Copyright Statement

The digital copy of this thesis is protected by the Copyright Act 1994 (New Zealand).

This thesis may be consulted by you, provided you comply with the provisions of the Act and the following conditions of use:

- Any use you make of these documents or images must be for research or private study purposes only, and you may not make them available to any other person.
- Authors control the copyright of their thesis. You will recognize the author's right to be identified as the author of this thesis, and due acknowledgement will be made to the author where appropriate.
- You will obtain the author's permission before publishing any material from their thesis.

General copyright and disclaimer

In addition to the above conditions, authors give their consent for the digital copy of their work to be used subject to the conditions specified on the Library Thesis Consent Form and Deposit Licence.
Mathematical Programming-based Models and Methods for University Course Timetabling

A thesis submitted in fulfilment of the requirements for the degree of
Doctor of Philosophy in Operations Research

2015

Author: Antony E. Phillips

Supervised By: Prof. David M. Ryan
Prof. Matthias Ehrgott
Assoc. Prof. Cameron G. Walker

Department of Engineering Science
Faculty of Engineering
University of Auckland
University course timetabling is a large resource allocation problem faced by thousands of academic institutions worldwide. However, due to the size and complexity of modern universities, even finding a feasible timetable can be a computationally difficult problem. After decades of research, the development of algorithms for automated timetabling remains as an active research field, predominantly using metaheuristics.

In this thesis we develop new mathematical programming-based models and methods for university course timetabling problems. To address large practical problems, course timetabling can be decomposed into a time assignment followed by a room assignment.

For the time assignment problem, we define an integer programme which assigns each event (e.g. a lecture) to a time period in such a way that no conflicts occur. When generating a time assignment, we also consider how to ensure that a feasible room assignment will exist. This is shown to be complex for practical problems which feature an arbitrary set of room types.

For the room assignment problem, we define a set packing integer programme within a lexicographic optimisation process, to consider several quality measures. An analysis of the matrix structure for this model yields insights into the practical
difficulty of addressing different room assignment quality measures.

We also study the minimal perturbation problem, where infeasibility in an existing timetable must be repaired with the least possible disruption. We define a heuristic which uses an integer programme to resolve infeasibility within an expanding neighbourhood of possible perturbations. In practice, this algorithm can be applied in the construction and maintenance of a timetable.

The results of our methods are promising. In addition to validating against established benchmarks, we demonstrate the ability to generate high quality solutions for practical problems at the University of Auckland and the Technical University of Denmark.

The applications of this work are extended by studying the analytics capability of our timetabling algorithms, which further leverage the use of exact optimisation. In addition to conducting scenario analyses, our algorithms are used to explore the quality tradeoffs inherent to the timetabling process. The provision of analytics to timetablers has significant potential for future research in this field.
Acknowledgements

First and foremost, I would like to express sincere gratitude to my main supervisor, Professor David Ryan. From the original conception of this project to the present day, his continual enthusiasm for our research and faith in my ability has served as the strongest motivation. David, I have benefitted extraordinarily from working with you.

I also greatly appreciate the numerous contributions of my co-supervisors, Professor Matthias Ehrgott and Associate Professor Cameron Walker, throughout the stages of this research. I look forward to working with you both again in the future.

Many other academic staff have also supported this work. Dr. Hamish Waterer from the University of Newcastle has played a key role in the initial stages, and provided guidance in publishing my first journal article. The entire operations research group in the Department of Engineering Science has provided significant encouragement throughout this project. In particular, I acknowledge the input of Associate Professor Andrew Mason, Dr. Michael O’Sullivan and Dr. Andrea Raith. Finally, I thank Professor Jesper Larsen and the Management Science group at DTU in Denmark, who showed exceptional hospitality in twice hosting me as part of OptALI research collaboration visits.

I would like to thank Darren Woodward and Jin Koo Niersbach from the Timetable
Services Office at the University of Auckland for their assistance with acquiring and interpreting the timetabling data.

To my friends at university, in particular Hamid Abbasi, Simon Bull, Guillermo Cabrera, Imran Ishrat, Ethan Liu, Jon Pearce, Olga Perederieieva and Oddo Zhang, I have very much enjoyed working alongside such an interesting and affable group of people.

To my friends in the outside world, in particular Brad Hadley, Michael Hoksbergen, Joel Kranz, Vignesh Kumar, Alex MacDonald, Amar Mehta, John Newland, Gavin Ryan and Aidan Thorp, each of you gentlemen deserve many beans.

To my family, in particular my parents Alison and Paul Phillips, you have been exceptionally supportive of all my academic endeavours, which have culminated in this work. This thesis is dedicated to you.

Finally, to Vanessa Williams, your support and companionship throughout this project has made a world of difference.
Contents

List of Terminology xv
List of Notation xvii

1 Introduction 1
1.1 Motivation ................................................. 1
1.2 Methodology .............................................. 4
1.3 Thesis Outline ............................................ 7
1.4 Contribution ............................................... 8

2 University Course Timetabling 11
2.1 University Course Timetabling Formulations ..................... 12
2.2 Quality in Course Timetabling ................................ 19
2.3 Literature Review .......................................... 22
  2.3.1 Previous Surveys ....................................... 24
  2.3.2 Heuristic-based Approaches ............................ 27
  2.3.3 Optimisation-based Approaches ......................... 32
2.4 Discussion .................................................. 41

