c}, \quad \forall s \in S, c \in C, t \in T, \tag{2.4.39}$$

$$\beta^{\text{onl}}_{s,c,t} \leq \alpha^{\text{onl}}_{s,c}, \quad \forall s \in S, c \in C, t \in T, \tag{2.4.40}$$

$$\beta^{\text{inp}}_{s,c,t} \leq y^{T}_{c,t}, \quad \forall s \in S, c \in C, t \in T, \tag{2.4.41}$$

$$\beta^{\text{onl}}_{s,c,t} \leq y^{T}_{c,t}, \quad \forall s \in S, c \in C, t \in T, \tag{2.4.42}$$

$$\beta^{\text{inp}}_{s,c,t} \geq \alpha^{\text{inp}}_{s,c} + y^{T}_{c,t} - 1, \quad \forall s \in S, c \in C, t \in T, \tag{2.4.43}$$

$$\beta^{\text{onl}}_{s,c,t} \geq \alpha^{\text{onl}}_{s,c} + y^{T}_{c,t} - 1, \quad \forall s \in S, c \in C, t \in T. \tag{2.4.44}$$

Detect if a student has enough travel time between classes

For the objective z_3 described in Equation 2.4.3 to correctly detect travel time issues, the h_{s,c_1,c_2} variables again need to be linked to the assignment of student $s \in S$. This is done in a similar way to overlap detection and involves first adding the following constraints:

$$\beta^{\text{inp}}_{s,c,t} \times y^{R}_{c,r} = \gamma_{s,c,r,t}, \quad \forall s \in S, c \in C, t \in T, r \in R \setminus \{r^*\}, \tag{2.4.45}$$

$$\beta^{\text{onl}}_{s,c,t} \times y^{R}_{c,r^*} = \gamma_{s,c,r^*,t}, \quad \forall s \in S, c \in C, t \in T. \tag{2.4.46}$$

Then, for every student $s \in S$ and each $(c_1, c_2) \in C_{C_s}$, where C_{C_s} is the pair-wise combinations of classes that the student can take, add the following constraints:

$$D_2[r_1, r_2, t_1, t_2](\gamma_{s,c_1,r_1,t_1} + \gamma_{s,c_2,r_2,t_2}) \leq 1 + h_{s,c_1,c_2},$$
$$\forall t_1 \in T_{c_1}, t_2 \in T_{c_2}, r_1 \in R_{c_1}, r_2 \in R_{c_2}. \quad (2.4.47)$$

These constraints ensure that h_{s,c_1,c_2} is equal to one if student s does not have enough time to travel between classes c_1 and c_2. Once again these are non-linear constraints so Constraints 2.4.45 and Constraints 2.4.46 are replaced with the following:

$$\gamma_{s,c,r,t} \leq \beta^{\text{inp}}_{s,c,t}, \quad \forall s \in S, c \in C, t \in T, r \in R \setminus \{r^*\}, \quad (2.4.48)$$

$$\gamma_{s,c,r,t} \leq y^{R}_{c,r}, \quad \forall s \in S, c \in C, t \in T, r \in R \setminus \{r^*\}, \quad (2.4.49)$$

$$\gamma_{s,c,r,t} \geq \beta^{\text{inp}}_{s,c,t} + y^{R}_{c,r} - 1, \quad \forall s \in S, c \in C, t \in T, r \in R \setminus \{r^*\}, \quad (2.4.50)$$

$$\gamma_{s,c,r^*,t} \leq \beta^{\text{onl}}_{s,c,t}, \quad \forall s \in S, c \in C, t \in T, \quad (2.4.51)$$

$$\gamma_{s,c,r^*,t} \leq y^{R}_{c,r^*}, \quad \forall s \in S, c \in C, t \in T, \quad (2.4.52)$$

$$\gamma_{s,c,r^*,t} \geq \beta^{\text{onl}}_{s,c,t} + y^{R}_{c,r^*} - 1, \quad \forall s \in S, c \in C, t \in T. \quad (2.4.53)$$

2.5 Solution method

The solution method used in this chapter involves a preprocessing stage and then a stage that solves the three objectives in a certain order.

2.5.1 Preprocessing steps

The full mathematical formulation includes some variables and constraints that are not necessary and therefore can be removed from the model. In this section, some of the steps taken to remove redundant variables and constraints are outlined. This does not necessarily remove all redundancies, and whilst more sophisticated reduction methods exist (Holm et al., 2022), these are not employed in this chapter.

Variables that can be removed

The first collection of variables that can be omitted from the model includes $x_{c,r,t}$ variables for each $c \in C$ where either $t \in T \setminus T_c$ or $r \in R \setminus R_c$. This is because Constraints 2.4.7 and 2.4.8 force these variables to be zero. Similarly, if $D_0[r,t] = 1$ for some pairing (r,t) where $t \in T$ and $r \in R$ then for all classes $c \in C$ it is the case that Constraint 2.4.9 is forcing $x_{c,r,t} = 0$ and therefore these variables can be removed from the model.

The second collection of variables that can be omitted are from the student sectioning part of the model. $n_{s,k} = 0$ for $s \in S$ and $k \in K \setminus K_s$ therefore these variables can be removed. Furthermore, any variable with $k \in K \setminus K_s$ in the indexing for a given student $s \in S$ can be removed as these will be forced to zero. Finally, any variable with $c \in C_{p,f,k}$ for $k \in K \setminus K_s$ in the indexing can be removed. Essentially, what is meant by this is that any variable relating to a course that a student does not want to attend is removed from the model.

Overlap and travel time constraints

For any two timesets $t_1 \in T$ and $t_2 \in T$, it is always true that $D_1[t_1,t_2] \leq D_2[r_1,r_2,t_1,t_2]$ for any choice of r_1 and r_2 from the set R. Therefore, if two timesets that overlap are identified and Constraints 2.4.38 to 2.4.44 are included in the model, there is no need to include Constraints 2.4.47 to 2.4.53 for those two timesets. There is a similar process for Constraints 2.4.20 where if two timesets overlap, there is no need to include the constraint for every room combination. We only need to constrain the time assignment using the y^T variables in this case.

Furthermore, when considering a pair of classes c_1 and c_2, it is possible that for a pair of timesets $(t_1,t_2) \in T_{c_1} \times T_{c_2}$ we have that $D_2[r_1,r_2,t_1,t_2] = 0$ for any pair of rooms $(r_1,r_2) \in R_{c_1} \times R_{c_2}$. This means there is no need to include Constraints 2.4.47 to 2.4.53 for this timeset pair (for this pair of classes). These two checks help reduce the total number of constraints, especially if classes have a large number of teaching spaces they could be assigned to.

A final check that is employed is that we do not check for scheduling issues for pairs of classes that we know cannot be attended together. For example, students can only participate in one class from a subpart (Constraint 2.4.29); therefore, we will never have problems with a pair of classes from the same subpart. The impact of this preprocessing step is dependent on the course and module structure assumed.

2.5.2 Objective ordering and solving

The solution method involves solving three single-objective problems sequentially. The objectives are ordered based on importance. This is also known as lexicographic optimisation. One such ordering is as follows:

1. Maximise the total number of elective module requests met (z_1).
2. Minimise the total number of deviations from mode requests (z_2).
3. Minimise the total number of student scheduling issues (z_3).

In this case, the approach would be to first maximise z_1 using a commercial solver. Denote the value of this solution as z_1^*. The constraint $z_1 = z_1^*$ is added, and the model is solved to maximise z_2. Similarly, denote the value of this solution as z_2^*. We add the constraint $z_2 = z_2^*$ to the model and minimise z_3.

In the Section 2.6, for each dataset or instance used, all six orderings of the three objectives described in Section 2.4 are solved. This approach with these orderings produces the extreme points of the Pareto frontier, where at least one of the objective functions takes the best value subject to the model constraints.

This is beneficial as it illustrates the range of objective function values and hints at the dynamics and trade-offs between multiple objectives. However, we do not gain information about compromise solutions (solutions between these six points) that would also be Pareto optimal. This drawback is addressed in Chapter 3.

Whilst this approach yields exact results, the drawback is that it is computationally expensive in both time and physical resources. This is because each objective function that gets added to the model further restricts the model, making the optimisation of the next objective function more difficult. In this chapter, we do not report the timings of this exact optimisation approach. The key feature of interest is the interactions between objectives for different instances.

2.6 Computational experiments

In this section, the experimental data used to test the model is described, and computational experiments are performed to demonstrate the use of the model. All experiments were completed on an internal computing node running Ubuntu 22.04.1 LTS with an Intel Xeon Gold 6348 CPU running at 2.60GHz and 528GB of RAM. The model was implemented in Python 3.9.16, and solutions were found using Gurobi Optimizer version 10.0.2.

2.6.1 Experimental data

The experimental data has been taken from the ITC-2019 competition (Müller et al., 2018). These datasets are based on data taken from real-world universities. These provide the model with most of the information needed. However, since these datasets do not explicitly consider the online space, there are features of the model presented in this chapter that require specifying for each experiment. This includes:

- The value of d^* in the matrix A.
- The value of π_s for each student $s \in S$.

- The value of k_s^{cap} for each student $s \in S$.
- The set R^h that identifies which physical rooms are suitable for hybrid teaching.

In our experiments, we assume that d^* is one and a half times bigger than the largest distance between physical rooms. We chose this number as it roughly estimates the travel time from student accommodation to campus for some universities in the UK we are familiar with. In practice, this value will vary depending on the university. Setting the distance to this value also makes switching between modes undesirable.

To align the data with the motivation for this model, we are also creating a shortage of capacity in physical rooms. This shortage reflects a situation similar to that of the COVID-19 pandemic. Barnhart et al. (2022) quoted that during this period they had a fourfold reduction in physical space on campus, and therefore we are reducing the capacity of each room in the datasets by 75%. Applying this reduction to the four instances means that room capacity is the limiting factor to the total in-person attendance.

It is also assumed that every student has one of three preferences: online teaching, in-person teaching or no preference (π_s equal to -1, 1 or 0 respectively). A systematic way of modifying the data is applied. In the ITC-2019 data, each student has an "ID" number. If a student's ID number is a multiple of three, they prefer the online mode. If the remainder after dividing their ID number by three is two, they have no preference. All other students prefer the in-person mode.

In the ITC-2019 competition, students are assigned to every module they request. This is equivalent to treating every module as compulsory. To demonstrate the objective of maximising the number of elective module requests met, we assume in our experiments that all modules are elective. This means there is no requirement to assign a student to any of the modules they request. The value of k_s^{cap} for each $s \in S$ is the length of the list of requests for that student.

Each "SameAttendee" distribution constraint used to model staff in the ITC-2019 problem has an associated list of classes (Müller et al., 2018). The class lists from the required distributions of this type in the data are used as part of our staff constraints. In particular, each distribution constraint in the data is a staff member $h \in H$ and the class list forms the set C_h used in Constraints 2.4.20.

Finally, we need to specify the set R^h. It is assumed that a class can only happen in the hybrid mode if the physical room assigned to it has a capacity of 30 or more people. In particular, if $cap(r) \geq 30$ for $r \in R$ then $r \in R^h$. The assumption is based on the observation that only the larger lecture halls at universities have the correct audio-visual equipment for hybrid teaching.

The four ITC-2019 instances used in this paper are from various stages of the competition and are currently archived on the competition website. Table 2.6.1 records the name of each instance and the critical features of that instance. To reduce the size of the problem, a subset of students from each instance is used in the experiment. This is defined by the ID of the first student in the subset and how many students are taken after that student. For example, with the instance *mary-fal18*, Table 2.6.1 says that the "Start" is equal to 600 and "Count" is equal to 400, meaning only students 600 to 999 are considered.

Whilst other ITC-2019 instances could have been included, the four instances selected sufficiently demonstrate the behaviour of the model and how the novel element of explicitly including hybrid teaching can behave differently when using different instances. Furthermore, maintaining focus on a limited number of instances allows for a more detailed discussion about each one.

Instance one: wbg-fal10

It is possible to solve any ordering of objectives for *wbg-fal10* as it is a small instance. Table 2.6.2 provides the objective values for the six possible orderings. It can be seen that only z_1 (the number of elective module requests met) and z_2 (the number of

Table 2.6.1: Instances from the ITC-2019 with their key features. "Start" and "Count" indicate the subset of students used in the experiment. Column $|R^h \setminus \{r^*\}|$ is based on our assumptions about hybrid rooms

Instance	$\|S\|$	$\|K\|$	$\|C\|$	$\|T\|$	$\|R\|$	$\|R^h \setminus \{r^*\}\|$	Start	Count
wbg-fal10	19	21	150	154	8	0	1	19
muni-fsps-spr17*	865	48	124	1,953	45	40	600	100
mary-fal18*	5,051	170	357	503	94	35	600	400
pu-cs-fal07*	2,002	34	159	182	14	4	1,000	1,000

mode request deviations) influence each other. Table 2.6.3 shows that lexicographically optimal solutions for this instance either meet all the student mode preferences or offer all requested modules.

Instance two: pu-cs-fal07*

In the instance *pu-cs-fal07**, 1,000 students are considered. Like *wbg-fal10*, no student conflicts arise in this instance, and the only objectives that appear to influence each other are z_1 and z_2. It can be seen from Table 2.6.2 that the lexicographically optimal solutions have one of two objective values. Looking at the class attendance, we see that the ratio of in-person to online attendance is about 1:2, suggesting that when students have no preference for a mode, they get placed into the online mode.

Instance three: muni-fsps-spr17*

Only 100 students from *muni-fsps-spr17** are used; however, each student attends nearly eight times the number of classes on average than in *pu-cs-fal07**. When z_1 is prioritised over z_2, there is a larger increase in z_2 than in *pu-cs-fal07**. This suggests a stronger link between the two objectives in this instance than in *pu-cs-fal07**. There is some relationship between z_2 and z_3 also.

Instance four: mary-fal18*

The *mary-fal18** instance considers 400 students. Table 2.6.2 shows that the lexicographically optimal solutions have one of five objective values with only six possible orderings. That makes this instance a good instance to demonstrate how objective ordering can influence the solution.

Orderings prioritising z_1 lead to solutions with the maximum amount of module requests met. The best possible values of z_2 and z_3 can be attained if we optimise those first (in either order); however, when z_1 is optimised before one or both of these objectives, this cannot be done. This shows that z_1 influences objectives directly and influences relations between objectives (compare orderings z_1, z_2, z_3 with z_1, z_3, z_2).

Figure 2.6.2 shows that if matching the mode preference (z_2) is prioritised over module requests (z_1), then fewer requests can be met. Figure 2.6.1 shows clearly that prioritising z_3 over z_1 removes conflicts but also shows that matching students' mode requests can reduce the number of conflicts (likely due to its direct influence on the number of modules attended by students indicated by Figure 2.6.2).

When students do not request many modules (e.g. master's students) or request too many modules (e.g. students enrolled on modules just for interest), conflicts may be unavoidable. Therefore, minimising the number of conflicts before maximising module attendance can lead to that student attending as low as 0-20% of their requested classes. Similarly, most of a student's requested modules may only be offered in a mode contradicting a student's preference, leading again to a low percentage of classes being attended if mode preference is prioritised over module requests.

2.7 Analysis and extensions

We believe that considering the hybrid teaching format presents new challenges that have not been fully researched in the timetabling literature. Some of these challenges

Table 2.6.2: Objective values for each instance and their six potential objective orderings. z_4 is the number of students who switch between online and in-person classes twice or more on the same day (measured using the found solution, not optimised)

		Objective			Classes attended			Switch
	Ordering	z_1	z_2	z_3	Total	In-person	Online	z_4
wbg-fal10	z_1,z_2,z_3	97	43	0	199	27	172	5
	z_1,z_3,z_2	97	43	0	199	27	172	5
	z_2,z_1,z_3	71	0	0	142	16	126	1
	z_2,z_3,z_1	71	0	0	142	17	125	2
	z_3,z_1,z_2	97	43	0	199	27	172	6
	z_3,z_2,z_1	71	0	0	142	17	125	2
muni-fsps-spr17*	z_1,z_2,z_3	980	106	8	1,552	567	985	13
	z_1,z_3,z_2	980	114	0	1,552	535	1017	11
	z_2,z_1,z_3	926	0	0	1,366	503	863	15
	z_2,z_3,z_1	926	0	0	1,366	450	916	8
	z_3,z_1,z_2	980	114	0	1,552	562	990	11
	z_3,z_2,z_1	926	0	0	1,366	450	916	8
mary-fal18*	z_1,z_2,z_3	1,597	109	36	1,613	517	1,096	13
	z_1,z_3,z_2	1,597	111	34	1,613	513	1,100	13
	z_2,z_1,z_3	1,498	0	26	1,509	536	973	2
	z_2,z_3,z_1	1,480	0	0	1,480	544	936	5
	z_3,z_1,z_2	1,571	98	0	1,571	514	1,057	13
	z_3,z_2,z_1	1,480	0	0	1,480	544	936	5
pu-cs-fal07*	z_1,z_2,z_3	1,226	11	0	1,611	567	1,044	0
	z_1,z_3,z_2	1,226	11	0	1,611	595	1,016	0
	z_2,z_1,z_3	1,220	0	0	1,599	665	934	0
	z_2,z_3,z_1	1,220	0	0	1,599	641	958	0
	z_3,z_1,z_2	1,226	11	0	1,611	560	1,051	0
	z_3,z_2,z_1	1,220	0	0	1,599	641	958	0

Table 2.6.3: Key metrics for individual students in tabular form. "% Electives" is the percentage of elective modules a student is assigned from their list. "# Conflicts" represents the number of scheduling issues a student has. "% Mode" is the percentage of classes a student is assigned that are in their preferred mode

		% Electives			# Conflicts			% Mode		
	Ordering	Max.	Min.	Avg.	Max.	Min.	Avg.	Max.	Min.	Avg.
wbg-fal10	z_1,z_2,z_3	100	100	100.00	0	0	0.00	100	25	77.42
	z_1,z_3,z_2	100	100	100.00	0	0	0.00	100	30.77	77.93
	z_2,z_1,z_3	100	0	73.51	0	0	0.00	100	100	100.00
	z_2,z_3,z_1	100	0	73.51	0	0	0.00	100	100	100.00
	z_3,z_1,z_2	100	100	100.00	0	0	0.00	100	25	77.90
	z_3,z_2,z_1	100	0	73.51	0	0	0.00	100	100	100.00
muni-fsps-spr17*	z_1,z_2,z_3	100	100	100.00	1	0	0.08	100	56.52	94.97
	z_1,z_3,z_2	100	100	100.00	0	0	0.00	100	52.17	94.62
	z_2,z_1,z_3	100	73.33	95.75	0	0	0.00	100	100	100.00
	z_2,z_3,z_1	100	73.33	95.75	0	0	0.00	100	100	100.00
	z_3,z_1,z_2	100	100	100.00	0	0	0.00	100	52.17	94.62
	z_3,z_2,z_1	100	73.33	95.75	0	0	0.00	100	100	100.00
mary-fal18*	z_1,z_2,z_3	100	100	100.00	6	0	0.09	100	0	94.62
	z_1,z_3,z_2	100	100	100.00	6	0	0.09	100	0	94.54
	z_2,z_1,z_3	100	0	95.08	6	0	0.07	100	100	100.00
	z_2,z_3,z_1	100	0	94.36	0	0	0.00	100	100	100.00
	z_3,z_1,z_2	100	66.67	98.83	0	0	0.00	100	0	94.96
	z_3,z_2,z_1	100	0	94.36	0	0	0.00	100	100	100.00
pu-cs-fal07*	z_1,z_2,z_3	100	100	100.00	0	0	0.00	100	0	99.66
	z_1,z_3,z_2	100	100	100.00	0	0	0.00	100	0	99.66
	z_2,z_1,z_3	100	0	99.67	0	0	0.00	100	100	100.00
	z_2,z_3,z_1	100	50	99.70	0	0	0.00	100	100	100.00
	z_3,z_1,z_2	100	100	100.00	0	0	0.00	100	0	99.66
	z_3,z_2,z_1	100	50	99.70	0	0	0.00	100	100	100.00

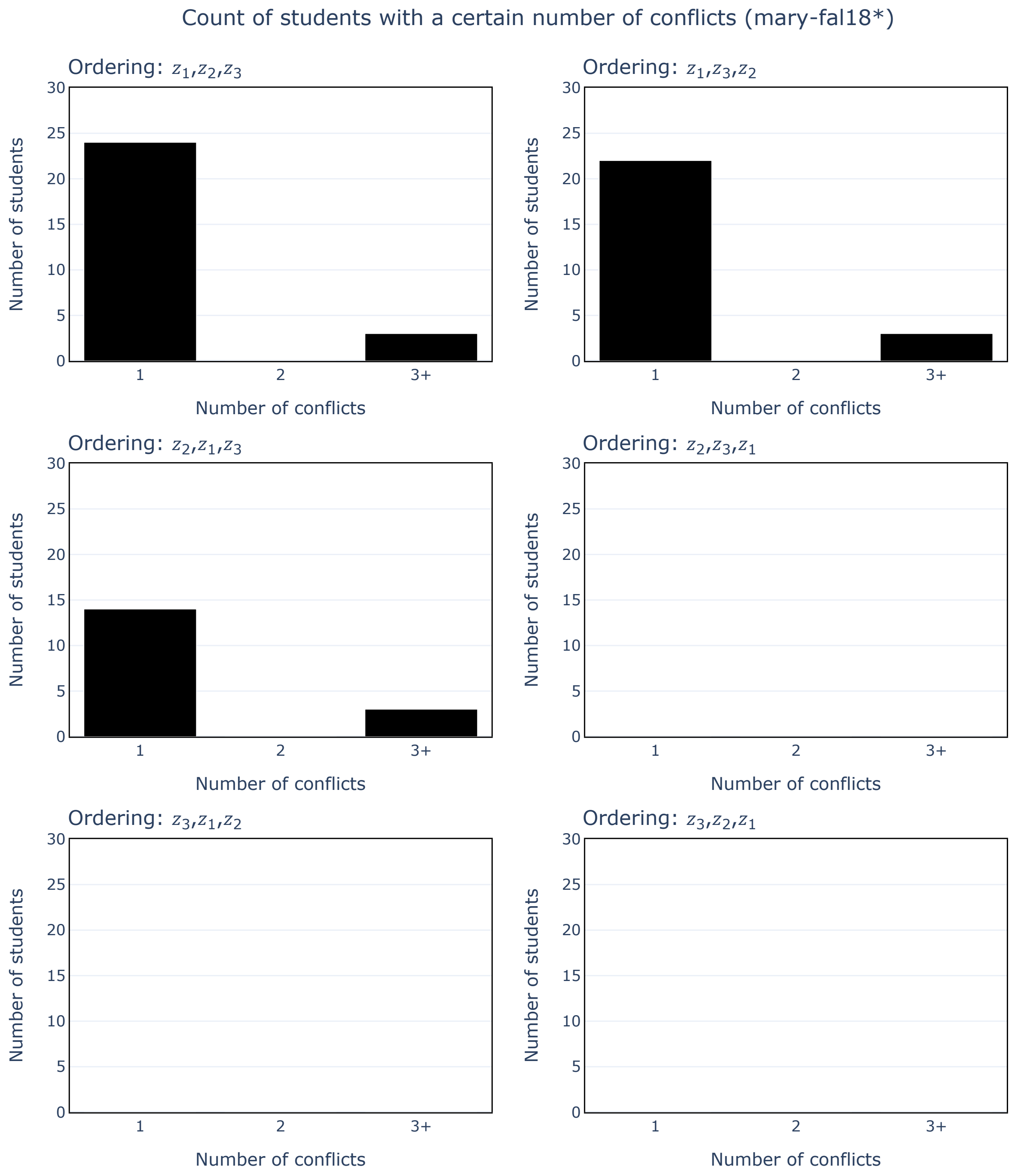


Figure 2.6.1: Count of students who experience one, two or three plus conflicts with their timetable. The majority of students do not experience any conflicts; therefore, the zero bar is omitted