3 Course Timetabling Datasets 47
3.1 The University of Auckland ................................. 47
  3.1.1 Background ............................................ 48
  3.1.3 2011 – 2013 Timetable Construction Process .......... 53
  3.1.4 Discussion .............................................. 56
3.2 The Technical University of Denmark .......................... 57
  3.2.1 Background ............................................ 57
  3.2.2 2011 – 2012 Timetable Construction Process .......... 58
  3.2.3 Discussion .............................................. 60
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3</td>
<td>University of Udine Benchmarks</td>
<td>61</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Background</td>
<td>62</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Discussion</td>
<td>65</td>
</tr>
<tr>
<td>4</td>
<td>The Time Assignment Problem</td>
<td>69</td>
</tr>
<tr>
<td>4.1</td>
<td>Introduction</td>
<td>70</td>
</tr>
<tr>
<td>4.2</td>
<td>Room Assignment Feasibility</td>
<td>73</td>
</tr>
<tr>
<td>4.3</td>
<td>Mathematical Model</td>
<td>77</td>
</tr>
<tr>
<td>4.4</td>
<td>Results for UoA 2013</td>
<td>79</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Time Assignment Quality</td>
<td>79</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Room Assignment Feasibility</td>
<td>82</td>
</tr>
<tr>
<td>4.5</td>
<td>Discussion</td>
<td>84</td>
</tr>
<tr>
<td>5</td>
<td>The Classroom Assignment Problem</td>
<td>87</td>
</tr>
<tr>
<td>5.1</td>
<td>Introduction</td>
<td>88</td>
</tr>
<tr>
<td>5.2</td>
<td>Mathematical Model</td>
<td>93</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Example Problem</td>
<td>94</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Notation</td>
<td>96</td>
</tr>
<tr>
<td>5.2.3</td>
<td>Integer Programming Formulation</td>
<td>98</td>
</tr>
<tr>
<td>5.2.4</td>
<td>Measures of Solution Quality</td>
<td>100</td>
</tr>
<tr>
<td>5.3</td>
<td>Computational Difficulty</td>
<td>103</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Event-based Problems</td>
<td>104</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Event-based Problems with Contiguous Room Stability</td>
<td>104</td>
</tr>
<tr>
<td>5.3.3</td>
<td>Pattern-based Problems</td>
<td>109</td>
</tr>
<tr>
<td>5.3.4</td>
<td>Lexicographic Optimisation Constraints</td>
<td>113</td>
</tr>
<tr>
<td>5.4</td>
<td>Results for UoA Problems</td>
<td>114</td>
</tr>
<tr>
<td>5.4.1</td>
<td>UoA 2010</td>
<td>114</td>
</tr>
<tr>
<td>5.4.2</td>
<td>UoA 2013</td>
<td>116</td>
</tr>
<tr>
<td>5.4.3</td>
<td>Discussion</td>
<td>119</td>
</tr>
<tr>
<td>5.5</td>
<td>Results for DTU Problems</td>
<td>119</td>
</tr>
<tr>
<td>5.5.1</td>
<td>Adaptation of the Model</td>
<td>120</td>
</tr>
<tr>
<td>5.5.2</td>
<td>DTU 2011–2012</td>
<td>121</td>
</tr>
<tr>
<td>5.5.3</td>
<td>Discussion</td>
<td>123</td>
</tr>
<tr>
<td>5.6</td>
<td>Results for Udine Benchmarks</td>
<td>125</td>
</tr>
<tr>
<td>5.6.1</td>
<td>Udine UD2</td>
<td>125</td>
</tr>
</tbody>
</table>
## 6 Minimal Perturbation Problems

6.1 Introduction .................................................. 136
6.2 Expanding Neighbourhood Algorithm .......................... 140
6.3 Event-based Neighbourhood Model ............................ 143
6.4 Course-based Neighbourhood Model ........................... 146
6.5 Defining the Neighbourhood .................................. 147
   6.5.1 Neighbourhood Time Periods ............................ 148
   6.5.2 Neighbourhood Rooms .................................. 150
6.6 Results for Construction Phase ............................... 153
   6.6.1 UoA 2010 ................................................. 154
   6.6.2 UoA 2013 ................................................. 158
6.7 Results for Enrolment Phase ................................. 160
   6.7.1 UoA 2010 Over-enrolment ............................... 160
   6.7.2 UoA 2013 Loss of Room Availability .................... 164
6.8 Discussion .................................................... 166

## 7 Timetabling Analytics

7.1 Introduction .................................................. 172
7.2 Benchmarking for UoA 2013 ................................. 173
   7.2.1 Solution Comparison .................................... 173
   7.2.2 Data Issues ............................................ 176
7.3 Scenario Analysis ............................................. 177
   7.3.1 Mid-term Planning ...................................... 178
   7.3.2 Long-term Planning ..................................... 179
7.4 Multi-Objective Analysis .................................... 181
   7.4.1 Lexicographic Optimisation ............................ 181
   7.4.2 Max-ordering Optimisation ............................. 181
   7.4.3 Weighted Sum Scalarisation ............................ 184

## 8 Conclusion

8.1 Key Results .................................................. 191
8.2 Future Work .................................................. 194
8.3 Discussion .................................................... 196
List of Figures

2.1 University Course Timetabling as part of Scheduling 12
3.1 2010 UoA Timetabling Process 54
3.2 DTU Campus Size 62
4.1 Room and Event Size Hierarchy 75
4.2 Room and Event Size and Attribute Set Hierarchy 76
5.1 Formulations for Event-Based Room Assignment 91
5.2 Room Assignment with Contiguous Room Stability 92
5.3 Room Assignment with Course Room Stability 93
5.4 Robustness Function 104
5.5 Set Packing Constraint Matrix 106
5.6 Independent Set Formulation of Example 2 107
5.7 Set Packing Partial Constraint Matrix 110
5.8 Independent Set Formulation of Example 3 111
5.9 UD2 Solve Time vs Room Utilisation and Connectedness 130
7.1 Complete Course Timetabling System 173
7.2 Supported Nondominated Points (SR vs RP) UoA 2013 186
7.3 Supported Nondominated Points (SR vs RS) UoA 2013 188
List of Tables

2.1 Quality Measures in University Course Timetabling ............... 22
2.2 Differentiating Factors in Timetabling Literature .................. 24

3.1 UoA 2010 Dataset Problem Size .............................. 50
3.2 Unpaired vs Paired Courses .................................... 52
3.3 Time Stability vs Time Equity .................................. 52
3.4 UoA 2013 Dataset Problem Size .............................. 55
3.5 Udine Benchmark Dataset Characteristics ......................... 64

4.1 Limitations of Strict Time Stability .......................... 72
4.2 Limitations of Strict Time Stability (reduced day gaps) ........ 73
4.3 UoA 2013 Basic Time Assignment Results .................... 81
4.4 UoA 2013 Full Time Assignment Results ..................... 83

5.1 Course and Event Data ......................................... 95
5.2 Room Data ....................................................... 95
5.3 Processed Problem Data ....................................... 96
5.4 A Feasible Solution ............................................ 96
5.5 Time Periods and Feasible Rooms ............................ 105
5.6 Time Periods and Feasible Rooms ............................ 110
5.7 UoA 2010 Starting Room Assignment Results .................. 115
5.8 UoA 2010 Final Room Assignment Results .................... 116
5.9 UoA 2013 Starting Room Assignment Results .................. 117
5.10 UoA 2013 Timetable Room Assignment Results ............... 118
5.11 DTU Room Assignment Results .............................. 122
5.12 DTU Room Assignment Quality Comparison .................... 123
5.13 Results on Müller’s UD2 (ITC2007) Time Assignments .......... 127
5.14 Results on Shaerf’s UD5 Time Assignments .................... 132
6.1 Course-based Perturbation Example .......................... 150
6.2 UoA 2010 Event-based Timetable Construction Results .......... 156
6.3 UoA 2010 Course-based Timetable Construction Results ........ 157
6.4 UoA 2013 Event-based Timetable Construction Results .......... 159
6.5 UoA 2013 Time Assignment Disruption .......................... 160
6.6 Scenario Changes to Course Enrolments .......................... 161
6.7 Event-based Solution ............................................. 162
6.8 Course-based Solution ............................................. 164
6.9 UoA 2013 Loss of Room Availability Results ..................... 165

7.1 UoA 2013 Solution Quality Comparison .......................... 174
7.2 UoA 2013 Course Room Stability Fairness ........................ 183
<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Course</td>
<td>A set of related events with a common group of students and teacher</td>
</tr>
<tr>
<td>Curriculum</td>
<td>A set of courses in which a common group of students are enrolled</td>
</tr>
<tr>
<td>Event</td>
<td>A meeting between staff and students (usually a lecture, lab or tutorial), which requires the use of one room for the duration of one time period</td>
</tr>
<tr>
<td>Long Event</td>
<td>A subset of two or more events for a given course, which must be taught in contiguous time periods</td>
</tr>
<tr>
<td>Pattern</td>
<td>A subset of events for a given course which are assigned to be taught in the same room</td>
</tr>
<tr>
<td>Room/Classroom</td>
<td>A physical location which is available for teaching in at least one time period</td>
</tr>
<tr>
<td>Section</td>
<td>An event which is taught concurrently with other events (also sections) from the same course, each in a different room</td>
</tr>
<tr>
<td>Session (DTU)</td>
<td>A block of either two morning time periods (8am–10am and 10am–12pm) or two afternoon time periods (1pm–3pm and 3pm–5pm)</td>
</tr>
<tr>
<td>Teacher</td>
<td>A staff member involved with teaching one or more events</td>
</tr>
<tr>
<td>Time period</td>
<td>The allotment of time prescribed for each event, most commonly one hour</td>
</tr>
</tbody>
</table>
List of Notation

\(E\)  all events
\(E_{carr}\)  events of courses in curriculum \(carr\)
\(E_F\)  events which are the first of a long event, or not part of a long event
\(E_N\)  events which are in neighbourhood \(N\)
\(E_{teac}\)  events which are taught by teacher \(teac\)
\(E_{rmgp}\)  events which are associated with room group \(rmgp\)
\(E_{tr}\)  events suitable for assignment to time period \(t\) and room \(r\)
\(e - 1\)  the event preceding \(e\) in a long event
\(P\)  all patterns
\(\mathcal{P}\)  a subset of all patterns
\(P_c\)  patterns of events from course \(c\)
\(P_e\)  patterns which include event \(e\)
\(P_{rt}\)  patterns which include events suitable for room \(r\) in time period \(t\)
\(C\)  all courses
\(C_N\)  courses which include an event in \(E_N\)
\(C_{stab}\)  courses which request time stability for their events
\(c_e\)  the course of which event \(e\) is part
\(CU\)  all curricula
\(\mathcal{CU}\)  a subset of all curricula
\(CU_c\)  curricula which include the course of event \(e\)
\(CU_N\)  curricula which include a course with an event in \(E_N\)
\(TE\)  all teachers
\(T\)  all time periods
\(T_e\)  time periods suitable for event \(e\)
\(T_p\)  time periods assigned to events of pattern \(p\)
\(T_R\)  time periods for which room \(r\) is available
\(T_N\)  time periods which are in neighbourhood \(N\)
\(T_d\)  time periods on day \(d\)
\(t_e\)  the time period assigned to event \(e\)
\(t - 1\)  the time period preceding \(t\) on the same day
\(D\)  all days of the timetabling domain
\( D_N \)  
all hours of the day  
\( H \)  
hours of neighbourhood time periods  
\( h_t \)  
the hour of time period \( t \)  
\( R \)  
all rooms  
\( R_p \)  
rooms suitable for events in pattern \( p \)  
\( R_c \)  
rooms suitable for events of course \( c \)  
\( R_t \)  
rooms available in time period \( t \)  
\( R_N \)  
rooms which are in neighbourhood \( N \)  
\( R_{bldg} \)  
rooms situated within building \( bldg \)  
\( R_{rmgp} \)  
rooms in room feasibility group \( rmgp \)  
\( R_{et} \)  
rooms suitable for event \( e \) and available in time period \( t \)  
\( BLDG \)  
all buildings  
\( RMGP \)  
all room feasibility groups  
\( A \)  
all room attributes  
\( att_r \)  
attributes of room \( r \)  
\( att_c \)  
attributes required by events of course \( c \)  
\( size_r \)  
maximum student capacity of room \( r \)  
\( size_c \)  
expected number of students in events of course \( c \)  
\( length_c \)  
number of events in course \( c \)  
\( length_p \)  
number of events in pattern \( p \)  
\( Pref(c, t) \)  
preference for an event of course \( c \) to be taught in time period \( t \)  
\( Pref(c, r) \)  
preference for an event of course \( c \) to be taught in room \( r \)  
\( Suit(c, r) \)  
suitability for an event of course \( c \) to be taught in room \( r \)  
\( x_{et} \)  
boolean variable for whether event \( e \) is held in time period \( t \)  
\( x_{etr} \)  
boolean variable for whether event \( e \) is held in time period \( t \) and room \( r \)  
\( x_{pr} \)  
boolean variable for whether all events of pattern \( p \) are held in room \( r \)  
\( y_{ch} \)  
boolean variable for whether any event of course \( c \) is held in hour \( h \)  
\( h_{\text{earliest}} \)  
integer variable for the earliest hour of the day used by curriculum \( curr \) across all weekly events  
\( h_{\text{latest}} \)  
integer variable for the latest hour of the day used by curriculum \( curr \) across all weekly events
This form is to accompany the submission of any PhD that contains research reported in published or unpublished co-authored work. **Please include one copy of this form for each co-authored work.** Completed forms should be included in all copies of your thesis submitted for examination and library deposit (including digital deposit), following your thesis Acknowledgements.