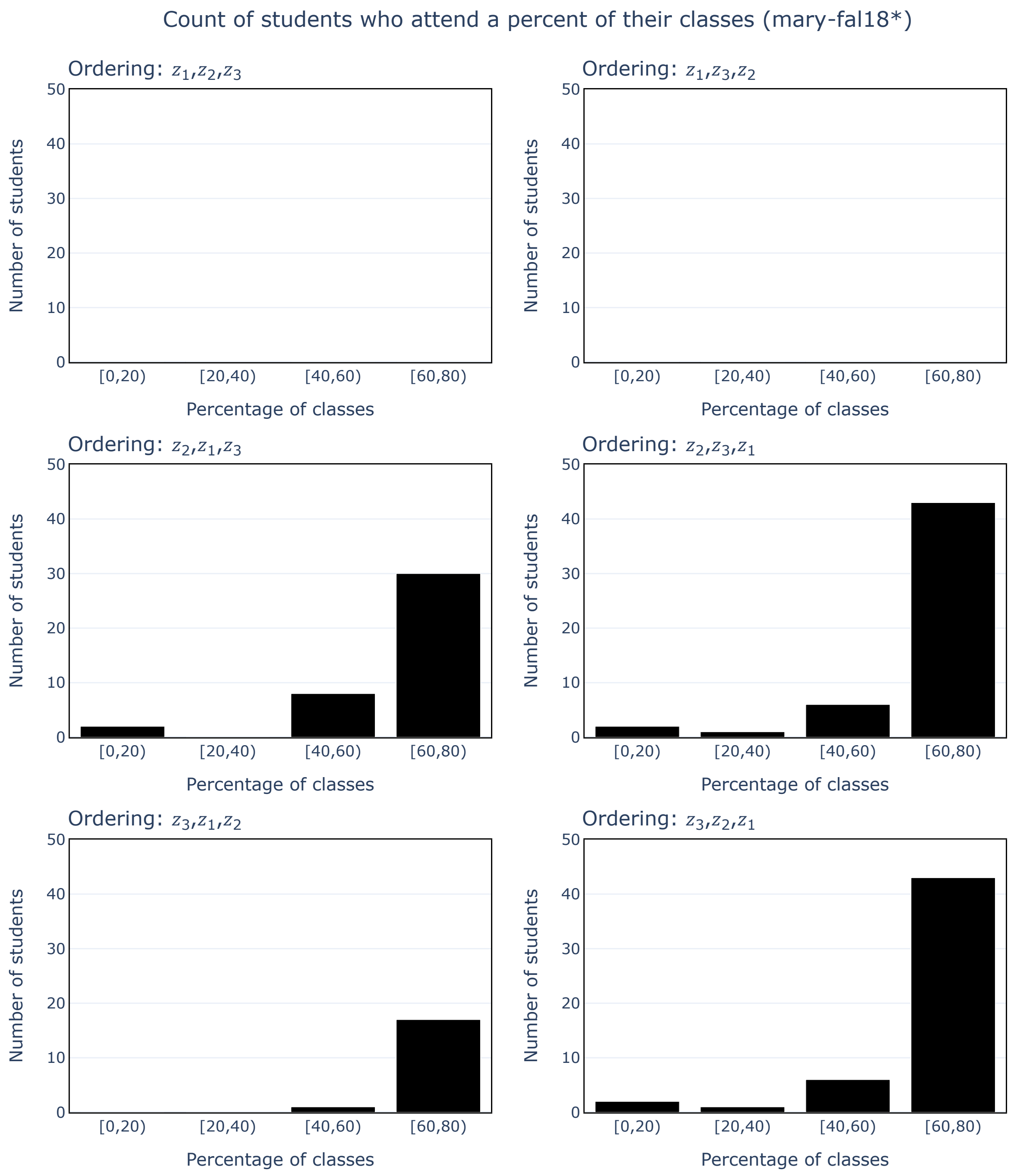


Figure 2.6.2: Count of students who have a certain percentage of their classes met. They are aggregated into ranges so that the bars are an appropriate size. The $[80, 100]$ range is omitted, but contains the remaining students

are discussed here.

2.7.1 Impact of including hybrid teaching

From Table 2.6.2, it can be seen that many classes are attended online. This demonstrates how well the model presented in this chapter satisfies the mode preferences. By construction, one-third of the student population in the instances preferred in-person teaching, and the other two-thirds either had no preference or preferred online teaching. This is roughly represented in the ratio of classes attended in-person to classes attended online. The results in the previous section also suggest that the model takes advantage of the students who have no preference and assigns them to the more flexible and high-capacity online space. This alleviates pressure on physical resources while delivering the same amount of teaching. From a modelling point of view, the inclusion of hybrid teaching adds more flexibility to the timetabling process because a purely online timetable is not impacted by travel times and room capacity in the same way as a purely in-person timetable. In practice, it would be up to a practitioner to decide to what extent the online space should be used.

2.7.2 Impact of objective ordering

If a university has preconceived thoughts on how the objectives should be ordered, then this model can cater to those preferences. This model can also be used to gain more information about the problem. Trying different orders of objectives can be used to gain insight into how and to what extent the objectives influence each other.

One example is that in the instance *mary-fal18**. It was identified here that different orderings typically resulted in differing solutions. A university may have an idea of an objective order they want, but find that a different ordering is better suited to their needs.

Table 2.6.2 shows that differences in the solutions generated by different orderings might be very different or only have subtle differences. This demonstrates that whilst the ordering of objectives does impact the difference in solutions, it seems that the extent of the difference is unique for each instance.

2.7.3 Model extensions

Compulsory modules and credit loads

Constraints 2.4.23 describe compulsory modules. The model becomes infeasible if a student cannot be offered a mandatory module. This is not a useful finding in practice as universities ultimately need a timetable (Sørensen and Dahms, 2014). One approach would be to relax the constraint and instead attach a high penalty for not ensuring every student is assigned to their mandatory modules. Including hybrid teaching offers flexibility as it facilitates a meaningful two-stage approach of time assignment and then room assignment (Barnhart et al., 2022).

It is common for a student to not only have mandatory or compulsory modules but to be required to attend a certain total number of modules, making up a credit load. The model presented in this chapter could be extended so that students are assigned enough modules to meet a specific credit load. This would require students to provide an extended list of elective modules so there are sufficient options.

Fairness and extended preferences

All the objective functions present in this chapter are aggregate measures, so there is no concept of “fairness” when assigning students. For example, if two students request two modules, then a solution where one student is assigned to two modules and the other is assigned to none is equivalent in objective value to a solution where both students are assigned to one module each. The latter solution is arguably “fairer”.

A similar situation can happen with preferences. It is possible that a solution could assign students so that some students attend the mode they prefer for every class, whilst some students attend none of their classes in their preferred mode. Currently, each student has a mode preference for all modules. The model could be extended to consider a student's mode preferences for each module or even mode preferences for each class. This extension makes the problem more true to life but exacerbates the issues surrounding fair assignments.

Controlling mode of attendance

Universities may want control over what mode students study in. The mode that students attend their classes relates heavily to where students choose to live, how busy the campus is, and how resources are used on campus at the expense of the university.

For example, if a university has on-site student accommodation, it may aim to have a minimum percentage of classes attended in person. A university would do this to make renting this accommodation a more appealing choice for students. On the other hand, a university may wish to limit the percentage of teaching done in person. This may be to reduce student density on campus, either to reduce the spread of infection or to reduce the amount of energy and money spent on lighting and heating buildings at the university.

More parameters and constraints would need to be introduced to model this extension, but it would be useful for a practitioner to investigate the effect of changing these parameters.

Switching between modes of study

A timetable where students and staff make multiple switches between modes on the same day is undesirable. Multiple switches could mean more traffic on and around the university, which has both a health and environmental impact (potential spread of disease and increase in vehicle emissions). Whilst many students only attend a single mode of study or switch between the two modes only once, Table 2.6.2 shows that without any optimisation, there are students who switch between modes twice or more on the same day. An extension to this model would be to have an objective to minimise the number of switches that each student makes in a working day. However, research currently underway indicates that solving this problem is computationally challenging.

2.8 Conclusion

Several gaps in the university module timetabling literature are identified at the beginning of this chapter. To address these gaps, this chapter presents a formal and mathematical description of a multi-objective post-enrolment timetabling model with student sectioning and considerations for hybrid teaching. The novel feature of this work is the explicit modelling of hybrid classes, which have an in-person and online element occurring at the same time. The assignment of students in this model is demand-driven. Individual students request the modules they want to take, and the model creates a timetable that tries to satisfy these requests. It also tries to ensure students attend their teaching in the mode that they prefer. This is demonstrated in a series of computational experiments using modified benchmark data from the 2019 International Timetabling Competition. The modifications are so we can demonstrate the novel features under a setting where these features would be most relevant (for example, severely reduced capacity). In these experiments, an exact lexicographic solution method is used to show that solutions to this model depend on both the input data and the ordering of the objectives. This observation shows that the model can be used by practitioners who have different strategic priorities and are based at different universities. Finally, a list of potential extensions to this model is presented. Items on this list are starting points for future research. Exploring these extensions would benefit both the timetabling community and the wider large-scale integer optimisation community.

Chapter 3

Multi-objective Fix-and-Optimise Matheuristic for University Course Timetabling

3.1 Introduction

Many real-world optimisation problems are modelled as single-objective optimisation problems when a multi-objective model would be more appropriate. Sometimes this is an easy task, especially when all the objectives can be expressed in the same measure, such as cost. For the university course timetabling problem (UCTTP), many possible objectives that could be optimised are incommensurable. For example, it is not sensible to compare maximising room utilisation with minimising the length of the working day. However, keeping these objectives separate presents new challenges, such as deciding which objectives take priority over others.

To exacerbate this issue, the decisions made when solving the UCTTP do not impact goods, vehicles, or other machinery. The decisions made impact people. Consequently, there will often be no agreed-upon ordering of objectives and finding a “one-size-fits-all” solution becomes impossible. Nobody else in the literature has summed it up better than Carter (2001), who claimed that “practical course timetabling is 10% graph theory and 90% politics!”.

Given this paradigm, is it appropriate to keep developing algorithms that focus on producing a single high-quality solution? While this remains an important area of research, we argue that solving the UCTTP extends beyond the operational task of generating a single timetable. Specifically, the university timetabling literature has not yet adequately addressed the following interrelated research questions:

- How can we find a meaningful and useful range of objective values for multiple objective functions?
- How can we analyse the interactions between incommensurable objectives?
- How can we efficiently generate a set of alternative solutions for decision-makers?

Without a clear range of objective values, an understanding of trade-offs between objectives, and a diverse set of alternative solutions, decision-making becomes constrained and suboptimal. For large-scale UCTTPs, addressing these questions with exact methods is computationally infeasible, and incommensurable objectives make it difficult to design appropriate fitness functions for a genetic algorithm to optimise. Developing a method that overcomes these difficulties will result in more adaptable and practically useful timetabling solutions.

To attempt to address these questions, the objectives of this chapter are the following: (i) propose a method for partitioning the problem and using this to find bounds on objective values, (ii) apply a matheuristic to optimise a large multi-objective university course timetabling problem lexicographically, and (iii) present a method for exploring a multi-objective Pareto front and describe how this could be used in a strategic decision-making context.

The rest of the chapter is structured as follows. In Section 3.2, a review of existing work in the context of university timetabling is provided. In Section 3.3, a brief description of the problem we are applying the method to is given, along with a reminder of the key elements of the mathematical formulation for this problem. A method for partitioning and finding objective bounds is given in Section 3.4. The main solution methods are described in Section 3.5 and Section 3.6. Finally, Section 3.7 summarises the work done in this chapter.

3.2 Related work

The study of the related work will be structured as follows. First, we will make note of review papers that discuss all approaches, ranging from exact approaches to genetic algorithms. Key papers discussing interesting methods or applications will also be highlighted here. Secondly, we discuss the development and application of metaheuristics and matheuristics to the UCTTP. Finally, methods used for multi-objective optimisation of the UCTTP are discussed.

Literature reviews and key papers

One of the most up-to-date reviews of solution methods used for the UCTTP is given by Chen et al. (2021). Other methodology surveys for the UCTTP include Babaei et al. (2015) and Alghamdi et al. (2020). For surveys focused on heuristic solution methods for the UCTTP, Lewis (2008) looks at metaheuristic approaches and Pillay (2016) looks at hyper-heuristic approaches. The final survey papers we would like to mention are Bettinelli et al. (2015), where methods used as part of the third track in the International Timetabling Competition (ITC) of 2007 (Di Gaspero et al., 2007) are assessed and Müller et al. (2024), where a similar analysis is done in the context of the ITC-2019.

The key papers we will now mention are highlighted due to their influence on the ideas in this paper. The first is Carter (2001), who was motivated to publish details of their course timetabling and student scheduling system due to a lack of literature at the time. One key aspect of this work is sectioning students based on the similarity of their course requests to reduce the chance of conflicts. Clustering students based on similarity is used primarily in Section 3.4, although it also forms part of the neighbourhood selection in Sections 3.5 and 3.6.

The second paper is Lindahl et al. (2017), which takes a strategic view of university timetabling. They form three integer programs with different quality measures as objectives and investigate pair-wise interactions between these objectives using an ϵ-constraint approach. Section 3.6 of this chapter is inspired by this approach, where a method for approximating interactions between different objectives is presented.

The third paper is Mikkelsen and Holm (2022). This paper uses a sophisticated fix-and-optimise routine to solve instances for the ITC-2019. This procedure works by repeatedly fixing a large number of variables and optimising the sub-problem created from this process, therefore avoiding the computational burden of solving the entire problem. Whilst a fix-and-optimise routine is used in this chapter (Sections 3.5 and 3.6), our choice of neighbourhood size varies depending on what we are trying to achieve.

Use of metaheuristics

Heuristic approaches have existed for over 50 years (Simon and Newell, 1958). The term “metaheuristic”, often used to describe a framework of heuristics, was coined in the 1980s (Glover, 1986). These methods are popular because they can find good solutions to difficult problems quickly.

As the UCTTP is an NP-hard problem (Babaei et al., 2015), many papers utilise metaheuristic solution methods. There are too many to list exhaustively here; however, we name a couple to illustrate the variety of methods. Landa-Silva and Obit (2008) use a great-deluge algorithm to search neighbouring solutions and select improving solutions or solutions that only deteriorate slightly. De Causmaecker et al. (2009) use a move-based metaheuristic to construct and modify timetables. Lewis and Thompson (2015) use a two-stage approach where the first stage inserts events into the timetable and the second uses a simulated-annealing approach (Kirkpatrick et al., 1983) and neighbourhood operators to modify the solution.

There are many ways of combining metaheuristic approaches into a single approach. For a good example of this in the context of UCTTP, see Müller (2008).

Use of matheuristics

Due to the improvement of mixed-integer program (MIP) solvers, some problems that required metaheuristic methods can now be solved exactly. Some researchers of university course timetabling problems now formulate an MIP and use a solver (for example, Gonzalez et al. (2018)). Other exact methods, such as cuts (Fonseca et al., 2017) and graph theoretic techniques (Holm et al., 2022), can be used to tighten formulations of MIPs or remove redundant constraints/variables.

However, large MIPs are still hard to solve, and therefore, people leverage the benefits of exact and heuristic approaches using matheuristic approaches. Matheuristics are metaheuristics that use elements of exact methods. One example in the context of university timetabling is Méndez-Díaz et al. (2016) where a relaxation of an integer linear program (ILP) is used to determine what assignments to fix when solving the original ILP. Another is the already highlighted approach in Mikkelsen and Holm (2022) where neighbourhoods are defined by selecting a group of variables to unfix, and these variables are optimised using the commercial solver Gurobi.

Multi-objective optimisation

Many models in the literature have understood that the UCTTP is a multi-objective problem in some capacity. One early example of a multi-objective model is in Badri (1996). This model, and many that follow (such as Daskalaki et al. (2004) and Al-Yakoob and Sherali (2007)), use a weighted-sum approach where the objective weights indicate preferences. However, simple examples can be constructed to demonstrate the flaws of treating preferences in this way (Miettinen, 2014). Marler and Arora (2010) identify fundamental deficiencies with this approach, indicating that the weighted-sum approach may not be suitable for a multi-objective UCTTP.

One interesting approach in the literature is the subcost-guided search seen in Wright (2001). The problem being solved is a school timetabling problem with a weighted-sum objective; however, each element of the objective is considered a "subcost". A modified simulated annealing approach is then used, where solutions with a worse overall objective value can be selected by chance or because an improvement in a specific subcost was found. This adjustment means the search for diverse solutions is more guided and not reliant on probability alone.

An example in the literature where an ϵ-constraint method is used instead of a weighted-sum approach is Lindahl et al. (2017). In this paper, this method is applied to bi-objective programs to investigate interactions between the objectives without needing weights or scaling. The drawback of this method is that it requires solving many single-objective problems sequentially, each of which could be difficult to solve. To solve all orderings of n objectives, $n \times n!$ single-objective problems must be solved (Ehrgott, 2005).

Both ϵ-constraint methods and weighted-sum methods require some scalarization, where the problem is transformed to a single-objective problem. Approaches with less need for scalarization for multi-objective optimisation include population-based algorithms such as multi-objective simulated annealing and genetic algorithms (GAs)

(Keller, 2017). Recent literature suggests that genetic algorithms, often hybridised with local search techniques, represent a popular approach in practical multi-objective applications (Chen et al., 2021). GAs are good at multi-objective applications because they maintain a diverse population and can explore multiple trade-offs simultaneously by making many mutations over many generations of the population.

One method presented in the domain of UCTTP is Gülcü and Akkan (2020), where the aim is to produce robust timetables. Of course, the main drawback of these approaches is that they do not provide information on how far from optimality the solutions generated are. In single-objective optimisation, it is important to have bounds to indicate why your solutions are good. Bounds are even more critical in multi-objective optimisation as they define the feasible objective space containing all Pareto optimal solutions (Miettinen, 1998), providing decision-makers with insight into the range of possible trade-offs.

3.2.1 Contributions

This review of the related work identifies key gaps in the literature regarding multi-objective variants of the UCTTP. The first gap is the absence of an effective approach for bounding objective values, not only individually but also considering the interdependencies between multiple objectives. The second gap is the lack of a multi-objective matheuristic applied to the UCTTP, specifically for identifying alternative solutions that can support operational and strategic decision-making. With these gaps in mind, the key contributions of this chapter are:

- A novel partitioning approach for students, leveraging clustering and graph-theoretic techniques, to compute bounds on objective values.
- A matheuristic method for identifying the extreme points of a Pareto frontier.
- An adaptation of this matheuristic method that can approximate the remainder

of the Pareto frontier, while integrating decision-maker preferences.

3.3 Problem description

The problem addressed in this chapter is the binary program described in Chapter 2. As demonstrated in Chapter 2, data from the ITC-2019 timetabling competition (Müller et al., 2018) can be used as input data. The novel contribution of this chapter was a UCTTP model that explicitly incorporated hybrid teaching and objectives related to this aspect of university timetabling. Specifically, that chapter examined the trade-offs among three objectives:

z_1: Maximise the total number of elective module requests met.

z_2: Minimise the total number of deviations from mode requests.

z_3: Minimise the total number of student scheduling issues.

In Chapter 2, all lexicographic orderings of these three objectives were solved to optimality for four modified instances from the ITC-2019 competition. The conclusion drawn from this work was that solving larger instances with more objectives requires a more advanced method. This multi-objective problem demands a solution approach that can:

1. Efficiently solve lexicographic orderings of objectives.

2. Simultaneously explore trade-offs between multiple objectives.

3.3.1 Experimental data

The experimental data used in this chapter consists of benchmark datasets from the ITC-2019 competition (Müller et al., 2024). These datasets undergo the same modifications outlined in Chapter 2. Students have been removed from three instances and are referred

Table 3.3.1: Instances from the ITC-2019 with student count, course count, class count, timeset count, room count and a reason for its inclusion in the test set. Instances marked with a * are the reduced instances from Chapter 2

Instance	$\|S\|$	$\|K\|$	$\|C\|$	$\|T\|$	$\|R\|$	Reason instance is included
wbg-fal10	19	21	150	154	8	Used in Chapter 2
muni-fsps-spr17*	100	48	124	1,953	45	Used in Chapter 2
mary-fal18*	400	170	357	503	94	Used in Chapter 2
muni-fsps-spr17	865	226	561	1,953	45	Full size of muni-fsps-spr17*
pu-cs-fal07*	1,000	34	159	182	14	Used in Chapter 2
pu-cs-fal07	2,002	44	174	182	14	Full size of pu-cs-fal07*
mary-fal18	5,051	540	951	503	94	Full size of mary-fal18*
pu-d5-spr17	13,497	212	1,061	338	85	Medium student count
pu-llr-spr17	27,018	687	1,001	993	76	Large student count

to as "reduced instances". Table 3.3.1 provides an overview of the problem instances explored in this chapter.

We do not include all instances from the ITC-2019 competition. Instead, we focus on nine problem instances to demonstrate the effectiveness of the proposed method. Four of these instances were chosen because they were analysed in Chapter 2, allowing us to use the objective values found in that work to assess the method's performance. The remaining five instances were selected to ensure a diverse set of university sizes, enabling us to showcase the scalability of the approach.

3.4 Lexicographic bounds

As discussed in previous literature (for example, Carter (2001)), clustering students can aid in solving the timetabling problem or help in finding bounds. In this chapter, students are clustered based on the similarity of their module requests.

Due to the size and complexity of large UCTTP problems, solving them exactly to optimality is often impractical. Therefore, it is necessary to obtain bounds on the objective values to assess how close the solutions produced by the matheuristic are to the optimal solutions. These bounds can be approximately obtained through the following three-step process:

1. Partition the problem into manageable sub-problems.
2. Solve each smaller sub-problem independently.
3. Aggregate the objective values from the sub-problems to derive a bound.

3.4.1 Partitioning by clustering

When partitioning students into blocks of students, it is important to keep students with similar or the same module requests together in the same block, as this is where the most significant interactions occur. For example, room capacity violations happen only when students are scheduled for the same classes.

The partitioning approach in this chapter involves a two-step process: a "top-down" method that splits students into partially module-independent blocks, followed by a "bottom-up" divide-and-cluster method to refine these blocks into more manageable sub-problems. Finally, the blocks are combined to ensure all blocks have roughly the same number of students.

An essential part of this method is defining a measure of "distance" or "similarity" between any two students.

Distance between students

The "distance" between students $s_1 \in S$ and $s_2 \in S$ is defined as:

$$d(s_1, s_2) = 1 - \frac{|K_{s_1} \cap K_{s_2}|}{|K_{s_1} \cup K_{s_2}|} \tag{3.4.1}$$

This is identical to the Jaccard distance between the sets K_{s_1} and K_{s_2}, which is known to satisfy all the properties of a metric (Kosub, 2019). In particular, it is worth noticing that for any two students $s_1, s_2 \in S$, the following properties hold:

1. $d(s_1, s_2) = 1$ if and only if s_1 and s_2 do not have any of the same module requests.
2. $d(s_1, s_2) = 0$ if and only if s_1 and s_2 have exactly the same module requests.
3. $d(s_1, s_2) = d(s_2, s_1)$ for all s_1 and s_2.

By using the distance defined in Equation 3.4.1, we can produce an $|S| \times |S|$ matrix where the entry on the ith row and jth column is $d(s_i, s_j)$.

Maximum block size

Let B be the set of blocks forming a partition of S. Each block, $b \in B$, induces a sub-problem of the original UCTTP where: (i) the only students in the sub-problem are those in that block and (ii) the only modules in the sub-problem are those requested by students in that block.

The bound finding procedure requires these sub-problems to be tractable in size. Let $B^{\max}$ be the largest size a block of a partition can be. In this chapter, the value of $B^{\max}$ for each instance is the smallest of the following:

1. The number of requests for the most popular module.
2. The capacity of the largest physical space in the instance.

Partitioning into blocks using connected components

A graph can be constructed where each student is a node and each edge represents whether two students have a module request in common. This graph would have an adjacency matrix such that the entry on the ith row and jth column is one if $d(s_i, s_j) < 1$ and zero if $d(s_i, s_j) = 1$.

The collection of connected components of this graph is equivalent to the collection of sets of students who have at least one module request in common with another student in the same group. Identifying all connected components is identical to partitioning students into module-independent blocks. For example, one block may contain only humanities students, and another may only contain science students.

These blocks in practice may be larger than the defined $B^{\max}$. For example, many Science, Technology, Engineering, and Mathematics (STEM) subjects have overlapping modules and would be in the same partition block. Let $tol \in [0, 1]$ be the tolerance for module similarity. A more general adjacency matrix, A^{tol}, can now be defined as:

$$A^{tol}_{i,j} = \begin{cases} 1 & d(s_i, s_j) < 1 - tol \\ 0 & \text{Otherwise.} \end{cases} \tag{3.4.2}$$

By gradually increasing the value of tol, it is possible to partition blocks created using a lower tolerance into multiple blocks. Algorithm 1 outlines exactly what tolerances we use and the termination criteria of this partitioning.

Algorithm 1 Graph theoretic partition (GTP)

```
1:  function GTP($S$, $B^{\max}$, $\alpha \in \mathbb{Z}_{>0}$)
2:      $B \Leftarrow \{S\}$
3:      $tol \Leftarrow 0$
4:      while $tol < 1$ do
5:          $B^{\text{new}} \Leftarrow \emptyset$
6:          for $b \in B$ do
7:              if $|b| \leq \alpha B^{\max}$ then
8:                  $B^{\text{new}} \Leftarrow B^{\text{new}} \cup \{b\}$
9:              else
10:                 $B^{\text{new}} \Leftarrow B^{\text{new}} \cup \text{GTP-B}(tol, b)$
11:             end if
12:         end for
13:         $tol \Leftarrow tol + 0.1$
14:         $B \Leftarrow B^{\text{new}}$
15:     end while
16:     return $B$
17: end function
```

Algorithm 2 GTP Block Specific (GTP-B)

```
1: function GTP-B($tol$, $b$)
2:     Construct $A^{tol}$ using Equation 3.4.2
3:     Construct graph $G$ where:
4:         Nodes of G $\Leftrightarrow s \in b$
5:         Edges of G $\Leftrightarrow A^{tol}_{i,j} = 1$ where $i, j \in b$
6:     Suppose $G$ has $k$ connected components
7:     Component $i$ has set of nodes $b_i$
8:     return $\{b_1, \ldots, b_k\}$
9: end function
```

Clustering students into blocks

After a partition is produced using the graph-theoretic approach, the blocks can be split further using agglomerative hierarchical clustering. Agglomerative hierarchical clustering starts by placing each student into their own block and then groups blocks based on distance until the desired number of clusters is reached (Gan et al., 2007). This step tries to ensure that each block contains students who request the same or similar modules. Algorithm 3 outlines the details of this process.

Combining blocks

Finally, smaller blocks are combined with larger ones to produce a partition with fewer blocks, simultaneously ensuring each block is smaller than $B^{\max}$. Whilst this undoes some of the effort to keep students with differing module requests separate, if the combined blocks can be solved exactly, then this is better for the bound. Algorithm 5 details how this is done.

Finalising the partition

The following steps summarise how the student partitions are created:

1. Algorithm 1 splits students into blocks using graph theory methods (Using Algorithm 2 as a subroutine).

Algorithm 3 Partition block $b \in B$ using clustering (CLB)

```
function CLB($b$, $B^{\max}$)
    $B^{\text{fin}} \Leftarrow \{\}$
    while True do
        if $|b| \leq B^{\max}$ then
            $B^{\text{fin}} \Leftarrow B^{\text{fin}} \cup \{b\}$
            return $B^{\text{fin}}$
        end if
        $c^{\max} \Leftarrow \lfloor |b| - B^{\max} + 1 \rfloor$
        $c^{\min} \Leftarrow \lfloor B^{\max} / c^{\max} \rfloor + 1$
        while $c^{\text{curr}} \neq c^{\min}$ do
            $c^{\text{curr}} \Leftarrow \lfloor (c^{\max} - c^{\min})/2 \rfloor$
            $B^{\text{new}} \Leftarrow \text{CLB-N}(b, c^{\text{curr}})$
            $b^* \Leftarrow \arg\max_{b \in B^{\text{new}}} |b|$
            if $|b^*| \leq B^{\max}$ then:
                $c^{\max} \Leftarrow |B^{\text{new}}|$
            else
                $c^{\min} \Leftarrow |B^{\text{new}}|$
            end if
        end while
        $B^{\text{fin}} \Leftarrow B^{\text{fin}} \cup \{b^*\}$
        $b \Leftarrow b \setminus b^*$
    end while
end function
```

Algorithm 4 Partition block $b \in B$ into n (CLB-N)

```
function CLB-N($b$, $n$)
    Construct matrix $D$ where $D_{i,j} = d(s_i, s_j)$
    Use agglomerative clustering on $b$
    Stop when the $b$ is clustered into $n$ parts
    return clustering
end function
```

Algorithm 5 Combine blocks (COB)

```
function COB($B$, $B^{\max}$)
    $k \Leftarrow |B|$
    $B^{\text{new}} \Leftarrow \emptyset$
    $J \Leftarrow \emptyset$
    for $i$ from 1 to $k$ do
        $b \Leftarrow \emptyset$
        for $j$ from 1 to $k$ do
            if $|b \cup b_j| \leq B^{\max}$ and $j \notin J$ then
                $J \Leftarrow J \cup \{j\}$
                $b \Leftarrow b \cup b_j$
            end if
        end for
        if $0 < |b|$ then
            $B^{\text{new}} \Leftarrow B^{\text{new}} \cup \{b\}$
        end if
    end for
    return $B^{\text{new}}$
end function
```

2. Algorithm 3 splits the blocks within this partition using clustering (Using Algorithm 4 as a subroutine).

3. Algorithm 5 combines small blocks into larger blocks.

The partition produced using this process will be called the "true partition". Table 3.4.1 lists the instances from Table 3.3.1 and details the number of requests for the most requested module and the capacity (before reduction, as discussed in Chapter 2) of the largest room. This table also outlines the value of α used in Algorithm 1 and the number of blocks in the resulting partition.

We define the "naive partition" as the partition of students constructed by placing the first $B^{\max}$ students in the first block, the next $B^{\max}$ students in the second block, and so on. This partition will be used to assess how effectively the "true partition" keeps similar students together. For each instance and partitioning approach, the mean average distance between students is calculated. If the true partition has a lower mean average distance than the naive partition, then this indicates that similar students are

Table 3.4.1: Instances with number of requests for the most requested module, the capacity of the largest room, the maximum block size, the multiplier for Algorithm 1 and the partition size

Instance	Requests	Capacity	$B^{\max}$	α	$\|B\|$
wbg-fal10	12	4	4	4	5
muni-fsps-spr17*	36	126	36	4	4
mary-fal18*	56	100	56	4	8
muni-fsps-spr17	203	126	126	4	8
pu-cs-fal07*	249	61	61	4	18
pu-cs-fal07	535	61	61	4	34
mary-fal18	446	100	100	4	51
pu-d5-spr17	1,037	61	61	4	222
pu-llr-spr17	1,672	480	480	4	57

grouped more effectively in the true partition. Table 3.4.2 shows that in nearly all cases, the true partition manages to split students while keeping similar students together better than the naive partition.

For *wbg-fal10*, the two partitions yield similar values, with the naive approach performing slightly better. Since this instance includes only 19 students and the partitioning is attempting to create equal-sized blocks, the clustering algorithm may forcibly group dissimilar points, leading to "arbitrary merging" (Hamerly and Elkan, 2004). As a result, the naive approach performs comparably to the true partitioning method.

3.4.2 Finding bounds

This section describes how the partition of students can be used to find bounds for the full lexicographic problem described in Chapter 2.

Suppose that there are $k > 1$ objective functions $z_1, \ldots, z_k$ to be minimised lexicographically and that the partition B contains n blocks of students. For each block $b_i \in B$ where $i \in \{1, \ldots, n\}$, minimise z_1 and denote the optimal value as z_1^i. Then a

Table 3.4.2: Instance and the mean average distance between students, denoted as $\mathbb{E}(d)$, in each block for both partitioning approaches

	wbg-fal10	muni-fsps-spr17*	mary-fal18*	muni-fsps-spr17	pu-cs-fal07*	pu-cs-fal07	mary-fal18	pu-d5-spr17	pu-llr-spr17
$\mathbb{E}(d)$ naive partition	0.72	0.53	0.76	0.80	0.83	0.83	0.94	0.96	0.99
$\mathbb{E}(d)$ true partition	0.73	0.19	0.62	0.64	0.36	0.20	0.88	0.40	0.92

bound on z_1 is given by the following:

$$z_1 \geq \sum_{i=1}^{n} z_1^i. \tag{3.4.3}$$

Assume now that this bound is tight, meaning there exists a solution to the full problem such that Equation 3.4.3 is an equality. If we fix the values of z_1^i for each $i \in \{1, \ldots, n\}$ and minimise the values of z_j^i for some choice of $j \in \{2, \ldots, k\}$ then the following bound holds:

$$z_j \geq \sum_{i=1}^{n} z_j^i. \tag{3.4.4}$$

Notice that the choice of z_1 being the first to be optimised was arbitrary, implying that it is possible to apply this procedure by starting by bounding z_j with Equation 3.4.3 for any $j \in \{2, \ldots, k\}$.

The requirement of Equation 3.4.3 being tight for Equation 3.4.4 to hold is a strong assumption in general. This technique is useful in settings where the full problem is computationally intractable, and it is possible to prove or verify that Equation 3.4.3 is tight. For example, when students have no mandatory module requirements, a conflict-free solution trivially exists. In this scenario, tractable sub-problems can be constrained to maintain zero student conflicts while optimising for the maximum number of accommodated module requests. Equation 3.4.4 holds, meaning that the sum over the sub-problems of the number of accommodated module requests provides an upper bound for the whole problem.

We use this approach to find bounds for the instances and objectives outlined in Section 3.3 using partitions generated by following the steps outlined earlier in this section.

Finding bounds for solved instances

In Chapter 2, lexicographic orderings for objectives z_1, z_2 and z_3 were found using an exact approach for reduced instances. These solutions can be used to assess how the bound-finding approach described above performs because they are known to be optimal.

Table 3.4.3 compares the objective values with the derived bounds. The sense of the objective is given to highlight whether the bound is a lower bound or an upper bound. This table shows that many of the derived bounds are close to, if not exactly equivalent to, the optimal objective values for each lexicographic order. 63% of derived bound values match the corresponding optimal value. By defining a gap between bound and solution in the same way as Gurobi (Gurobi Optimization, LLC, 2025), that is

$$\text{gap} = \frac{|\text{Solution} - \text{Bound}|}{|\text{Solution}| + \epsilon}, \tag{3.4.5}$$

where ϵ is a small number, the average gap for exact/bound pairs that do not match is 0.35.

These results demonstrate that partitioning and clustering based on student module requests can produce subproblems that still capture the important relations between the objectives of interest. This leads to finding good objective bounds using sub-problems that are easier to solve than the original problem.

Figure 3.4.1 compares the computational time required by the Gurobi solver for the exact approach versus the bound-finding approach. The bound-finding procedure is substantially faster, demonstrating its value in strategic decision-making contexts where understanding objective trade-offs is more important than computing exact optimal solutions.

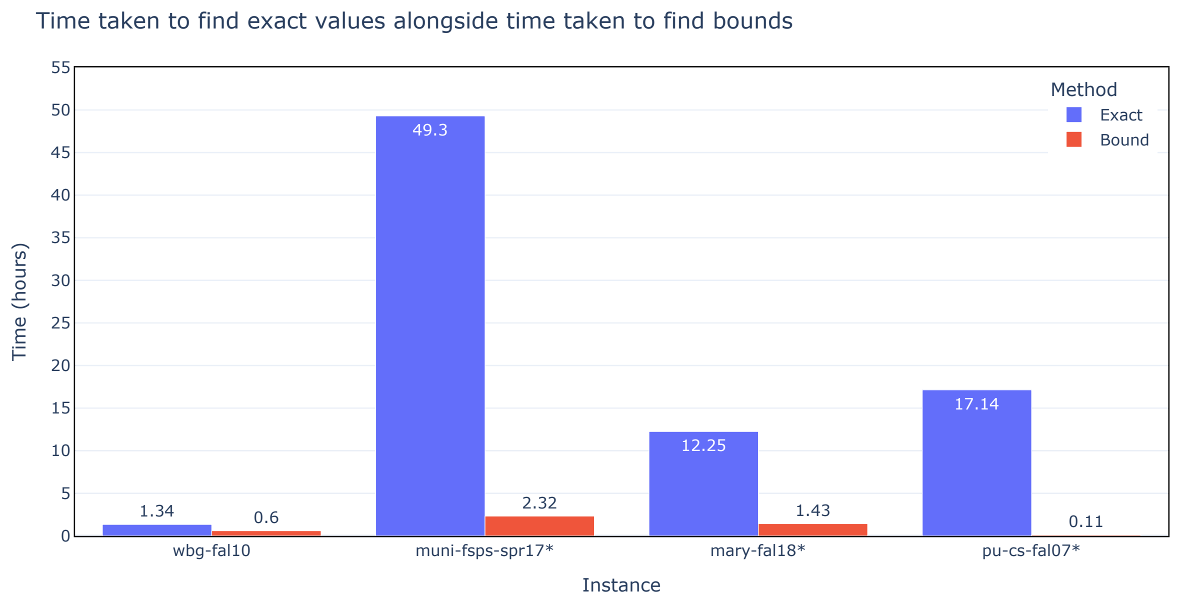


Figure 3.4.1: Plot comparing the time Gurobi spent solving models for the exact approach in Chapter 2 and the time Gurobi spent solving models for the bound finding procedure

Finding bounds for unsolved instances

For instances that cannot be efficiently solved to optimality using exact methods, obtaining bounds on the objective values is crucial to evaluate the quality of the solutions found by the matheuristic method. We have confirmed that the bound-finding procedure provides reliable estimates for any ordering of objectives, based on the exact results from Chapter 2. Consequently, we will compute bounds for the remaining instances listed in Table 3.3.1, using the partitions summarised in Table 3.4.1. This information will enable us to assess the performance of the matheuristic.

Table 3.4.3: Comparison of exact results from Chapter 2 and the bounds found using the method defined in Section 3.4.2

		max z_1		min z_2		min z_3	
	Ordering	Exact	Bound	Exact	Bound	Exact	Bound
wbg-fal10	z_1,z_2,z_3	97	97	43	19	0	0
	z_1,z_3,z_2	97	97	43	19	0	0
	z_2,z_1,z_3	71	80	0	0	0	0
	z_2,z_3,z_1	71	80	0	0	0	0
	z_3,z_1,z_2	97	97	43	0	0	0
	z_3,z_2,z_1	71	97	0	0	0	0
muni-fsps-spr17*	z_1,z_2,z_3	980	980	106	96	8	0
	z_1,z_3,z_2	980	980	114	96	0	0
	z_2,z_1,z_3	926	930	0	0	0	0
	z_2,z_3,z_1	926	930	0	0	0	0
	z_3,z_1,z_2	980	980	114	0	0	0
	z_3,z_2,z_1	926	980	0	0	0	0
mary-fal18*	z_1,z_2,z_3	1,597	1,597	109	91	36	34
	z_1,z_3,z_2	1,597	1,597	111	91	34	34
	z_2,z_1,z_3	1,498	1,517	0	0	26	0
	z_2,z_3,z_1	1,480	1,517	0	0	0	0
	z_3,z_1,z_2	1,571	1,571	98	0	0	0
	z_3,z_2,z_1	1,480	1,571	0	0	0	0
pu-cs-fal07*	z_1,z_2,z_3	1,226	1,226	11	6	0	0
	z_1,z_3,z_2	1,226	1,226	11	6	0	0
	z_2,z_1,z_3	1,220	1,224	0	0	0	0
	z_2,z_3,z_1	1,220	1,224	0	0	0	0
	z_3,z_1,z_2	1,226	1,226	11	0	0	0
	z_3,z_2,z_1	1,220	1,226	0	0	0	0

3.5 Multi-objective lexicographic optimisation

Multi-objective lexicographic optimisation involves solving single-objective optimisation problems sequentially. Each subsequent problem becomes progressively more challenging due to constraints that enforce objective functions to take values obtained by earlier problems.

3.5.1 Implied binary variables

As mentioned in Chapter 1, a common approach used in solvers for tackling binary programmes is an approach called branch-and-bound. Reducing the number of binary and integer variables from a model will significantly reduce the possible size of the branch-and-bound tree.

Whilst the model described in Chapter 2 mathematically represents the problem, it also contains significantly more binary variables than are needed. Some of the constraints in the model would force certain variables to take values in the set $\{0, 1\}$ even if they were defined to be continuous. These are known as "implied binary" variables. Some variables can be relaxed provided they are constrained to be in the set $[0, 1]$ or to be non-negative. Table 3.5.1 outlines the relaxed variable, a lower and upper bound, and the constraints or objective function reasons that ensure that it is binary.

Table 3.5.1: Variable to be relaxed, lower (LB) and upper (UB) bound, and justification

Variable	LB	UB	Constraint set
γ	$-\infty$	∞	Equations 2.4.45 and 2.4.46
β^{inp}	$-\infty$	∞	Equation 2.4.36
β^{onl}	$-\infty$	∞	Equation 2.4.37
τ	0	∞	Minimisation and Equations 2.4.34 and 2.4.35
h	0	∞	Minimisation and Equations 2.4.47 and 2.4.38
a	0	1	Equations 2.4.30
b	0	1	Equations 2.4.29
n	0	1	Equations 2.4.27

3.5.2 Single-objective matheuristic

This section outlines a method for solving a single-objective problem, which forms the foundational routine of the solution process. When we refer to "solving for an objective" in this chapter, we are implying the application of this single-objective fix-and-optimise routine.

Initial solutions

An initial solution is needed for this method. The absence of compulsory modules for students simplifies this step. If there are no required modules, then a blank solution automatically satisfies all hard constraints and can serve as an initial solution.

In practice, students typically have compulsory or required modules. One approach to find an initial solution in this case is to relax the constraints on these modules being compulsory and treat them as an objective. The single-objective matheuristic presented in this chapter could be used to minimise the number of constraint violations until a feasible solution is found.

Neighbourhoods

In each iteration, a subset of variables is selected to be fixed at their current solution values. The neighbourhood of a solution is defined as the set of solutions reachable by optimising only the unfixed variables while keeping the fixed variables constant. Since neighbourhoods are determined by the choice of which variables to fix, we use the term "neighbourhood" interchangeably to refer to both the set of unfixed variables and the corresponding set of reachable solutions.

High-level framework

The algorithm takes as input the problem instance and can be configured using several parameters. The first is the maximum number of iterations the algorithm will run for.

The second is the "lowest maximum" time an iteration is allowed to run, which is a lower bound on the soft iteration time limit. Without this bound, the algorithm could maintain a small neighbourhood size for hard-to-solve instances and not make progress (see Algorithm 6).

The next set of parameters governs the size of neighbourhoods. The first is the starting size of the neighbourhoods, the second is the maximum increase in neighbourhood size per iteration, and the final parameter is the maximum size of the neighbourhood. All these are expressed as percentages of the total number of students in the instance. For example, in an instance with 100 students, a neighbourhood size of 0.1 corresponds to a neighbourhood containing 10 students.

Additionally, there are parameters to handle stagnation in the solution search. The first parameter specifies the number of consecutive iterations that must result in the same objective value before the search is considered stagnant. The second defines how many previous solutions are retained in memory, and the third determines how many times the algorithm is allowed to return to previous solutions.

Mathematical notation

The following list outlines the notation that will be used for describing the method.

B^{prob}:	Probability of selecting students from a random $b \in B$.
I:	Maximum number of iterations.
S^{unfix}:	Set of students that are "unfixed".
S^{fixed}:	Set of students that are "fixed".
I^T:	Lowest maximum time an iteration can run for in seconds.
N^{start}:	Starting neighbourhood size.
N^{curr}:	Current neighbourhood size.

N^{+}:	Maximum increase in neighbourhood size.
$N^{\max}$:	Maximum neighbourhood size.
X^{curr}:	Number of iterations of getting the same objective value.
$X^{\max}$:	Number of iterations of getting the same objective value to claim search has stagnated.
X^{sol}:	Number of stored solutions.
R^{curr}:	Number of resets to previous solution.
$R^{\max}$:	Maximum number of resets to previous solution.

Speed benchmarking

To estimate the time required to solve a neighbourhood of a certain size, the algorithm begins with a benchmarking step. During this step, a percentage of students are randomly selected, and for each student, all lexicographic orderings of the objectives are solved. The time taken for each student is recorded, and the average time is then calculated. This average time is denoted as I^S and enables the iteration threshold time to be adjusted based on the size of the neighbourhoods.

Neighbourhood selection and solving

The choice of neighbourhood involves two decisions: the size of the neighbourhood and the fixed/unfixed elements within it. Both factors significantly impact the algorithm's performance.

Large neighbourhoods with many unfixed elements require more computational resources to store and solve (with the extreme case being no fixed elements), resulting in longer solution times. In contrast, small neighbourhoods are less computationally demanding and faster to solve, but are more constrained by the fixed elements, limiting potential solution improvement.

For the first iteration, the neighbourhood size is N^{start}. Algorithm 6 details how the neighbourhood sizing is adjusted for later iterations based on the previous iteration. Algorithm 7 details how students are selected. When a student partition is not available, let $B = \{S\}$.

In an iteration, once Algorithm 7 has partitioned S into S^{unfix} and S^{fixed}, the model described in Chapter 2 is built. Variables relating to students in S^{fixed} are fixed at the values given by the current solution. This ensures that the timetables for every $s \in S^{\text{fixed}}$ do not change. The model is then solved. In our case, we use Gurobi Optimizer version 12.0.1 in Python 3.12.7 running on an Intel Xeon Gold 6348 CPU with a 2.60GHz speed and 528GB of RAM.

Algorithm 6 Adjust neighbourhood size

```
1: Let $t$ be the time taken for the iteration to finish
2: function AdjustSize($t$, $N^{\text{curr}}$)
3:     $t^{\max} \Leftarrow \max(\lfloor N^{\text{curr}}|S|I^S \rfloor, I^T)$
4:     if $t \geq t^{\max}$ and $N^{\text{curr}} = N^{\text{start}}$ then
5:         Stop optimisation.
6:     else if $t \geq t^{\max}$ then
7:         $N^{\text{curr}} \Leftarrow \max(N^{\text{curr}} - 2N^{+}, 1/|S|)$
8:     else
9:         $N^{\text{curr}} \Leftarrow \min(N^{\text{curr}} + N^{+}, N^{\max})$
10:    end if
11:    return $N^{\text{start}}$, $N^{\text{curr}}$, $N^{+}$
12: end function
```

Shuffling and annealing

To move the solution away from a local optimum, the option of returning to a solution with a worse objective is allowed. When and how this is done depends on the parameters chosen. Algorithm 8 specifies how we iterate the value X^{curr} depending on the objective value between iterations, and Algorithm 9 specifies how we use this value to determine

Algorithm 7 Select neighbourhood

```
1: function SelectStudents($N^{\text{curr}}$, $S$, $B$, $B^{\text{prob}}$)
2:     Let $p$ be random number in $[0, 1]$
3:     if $p \leq B^{\text{prob}}$ then
4:         $S^{\text{unfix}} \Leftarrow s_1, \ldots, s_{N^{\text{curr}}} \in_R S$
5:         $S^{\text{fixed}} \Leftarrow S \setminus S^{\text{unfixed}}$
6:     else
7:         $S^{\text{unfix}} \Leftarrow \emptyset$
8:         while $|S^{\text{unfix}}| < N^{\text{curr}}$ do
9:             Choose $b \in B$
10:            if $|S^{\text{unfix}}| + |b| < N^{\text{curr}}$ then
11:                $S^{\text{unfix}} \Leftarrow S^{\text{unfix}} \cup b$
12:            else
13:                $N^{\text{part}} \Leftarrow N^{\text{curr}} - |S^{\text{unfix}}|$
14:                $S^{\text{part}} \Leftarrow s_1, \ldots, s_{N^{\text{part}}} \in_R b$
15:                $S^{\text{unfix}} \Leftarrow S^{\text{unfix}} \cup S^{\text{part}}$
16:            end if
17:        end while
18:        $S^{\text{fixed}} \Leftarrow S \setminus S^{\text{unfix}}$
19:    end if
20:    return $S^{\text{unfix}}$, $S^{\text{fixed}}$
21: end function
```

if we revert to a previous solution and iterate R^{curr} or continue with the current solution. In this thesis, line 5 of Algorithm 9 uses the word "choose" to mean a random selection of a single solution from the previous solutions with uniform probability. This could be replaced with a more sophisticated selection scheme.