Please indicate the chapter/section/pages of this thesis that are extracted from a co-authored work and give the title and publication details or details of submission of the co-authored work.

**Chapter 5**

**Published Journal Article:**


**Nature of contribution by PhD candidate:** Writing, implementation of all software, adaptation of integer programming model, comprehensive theoretical analysis, generation of results for all problems (UoA 2010, UoA 2013, UoD UoD, UoD UoP, DTU), discussion

**Extent of contribution by PhD candidate (%)**

<table>
<thead>
<tr>
<th>CO-AUTHORS</th>
<th>Nature of Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dr. Hamish Waterer</td>
<td>Initial written report defining integer programme, Editing</td>
</tr>
<tr>
<td>Professor Matthias Ehrigott</td>
<td>Research supervision, Editing</td>
</tr>
<tr>
<td>Professor David Ryan</td>
<td>Research supervision, Editing</td>
</tr>
</tbody>
</table>

**Certification by Co-Authors**

The undersigned hereby certify that:

- the above statement correctly reflects the nature and extent of the PhD candidate’s contribution to this work, and the nature of the contribution of each of the co-authors; and
- in cases where the PhD candidate was the lead author of the work that the candidate wrote the text.

<table>
<thead>
<tr>
<th>Name</th>
<th>Signature</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antony Phillips</td>
<td>[Signature]</td>
<td>27/05/15</td>
</tr>
<tr>
<td>Dr. Hamish Waterer</td>
<td>[Signature]</td>
<td>27/05/15</td>
</tr>
<tr>
<td>Professor Matthias Ehrigott</td>
<td>[Signature]</td>
<td>26/5/15</td>
</tr>
<tr>
<td>Professor David Ryan</td>
<td>[Signature]</td>
<td>26/5/15</td>
</tr>
</tbody>
</table>

Last updated: 25 March 2013
# Co-Authorship Form

This form is to accompany the submission of any PhD that contains research reported in published or unpublished co-authored work. Please include one copy of this form for each co-authored work. Completed forms should be included in all copies of your thesis submitted for examination and library deposit (including digital deposit), following your thesis Acknowledgements.

Please indicate the chapter/section/pages of this thesis that are extracted from a co-authored work and give the title and publication details or details of submission of the co-authored work.

Chapter 6

Published Refereed Conference Proceedings:

Submitted Journal Article:

| Nature of contribution by PhD candidate | Writing, implementation of all software, definition of integer programming model, development of neighbourhood algorithm, generation of results for all problems (UA 2010, UoA 2013), discussion |
| Extent of contribution by PhD candidate (%) | 95 |

## Co-Authors

<table>
<thead>
<tr>
<th>Name</th>
<th>Nature of Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Associate Professor Cameron Walker</td>
<td>Research supervision, Editing</td>
</tr>
<tr>
<td>Professor Matthias Ehrhart</td>
<td>Research supervision, Editing</td>
</tr>
<tr>
<td>Professor David Ryan</td>
<td>Initial concept, Research supervision, Editing</td>
</tr>
</tbody>
</table>

## Certification by Co-Authors

The undersigned hereby certify that:

- the above statement correctly reflects the nature and extent of the PhD candidate’s contribution to this work, and the nature of the contribution of each of the co-authors; and
- in cases where the PhD candidate was the lead author of the work that the candidate wrote the text.

<table>
<thead>
<tr>
<th>Name</th>
<th>Signature</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antony Phillips</td>
<td></td>
<td>27/05/15</td>
</tr>
<tr>
<td>Associate Professor Cameron Walker</td>
<td></td>
<td>28/05/15</td>
</tr>
<tr>
<td>Professor Matthias Ehrhart</td>
<td></td>
<td>28/05/15</td>
</tr>
<tr>
<td>Professor David Ryan</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Last updated: 25 March 2013
1.1 Motivation

University course timetabling is a large resource allocation problem faced by thousands of academic institutions worldwide. Due to the significant investment required to employ staff and provide teaching facilities, these resources should be well-utilised. The timetable can be considered to be the production plan of a university, as it specifies the use of teaching facilities for thousands of students and staff.

Generating a timetable requires the consideration and understanding of many components within the complex and diverse university system. Broadly, this includes the interaction between students, teaching staff, courses, faculties, available time periods and rooms. An advanced timetabling system should be able to find a high quality solution within a relatively short timeframe, and may even include sophisticated analytics capabilities.

Modern university course timetabling is classified as an “NP-hard” optimisation problem, which means no known algorithm or process is guaranteed to find the “best” timetable solution within a reasonable timeframe. As a result, practi-
1.1. MOTIVATION

cal timetabling is an active research field where a diverse range of proposed algorithms vary substantially in performance (i.e. the quality of solution for a given solve time). The most sophisticated methods can be either heuristic or optimisation-based (e.g. mathematical programming), and utilise recent advances in algorithmics and modern computational power.