Stopping criteria

There are four stopping criteria for this method:

1. **Iteration limit reached:** Once all the iterations are completed, the best solutions found up to that point are returned.

2. **Objective value equals bound:** The solution is optimal if the objective value meets the bound. Therefore, the algorithm stops, and the solution is returned.

Algorithm 8 Update X^{curr}

```
1: function UpdateXcurr($X^{\text{curr}}$)
2:     if on the first iteration then
3:         $X^{\text{curr}} = 0$
4:         return $X^{\text{curr}}$
5:     else
6:         Let $z_{-1}$ be the previous objective value.
7:         Let $z_0$ be the current objective value.
8:         if $z_{-1} \neq z_0$ then
9:             $X^{\text{curr}} = 0$
10:            return $X^{\text{curr}}$
11:        else
12:            $X^{\text{curr}} \Leftarrow X^{\text{curr}} + 1$
13:        end if
14:    end if
15: end function
```

Algorithm 9 Update R^{curr}

```
1: function UpdateRcurr($X^{\text{curr}}$, $X^{\max}$, $X^{\text{sol}}$, $R^{\text{curr}}$)
2:     if $X^{\text{curr}} < X^{\max}$ then
3:         return $X^{\text{curr}}$, $R^{\text{curr}}$
4:     else
5:         Choose one of $X^{\text{sol}}$ previous solutions.
6:         Set solution as the current solution.
7:         $R^{\text{curr}} \Leftarrow R^{\text{curr}} + 1$
8:         $X^{\text{curr}} \Leftarrow 0$
9:         return $X^{\text{curr}}$, $R^{\text{curr}}$
10:    end if
11: end function
```

3. $N^{\mathbf{curr}}$ **equal to** 1**:** If the neighbourhood size is 1.0 (100% of students are unfixed), then the entire problem has effectively been solved exactly, and therefore the solution found is optimal.

4. $R^{\mathbf{curr}}$ **greater than** $R^{\mathbf{max}}$**:** If the search has been stagnant more than times than allowed, then the algorithm is stopped and the best solutions found up to that point are returned.

3.5.3 Lexicographic matheuristic

The advantage of a fix-and-optimise routine is that constraints not violated by the current solution can be added dynamically at any iteration. This flexibility allows for constraining the current objective value and optimising another objective within the same framework. This is precisely how the single-objective matheuristic is extended to solve the problem lexicographically. Figure 3.5.1 illustrates how the matheuristic is used to solve a lexicographic ordering.

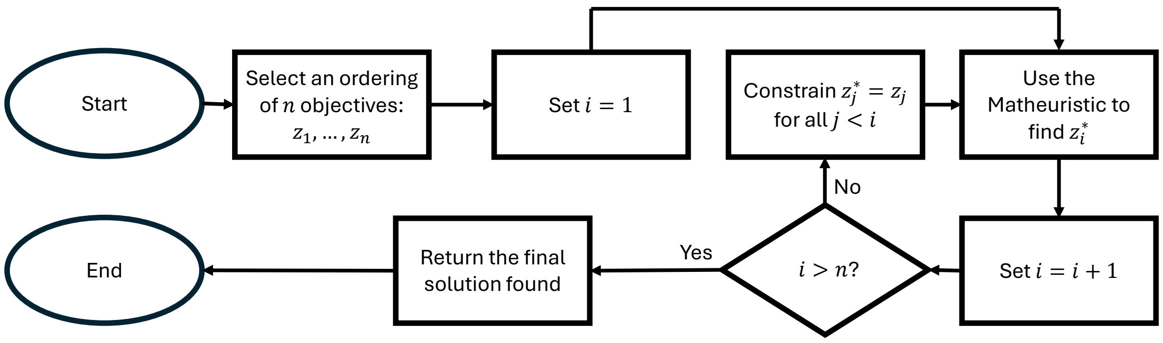

Start
Select an ordering of n objectives: $z_1, \dots, z_n$
Set $i = 1$
Constrain $z_j^* = z_j$ for all $j < i$
Use the Matheuristic to find z_i^*
No
End
Return the final solution found
Yes
$i > n$?
Set $i = i + 1$

Figure 3.5.1: Flowchart providing an overview of how the matheuristic can be used to solve a lexicographic ordering of objectives

Experiments with exact bounds

For the four instances used in Chapter 2, every ordering of objectives z_1, z_2 and z_3 was optimised lexicographically and exactly. Therefore, the results of the lexicographic

Table 3.5.2: Parameters for the matheuristic used in the experiments finding lexicographic solutions

	Matheuristic lexicographic algorithm parameter								
Instance	I	I^T	N^{start}	N^{+}	N^{max}	X^{max}	X^{sol}	R^{max}	B^{prob}
wbg-fal10	80	300	0.01	0.01	0.21	3	20	5	0.5
muni-fsps-spr17*	80	300	0.01	0.01	0.12	3	20	5	0.5
mary-fal18*	80	300	0.01	0.01	0.08	3	20	5	0.5
muni-fsps-spr17	20	300	0.01	0.01	1.0	3	20	5	0.5
pu-cs-fal07*	80	300	0.01	0.01	0.5	3	20	5	0.5
pu-cs-fal07	20	300	0.01	0.01	1.0	3	20	5	0.5
mary-fal18	20	300	0.01	0.01	1.0	3	20	5	0.5
pu-d5-spr17	20	300	0.01	0.01	1.0	3	20	5	0.5
pu-llr-spr17	20	300	0.01	0.01	1.0	3	20	5	0.5

matheuristic can be compared with the bounds found in the previous section and the exact results found in Chapter 2. The parameters for the matheuristic used in these experiments are specified in Table 3.5.2.

To prevent the matheuristic from expanding the neighbourhood to the entire problem, essentially solving it exactly, the neighbourhood size for the reduced variants is constrained. For *pu-cs-fal07**, *muni-fsps-spr17** and *mary-fal18**, the maximum neighbourhood size is set to be 0.5, 0.12 and 0.08, respectively. These values correspond to the ratio of students remaining after the instance reduction to the total number of students in the original instance. For example, Table 3.3.1 states that *mary-fal18** has 400 students and *mary-fal18* has 5,051. $400/5051 \approx 0.08$, so this is the maximum neighbourhood size for *mary-fal18**. For *wbg-fal10*, which only contains 19 students, the neighbourhood size is limited to at most the size of the largest block identified in the bound-finding procedure. This results in a maximum neighbourhood size of 0.21 (4 out of 19 students).

Table 3.5.3 presents the objective values obtained for each instance and objective ordering using the exact approach and the matheuristic method described in this chapter.

These results show that the matheuristic generally produces values of the same order of magnitude as the exact approach, except for the number of student conflicts in certain orderings for *muni-fsps-spr17** and *mary-fal18**. By using Equation 3.4.5 and ignoring infinite values, the average gap between the known optimal values and the matheuristic values for *wbg-fal10*, *muni-fsps-spr17**, *mary-fal18**, and *pu-cs-fal07** are 0.11, 0.49, 0.45 and 0.14, respectively. These results indicate that the matheuristic produces solutions of sufficient quality for practical use.

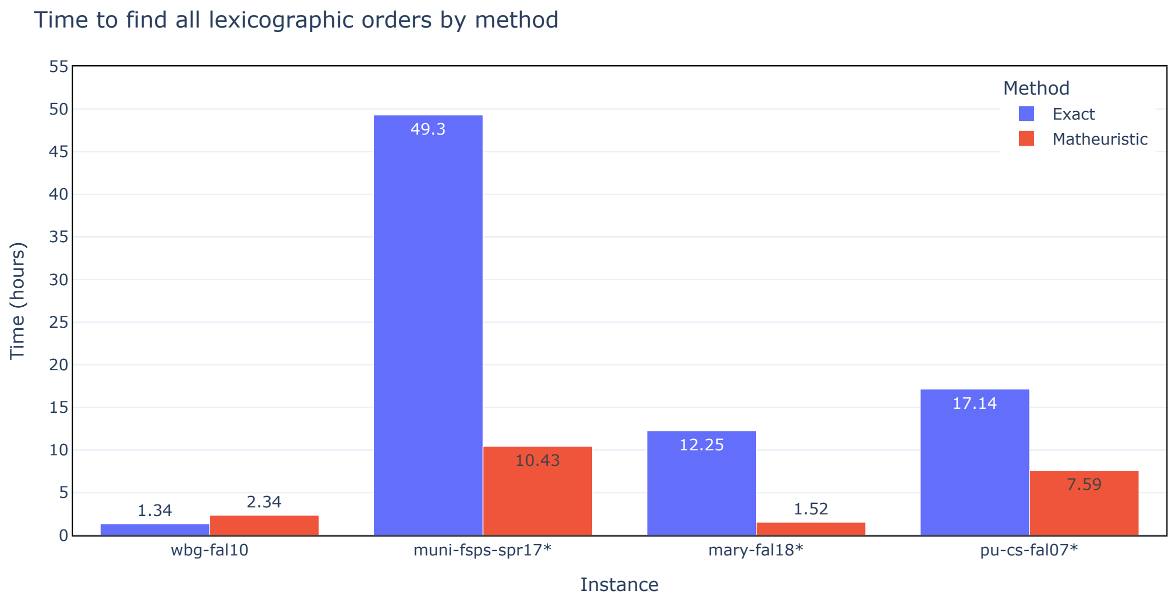


Figure 3.5.2: Plot comparing the time Gurobi spent solving models for the exact approach in Chapter 2 and the time Gurobi spent solving models for the matheuristic approach

Figure 3.5.2 compares the total time spent using the Gurobi solver for both methods across instances, demonstrating that the matheuristic is significantly faster than the exact approach. While the exact method guarantees optimality, the large time differences (e.g., for *muni-fsps-spr17**) suggest that even for generating rough initial solutions, the matheuristic is preferable due to its efficiency.

However, in the case of *wbg-fal10*, the matheuristic took longer than the exact approach. One reason is that this relatively small instance is not challenging for the

Table 3.5.3: Comparison of lexicographic matheuristic results and exact results from Chapter 2

		max z_1		min z_2		min z_3	
	Ordering	Exact	Matheuristic	Exact	Matheuristic	Exact	Matheuristic
wbg-fal10	z_1,z_2,z_3	97	61	43	17	0	0
	z_1,z_3,z_2	97	96	43	40	0	0
	z_2,z_1,z_3	71	58	0	0	0	0
	z_2,z_3,z_1	71	63	0	0	0	0
	z_3,z_1,z_2	97	85	43	24	0	0
	z_3,z_2,z_1	71	63	0	0	0	0
muni-fsps-spr17*	z_1,z_2,z_3	980	486	106	80	8	31
	z_1,z_3,z_2	980	471	114	142	0	90
	z_2,z_1,z_3	926	451	0	0	0	22
	z_2,z_3,z_1	926	356	0	0	0	0
	z_3,z_1,z_2	980	318	114	7	0	0
	z_3,z_2,z_1	926	287	0	0	0	0
mary-fal18*	z_1,z_2,z_3	1,597	883	109	5	36	2
	z_1,z_3,z_2	1,597	788	111	4	34	0
	z_2,z_1,z_3	1,498	950	0	0	26	40
	z_2,z_3,z_1	1,480	834	0	0	0	0
	z_3,z_1,z_2	1,571	794	98	0	0	0
	z_3,z_2,z_1	1,480	860	0	0	0	0
pu-cs-fal07*	z_1,z_2,z_3	1,226	1,185	11	0	0	0
	z_1,z_3,z_2	1,226	1,206	11	9	0	0
	z_2,z_1,z_3	1,220	1,210	0	0	0	1
	z_2,z_3,z_1	1,220	1,189	0	0	0	0
	z_3,z_1,z_2	1,226	1,223	11	23	0	0
	z_3,z_2,z_1	1,220	1,197	0	0	0	0

Gurobi solver. Another reason is that the search for a better solution would often become stagnant. Consequently, the algorithm would return to a previous solution and resume from there unless the number of resets had been reached (Criteria 4). Furthermore, Table 3.5.2 specifies that $N^{\text{curr}} \leq N^{\text{max}} = 0.21$, preventing the algorithm from meeting stopping criteria 3 and limiting what solutions can be reached within an iteration.

Objective value trace plots of the first ordering of objectives are given in Figure 3.5.3. The sharp spikes in the plots indicate where the matheuristic has returned to a previous solution. This figure also highlights two key trends: (i) the searches typically get stuck around the same place before resetting, and (ii) improving the objective value becomes increasingly difficult as the algorithm progresses down the lexicographic ordering (especially for *muni-fsps-spr17**).

A potentially confusing feature of the plots in Figure 3.5.3 is that in some of the plots for the objective functions z_2 and z_3, the matheuristic appears to beat the optimal value. The key observation to make is that z_1 never reaches the optimal value. Constraining z_1 to be at least as good as what the matheuristic found is not as restrictive as constraining z_1 to be optimal. This allows later objectives to perform better than optimal. For plots in the middle and on the right of Figure 3.5.3, the optimality line indicates the "optimal objective value assuming the previous solves were optimal".

Experiments with estimated bounds

For the five instances identified in Table 3.3.1 not analysed in Chapter 2, we unfortunately lack exact objective values for any ordering of the objectives z_1, z_2 and z_3. Due to the prohibitive computational cost of obtaining these values, we compare the results of the lexicographic matheuristic method with the bounds determined by the bound-finding procedure described in Section 3.4.

Unlike the previous experiments, the neighbourhood size in this experiment can grow to the full size of the instance. However, the neighbourhood size is still constrained by

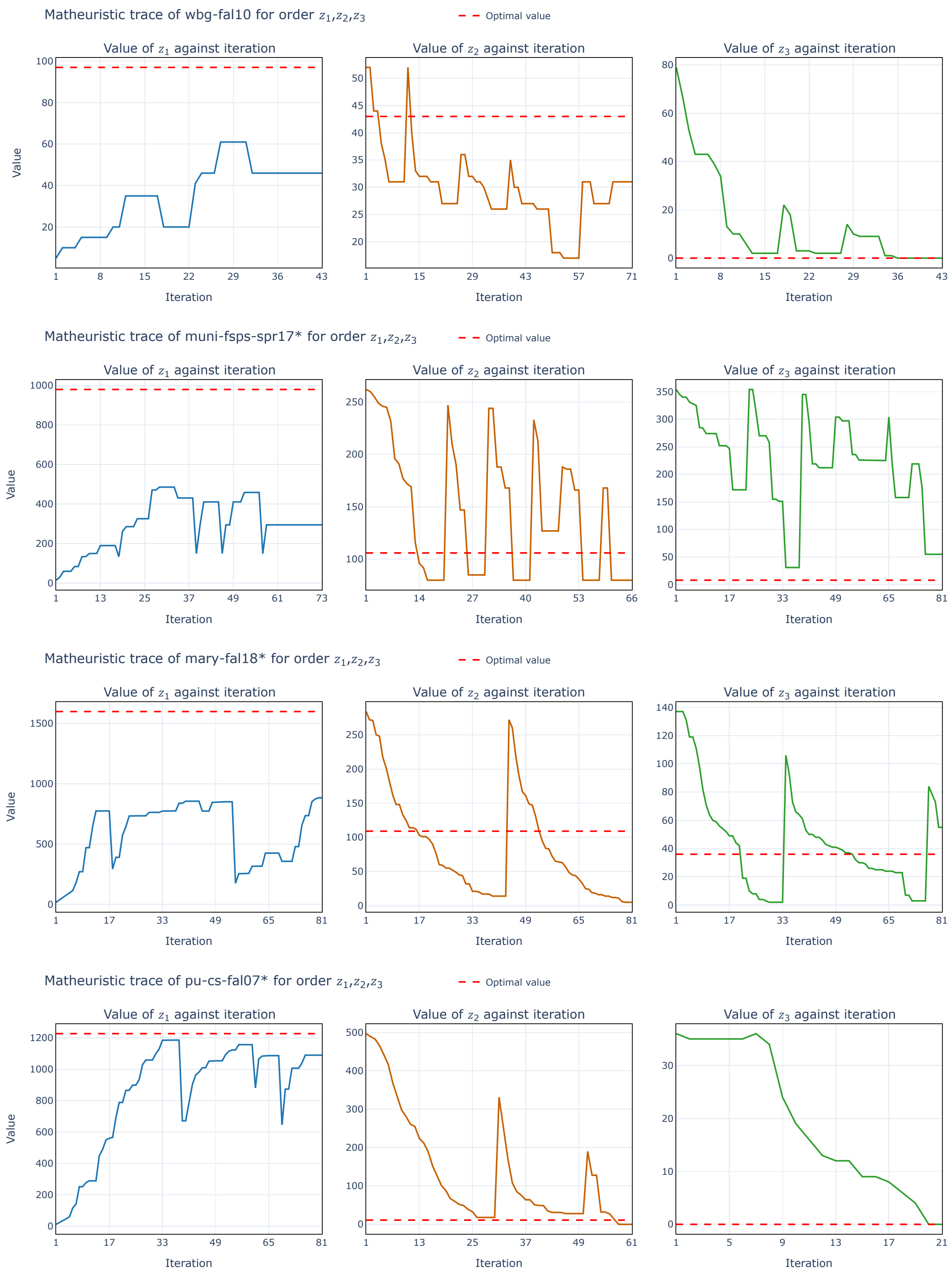


Figure 3.5.3: Trace plots of objective value against iteration for the matheuristic results of the first ordering for each instance in Table 3.5.3

the threshold time. The parameters used for testing the lexicographic matheuristic are outlined in Table 3.5.2.

Table 3.5.4 presents the objective values obtained for each instance using the matheuristic approach described in this chapter, alongside the bounds identified through the procedure also detailed in this work. It is evident from this table that these instances are larger than the instances that could be solved to optimality exactly. The smallest upper bound on z_1 across these instances is 50% greater than the largest upper bound on the reduced instances.

The results in Table 3.5.4 also demonstrate that these instances were more challenging to solve. The average gap between the derived bounds and the matheuristic values for *muni-fsps-spr17*, *pu-cs-fal07*, *mary-fal18*, *pu-d5-spr17*, and *pu-llr-spr17* are 0.19, 1.08, 0.25, 1.51 and 12.37, respectively. However, it is important to note that these instances were run for fewer iterations.

3.6 Frontier approximation and objective trade-off

A set of lexicographic solutions can help decision-makers identify conflicting objectives and understand the trade-offs involved in selecting a final solution. However, relying solely on extreme or lexicographic solutions may not clearly illustrate what a balanced trade-off solution would look like.

To address this, we generate an approximation of the Pareto frontier, which provides a more comprehensive view of how objectives interact. In this section, we apply a variant of the matheuristic described in Section 3.5 to construct this approximation.

Before describing the frontier search algorithm, we introduce the concept of ideal and nadir points. These two points are defined formally in Ehrgott (2005); however, we define them using lexicographic optimisation concepts.

The ideal point is the vector of values corresponding to the values that each objective

Table 3.5.4: Comparison of bound values with lexicographic matheuristic results

		$\max z_1$		$\min z_2$		$\min z_3$	
	Ordering	Bound	Matheuristic	Bound	Matheuristic	Bound	Matheuristic
muni-fsps-spr17	z_1,z_2,z_3	6,715	4,389	919	380	0	224
	z_1,z_3,z_2	6,715	4,988	919	756	0	507
	z_2,z_1,z_3	6,087	3,788	0	0	0	61
	z_2,z_3,z_1	6,087	4,506	0	0	0	0
	z_3,z_1,z_2	6,715	4,690	0	551	0	0
	z_3,z_2,z_1	6,715	3,949	0	0	0	0
pu-cs-fal07	z_1,z_2,z_3	2,393	1,908	32	287	0	1
	z_1,z_3,z_2	2,393	1,715	32	195	0	3
	z_2,z_1,z_3	2,368	1,397	0	0	0	7
	z_2,z_3,z_1	2,368	1,341	0	0	0	0
	z_3,z_1,z_2	2,393	1,636	0	116	0	0
	z_3,z_2,z_1	2,393	1,473	0	0	0	0
mary-fal18	z_1,z_2,z_3	21,017	14,618	1,033	2,046	0	220
	z_1,z_3,z_2	21,017	13,929	1,033	1,331	0	372
	z_2,z_1,z_3	20,039	12,660	0	0	0	171
	z_2,z_3,z_1	20,039	12,092	0	0	0	0
	z_3,z_1,z_2	21,017	13,663	0	723	0	0
	z_3,z_2,z_1	21,017	12,481	0	0	0	0
pu-d5-spr17	z_1,z_2,z_3	19,614	16,130	3,602	4,064	12	155
	z_1,z_3,z_2	19,614	16,289	3,602	4,070	12	138
	z_2,z_1,z_3	16,430	12,556	0	0	0	79
	z_2,z_3,z_1	16,430	12,306	0	0	0	0
	z_3,z_1,z_2	19,603	14,835	0	1,958	0	0
	z_3,z_2,z_1	19,603	11,944	0	0	0	0
pu-llr-spr17	z_1,z_2,z_3	81,549	60,265	1,215	19,232	78	6,598
	z_1,z_3,z_2	81,549	60,566	1,215	20,037	78	6,499
	z_2,z_1,z_3	80,384	56,128	0	0	0	5,686
	z_2,z_3,z_1	80,384	55,254	0	0	0	0
	z_3,z_1,z_2	81,472	57,733	0	18,934	0	0
	z_3,z_2,z_1	81,472	56,960	0	0	0	0

function takes when optimised first in any ordering of objectives. The nadir point is the vector of values corresponding to the "worst" value that each objective function takes across all orderings where that objective function is last in the ordering. This point can be thought of as the "best-worst-case" point.

For example, using values from Table 2.6.2, it can be seen that the instance *wbg-fal10* has the ideal point $(z_1, z_2, z_3) = (97, 0, 0)$ and the nadir point $(z_1, z_2, z_3) = (71, 43, 0)$.

3.6.1 Frontier search algorithm

This approach differs from the matheuristic in Section 3.5 in two key ways. First, unlike large neighbourhood search strategies, where larger neighbourhoods result in significant objective value changes, this algorithm intentionally keeps neighbourhood sizes small to limit shifts in objective values. Second, neighbourhood selection is no longer the only decision made at each iteration. Instead, each iteration requires answering the following: (i) What objective should be optimised? and (ii) Which objectives should be constrained? Figure 3.6.1 provides an overview of the method.

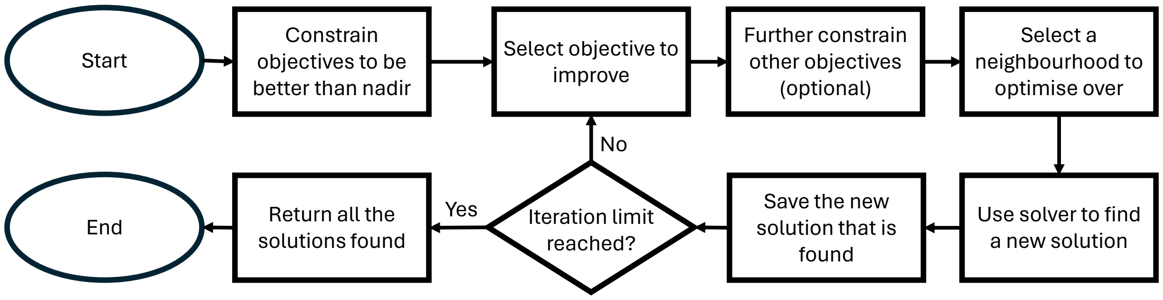

Start
Constrain objectives to be better than nadir
Select objective to improve
Further constrain other objectives (optional)
Select a neighbourhood to optimise over
Use solver to find a new solution
Save the new solution that is found
Iteration limit reached?
No
Yes
Return all the solutions found
End

Figure 3.6.1: Flowchart providing an overview of the frontier search procedure

Mathematical notation

I: Maximum number of iterations.

D: Set of starting solutions.

R: Repeats for each starting solution.

Z: Set of objective functions.

i_z: The ideal value for objective function $z \in Z$.

n_z: The nadir value for objective function $z \in Z$.

Starting solutions

Any feasible solution set can be used for this procedure. In this chapter, we use lexicographic order solutions (exact where possible) as starting points. Since the problem defined in Chapter 2 involves three objectives, the initial set contains at most six points. These solutions lie at the extreme ends of the Pareto front, increasing the likelihood that the algorithm effectively explores the entire frontier.

Neighbourhood selection and size

Similar to the matheuristic in Section 3.5, the method takes as input N^{start}, N^{+} and $N^{\max}$. The main difference is that both N^{+} and $N^{\max}$ are more effective if they are kept small, especially the value for $N^{\max}$. The neighbourhood size adjustment is the same as in Algorithm 6 and the neighbourhood selection is the same as in Algorithm 7.

Objective selection

An objective to optimise is selected in each iteration. This is done randomly, where the probability of choosing an objective function for optimisation is denoted as p_z for each $z \in Z$.

If certain objectives are of particular interest to the decision-maker, the probability values can be adjusted to prioritise their exploration. A more advanced objective selection scheme could further enhance this process by dynamically adapting probabilities based on search status, ensuring a more efficient and targeted exploration of the Pareto frontier. However, this is beyond the scope of this thesis.

Constraining objectives

In each iteration, only one objective is optimised, while the remaining objectives are allowed to change freely. Although these changes are expected to be minimal, they will typically move in a worsening direction.

To prevent excessive deterioration, each objective is constrained to be no worse than the objective value given by the approximate nadir point, as determined by the lexicographic solutions. In particular:

- If maximising $z \in Z$, then ensure $z \geq n_z$,
- If minimising $z \in Z$, then ensure $z \leq n_z$.

By leveraging the fix-and-optimise approach, additional constraints can be introduced dynamically to better control the search process. For instance, objectives not currently being optimised could be restricted within a decision-maker-defined range. This chapter employs a straightforward constraint strategy, with more advanced schemes not being covered in this work.

Stopping criteria and post-processing

The frontier search stops when the algorithm has completed $I \times R$ iterations for each starting point $d \in D$. The final step that the algorithm performs is a post-processing step that removes any dominated solutions from the overall set of solutions found.

3.6.2 Frontier search experiments

Evaluating the quality of a frontier search algorithm requires two key components:

1. An accurate approximation of the true Pareto frontier.
2. Metrics to compare the search results against this frontier.

Comparison frontier

Let $I^Z = (i_{z_1}, \ldots, i_{z_k})$ and $N^Z = (n_{z_1}, \ldots, n_{z_k})$ denote the ideal point and nadir point respectively. The results in Table 3.5.3 can be used to derive both of these points.

Algorithm 10, which we call the exact frontier algorithm (EFA), takes the ideal point, the nadir point and the instance as input and returns the Pareto frontier. The algorithm also takes an optional parameter, $res \in [0, 1]$, indicating the "resolution" of the frontier. $res = 1$ being the full frontier and $res < 1$ being an approximation of the frontier.

Algorithm 10 Exact frontier algorithm (EFA)

```
1:  function EFA($I^Z$, $N^Z$, $res$)
2:      for $j \in \{1, \ldots, k\}$ do
3:          $n \Leftarrow |i_{z_j} - n_{z_j}|$
4:          $n \Leftarrow \max(\lfloor res \times n \rfloor, 2)$
5:          $P_j \Leftarrow$ Set of $n$ equally spaced values between $i_{z_j}$ and $n_{z_j}$.
6:      end for
7:      Generate grid $P$ using all $P_j$ sets.
8:      $F \Leftarrow \emptyset$
9:      for $p \in P$ do:
10:         if $p$ feasible then
11:             $F \Leftarrow F \cup \{p\}$
12:         end if
13:     end for
14:     Remove dominated points in $F$
15:     return $F$
16: end function
```

Comparison metrics

Li and Yao (2019) present a selection of metrics for comparing multi-objective solution sets and Pareto frontiers, identifying four key aspects of frontier quality: convergence, cardinality, spread, and uniformity.

Given that our experiments involve only three dimensions and a known nadir point, the hyper-volume (HV) metric (Zitzler and Thiele, 1998) is a suitable choice. This metric simultaneously evaluates convergence, cardinality, and spread. We compute HV

Table 3.6.1: Parameters for the matheuristic used in the experiments approximating the Pareto frontier. Also included is the *res* parameter for EFA (Algorithm 10)

	Matheuristic frontier search algorithm parameter							EFA
Instance	I	I^T	R	N^{start}	N^+	$N^{\max}$	B^{prob}	res
wbg-fal10	40	300	3	0.01	0.01	0.05	0.5	0.33
muni-fsps-spr17*	40	300	3	0.01	0.01	0.05	0.5	0.05
mary-fal18*	40	300	3	0.01	0.01	0.05	0.5	0.10
pu-cs-fal07*	40	300	3	0.01	0.01	0.05	0.5	0.66

using the implementation in the Python package *pymoo* (Blank and Deb, 2020), with the nadir point as the reference point.

Additionally, we employ the spacing (SP) metric proposed by Schott (1995), which measures the variation in distances between solutions within a set. We implemented SP ourselves using the equations provided by Li and Yao (2019).

Experiment setup

To demonstrate the frontier search algorithm, the four instances from Chapter 2 with known exact lexicographic solutions will be used. The starting set D that will be used will be the set of those solutions. The nadir point is known as we can take the worst objective value seen across the solutions in D for each objective function because the points in D are proven extreme points. Finally, we set $p_z = 1/|Z|$ for all $z \in Z$. Other experiment parameters are given in Table 3.6.1.

Results

After running the frontier search algorithm and the exact frontier algorithm for the parameters in Table 3.6.1, we have generated new solutions for each instance. Figure 3.6.2 shows the objective values of the solutions from the search algorithm alongside the objective values of the lexicographic solutions.

For *wbg-fal10* and *pu-cs-fal07**, we can see that one objective is the same for all solutions. Therefore, we can reduce the frontier to two dimensions. Figure 3.6.3 compares the search results with the frontier found using Algorithm 10 for these two-dimensional frontiers. For *muni-fsps-spr17** and *mary-fal18** all three objectives change. Figure 3.6.4 compares the search results with the frontier found using Algorithm 10 for these three-dimensional frontiers.

Table 3.6.2 records the cardinality, the hyper-volume score and the spacing metric of each frontier. The final entries are ratios of the HV scores and the SP scores.

By looking at the ratio of the HV scores, we can see that the search algorithm is around two-thirds the score of the frontier generated using Algorithm 10. This suggests that the frontiers generated by the search are not as effective as those found by the EFA. Note that for *muni-fsps-spr17**, Algorithm 10 did not produce any new solutions, and so the hyper-volume for the approximation of the exact frontier is zero, and the hyper-volume ratio is infinite.

By looking at the ratio of the SP scores, we can also see that the points in the frontiers generated by the matheuristic search are less uniform. It does not appear that either method produces significantly more non-dominated points than the other.

As with the lexicographic matheuristic in Section 3.5, the frontier search algorithm is much faster computationally than the exact frontier algorithm. The EFA produces a better frontier but takes days to complete even an approximation of the frontier, whereas FSA takes only hours.

3.6.3 Trade-offs and decision-making

The plots in Figure 3.6.2 identify regions where objectives conflict, therefore providing an understanding of the trade-offs involved between different objectives. For *muni-fsps-spr17** and *mary-fal18**, it can be seen that minimising z_2 (Deviation from mode preferences) is in conflict with minimising z_3 (Number of student conflicts). For *wbg-fal10*

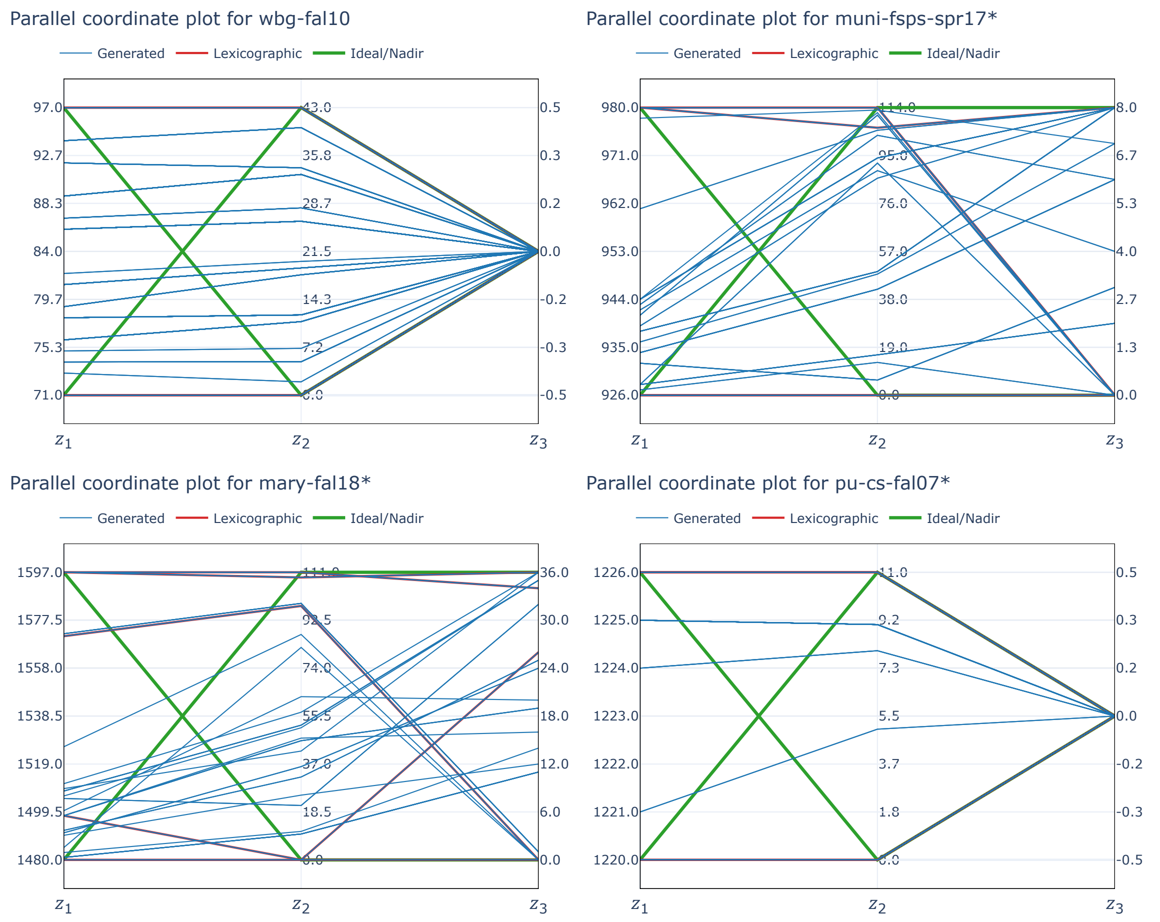


Figure 3.6.2: Parallel coordinate plots containing lines representing the generated solutions, lexicographic solutions and the ideal/nadir solution values

and *pu-cs-fal07* maximising z_1 conflicts with minimising z_2.

The Pareto front approximations in Figures 3.6.3 and 3.6.4 provide similar insights into trade-offs in objectives. The benefit of viewing solutions in this way is that it shows solution diversity and also highlights which objectives are more sensitive to changes.

By using both figure types to examine the solutions generated by the matheuristic method and their associated objective value trade-offs, decision-makers are empowered to make more informed choices, choosing the timetabling solution that aligns with both operational needs and stakeholder preferences at a strategic level.

Table 3.6.2: Summary metrics for the frontier found by the search algorithm (F^S) and the EFA (F^{EFA})

	F^S			F^{EFA}			Ratio (F^S/F^{EFA})	
Instance	$\|F^S\|$	HV	SP	$\|F^{EFA}\|$	HV	SP	HV	SP
wbg-fal10	15	482	1.19	8	553	2.46	0.87	2.46
muni-fsps-spr17*	18	5,191	5.11	3	0	87.76	∞	87.76
mary-fal18*	21	92,306	14.15	22	206,205	6.41	0.45	6.41
pu-cs-fal07*	5	17	2.05	4	22	0.00	0.77	0.00

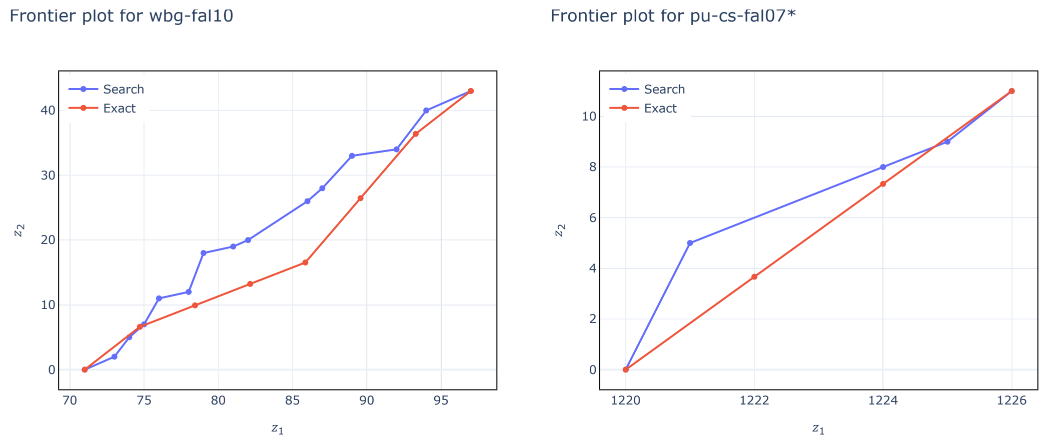


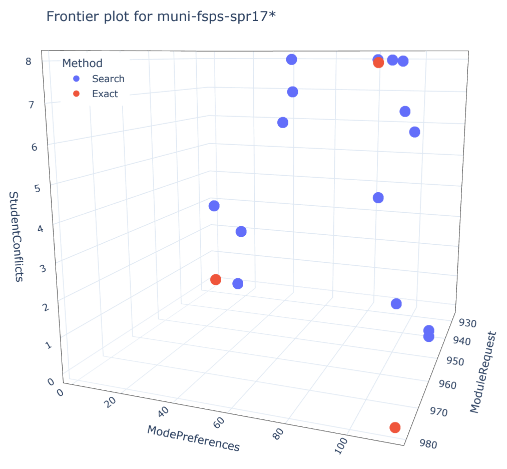
Figure 3.6.3: 2D frontier plots for the instances *wbg-fal10* and *pu-cs-fal07**

3.7 Conclusion

This chapter reviews the field of university course timetabling and identifies a gap in the methods used for solving a multi-objective UCTTP, where purely heuristic approaches dominate. To address this, we extend a popular matheuristic traditionally used for single-objective problems to enable it to approximate the Pareto frontier for a multi-objective setting.

To achieve this, we first introduce a novel method for partitioning students to establish objective bounds. We then define a single-objective fix-and-optimise matheuristic and use it to solve lexicographic orderings of objectives. After demonstrating its effectiveness in handling large-scale problems, we adapt the method to approximate a Pareto frontier.

The key takeaways from this chapter are:



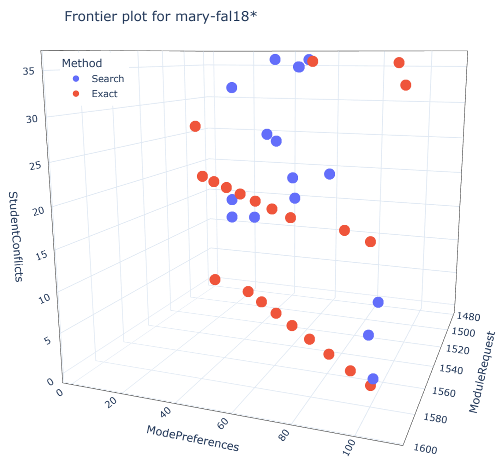


Figure 3.6.4: 3D frontier plots for the instances *muni-fsps-spr17** and *mary-fal18**

- Partitioning students is useful for bounding objective values and can also inform neighbourhood structures for matheuristic algorithms.
- The fix-and-optimise matheuristic can handle large UCTTP instances and, when implemented correctly, is capable of being extended for multi-objective optimisation.

Finally, this work raises important research questions for further exploration:

- What is the best way to allocate computational resources when solving lexicographic orderings of objectives?
- How can the method in this chapter be further refined to improve the Pareto frontier search?
- How can a dynamic constraint-handling scheme or objective-selection scheme enhance search efficiency and solution quality?

By addressing these questions, future research can enhance the applicability of matheuristics in multi-objective university timetabling, leading to more adaptable and practical decision-support tools.

Chapter 4

Evaluating University Timetabling Policies Using Real-World Institutional Data

4.1 Introduction

The higher education (HE) sector in the United Kingdom (UK) is a sector that has a revenue in the billions of pounds. Data for 2023/24 shows that the total income of the sector was £52 billion (Perrott, 2025b). Tuition fees and education contracts make up over 52% of the total income, with Figure 4.1.1 illustrating how this income source dominates other sources of income such as research grants and investments.

Therefore, it makes sense for universities to attract as many students as they can to maximise this income. There are many ways in which a university can attract new students. When students were surveyed, one of the top reasons for selecting a university was if that university offered high-quality teaching (Bhardwa, 2017).

The definition of "high-quality" teaching is subjective; however, studies have identified that factors such as the quality of instruction, classroom climate and classroom management are significant factors in improving student outcomes (Coe et al., 2014). In the context of universities, this means hiring knowledgeable staff and providing state-of-the-art teaching resources. The drawback of this approach is the finances involved.

Despite the seemingly increasing income of the higher education sector in the UK, Figure 4.1.2 shows that the total expenditures of higher education providers in the UK nearly match and occasionally exceed the total income. Data for 2023/24 shows that £18 billion went towards staff wages and £20 billion was spent on operating expenses (Perrott, 2025a). This strongly motivates the need for careful staff and resource management.

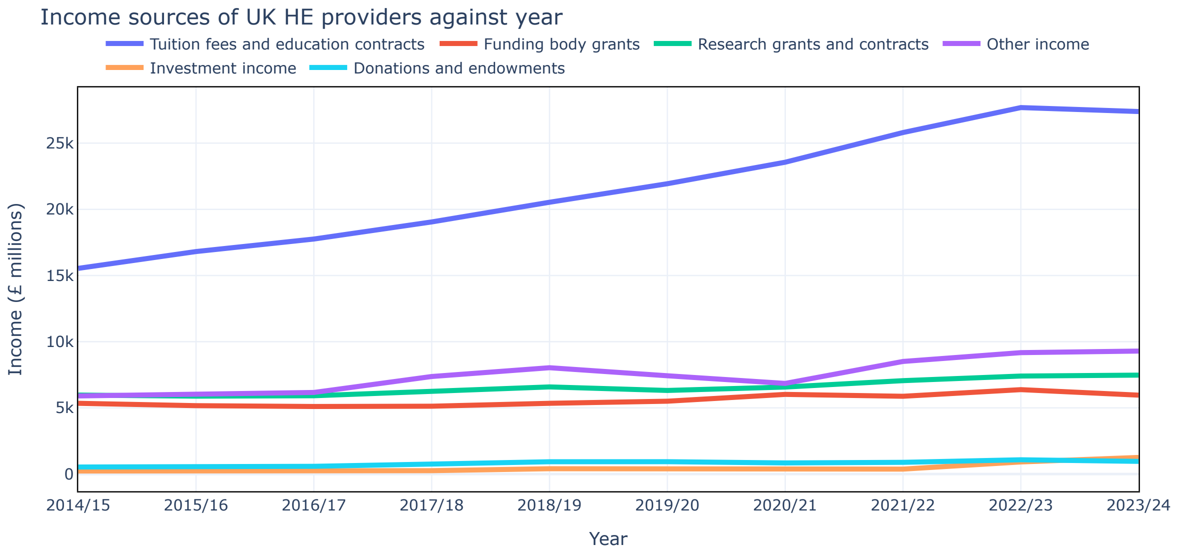


Figure 4.1.1: The income sources for higher education providers in the United Kingdom (Perrott, 2025b)

One of the most central operational elements at any university is the teaching timetable. This timetable dictates what staff, students and resources are doing at any given time. The construction of this timetable is restricted by policies at the university. Consequently, modifying any university policy could have either a positive or detrimental impact on operational costs.

Arrival at this conclusion motivates the following research questions:

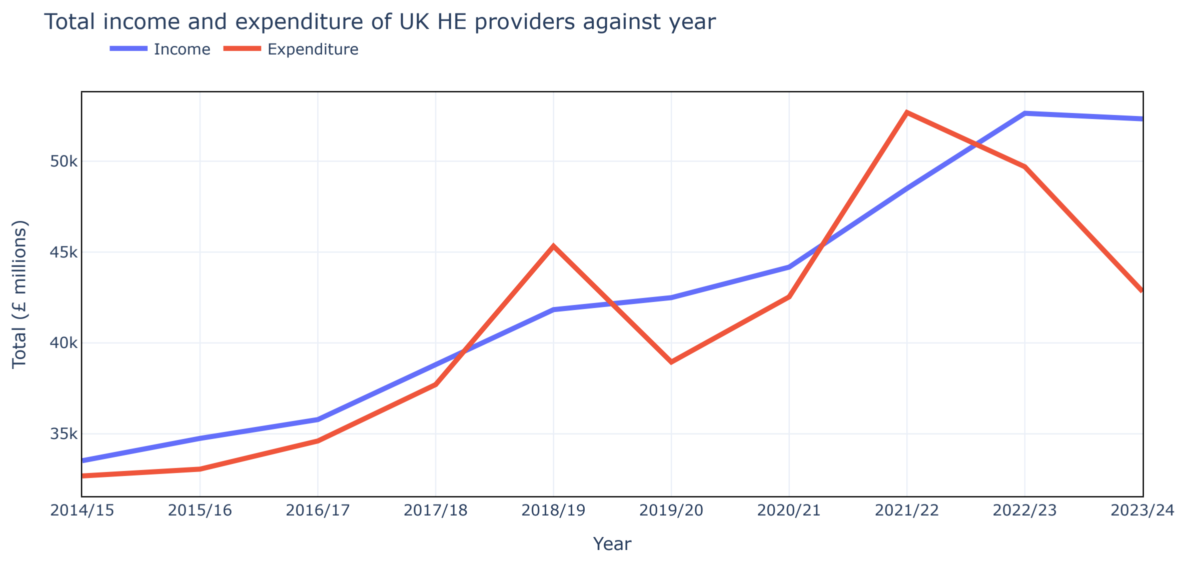


Figure 4.1.2: The total income and total expenditure for higher education providers in the United Kingdom (Perrott, 2025b,a)

- Can existing timetables be used to identify specific policies problematic to university operation?
- Can changes in particular policies be modelled so that these changes can be incorporated into a timetabling model?
- Can the resulting timetables produced by this timetabling model be used to assess the overall impact of various policy changes?