However, despite significant progress into the development of algorithms for automated timetabling, simple rules-based or manual processes are commonly used in practice. Manually deciding the time and room assignment for thousands of weekly events (e.g. lectures) is an extremely difficult task. Ultimately, it is not possible for the timetabler to consider the effect of each individual decision on the overall timetable. This may result in it being difficult to even find a feasible timetable, let alone one which adheres to measures of quality and utilisation of space. Such systems also require a substantial amount of time and effort from timetabling administrators and faculty representatives (often academic staff) to generate a timetable.

The specific definition of timetable quality is known to vary significantly between universities, and between different stakeholders at the same university. For a given university, we can consider the concept of quality as it relates to the priorities of the students, teaching staff and administrators. For students, this includes assigning events to rooms which are near to the respective faculty buildings, and consideration of the load or structure of an individuals daily timetable. To enhance student choice, a sometimes-overlooked quality consideration relates to teaching certain groups of courses (e.g. electives) at non-overlapping times.

A practical example of the importance of student choice is reported by van den Broek et al. (2009). In this case, an antiquated manual timetabling process was unable to find a satisfactory solution for students, after the size and complexity of the problem had increased year-after-year. The problem was resolved after a new sophisticated algorithmic approach was developed, which was able to optimise the
time assignment to maximise student choice.

For teaching staff, many quality considerations are shared with those of students. However, an additional consideration may involve requests to teach in particular rooms, or in particular time periods. For example, some staff develop an affinity for specific rooms, or would prefer their teaching events to be clustered together. Schimmelpfeng and Helber (2007) detail the implementation of a practical timetabling system where staff requests are particularly important. In this case, the implemented algorithm was able to exceed staff expectations by honouring nearly all such requests.

Finally, we consider quality as it relates to the interests of university administrators. This can include achieving a high utilisation of teaching spaces, leaving under-utilised rooms available to be re-purposed. Fizzano and Swanson (2000) develop a practical algorithm to find a feasible timetable which uses as few rooms as possible.

In addition to the quality of the timetabling solution, a timetabling system may be assessed based on the number of working days required for timetablers and faculty to construct a feasible timetable. This is mentioned by Stallaert (1997) where administrators were particularly impressed with how much time and effort were saved relative to finding a comparable solution manually.

Another major benefit of an automated timetabling system is the capability for analytics. This includes providing an understanding of the bottlenecks in the university system, and the ability to conduct scenario analysis for administrative planning. Schimmelpfeng and Helber (2007) describe their timetabling algorithm as an “extremely powerful instrument to analyze the use of resources in the timetabling process”.

Carter (2001) provides a particularly positive account of all the mentioned benefits from implementing a new timetabling system. This system simultaneously
provided for substantially improved solutions, a reduction in time spent to generate a solution, and a useful analytics tool.

In summary, despite the technical challenges of designing and implementing a sophisticated timetabling system, there are substantial benefits for all stakeholders in the university. This includes intangible improvements in the satisfaction of staff and students, and the financial benefits of a reduced timetabling cost and improved decision making.

1.2 Methodology

The focus of this research is to develop a set of algorithms which can be used for complex course timetabling. The aim is to find a high quality feasible solution within a reasonable timeframe, to fit into a practical timetabling process. This means the method must scale efficiently to the size of large modern universities, and be versatile in modelling capability. We aim to utilise the techniques of mathematical programming-based methods and demonstrate their applicability in this domain.

As our primary motivating example, we address the timetabling problem at the University of Auckland. This problem involves over 33000 equivalent full-time students, and nearly 5000 equivalent full-time staff (The University of Auckland, 2015), which is classified as “extra large” by international standards (QS Intelligence Unit, 2015).

Due to the size and complexity of modern timetabling problems, finding the “optimal” timetable solution is generally considered to be impractical. Moreover, due to the dynamic nature of timetabling data and the ambiguity in defining quality, the concept of “optimality” is not necessarily meaningful.

We aim to find a solution which is feasible (i.e. satisfies all the operational constraints of the university), and which is of a high quality (i.e. meets preferences of
students, staff and administrators). This is achieved by using mathematical programming models within a broader heuristic framework. Specifically, the course timetabling problem is decomposed into the subproblems of time assignment, room assignment, and minimal perturbation. The overall methodology can be considered to be heuristic in nature, because the subproblems are each solved as self-contained components. However, within each of these subproblems, tractable-sized integer programming models are formulated and solved. Our approach is designed to leverage the advantage of exact methods, such as a certainty of the feasibility (or infeasibility) of each model.

In addition to improving the computational tractability, we note two additional benefits to using a modular approach for course timetabling. Firstly, we are able to customise the method used for each subproblem, based on the unique properties and priorities for this type of problem. We also investigate the computational properties of each particular problem to understand in what circumstances they will be easy or difficult.

Secondly, a modular approach makes our method applicable to a diverse range of timetabling problems. For example, the timetabling problem at the Technical University of Denmark only requires that we solve a room assignment problem, for a given fixed time assignment.

In contrast to our methodology, much of the timetabling literature proposes a single algorithm (typically a metaheuristic) to tackle the time and room assignment simultaneously. Furthermore, literature surveys have noted that very few timetabling algorithms are developed in a practical context, and more focus is needed on the real-world complications of implementing a timetabling system. For the timetabling studies which are conducted in a practical context (see Section 2.3), few deal with a very large university, and instead solve the timetabling problem for a single department.
The involvement of both faculty representatives and timetabling staff is essential to our methodology. The ability to find a high quality timetable is clearly dependent on input from faculties in terms of quality preferences. The most desirable situation is for faculties to compose an ideal time assignment for their courses (without consideration of rooms), which allows them to express any depth of complexity into their particular requirements. This provides timetablers with a high quality starting point, although modification will be required to find room feasibility. In implementing their timetabling system, Schimmelpfeng and Helber (2007) noted that staff requested to teach in time periods similar to what had “worked” in previous years. Their system was able to meet these preferences, as well as other quality measures, and generate better schedules than ever used before.