Attempting to address these questions, the objectives of this chapter are the following: (i) reshape real-world university timetabling data into a format known within the literature, forming a novel university timetabling benchmark instance and approximation of the university, (ii) use this approximation and domain knowledge from the university timetabling department to identify policies causing operational bottlenecks, and (iii) model and test changes to these policies, using change in quality of the resulting timetables as a proxy for measuring overall impact.

The rest of this chapter is structured as follows. In Section 4.2, a review of existing work in the context of university timetabling is provided. In Section 4.3, we discuss the real-world data and how this has been transformed into a standard format. Section 4.4 involves pairing up the data with the timetabling model presented in Chapter 2 and using the methods in Chapter 3 to verify that it produces timetables that match the real-world ones. Section 4.5 takes several university policies in turn; first describing them and then modelling them to be compatible with the model from Chapter 2. These policies, or changes to them, are then assessed based on the resulting timetables. Finally, Section 4.6 summarises the work done in this chapter.

4.2 Related work

A paper that provides an overview of areas in education where operational research has been applied is Johnes (2015). In particular, the paper is split into parts that separate planning, efficiency and scheduling to review methods for each part. In their conclusion, they emphasise the need for researchers to develop solutions that enhance educational efficiency, particularly given the rapid evolution in educational delivery methods.

For a review focused on methods for solving some specification of the university course timetabling problem (UCTTP), see Chen et al. (2021). For a review of educational timetabling problems and available benchmarking datasets, see Ceschia et al. (2023). For the remainder of this review, we focus on individual cases where operational research methods have been used to model or assess a particular policy or to investigate aspects of the timetabling procedure.

The first “policy” we identify is how universities would schedule activities in the past. Typically, timetabling was done manually, and timetabling practitioners would use the previous schedule as a starting point, leading to inefficiencies. Both Daskalaki et al. (2004) and Carter (2001) present methods to challenge this concept and promote automatic timetabling that starts from scratch.

Schimmelpfeng and Helber (2007) use an integer programming timetabling model to assess if a feasible schedule exists when particular rooms are unavailable. Their case study, focusing on an individual school within a university, helped identify the teaching spaces that the department could relinquish, albeit with some impact on the quality of the timetable. This was modelled by changing the input data.

Dammak et al. (2008) present an approach that implicitly improves various soft constraints relating to “compactness”, a term typically used to describe the quantity of free time between activities and how this is arranged in the schedule. Comparisons with timetables created by hand are made, ruling in favour of the heuristic approach used.

Mühlenthaler and Wanka (2016) modify the objective function of the timetabling problem described in Di Gaspero et al. (2007) in various ways to incorporate “fairness”. They then utilise this model to investigate the trade-off between timetable efficiency and fairness.

Vermuyten et al. (2016) aim to manage congestion in corridors at the Faculty of Economics and Business at KU Leuven Campus. They claim that the current timetables are partially the cause of this problem. Therefore, they incorporate a model of the student flow into a mixed integer program (MIP) for timetabling. They found that the timetables generated from doing this significantly reduced congestion.

The model presented in Gonzalez et al. (2018) allowed the United States Air Force Academy to switch from an “alternating-day approach” to a “repeated-week pattern”. This model was motivated by changing the timetabling system and the cost savings from changing the format. They also found other benefits, such as being able to combine multiple classes into a single class, leading to further efficiency gains.

Lindahl et al. (2017) focus on combining MIP models to be able to trade off between different objectives. They present three different models aiming to achieve a certain objective that utilise common variables. Pair-wise combinations of these models are used to show that the objectives of each model impact each other.

Barnhart et al. (2022) developed an MIP to investigate the impact on timetabling if the Massachusetts Institute of Technology moved from a two-semester calendar to a three-semester calendar. They then presented the findings of this to the timetabling department. The model they used specifically focuses on best utilising available resources, all with reduced capacity due to the 2020 COVID pandemic restrictions.

Gogos et al. (2022) focus on a more operational problem relating to university course timetabling. Much like Mühlenthaler and Wanka (2016), one of the aspects of this work is to find a fair solution, except they use both a combination of objective function changes and constraint additions to achieve this.

From this review of the literature, we can see that there are a few key ways in which timetabling practitioners can assess policy and other changes to the timetabling procedure.

- Create a new model to replace outdated models.
- Sensitivity analysis by modifying input parameters.
- Modifying model aspects such as the objective function or constraints.
- Combine multiple models into a single model.

4.2.1 Contributions

A significant gap in the literature is the fact that, in general, authors tend to choose a single approach to tackle strategic and tactical problems in university course timetabling. Another gap that exists is a lack of direct comparison between approaches and analysis on which are appropriate in what contexts. A final gap is that there is a lack of papers making their case study data available. As a minimum, authors could transform some or all of their data into the format of a benchmark instance when the raw data is sensitive. With these gaps in mind, the key contributions of this chapter are:

- A new benchmark instance for a large campus-based university in the United Kingdom and a thorough description of its creation.

- A demonstration of the ways to analyse such an instance to identify timetabling bottlenecks caused by university policy or infrastructure.
- A case study using the new instance where we methodically select and combine techniques seen in the literature to assess the impact of changing policies, concluding when to use each one.

4.3 Data processing

The data provided for this work was the academic timetable for every student enrolled during 2023/24 at Lancaster University (LU), a university based in the UK. The data is accessed through the program Scientia Syllabus Plus (SS+), which is the original software that produced the timetables.

This section outlines how the data was extracted and transformed into the XML format seen in the International Timetabling Competition 2019 (ITC-2019) (Müller et al., 2024). SS+ allows for the export of a lot of the data required for this process; however, some key elements needed to construct an instance file in the same format as the ITC-2019 were not obvious at first.

4.3.1 Raw data

The raw data was exported into three separate spreadsheet files. The first spreadsheet contains location information, the second contains student information, and the third contains activity information. Table 4.3.1 outlines the columns present in each spreadsheet with a brief description.

Given a row of data, if a column could take multiple values, then SS+ would copy the row enough times so that each column value would have its own row. For example, if an activity was assigned two instructors, then the activity row would be copied, and the "Allocated staff member" column would be different for these two rows.

Table 4.3.1: Summary of the features seen in the raw data exported from Scientia Syllabus Plus

Data	Column name	Description
Room data	Name	ID of the room
	Capacity	Maximum number of people allowed in room
	Name of zone	Zone on campus where the space is situated
	Primary suitabilities	Key features of the space
Student data	Name of student	ID of the student
	Activity	Name of activity that student is attending
	Activity module	Name of module of the activity
Activity data	Name of activity	ID of activity
	Module of activity	Module of the activity
	Allocated staff member	Primary staff member assigned to activity
	Allocated location	Location the activity has been given
	Location suitabilities	Required features of a space for activity
	Planned size	Expected number of students attending
	Date	Date that the activity is happening
	Start	Time of day the activity starts
	Duration	Length of time that the activity takes

It is important to note that before any extraction begins, the raw data is filtered to remove “tricks” used by the timetabling team to include certain activities. This was done in consultation with the timetabling team at Lancaster University.

4.3.2 Room information

The room information needed for the ITC-2019 format is primarily the capacity, availability and the relative distance from the other rooms on the list. Capacity can be pulled directly from the room data. Unfortunately, the provided data did not outline whether rooms were unavailable at certain times. Therefore, it is assumed that rooms are available at all times.

This means that the final aspect of the room information to extract is the relative distance between rooms. The travel time between rooms within the same building is assumed to be negligible, and the travel time between two rooms in different buildings is the time taken to travel between those buildings.

OpenStreetMap (OSM) data for Lancaster University (OpenStreetMap contributors, 2025) was used to find the relative distance between buildings. Firstly, the coordinates of the centroid of each building are found. Secondly, the closest OSM node to each centroid is identified. Finally, the shortest distance between pairs of nodes is calculated. These values are taken as the relative distance.

As outlined by Table 4.3.1, each space has a list of primary suitabilities that indicate what the space is often used for and what equipment the room has. One suitability a space may have is “Lecture Capture or Stream”. Whilst the ability to capture does not imply the ability to stream, each space with that suitability is given the benefit of the doubt and a tag is attached to the room indicating that it is “hybrid capable”.

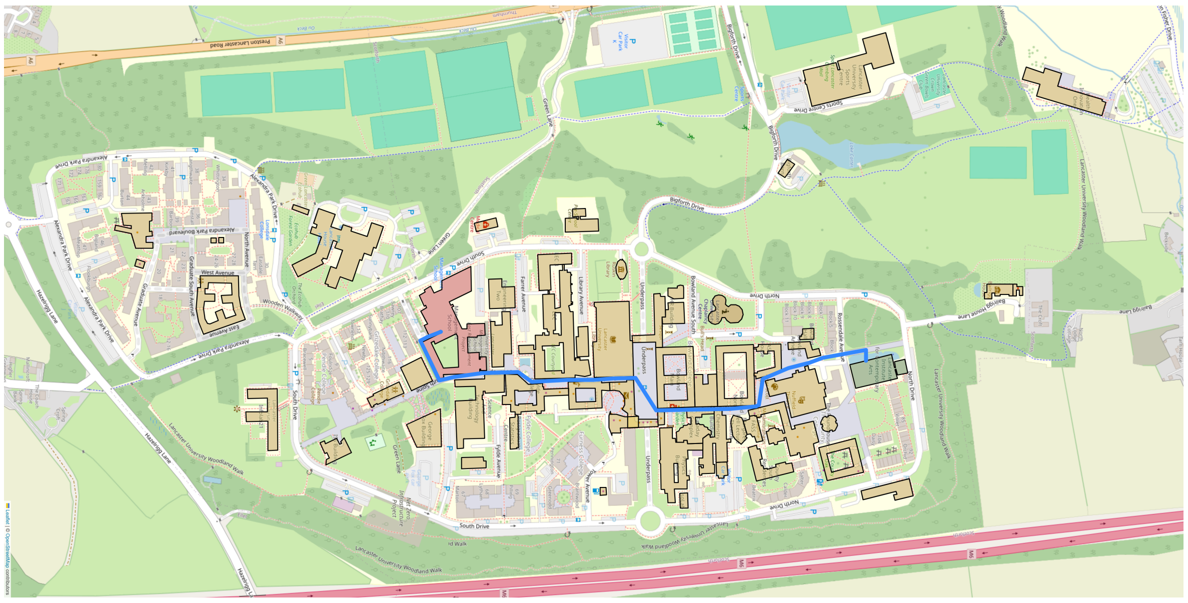


Figure 4.3.1: Map of the Lancaster campus at Lancaster University in the United Kingdom (OpenStreetMap contributors, 2025). The blue line shows the shortest route between the Lancaster Institute for the Contemporary Arts (LICA) building and the Management Sciences building (shaded green and red, respectively)

4.3.3 Student module requests

Student module requests are straightforward to identify. In the student data, there is a list of activities that each student attends. The module that each activity is a part of is given as part of the activity data. Therefore, the module request list for each student is simply the set of modules associated with the activities they are doing.

Some activities are filtered out based on advice from the LU timetabling team. Activities such as field trips are filtered out as these have typically been "hacked" into the schedule by scheduling an event in a location on campus for the entire day.

4.3.4 Module structure

For each module, we can infer the structure based on the activities that each student enrolled on that module attends. It is common for students to attend some of the same activities in a module, such as lectures, and then attend some combination of workshops and labs.

Using the language of the ITC-2019, each unique combination of activities that a student attends can be regarded as a configuration of that module, where each activity is a subpart of the configuration. To match the ITC-2019 formatting, we introduce a constraint set called "SameClass" where a list of activities in this constraint must happen at the same time and place. This allows us to relabel identical activity names to ensure that each activity name is unique.

For example, suppose a module has four activities. If Student A attends activities 1, 2, and 3, then this would be considered a configuration, and if Student B attends activities 1 and 4, this would be considered a different configuration. As activity 1 appears in two different subparts, the second appearance is relabelled to activity 5, and a "SameClass" constraint linking activity 1 and 5 is added to the instance.

The "SameClass" constraint applied to a pair of activities is identical to implementing the ITC-2019 "SameStart", "SameTime", "SameDays", "SameWeeks" and "SameRoom" constraints for that pairing.

4.3.5 Activity information extraction

Having been provided the 2023/24 Lancaster University schedule, it is possible to extract the room and timeset for each activity.

Room extraction

As Table 4.3.1 suggests, we are given the allocated location for each activity. There is a chance that multiple venues are assigned to the activity. In the case of multiple spaces, we take the space with the largest capacity.

Before extracting this location, we check that the capacity is larger than the planned size of the activity and the actual size of the activity in the original schedule. If the space fails this check, then the room is left unassigned.

Some activities are held online, and in this case, to maintain the formatting of the ITC-2019, the activity is labelled as not requiring a room.

Timeset extraction

The academic year can be split into time slots of a given length. By collecting all of the activities with a certain ID, it is possible to convert the dates of the activities along with the start time and duration of the activities on those dates into timeslots describing when that activity is taking place. This sufficiently meets the definition of "timeset" outlined in Chapter 2; however, we aim to format time in a similar way to the ITC-2019.

This format requires us to specify days and weeks in a binary string (for example, days with a string equal to "1001000" indicating activity is on Monday and Thursday) as well as start and length as starting timeslot and duration in timeslots, respectively. For the majority of activities, this is not a problem. Issues arise whenever the following occur:

- An activity has different start times for two or more sessions
- An activity has different durations for two or more sessions
- An activity has sessions on different days on different weeks

The resolution to these issues is to split the activity into separate sessions. For example, if an activity alternated each week between Monday and Thursday, then we would split the activity into the "Monday session" and the "Thursday session". Students attending this activity are then constrained to attend both of these separate sessions.

Extraction anomalies

In the model described in Chapter 2, it is not permitted for any activity to use the same space as another activity at the same time. This constraint is broken by the raw data (that is, the existing timetable) for some spaces. When the activities overlap in time perfectly, then they are essentially the same activity and are labelled accordingly. If the overlap is not perfect, then these activities are shortened slightly to eliminate the overlap or split into parts. The number of activities corrected in this way was negligible (less than 0.02% of the total number of activities).

4.3.6 Activity information generation

Given the extracted room and timesets for each activity, these can be used to generate or select new candidate rooms or timesets.

Room selection

The identification of candidate spaces for an activity involves looking for spaces with equal or greater capacity in the same zone as the extracted space. For each space fulfilling this criterion, the location suitabilities are checked against the space's primary suitabilities. If all of the location suitabilities are met, then the space is attached to the activity as an alternative space that can be used. If there are no suitable spaces identified and the activity has no location assigned from the previous room extraction step, then the activity is assumed to happen online.

Timeset generation

Generating new timesets for an activity involves modifying each of the individual elements of the ITC-2019 definition of a timeset. The most minor modification to a timeset is to create a new timeset for each possible starting time and keep all other elements identical. The second modification is to permute the days of the week. For example, if the original days string was “1000000” then we could generate a timeset with the string “0100000” or “0010000”. Likewise, the third modification is to permute the “1”s in the original string for the weeks the activity occurs. These permutations ensure that the activity still has the same number of contact hours per week.

4.3.7 Staff information

Each activity can have instructors assigned to it. Any given instructor will likely be assigned to multiple activities. Consequently, it is important to record this in our custom instance and ensure, where possible, that this list of activities does not overlap in time and that there is enough time to travel between the activities. These constraints are mathematically modelled in Chapter 2 by the constraints described in Equation 2.4.20.

4.3.8 Data summary

The instance created using the Lancaster University data includes timesets that contain timeslots 5 minutes in length. The raw data spans over 65 weeks, including each day of the week (Monday to Sunday), which the instance created reflects. Table 4.3.2 provides values that summarise the instance.

Table 4.3.2: Summary metrics of instance created using university data

Metric	Value
Number of activities	60,728
Number of modules	1,566
Number of spaces	640
Number of students	13,991
Number of timesets	35,539
Number of staff	410
Average modules per student	7.39
Average activities per staff	5.74

4.4 Model and data evaluation

Before using the data and the model to assess policy, we use subsets of the data and the model to verify that they can produce identical or better timetables. Using subsets of the data allows for an exact solution approach. The validation is a two-stage approach.

First, we fix the variables in the model to reflect the raw data. This forces students to attend activities listed in the "Student data" and the activities are forced to occur in the timeset and location outlined by the "Activity data". This assignment will be referred to as either the "default solution" or "initial solution". Only these assignment variables are fixed. Every other variable in the model (typically those relating to module structure or objectives) is allowed to change, subject to the constraints.

The objective values of this restricted model are then compared to the measured objective values of the original timetable to ensure they are the same. This will also indicate if the solution given by the raw data is feasible.

Secondly, we unfix the variables and allow the full model to run, potentially rearranging the students to improve the objective values. This gives us insight into the magnitude of potential improvement in the timetables and the computational demand of the model.

Note that all experiments in this chapter were completed on an internal computing node running Ubuntu 22.04.5 LTS with an Intel Xeon Gold 6348 CPU running at 2.60GHz and 528GB of RAM. The models were implemented in Python 3.12.7 with Gurobi Optimizer version 12.0.1 as our choice of solver (either directly or within the matheuristic applied in Section 4.5).

4.4.1 Notation and objectives

When summarising the size and contents of the subsets of the data chosen, the mathematical notation for sets of UCTTP elements is adopted from Chapter 2. When describing any methods applied to solve the problem, the notation from Chapter 3 will be used. In this chapter, we use the terms class and activity interchangeably.

Much like Chapter 2 and Chapter 3, the objective functions for the data validation experiments are the following:

z_1: Maximise the total number of elective module requests met.

z_2: Minimise the total number of deviations from mode requests.

z_3: Minimise the total number of student scheduling issues.

It is important to clarify that for these tests, it is still assumed that all modules are elective modules. Where this chapter differs from Chapter 2 and Chapter 3 is that we now assume that every student prefers in-person teaching to online teaching.

4.4.2 Student with the most modules

On average, each student is enrolled on seven modules. There is a student enrolled on 23 modules, who we will refer to as “Student A”. This student will form the basis for our first two evaluations of the data and model.

Single student

For the single student instance, denoted as *SA1*, we create an instance with only Student A and the modules that they are enrolled on. Table 4.4.1 provides the key metrics of this subset of data along with the objectives given by the fixed solution.

All students

For this instance, denoted as *SA2*, we extend *SA1* to include students enrolled on any of the modules that Student A is enrolled on. We do not include the modules of these students that Student A does not attend. Table 4.4.1 shows that there is no change in $|K|$, but increases in the other values. This is due to the additional configurations of the modules that other students take.

4.4.3 Module with the most students

On average, there are 65 students enrolled on a module. There is one module that has a significantly higher enrolment of 470 students. We will refer to this module as “Module A”. This module will form the basis for our final two evaluations of the data and model.

Single module

For this test, we create an instance, denoted as *MA1*, with only Module A and all of the students who are enrolled on this module. Table 4.4.1 confirms that only one module is present in the instance. It shows a greater number of activities and timesets than both *SA1* and *SA2*.

Related modules

For the final test, denoted as *MA2*, we extend *MA1* to include all of the modules that the students attending Module A are enrolled on. We still keep the same subset of the student population, that is, the 470 students enrolled on Module A. This increases the

Table 4.4.1: Instance metrics for each test defined in Sections 4.4.2 and 4.4.2 as well as the full instance. The objective values of z_1, z_2 and z_3 for the default solution are also presented

	Instance metrics					Fixed instance		
Test	$\|S\|$	$\|K\|$	$\|C\|$	$\|R\|$	$\|T\|$	z_1	z_2	z_3
SA1	1	23	52	404	480	23	12	7
SA2	690	23	202	282	510	1,117	773	151
MA1	470	1	1,594	54	520	470	2,350	87
MA2	470	39	6,409	436	4,076	2,054	3,748	9,521
Full instance	13,991	1,566	60,728	657	35,539	102,673	84,078	127,626

module count from 1 to 39 and significantly increases the number of activities in the instance.

4.4.4 Evaluation summary

For each instance, it was found that the expected objective values recovered from the initial solution matched those found by the model with the fixed assignments of activities and the fixed assignments of students to these activities. If the model did not represent the module structure and each objective accurately, then we would have seen a difference in these objective values. This test also helped us identify issues with the data processing (Section 4.3.5) as these issues would cause infeasibilities that could be isolated using the Irreducible Infeasible Subsystem (IIS) tool in Optimizer version 12.0.1 (Gurobi Optimization, LLC, 2025).

By unfixing the model for *SA1* and *SA2*, the objective value of z_3 could be improved from the initial solution, eliminating student scheduling issues from the instance. It was also found that there was often a trade-off between two or more objectives. Table 4.4.2 presents the objective values for each lexicographic ordering. This demonstrates that by using this model, the original timetable recovered from these instances can be improved, or alternative solutions favouring different objectives can be found.

Table 4.4.2: The objective values of z_1, z_2 and z_3 for the instances defined in Section 4.4.2 when solved lexicographically

	Ordering	z_1	z_2	z_3	Solver time (s)
SA1	z_1,z_2,z_3	23	12	0	14
	z_1,z_3,z_2	23	12	0	17
	z_2,z_1,z_3	14	0	0	4
	z_2,z_3,z_1	14	0	0	2
	z_3,z_1,z_2	23	12	0	12
	z_3,z_2,z_1	14	0	0	2
SA2	z_1,z_2,z_3	1,117	773	0	3,386
	z_1,z_3,z_2	1,117	773	0	3,272
	z_2,z_1,z_3	559	0	0	874
	z_2,z_3,z_1	559	0	0	529
	z_3,z_1,z_2	1,117	773	0	2,030
	z_3,z_2,z_1	559	0	0	388

By unfixing the model for *MA1* and *MA2*, we identified that solving the full instance would not be able to be done using an exact method. Even after giving the model several days on a computing node with lots of memory, the solver struggled to produce improved solutions. This was expected as Table 4.4.2 shows a large increase in solve time between *SA1* and *SA2*. These computational challenges faced when attempting to improve the initial solution for *MA1* and *MA2* motivate the need to apply the approaches described in Chapter 3 when handling the full instance. The methods in that chapter require a feasible initial solution, which is a requirement that is satisfied due to how the instance was constructed.

4.5 Policy evaluation

From the literature review in Section 4.2, several approaches to assess policy and other changes to the timetabling procedure were identified. This section aims to demonstrate how each of these techniques from the literature can be used. We use the processed data from Section 4.3, the model from Chapter 2 and methods from Chapter 3 to evaluate potential changes to common timetabling policies seen at universities. At the end of this section, we review our findings regarding the assessed policies and discuss the implications of these results on strategic planning at universities.

4.5.1 Curricula changes

One significant change to any university that impacts timetabling is any change to the curricula of different programmes. It has been observed that timetabling for high schools, where students typically follow rigid programmes of teaching, is a less challenging problem than for universities, where individual students have a large amount of flexibility.

Whilst this flexibility is attractive to students, trying to find a timetable that can facilitate this flexibility is difficult. Chapter 3 proposes a measure for how "related" any two students are based on their module selection, and a method for partitioning students based on this measure. This section demonstrates how that methodology can answer the following questions:

- How related are students currently?
- How related are students under a modified curriculum?
- How does this impact the timetabling process?

For each module in the "Module of activity" column of "Student data" (see Table 4.3.1), the specific programme is in the first four characters of the module, which indicates the school or theme. For example, "MSCI222" is a module under "Management Science".

After checking the modules of each student, the maximum number of schools/themes that any one student has is eight. Let $n \in \{1, \dots, 8\}$. For each n, a custom instance is created where only the top n schools/themes for each student are kept. For example, suppose for some student $s \in S$ that $K_s = \{\text{MSCI222}, \text{MSCI223}, \text{MATH240}\}$. If $n = 1$,

Table 4.5.1: Summary statistics for the partitions found

	Top n schools/themes		
	1	2	3
Block count	111	28	25
Largest block size	858	12,758	12,969
Smallest block size	1	3	3
Average block size	126	500	560

the set of requests would be reduced to $K_s = \{\text{MSCI222}, \text{MSCI223}\}$ as “MSCI” occurs twice and “MATH” occurs once.

For each new instance, the students are partitioned into module-independent groups. This means a student from a given group will share at least one module request with at least one other student in the same group, and two students from different groups will not share any module requests.

Figure 4.5.1 shows how the number of partitions changes subject to the choice of n. It was found that the partitions produced for $n \in \{3, \dots, 8\}$ were all identical. Therefore, Table 4.5.1 only reports the values of n such that $n \in \{1, 2, 3\}$. The largest blocks contain 91% and 93% of students for $n = 2$ and $n = 3$, respectively. This indicates that it is highly likely that any two students at a university are connected by some chain of modules, making a clean partition of the problem difficult to achieve. When $n = 1$, which represents the scenario where students only study modules from a single school/theme, the partition contains significantly more blocks, with the largest block containing only 6% of students.

These results suggest that even under scenarios of radical curriculum restructuring, students remain highly interconnected through overlapping module enrolments. Continued student interconnectedness creates additional constraints on shared resources, particularly staff assignments, room allocations, and timeslot availability. Whilst this conclusion is negative from a decomposition point of view, we have not found sufficient

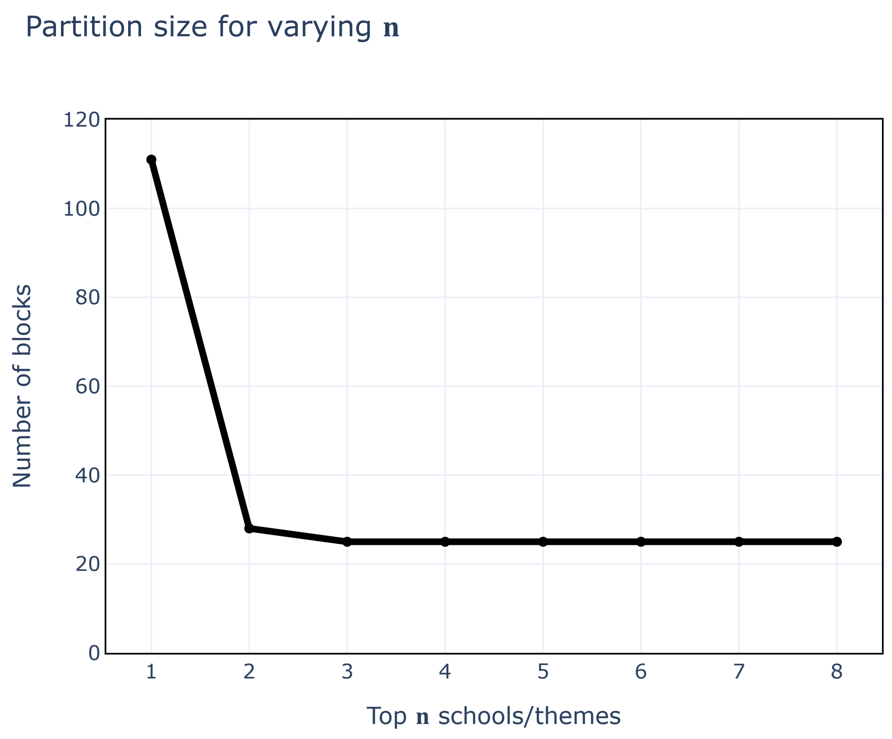


Figure 4.5.1: Line plot illustrating the number of blocks in the partition when the top n schools/themes of each student are kept

evidence to suggest that stopping students from taking an interdisciplinary programme of study will significantly benefit the timetabling process.

Two new lines of research arise from this section and the computational difficulties experienced in Section 4.4. The first would be to investigate limiting the number of potential configurations and subparts for each module to benefit timetable creation efficiency, instead of limiting student requests. This would reduce the size of the model described in Chapter 2 and hopefully make the model easier to solve. The second would be to apply minimum cut techniques to a graph where each node represents a student and each edge is weighted according to how related the students are. This could give rise to an alternative partitioning scheme to the one presented in Chapter 3.

4.5.2 Teaching hours

At many universities, there are core teaching hours. At Lancaster University, this is 09:00 to 18:00 Monday to Friday, except for Wednesday, where the hours are 09:00 to 13:00. The timetabling policy states that, under specific conditions, there may be credit-bearing teaching outside of these teaching hours.

Placing activities outside of these teaching hours negatively impacts students. The Wednesday afternoon gap is designed to be time for students to take part in extra-curricular activities like sports or society events. Placing events on the fringe of the teaching hours (early morning or late afternoon) can negatively impact students who have a significant commute. In the initial solution, 3,814 activities were scheduled outside of core teaching hours. This is 6% of the total number of activities but impacts 9,740 unique students (70%). Motivated by these figures, this section addresses the following questions:

- Can activities be moved to occur within teaching hours?
- What is the impact on infrastructure usage?

One approach to answering the first question is to attach an integer penalty to each timeset representing the number of meetings that fall outside of core hours. For example, if a timeset described a class that was held after 18:00 on a Monday and Tuesday for three weeks, the penalty would be as follows:

$$1(\text{events}) \times 2(\text{days}) \times 3(\text{weeks}) = 6. \tag{4.5.1}$$

The goal is to minimise the sum of these timeset penalties while not allowing the values of z_1, z_2, and z_3 to deteriorate. By applying the matheuristic from Chapter 3 (using the parameters specified in Table 4.5.3), we were able to reduce the number of problematic events from 3,814 to 3,751 and maintain the quality of the other objectives.

Whilst this seems like an improvement on the surface, post-optimisation analysis shows that the new solution uses 35 more physical spaces (a 13% increase). However, a more critical point is that the problematic sessions impact slightly more students, with 9,776 unique students being affected. This reveals a significant limitation of the above approach, underscoring the non-trivial nature of optimisation decisions in university timetabling, where intuitive improvements can yield counterproductive results.

4.5.3 Capacity relaxation

At many universities, a key restriction to the timetabling process is the availability of suitably sized teaching spaces. When processing the real-world data, we found over 400 activities where the planned size or actual size (whichever was larger) exceeded any of the suitable spaces. Figure 4.5.2 illustrates that this may be because there are many modules which need to share the limited number of high-capacity venues on campus.

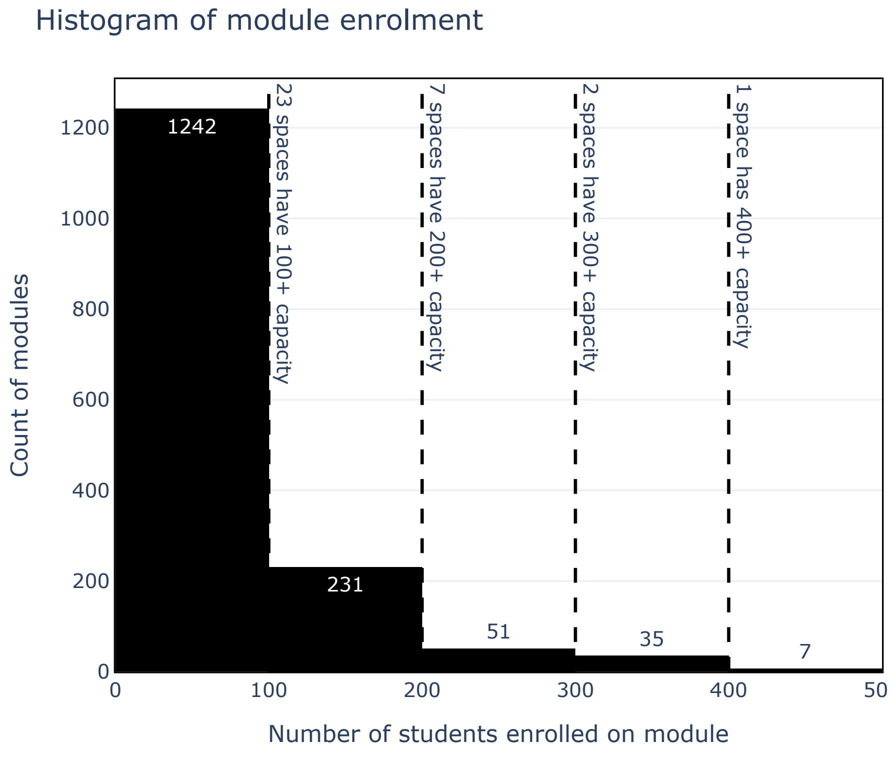


Figure 4.5.2: Histogram showing the enrolment of students on modules. Information on physical space capacity is also given to highlight the issue of limited resources

There are examples in the literature where room capacity is considered a soft constraint (see De Causmaecker et al. (2009)). The justification for this is that attendance typically peaks at the start of the semester and quickly drops, but eventually stabilises.

In our case, keeping the room capacity constraints as hard constraints creates two operational challenges. Firstly, the limited availability of large teaching spaces restricts the timetabling process. Secondly, activities continuing to occupy oversized rooms throughout the semester are an inefficient utilisation of available space. Attempting to anticipate attendance decline by assigning smaller rooms carries substantial risks. Most critically, underestimating attendance can result in insufficient space for all enrolled students, potentially compromising the quality of educational delivery.

Therefore, there is a clear trade-off between long-term efficiency and immediate student satisfaction. The questions we aim to address in this sub-section are:

- How many activities are over-subscribed?
- How necessary is a 1-1 seating policy?
- Can we timetable with a drop off in attendance in mind?

Capacity deviation

In this experiment, we aim to investigate the impact on the resulting timetables if the in-person class capacity is relaxed. This requires us to provide new candidate locations for all activities. For activities that were moved online, candidate rooms include those that were in the same zone as the original allocation and have enough capacity to fit the planned or actual attendance, whichever is smaller. This could mean that the original allocation becomes a candidate location. For activities with a feasible allocation, the same process is followed, increasing the number of alternative rooms for that class.

The original attendance of the activities that were moved online can be compared to the capacity of the largest candidate space. If subtracting the attendance from the capacity of the space produces a negative number, we refer to this as a "capacity deviation". The absolute values of these deviations are placed into a list, $\mathcal{C}^{\text{dev}}$, and ordered from lowest to highest. The value of zero is added to the start of this list.

Relaxing capacity

In Chapter 2, the constraints on the capacity of an activity are given by constraints of the form:

$$\sum_{s \in S} \alpha^{\text{inp}}_{s,c} \leq \sum_{r \in R_c \setminus \{r^*\}} cap(r) y^R_{c,r}, \quad \forall c \in C, \tag{4.5.2}$$

where $\alpha^{\text{inp}}_{s,c}$ is a binary variable indicating student $s \in S$ is attending activity $c \in C$ in person, $R_c \setminus \{r^*\}$ is the set of spaces for activity $c \in C$ not including the online space and $cap(r)$ is the capacity of space $r \in R$.

To relax the capacity, a new model variable is introduced. Let $\Delta_c \in \mathbb{N}_0$ indicate the amount that the capacity of class $c \in C$ is relaxed by. For all $c \in C$, the right hand side of Equation 4.5.2 is modified to be:

$$\Delta_c + \sum_{r \in R_c \setminus \{r^*\}} cap(r) y^R_{c,r}. \tag{4.5.3}$$

The maximum amount that any class can have its capacity relaxed can be done using the following constraints:

$$\Delta_c \leq \Delta, \quad \forall c \in C, \tag{4.5.4}$$

where Δ can either be an integer variable or a fixed value.

Capacity experiment

For each value $\delta \in \mathcal{C}^{\text{dev}}$ we set $\Delta = \delta$, constrain the objective functions for z_1, z_2 and z_3 to be no worse than the values identified in Section 4.4 and then minimise the total number of deviations from a student's preferred mode (z_2). Table 4.5.3 outlines the details of this experiment.

Table 4.5.2 presents the results of this experiment. These results show that relaxing the capacity of physical spaces can facilitate the improvement of meeting mode preferences. Table 4.5.2 also gives the number of activities that went above the original capacity of the space, which shows that as the relaxation level increases, the number of activities that take advantage of this also increases. Despite there always being at least one activity that takes advantage of the full relaxation, the average amount of capacity actually relaxed becomes a decreasing percentage of the full relaxation.

Table 4.5.2: Level of relaxation (Δ_c), total number of deviations from a student's preferred mode (z_2), number of activities with relaxed capacities, maximum relaxation and average relaxation

	Δ_c			
	0	1	13	28
z_2	83,230	81,714	80,680	79,816
Relaxed capacities	0	303	501	655
Maximum relaxation	0	1	13	28
Average relaxation	0	1	5.54	8.04

To investigate this phenomenon further, Figure 4.5.3 is a histogram of the total amount over capacity for each relaxed activity. This suggests that as the maximum allowable capacity relaxation increases, the utilisation of this relaxation decreases, with most activities requiring only modest capacity violations.

When considering the improvement in the mode preference objective whilst also taking Figure 4.5.3 into account, it is clear that modest relaxation in room capacity can greatly improve this objective.

The benefit of this approach over allowing capacity to be a soft constraint is that it controls the worst-case scenario. In practice, a timetabling practitioner could generate histograms, such as Figure 4.5.3, for each relaxation level and assess the most appropriate

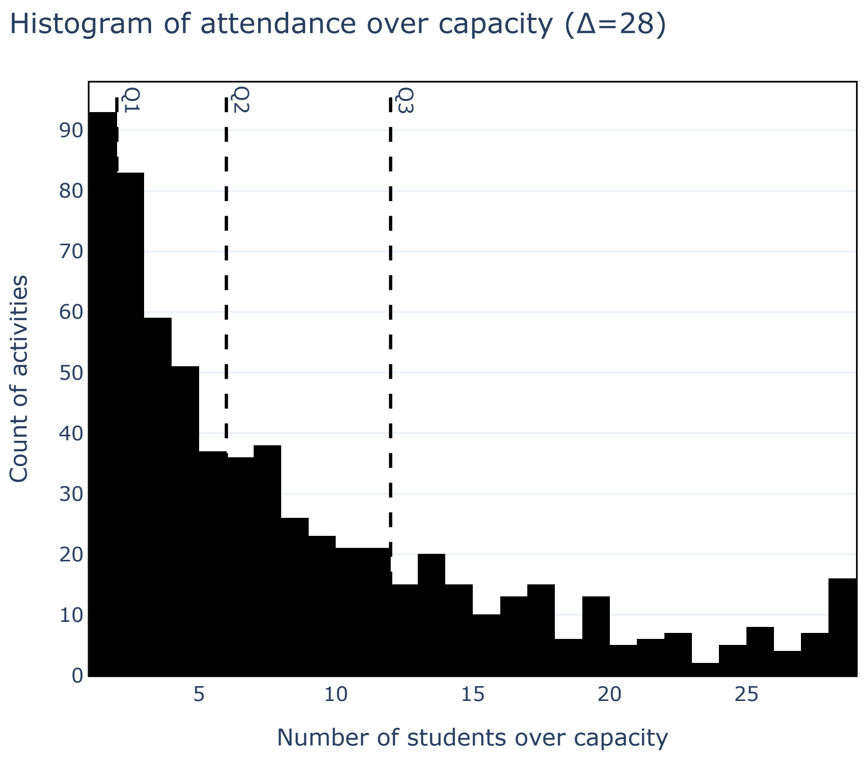


Figure 4.5.3: Histogram showing the number of students over capacity for each relaxed activity. Quantiles are marked using dashed lines

maximum acceptable relaxed capacity. The optimisation method could then be run, with the chosen Δ, for more iterations than outlined in Table 4.5.3. This would produce a solution closer to optimal and ensure that no activity is relaxed more than the maximum amount.

4.5.4 Hybrid teaching

In the previous section, we investigated how we should schedule if we expect a drop in student attendance. In this section, we investigate how to schedule when we expect a drop in available physical capacity.

In the 2020 COVID pandemic, university modules moved online to enable social distancing. However, coming out of the pandemic, some universities offered an option known as "hybrid teaching" where a portion of the class attends online and a portion attends in person. To ensure safe in-person attendance, students were required to spread out throughout the space. This, in some cases, led to a fourfold reduction in physical capacity (Barnhart et al., 2022). Figure 4.5.4 demonstrates that the original timetable for the real-world instance becomes infeasible under this level of capacity reduction.

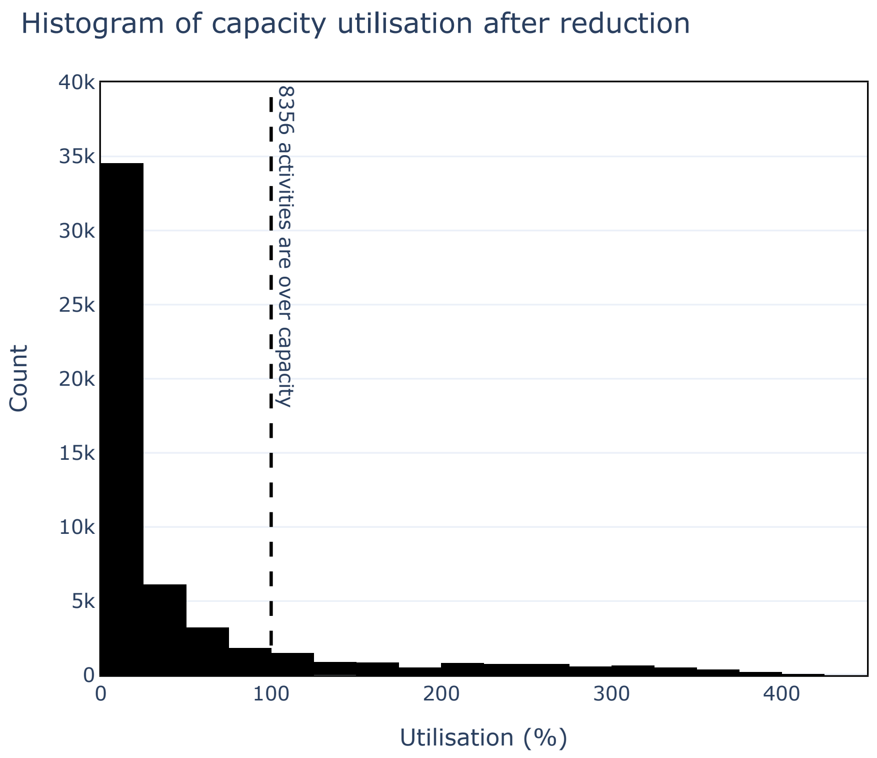


Figure 4.5.4: Histogram showing theoretical utilisation of a space after reducing physical capacity by 75%

Some universities, such as Lancaster University, have spaces with equipment that can record or broadcast lectures. These typically make up a small proportion of teaching spaces, with 19% of teaching spaces at Lancaster University able to capture or stream an activity. The questions we aim to address in this section are:

- Does allowing hybrid teaching fix the capacity issue?
- Can spaces to upgrade be identified?
- Can we minimise the number of spaces to upgrade?

Solution under scarcity

Reducing the capacity of all physical spaces by 75% makes the current solution infeasible. To remedy this, every single activity is moved online. The aim now is to return as many students to in-person teaching. In particular, we are optimising the total number of deviations from mode preferences (minimising z_2) whilst ensuring the number of module requests met and the number of student conflicts don't deteriorate from the values found in Section 4.4.

The reason this approach is taken rather than using a starting solution that takes advantage of some of the hybrid spaces is that this would require a decision to be made on how the students should be split. A poor choice of this split could potentially lead to more student conflicts than in the initial solution because of the large travel time between in-person and online.

After optimising using the parameters outlined in Table 4.5.3, only 50 activities were returned to in-person teaching and 5 activities operated in hybrid mode. This limited improvement can be attributed to several constraining factors: fixed student timetables that restrict potential modifications, insufficient algorithm iterations, and the fundamental constraint of reduced physical capacity that limits feasible improvements.

Figure 4.5.5 illustrates that the improvement in objective value does not plateau in the typical way that matheuristic approaches near optimality do. This suggests that more iterations would lead to a greater restoration of activities to in-person or hybrid teaching.

Solution allowing conversions

With the solution under scarcity, one potential limiting factor is that not every physical space is hybrid-compatible. However, converting a space to be hybrid-compatible can be costly, and therefore, the number of spaces converted should be limited. To minimise the number of conversions, a binary variable is introduced into the model to indicate if a

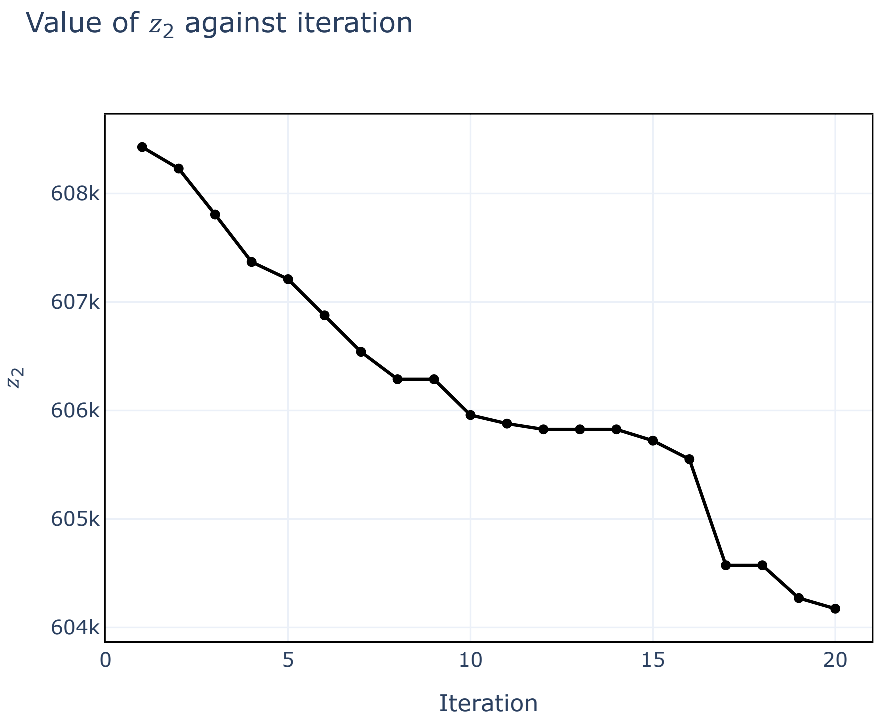


Figure 4.5.5: Plot of the objective value against iteration in the case of restricted resources

space has been converted. In particular, for $r \in R \setminus R^h$ where $R \setminus R^h$ is the set of spaces that are not hybrid compatible, $\nabla_r = 1$ indicates that a room has been converted.

In Chapter 2, constraints ensuring that a class was not hybrid if assigned a room that was not hybrid-compatible were the following:

$$y^R_{c,r^*} \leq 1 - \sum_{r \in R \setminus R^h} y^R_{c,r}, \quad \forall c \in C, \tag{4.5.5}$$

where $y^R_{c,r}$ is a variable indicating if activity $c \in C$ is assigned space $r \in R$.

Modifying this constraint to allow for conversions means replacing the constraints defined in Equation 4.5.5 with the following:

$$y^R_{c,r^*} \leq 1 - y^R_{c,r} + \nabla_r, \quad \forall c \in C, r \in R \setminus R^h \tag{4.5.6}$$

Now that the model has been modified in this way, we continue improving our solution from the last step of this experiment, allowing any room to be converted if it allows for an improvement in the objective. Once this improvement has stopped, we constrain the value of z_2 so that it does not deteriorate and minimise a new objective z_5 defined as:

$$z_5 = \sum_{r \in R \setminus R^h} \nabla_r \tag{4.5.7}$$

By minimising Equation 4.5.7, we find the most essential spaces to convert. These can further be ranked by looking at each conversion's impact. In particular, determining the number of students' mode preferences that are met by attending an activity in this space.

By optimising z_2 using parameters given in Table 4.5.3, further improvement in z_2 was achieved. There were 145 activities moved to in-person teaching (195 total) and 26 activities moved to hybrid teaching (31 total). Much like the previous stage of this experiment, it may have been beneficial to increase the number of iterations the algorithm would perform.