The timetabling staff remain in the role of facilitating the collection of high-integrity data, and using timetabling software with their judgement (based on knowledge about the university) to implement the best possible solution. The algorithms in this thesis provide powerful tools for this process, where it is possible to consider and evaluate the effects of various high-level decisions, such as the different measurements of quality.

Timetabling staff also have an ongoing role to maintain the timetable after it has been constructed and published. In addition to managing the ad-hoc room bookings, timetablers must address unexpected changes to room or staff availability throughout the semester. Such changes are considered inevitable in a practical university environment. We provide an advanced algorithm which enables timetablers to assess and implement a set of perturbations to restore feasibility while minimising disruption.

Finally, although we develop a complete set of algorithms for timetabling, we note that every timetabling system requires ongoing development as university requirements change, and techniques are refined. Carter (2001) reports that 40 person-years
of development were spent in development of their timetabling system (including the auxiliary components of user interfaces and databases), which was conducted while receiving feedback and making continuous improvements.

1.3 Thesis Outline

This thesis is structured as follows.

In Chapter 2 we provide a comprehensive overview of university timetabling in theory and practice. We outline the role of each timetabling subproblem, and address the concept of timetabling quality, including a detailed list of potential quality measures. Based on these definitions, we conduct a comprehensive literature review of automated timetabling algorithms. A discussion of this literature provides the context for the scope and significance of our work.

In Chapter 3 we introduce four timetabling datasets which are used to demonstrate our proposed algorithms. The datasets derive from the practical timetabling problems in 2010 and 2013 at the University of Auckland, 2011–2012 at the Technical University of Denmark, and the 2007 International Timetabling Competition. In particular, we focus on the 2013 problem at the University of Auckland as the primary motivating example throughout this thesis.

In Chapter 4 we propose a mathematical programming-based algorithm for finding a time assignment for a set of university courses. We demonstrate the conceptual tradeoffs between different quality measures. We also discuss the difficulty of ensuring a feasible room assignment, which is an important component of the time assignment problem.

In Chapter 5 we propose a mathematical programming-based algorithm for finding a room assignment for a set of university courses. The computational properties of this formulation are explored through an investigation of the matrix properties
of the integer programme. We provide comprehensive results for the application of this model to each dataset, and demonstrate how the method can be adapted for a diverse range of problems.

In Chapter 6 we propose a mathematical programming-based algorithm for repairing an infeasible timetable. We discuss how to define the mathematical models so they are able to resolve infeasibility with a minimal disruption to the remainder of the timetable, while remaining small and fast to solve. We demonstrate the use of this algorithm both for constructing a feasible timetable, and for addressing unexpected changes to the data which also cause infeasibility.

In Chapter 7 we examine the broader application of our proposed algorithms within university timetabling. Extending beyond the construction of a timetable, we consider the analytics capability of automated timetabling algorithms. This includes scenario analysis for administrative planning, modelling timetabling fairness, and exploring the quality tradeoffs inherent to the university system.

In Chapter 8 we conclude by highlighting the most important results of the thesis. The overall contributions of this thesis are discussed in a practical context, and we provide ideas for further work in this field.

1.4 Contribution

We propose techniques based on integer programming, for use on large problems with practical complexities. This ultimately makes each component of our method novel, as existing techniques are based on metaheuristics, or use integer programming for smaller, simplified problems.

We first propose a novel method for solving the time assignment problem. In particular, we investigate the difficulty of ensuring room feasibility when dealing with complex arbitrary sets of room features, which are found in practical problems.
1.4. CONTRIBUTION

We propose a novel method for solving the room assignment problem, which generalises existing methods. We expand upon existing theoretical findings into the computational difficulty of solving room assignment problems, allowing us to identify which quality measures are difficult to solve, and to estimate the difficulty in finding an optimal solution. We demonstrate the application of this method on practical data, as well as data from the 2007 International Timetabling Competition (ITC, 2007). Using the time assignment solutions provided by other researchers, we demonstrate significant improvements which can be made to the solutions within a short solve time. This provides a strong example of how existing metaheuristic processes can benefit by incorporating optimisation methods. This work has been published in Computers & Operations Research (Phillips et al., 2015).

We provide the first use of mathematical optimisation to solve the minimal perturbation problem in dynamic scheduling, and demonstrate its application broadly within the scope of practical timetabling. The importance of repairing an infeasible timetable, as an application of the minimal perturbation problem, has been noted to be an important research area within university timetabling (Burke et al., 2008b). This work has been published in the refereed proceedings of the PATAT (Practice and Theory of Automated Timetabling) conference (Phillips et al., 2014), and in the Annals of Operations Research (Phillips et al., 2016).

We also provide the first use of multi-objective optimisation to explore the trade-offs between multiple measures of quality within university course timetabling. The provision of timetabling analytics is a growing field of interest within the timetabling community.

In summary, although elementary mathematical programming models for university course timetabling were proposed as far back as the 1960s (Schmidt and Ströhlein, 1980), this thesis documents the first use on a large and modern practical problem. We empirically demonstrate the viability of mathematical programming-
based methods for use in practice, and present each algorithm as a generalised method which may be applied to a diverse set of problems. We also demonstrate the application of optimisation algorithms for a broader range of tasks than timetabling construction, such as repairing infeasible timetables, and the significant analytics capability.

To the broader operations research community, we demonstrate the successful integration of heuristic and optimisation methodologies, sometimes referred to as a “matheuristic” (Boschetti et al., 2009). We are able to solve a well-known operations research problem which is relevant to academic institutions worldwide.
University Course Timetabling

Educational timetabling is a broad field, encompassing university course timetabling, school course timetabling, and examination timetabling (Kingston, 2013a). Although there are similarities between these three problems, this chapter specifically addresses university course timetabling due to the wealth and diversity of literature relating to this problem. Educational timetabling itself exists under the umbrella of general timetabling (Burke et al., 2014), and ultimately all scheduling problems (Wren, 1996), as depicted in Figure 2.1. Many concepts and theoretical observations used in educational timetabling are drawn from literature on wider scheduling problems.