Post-optimisation analysis found that seven rooms had been converted into spaces capable of hybrid teaching. We were unable to reduce the number of converted rooms by minimising z_5. Figure 4.5.6 shows the impact of each room conversion on student and activity counts, identifying priority areas for hybrid teaching enhancements. Based on these results, the spaces *113* and *253* are both used for 11 activities and impact over 90 students and therefore should be prioritised for conversion over other spaces.

4.5.5 Experiment details

Table 4.5.3 outlines the matheuristic parameters used for each policy experiment. If there were multiple optimisations for a single experiment, then each optimisation used the same parameters each time.

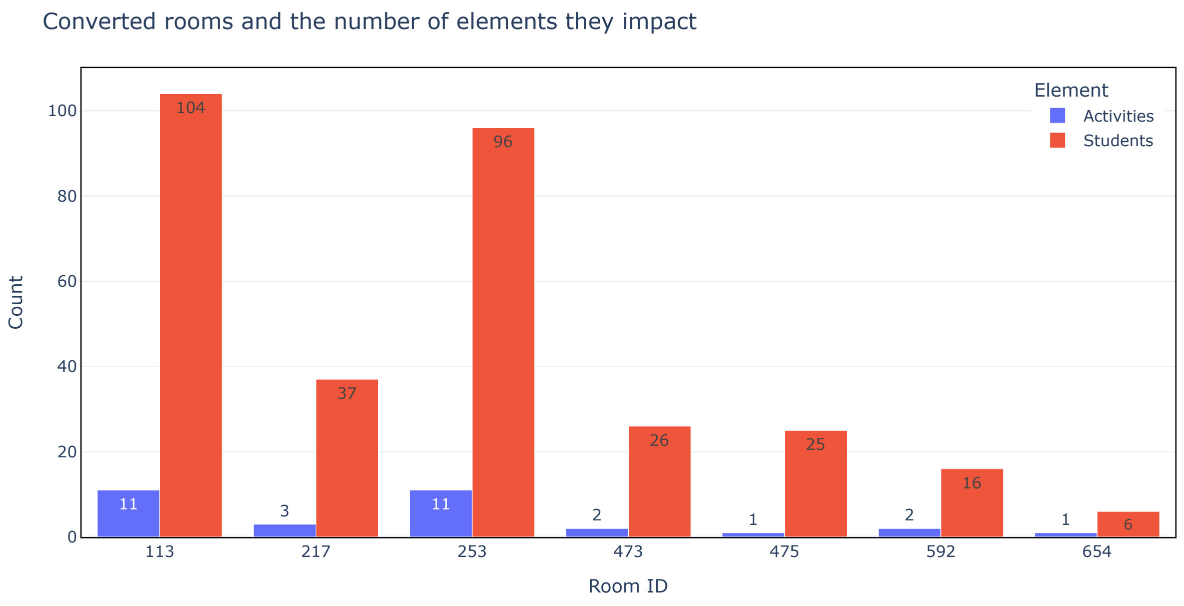


Figure 4.5.6: Converted rooms and the number of activities using them, and the number of students this impacts

Table 4.5.3: Parameters for the matheuristic used in policy evaluation experiments

	Matheuristic algorithm parameter								
Experiment	I	I^T	N^{start}	N^{+}	$N^{\max}$	$X^{\max}$	X^{sol}	$R^{\max}$	B^{prob}
Teaching hours	20	300	0.05	0.01	0.05	3	20	5	0.5
Capacity relaxation	20	300	0.005	0.005	0.015	3	20	5	0.5
Hybrid teaching	20	300	0.005	0.005	0.015	3	20	5	0.5

4.6 Conclusion

This chapter identifies several ways in which timetabling practitioners tackle both operational and strategic elements of the UCTTP. This work aimed to apply a collection of these techniques to a benchmark instance derived using data from a university campus based in the UK, consolidating examples of each technique in one body of work.

To address this gap, this chapter outlines the creation of a new ITC-2019 benchmark instance using real-world data. Four strategic problems were identified by either consulting with practitioners or by augmenting the data to create problematic scenarios to be addressed. Modelling and methodology elements were combined and then applied to the real-world instance to produce insights into solutions for these problems.

In particular, we can explicitly link the examples to the techniques named in the introduction of this chapter. The entirety of Chapter 2 is based on creating a new model, which is then used for three of the problems. The curriculum problem used modified data and the partitioning scheme given in Chapter 3, forming a rudimentary sensitivity analysis. The teaching hours and capacity relaxation problems used new or modified model constraints or new objective functions. Finally, the hybrid teaching problem used nearly all of the identified techniques.

The key takeaways from this chapter are:

- Converting real-world problems into standardised benchmark formats involves significant pre-processing challenges and requires careful preservation of problem characteristics.
- Optimisation methods that seem intuitively appropriate may generate solutions that are practically infeasible or operationally unacceptable.
- Strategic integration of techniques, with careful validation of practical feasibility, can yield substantial insights for complex, highly interdependent problems.

Finally, this work raises important research questions for further exploration:

- Can techniques to address strategic questions using mathematical optimisation be classified?
- Can specific strategic questions be paired with a particular approach?

By addressing these questions, we would result in a catalogue of methods paired up with optimisation techniques so that if a timetabling practitioner encounters a specific problem, they can quickly identify what approach to use. If the problem is truly unique, then the new approach developed could be classified and catalogued for future use.

Chapter 5

Conclusions

This thesis has addressed the challenges of university timetabling in the modern educational landscape, with particular focus on the emerging paradigm of hybrid teaching. Through the development of novel mathematical models, solution methods, and the use of real-world data, this research has contributed both theoretical advances and practical insights.

5.1 Contributions

The primary contributions of this thesis can be understood through the lens of the five key research questions established in the introduction:

Q1: Hybrid teaching integration

Chapter 2 addressed this question by developing the first mathematical formulation to explicitly incorporate hybrid teaching into university timetabling. The model introduces variables to capture student modes of participation (in-person versus online) and establishes three key objective functions: (i) Maximising module requests, (ii) minimising mode preference deviations, and (iii) minimising student conflicts. The research identified critical data requirements, including room capabilities, student mode preferences, and capacity constraints for both physical and online participation.

Q2: Student partitioning and objective bounds

Chapter 3 introduced a novel partitioning approach that clusters students based on their module requests, creating manageable subproblems whilst maintaining key problem structures. This partitioning strategy enables the computation of meaningful bounds on objective functions by solving smaller, isolated problems optimally. By providing limits on objective values, university administrators can better understand the feasibility of their goals and the trade-offs inherent in their scheduling decisions.

Q3: Lexicographic optimisation methods

In the first half of Chapter 3, a multi-objective fix-and-optimise matheuristic was developed for large-scale university timetabling problems. The method extends traditional single-objective matheuristics to handle lexicographic optimisation efficiently, incorporating the student partitioning scheme to guide neighbourhood selection. The experimental results demonstrated that this approach can handle realistic problem sizes while generating high-quality solutions across different objective orderings.

Q4: Multi-objective Pareto analysis

In the second half of Chapter 3, the methodology from the first half was extended to approximate Pareto fronts for multiple objectives, revealing significant interactions between competing goals. The Pareto front approximation enables decision-makers to visualise the full spectrum of possible solutions and understand the true cost of prioritising one objective over others.

Q5: Strategic policy assessment

Chapter 4 demonstrated how the models and methods developed in this thesis could be used for real-world policy evaluation using data from Lancaster University. The research showed how timetabling models can assess the impact of various policy changes and illustrated multiple approaches to policy evaluation.

Additional contributions

Beyond addressing the core research questions, this thesis has made one final contribution. As part of the work in Chapter 4, a realistic and large-scale benchmark instance based on Lancaster University data was created. This instance, formatted according to ITC-2019 standards, provides the timetabling community with a valuable resource for future research.

5.2 Further research

While this thesis has made significant contributions to university timetabling research, several important directions for future work have emerged. As each chapter summarises these individually, we will simply recap the themes of these directions:

1. **Modelling extensions:** Whilst the model is fairly representative of the UCTTP, there are certain features still to be included in this model. Chapter 2 outlines a few of these in detail. Chapter 4 applies some model extensions to tackle strategic decision-making problems, but this is not exhaustive by any means.

2. **Method enhancements:** The methods proposed in Chapter 3 use simple low-level procedures, and this is to provide a successful proof of concept for the method. Many of these procedures could be improved to enhance the methodology overall. Where this can be done is identified either within the text or in the conclusion of Chapter 3.

3. **Multi-objective methods:** The multi-objective approach presented in this thesis lacks interaction from a decision-maker. The method for searching the Pareto frontier currently does so in a stochastic fashion, treating all objectives equally. Better incorporation of decision-maker preferences for improving some objectives over others would mean a method that is more useful as a decision-support tool.

5.3 Final remarks

This thesis has demonstrated that university timetabling requires sophisticated mathematical modelling and solution approaches that can handle multiple objectives, emerging teaching trends, and complex stakeholder requirements. The explicit incorporation of hybrid teaching into timetabling models represents one crucial step forward in addressing modern educational challenges.

The multi-objective perspective adopted throughout this research reveals that effective timetabling is not merely about finding feasible solutions, but about understanding and managing trade-offs between competing goals. The methods developed in this work provide universities with tools for decision-making, transforming timetabling from an operational necessity into a strategic advantage.

As higher education continues to evolve, the need for efficient timetabling approaches will only grow. This thesis provides a foundation for addressing these challenges, but the journey toward optimal educational scheduling systems is far from complete. The future research directions outlined above represent opportunities for the academic community to build upon this work.

The ultimate goal remains unchanged: creating educational environments that maximise learning opportunities while efficiently utilising available resources. Through continued research and development in mathematical optimisation, multi-objective analysis, and decision support systems, this goal becomes increasingly achievable.

Bibliography

Aizam, N. A. H. and Caccetta, L. (2014). Computational models for timetabling problem. *Numerical Algebra, Control and Optimization*, 4(3):269–285.

Al-Yakoob, S. M. and Sherali, H. D. (2007). A mixed-integer programming approach to a class timetabling problem: A case study with gender policies and traffic considerations. *European Journal of Operational Research*, 180(3):1028–1044.

Alghamdi, H., Alsubait, T., Alhakami, H., and Baz, A. (2020). A review of optimization algorithms for university timetable scheduling. *Engineering, Technology & Applied Science Research*, 10(6):6410–6417.

Avella, P. and Vasil'Ev, I. (2005). A computational study of a cutting plane algorithm for university course timetabling. *Journal of Scheduling*, 8(6):497–514.

Aziz, N. L. A. and Aizam, N. A. H. (2018). A brief review on the features of university course timetabling problem. *AIP Conference Proceedings*, 2016(1).

Babaei, H., Karimpour, J., and Hadidi, A. (2015). A survey of approaches for university course timetabling problem. *Computers & Industrial Engineering*, 86:43–59.

Badri, M. A. (1996). A two-stage multiobjective scheduling model for [faculty-course-time] assignments. *European Journal of Operational Research*, 94(1):16–28.

Barnhart, C., Bertsimas, D., Delarue, A., and Yan, J. (2022). Course scheduling under sudden scarcity: Applications to pandemic planning. *Manufacturing & Service Operations Management*, 24(2):727–745.

Bettinelli, A., Cacchiani, V., Roberti, R., and Toth, P. (2015). An overview of curriculum-based course timetabling. *TOP*, 23(2):313–349.

Bhardwa, S. (2017). Why do students go to university and how do they choose which one?

Blank, J. and Deb, K. (2020). pymoo: Multi-objective optimization in python. *IEEE Access*, 8:89497–89509.

Broder, S. (1964). Final examination scheduling. *Commun. ACM*, 7(8):494–498.

Burke, E. K., Mareček, J., Parkes, A. J., and Rudová, H. (2012). A branch-and-cut procedure for the udine course timetabling problem. *Annals of Operations Research*, 194(1):71–87.

Carter, M. W. (2001). A comprehensive course timetabling and student scheduling system at the university of waterloo. In Burke, E. and Erben, W., editors, *Practice and Theory of Automated Timetabling III*, pages 64–82, Berlin, Heidelberg. Springer Berlin Heidelberg.

Ceschia, S., Di Gaspero, L., and Schaerf, A. (2023). Educational timetabling: Problems, benchmarks, and state-of-the-art results. *European Journal of Operational Research*, 308(1):1–18.

Chaudhuri, A. and De, K. (2010). Fuzzy genetic heuristic for university course timetable problem. *International Journal of Advances in Soft Computing and Its Applications*, 2.

Chen, M. C., Sze, S. N., Goh, S. L., Sabar, N. R., and Kendall, G. (2021). A survey of university course timetabling problem: Perspectives, trends and opportunities. *IEEE Access*, 9:106515–106529.

Coe, R., Aloisi, C., Higgins, S., and Major, L. (2014). What makes great teaching? review of the underpinning research. Technical report, Durham Research Methods Centre (DRMC). EPrint Processing Status: Full text deposited in DRO.

Cooper, T. B. and Kingston, J. H. (1995). The complexity of timetable construction problems. In *International Conference on the Practice and Theory of Automated Timetabling*.

Dammak, A., Elloumi, A., Kamoun, H., and Ferland, J. (2008). Course Timetabling at a Tunisian University: A case study. *Journal of Systems Science and Systems Engineering*, 17:334–352.

Daskalaki, S. and Birbas, T. (2005). Efficient solutions for a university timetabling problem through integer programming. *European Journal of Operational Research*, 160(1):106–120. Applications of Mathematical Programming Models.

Daskalaki, S., Birbas, T., and Housos, E. (2004). An integer programming formulation for a case study in university timetabling. *European Journal of Operational Research*, 153(1):117–135.

Davison, M., Kheiri, A., and Zografos, K. G. (2024). Modelling and solving the university course timetabling problem with hybrid teaching considerations. *Journal of Scheduling*.

De Causmaecker, P., Demeester, P., and Vanden Berghe, G. (2009). A decomposed metaheuristic approach for a real-world university timetabling problem. *European Journal of Operational Research*, 195:307–318.

de Werra, D. (1985). An introduction to timetabling. *European Journal of Operational Research*, 19(2):151–162.

Di Gaspero, L., McCollum, B., and Schaerf, A. (2007). The Second International Timetabling Competition (ITC-2007): Curriculum-based Course Timetabling (Track 3). Technical Report QUB/IEEE/Tech/ITC2007/CurriculumCTT/v1.0/1, DIEGM, University of Udine. Available from `http://www.cs.qub.ac.uk/itc2007/`.

Di Gaspero, L. and Schaerf, A. (2006). Neighborhood Portfolio Approach for Local Search Applied to Timetabling Problems. *Journal of Mathematical Modelling and Algorithms*, 5(1):65–89.

Ehrgott, M. (2005). *Multicriteria Optimization*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2nd ed. 2005. edition.

Fonseca, G. H., Santos, H. G., Carrano, E. G., and Stidsen, T. J. (2017). Integer programming techniques for educational timetabling. *European Journal of Operational Research*, 262(1):28–39.

Gan, G., Ma, C., and Wu, J. (2007). *Data Clustering: Theory, Algorithms, and Applications*. Society for Industrial and Applied Mathematics.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5):533–549. Applications of Integer Programming.

Gogos, C., Dimitsas, A., Valouxis, C., and Alefragis, P. (2022). Modeling a balanced commute educational timetabling problem in the context of teaching integer programming. In *2022 7th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM)*, pages 1–5.

Gonzalez, G., Richards, C., and Newman, A. (2018). Optimal Course Scheduling for United States Air Force Academy Cadets. *Interfaces*, 48(3):217–234.

Gurobi Optimization, LLC (2025). Gurobi Optimizer Reference Manual. `https://docs.gurobi.com/projects/optimizer/en/current`. Accessed: 2025-10-19.

Gülcü, A. and Akkan, C. (2020). Robust university course timetabling problem subject to single and multiple disruptions. *European Journal of Operational Research*, 283(2):630–646.

Hamerly, G. and Elkan, C. (2004). Learning the k in k-means. *Advances in Neural Information Processing Systems*, 17.

Helber, S. and Sahling, F. (2010). A fix-and-optimize approach for the multi-level capacitated lot sizing problem. *International Journal of Production Economics*, 123(2):247–256.

Holm, D. S., Mikkelsen, R. Ø., Sørensen, M., and Stidsen, T. J. R. (2022). A graph-based mip formulation of the international timetabling competition 2019. *Journal of Scheduling*.

Johnes, J. (2015). Operational Research in education. *European Journal of Operational Research*, 243(3):683–696.

Jourdan, L., Basseur, M., and Talbi, E.-G. (2009). Hybridizing exact methods and metaheuristics: A taxonomy. *European Journal of Operational Research*, 199(3):620–629.

Keller, A. A. (2017). *Multi-objective optimization in theory and practice. I, Classical methods*. Bentham eBooks, Sharjah, United Arab Emirates, 1st ed. edition.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.

Korte, B. H. and Vygen, J. (2019). *Combinatorial optimization : theory and algorithms*. Algorithms and combinatorics ; volume 21. Springer, Berlin, sixth edition. edition.

Kosub, S. (2019). A note on the triangle inequality for the jaccard distance. *Pattern Recognition Letters*, 120:36–38.

Landa-Silva, D. and Obit, J. H. (2008). Great deluge with non-linear decay rate for solving course timetabling problems. In *2008 4th International IEEE Conference Intelligent Systems*, volume 1, pages 8–11–8–18.

Lewis, R. (2008). A survey of metaheuristic-based techniques for University Timetabling problems. *OR Spectrum*, 30(1):167–190.

Lewis, R., Paechter, B., and McCollum, B. (2007). Post Enrolment based Course Timetabling: A Description of the Problem Model used for Track Two of the Second International Timetabling Competition. Technical Report A2007-3, Cardiff Business School, Cardiff University. Available from `http://www.cs.qub.ac.uk/itc2007/`.

Lewis, R. and Thompson, J. (2015). Analysing the effects of solution space connectivity with an effective metaheuristic for the course timetabling problem. *European Journal of Operational Research*, 240:637–648.

Li, M. and Yao, X. (2019). Quality evaluation of solution sets in multiobjective optimisation: A survey. *ACM Comput. Surv.*, 52(2).

Lindahl, M., Mason, A., Stidsen, T., and Sørensen, M. (2017). A strategic view of university timetabling. *European Journal of Operational Research*, 266.

Marler, R. T. and Arora, J. S. (2010). The weighted sum method for multi-objective optimization: new insights. *Structural and Multidisciplinary Optimization*, 41(6):853–862.

Miettinen, K. (1998). *Nonlinear Multiobjective Optimization*, volume 12 of *International Series in Operations Research & Management Science*. Springer, Boston, MA, 1 edition.

Miettinen, K. (2014). Introduction to some methods and applications of nonlinear multiobjective optimization. 39th Conference on The Mathematics of Operations Research 2014.

Mikkelsen, R. Ø. and Holm, D. S. (2022). A parallelized matheuristic for the international timetabling competition 2019. *Journal of Scheduling*.

Mühlenthaler, M. and Wanka, R. (2016). Fairness in academic course timetabling. *Annals of Operations Research*, 239(1):171–188.

Müller, T., Rudová, H., and Müllerová, Z. (2018). University course timetabling and International Timetabling Competition 2019. *Proceedings of the 12th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2018)*, pages 5–31.

Müller, T., Rudová, H., and Müllerová, Z. (2024). Real-world university course timetabling at the international timetabling competition 2019. *Journal of Scheduling*.

Méndez-Díaz, I., Zabala, P., and Miranda-Bront, J. J. (2016). An ILP based heuristic for a generalization of the post-enrollment course timetabling problem. *Computers & Operations Research*, 76:195–207.

Müller, T. (2008). Itc2007 solver description: A hybrid approach. *Annals of Operations Research*, 172:429–446.

OpenStreetMap contributors (2025). Way: Lancaster University (23347159). `https://www.openstreetmap.org/way/23347159`. Accessed: 2025-06-06.

Perrott, L. (2025a). What is the expenditure of HE providers? `https://www.hesa.ac.uk/data-and-analysis/finances/expenditure`. Accessed: 2025-10-19.

Perrott, L. (2025b). What is the income of HE providers? `https://www.hesa.ac.uk/data-and-analysis/finances/income`. Accessed: 2025-10-19.

Pillay, N. (2016). A review of hyper-heuristics for educational timetabling. *Annals of Operations Research*, 239(1):3–38.

Pisinger, D. and Røpke, S. (2010). *Large Neighborhood Search*, pages 399–420. Springer, 2 edition.

Post, G., Di Gaspero, L., Kingston, J., McCollum, B., and Schaerf, A. (2013). The third international timetabling competition. *Annals of Operations Research*, 239.

Post, G., Kingston, J. H., Ahmadi, S., Daskalaki, S., Gogos, C., Kyngas, J., Nurmi, C., Musliu, N., Pillay, N., Santos, H., and Schaerf, A. (2014). XHSTT: an XML archive for high school timetabling problems in different countries. *Annals of Operations Research*, 218(1):295–301.

Rudová, H., Müller, T., and Murray, K. (2011). Complex university course timetabling. *Journal of Scheduling*, 14(2):187–207.

Santos, H. G., Uchoa, E., Ochi, L. S., and Maculan, N. (2012). Strong bounds with cut and column generation for class-teacher timetabling. *Annals of Operations Research*, 194(1):399–412.

Schimmelpfeng, K. and Helber, S. (2007). Application of a real-world university-course timetabling model solved by integer programming. *OR Spectrum*, 29(4):783–803.

Schott, J. (1995). *Fault tolerant design using single and multicriteria genetic algorithm optimization.* PhD thesis, Massachusetts Institute of Technology, Dept. of Aeronautics and Astronautics.

Simon, H. A. and Newell, A. (1958). Heuristic problem solving: The next advance in operations research. *Operations Research*, 6(1):1–10.

Skiena, S. S. (2008). *The Algorithm Design Manual.* Springer Publishing Company, Incorporated, 2nd edition.

Sørensen, M. and Dahms, F. (2014). A two-stage decomposition of high school timetabling applied to cases in denmark. *Computers & Operations Research*, 43:36–49.

Standley, N. (2023). Nearly a third of university courses still have hybrid teaching. https://www.bbc.com/news/education-64130367. Accessed: 2025-10-30.

Steed, M. (2024). Hybrid schools will soon be the norm across the world. https://www.schoolmanagementplus.com/bursars-finance/hybrid-schools-will-soon-be-the-norm-across-the-world/. Accessed: 2025-10-30.

Stone, A. (2024). Teacher Shortage: Is Hybrid or Remote Teaching the Answer? https://edtechmagazine.com/k12/article/2024/12/teacher-shortage-perfcon. Accessed: 2025-10-30.

Tan, J. S., Goh, S. L., Kendall, G., and Sabar, N. R. (2021). A survey of the state-of-the-art of optimisation methodologies in school timetabling problems. *Expert Systems with Applications*, 165:113943.

UniTime (2023). *Universiy timetabling - Comprehensive academic scheduling solutions.*

Vermuyten, H., Lemmens, S., Marques, I., and Beliën, J. (2016). Developing compact course timetables with optimized student flows. *European Journal of Operational Research*, 251(2):651–661.

Welsh, D. J. A. and Powell, M. B. (1967). An upper bound for the chromatic number of a graph and its application to timetabling problems. *The Computer Journal*, 10(1):85–86.

Williams, H. (1999). *Model building in mathematical programming.* Wiley, New York, 4th ed. edition.

Wolsey, L. A. and Nemhauser, G. L. (2014). *Integer and Combinatorial Optimization.* Wiley Series in Discrete Mathematics and Optimization. Wiley, Hoboken, 1st ed. edition.

Wright, M. (2001). Subcost-guided search—experiments with timetabling problems. *Journal of Heuristics*, 7(3):251–260.

Yu, P. L. (1991). Habitual domains. *Operations Research*, 39(6):869–876.

Zitzler, E. and Thiele, L. (1998). Multiobjective optimization using evolutionary algorithms - a comparative case study. In Eiben, A. E., Bäck, T., Schoenauer, M., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature - PPSN V*, pages 292–301, Berlin, Heidelberg. Springer Berlin Heidelberg.