The university course timetabling problem should be considered as a practical “problem” in the broadest sense, rather than a specific well-defined operations research problem. A general introduction to the concepts of timetabling, and common methods used to solve such problems, can be found in the early work of de Werra (1985). Over the past 30 years, many other such introductions to university timetabling have been contributed (Bardadym, 1996; Burke et al., 1996, 1997; Carter and Laporte, 1998; Lewis, 2008; MirHassani and Habibi, 2013; Burke
et al., 2014) which are each able to add a valuable perspective.

This chapter provides a comprehensive coverage of the differing formulations for university course timetabling, and reviews the most relevant published literature to date. Section 2.1 gives a general introduction to course timetabling, and outlines the major subproblems which are commonly considered in practice. Most timetabling systems will only solve a subset of these problems. Section 2.2 outlines the most common measures of quality used within the timetabling community for each of the subproblems. Section 2.3 provides a comprehensive literature review, focussing particularly on the use of mathematical programming. Finally, in Section 2.4 we discuss common themes and trends in the literature, to justify and contextualise the direction of research in this thesis.
2.1 University Course Timetabling Formulations

University course timetabling typically involves assigning a time period and a room for each event (e.g., a lecture, lab, or tutorial) of each course. Additionally, it may also include broader decisions about whether to split courses into a number of sections per semester, and how to assign teachers to the courses.

These decisions are often dependent on when the timetable is constructed; either prior to student enrolments, or after student enrolments have taken place. This distinction defines whether a timetabling problem is curriculum-based or (post) enrolment-based (Lewis et al., 2007; Di Gaspero et al., 2007). In the former case, to avoid time-clashes it is necessary to specify groups of courses which are expected to be taken by students simultaneously. A group of such courses is known as a curriculum. In the latter case, any time-clashes between courses can be based on the actual student enrolment in any combination of courses.

The most general form of university course timetabling comprises one or more of the following components (adapted from Carter and Laporte, 1998; Rudová et al., 2011).

**Student Sectioning** Many university courses are taught at most once per semester, where each student attends a prescribed set of weekly events. However, some courses may require several offerings or sections in a given semester. This means students may study the course in any of the independently-taught sections, which are typically held in a different set of time periods. Courses may be split into sections to offer choice to the students, or more commonly, because the total course enrolment exceeds the capacity of the largest available room.

As part of enrolment-based timetabling, the student sectioning (or "student scheduling") problem determines how to assign students to particular sections
of their enrolled courses. Based on students’ course enrolments, the sectioning should ensure clashes are avoided. When this problem is required, it may be conducted as the first timetabling subproblem (Kingston, 2013a), after the time assignment (Müller and Murray, 2010), or as a final stage (Carter, 2001).

This problem also exists within curriculum-based timetabling where large courses must be divided into a number of sections. However, this is a much simpler problem, and is most akin to determining which courses to teach (often conducted manually by faculties). As the times are not yet assigned and there are no student enrolments, there is no consideration of potential clashes.

As noted by Rudová et al. (2011) and Pillay (2014), the student sectioning problem is not addressed by the majority of the timetabling literature.

**Teacher Assignment** This problem determines which teacher or teachers should be assigned for each component of each course. A course will commonly involve more than one teacher, particularly if it comprises a tutorial and/or lab component, in addition to the lecture component.

A teacher (e.g. lecturer, teaching assistant, laboratory technician) will typically teach events for more than one course. This problem is difficult if the time assignment is determined previously or concurrently. To suit a given time assignment, each teacher introduces constraints relating to time-clashes, and the total number of weekly and daily events.

In practice, faculties will often manually determine the teacher assignment for lecturers prior to any time assignments. This is not necessarily difficult, as there are a limited number of academic staff who are capable of lecturing for any given course. In a university setting this is particularly true, as opposed to broader or less-specialised educational timetabling. More flexibility may exist in assigning a teacher to the tutorial or lab components of a course, however,
this is still often determined by the faculty. In this case, considerations of
teacher clashes and workload are addressed in the time assignment.

This problem is also infrequently addressed in the literature, as many timetabling
implementations assume the teacher assignments are provided as an input, and
fixed. This problem is distinct from class-teacher timetabling which arises pre-
dominantly in the context of high-school timetabling.

**Time Assignment (event to time)** This problem determines the weekly time
period for each event, often subject to many complex constraints and ob-
jectives. These can be derived from the requirements of students, faculty
members and the administration. A common constraint relates to courses
which require events to be held in contiguous time periods, such as multi-hour
laboratory or tutorial sessions. We refer to each one-hour block as an event,
and the contiguous events collectively are referred to as a *long* event.

The time assignment problem also needs to consider the placement of events to
avoid conflicts within a curriculum, or within any teacher’s personal timetable.
Constraints may also relate to the daily workload and event pattern for stu-
dents and staff. However, between universities there is substantial variation
in which factors are considered as constraints, and which are considered ob-
jectives. We provide a comprehensive coverage of these factors in Table 2.1.

In the literature, this problem is commonly solved together with the classroom
assignment (see Section 2.3). However, in practice, it is often solved before
the room assignment (“times first, rooms second”), sometimes manually by
faculties. When the time assignment is solved first, additional constraints can
be used to ensure a feasible room assignment will exist for the proposed time
assignment.

**Classroom Assignment (event to room)** This problem determines which room
to use for each event. Each room is only suitable for a subset of the events, as determined by its capacity (or size) and attributes (e.g. demonstration bench). It is required that only one event can occupy a room at any time, which is not the case in examination timetabling. For a long event, it is often required that the same room is used for the entire duration, known as contiguous room stability.

As previously noted, much of the literature solves this problem together with the time assignment. Most other approaches perform the room assignment immediately after the time assignment, although it is possible to assign the rooms before the times (Mirrazavi et al., 2003).

**Minimal Perturbation** This problem arises when an existing timetable contains hard constraint violations, or *infeasibilities*. Instead of re-solving the entire timetabling problem, it is often preferable to perturb the infeasible timetable to restore feasibility. The objective is to resolve all the infeasibilities while minimising the disruption or perturbation to the remainder of the timetable.

This situation is most commonly caused by unexpected changes to the timetabling data after the timetable has been published (e.g. course enrolments, room availability, curricula requirements). However, it can also arise during the timetable construction. If the time assignment is conducted prior to the room assignment, it is possible that no feasible room assignment will exist without perturbing the time assignment.

This problem is particularly under-addressed in the automated timetabling literature. However, many practitioners acknowledge the need to make such perturbations (e.g. Burke et al., 2008b), as dynamic data is an inevitable part of university course timetabling. The minimal perturbation problem is important for curriculum-based timetabling, where the timetable is constructed
prior to enrolment data being available.

It is important to note that all of the subproblems are not necessarily applicable for every university. The problems of student sectioning, teacher assignment and minimal perturbation are commonly referred to in the literature, but less frequently addressed. Much of the literature, including that which addresses the international timetabling competitions, develops algorithms exclusively for the time and room assignment subproblems. These can be considered as the “core” subproblems of university course timetabling, and “course timetabling” is often used to refer specifically to these two subproblems.

The disproportionate levels of attention are likely to be because the time and room assignment problems are universally required, and the difficulty of the problems makes them suitable for the use of automation. By contrast, the difficulty and nature of the sectioning and teacher assignment problems are strongly dependent on a given university. For example, student sectioning is a difficult problem in some universities where the majority of courses have multiple sections. In other universities, very few (or even none) of the courses are taught in multiple sections. At the University of Auckland, each faculty only requires sectioning for a small number of (typically) entry-level courses. Similarly with teacher assignment, the majority of the literature considers the teachers as pre-assigned. As a result, algorithms for time and room assignment are more generalisable. The minimal perturbation problem is also generalisable from an algorithm standpoint, although comparison of results is extremely difficult without testing on a common dataset.

The distinction between curriculum-based timetabling and enrolment-based timetabling is made frequently in the literature. Although a particular subproblem can be either solved based on curricula or enrolment data, this does not imply that the timetabling process for a particular university uses only one of these types of timetabling. In many practical situations, both styles of timetabling will be required (Kingston,
Timetabling based on curricula is often a necessity, as the timetable construction process may take several months (including all data collection, processing, iterations, approvals, and publishing). Throughout this period, estimates are used for the number of students enrolling in each course, and for which courses need to be kept clash-free (i.e. those within a curriculum). Curricula data may be derived from faculty estimates of courses commonly taken together, and investigation of other less obvious groups of courses which have historically been taken together (e.g. by conjoint students, electives, split-part students). It would also be undesirable to require students to commit to their enrolment prior to knowing which weekly times each course will require, particularly for part-time students or others with outside commitments. At the University of Auckland, final student enrolment figures are only available once enrolment closes, which is two weeks after the start of semester. This type of timetabling is also referred to as “Master Timetabling” by Carter and Laporte (1998).

Timetabling based on enrolments may be used to construct a timetable, although typically only for a small or simple problem. In particular, if a university offers a small number of courses, or if timetabling is logically separable between departments (i.e. no shared facilities or split-students), and students are satisfied to enrol in advance. Depending on the number of students, it may even be possible to use student preferences in the timetabling process. However, for most large universities, enrolment-based timetabling is more limited in applicability. We address the use of enrolment-based timetabling in the minimal perturbation problem, where unexpected changes (usually related to the enrolment) have caused the existing timetable to become infeasible. In this case, we can consider actual enrolments, as curricula are no longer as important. This transition removes the constraints of honouring any curricula with no enrolled students, but it also adds the limitations of affecting
students enrolled in course combinations which were originally not part of any curriculum. If course conflicts are unavoidable, the severity can be weighted based on the number of students enrolled. However, there may be an additional consideration for which type of courses are affected. For example, it may be more acceptable to displace students enrolled in an elective course (which they could substitute), or a course which is offered more than once per year. This type of timetabling is also referred to as a “Demand-Driven” approach in Carter and Laporte (1998). Müller and Rudová (2014) argue that complex relationships between curricula, such as multiple sections of a course or compulsory and optional courses in a curriculum, should be represented with virtual enrolments and treated as an enrolment-based problem.

In this thesis, we focus on the most common timetabling case. It is usually necessary to use curriculum-based timetabling to construct a timetable long before semester commences, and then to make as few modifications at enrolment time as possible.

2.2 Quality in Course Timetabling

The core constraints (or “hard constraints”) for each of the five subproblems of university course timetabling, as presented in Chapter 2.1, tend to be relatively ubiquitous between different implementations in different universities.

However, the way quality within timetabling is prioritised and measured can differ substantially. A given quality measure such as room stability in classroom assignment (all events of a course are held in the same room) can be modelled as critically important i.e. as a hard constraint (e.g. Carter and Tovey, 1992), as an objective to be maximised (e.g. Di Gaspero et al., 2007), or unimportant (e.g. Schimmelpfeng and Helber, 2007).

Differing measurements of quality are partly a result of cultural differences in the
2.2. QUALITY IN COURSE TIMETABLING

way timetabling is viewed globally. Although, even within a single institution, there are multiple stakeholders with a unique set of interests in timetabling e.g. students, staff and administrators.

In Table 2.1 we present a comprehensive listing of quality measures which have appeared in course timetabling literature, and several which we introduce. The name of each quality measure indicates whether it applies to individual events, or entire courses, curricula, staff etc. Several of the quality measures use the term “soft” for curricula, preassignments or room attributes. This generalises the expressions to exist in the context of quality, implying that a feasible timetable may break these (typically hard) constraints by incurring a penalty to quality. Additionally note that many of the quality measures which apply to curricula also apply to staff. This is because a curriculum also represents the group of students who are taking that set of courses.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event: Time assignment</td>
<td>All events should be assigned to a time period</td>
</tr>
<tr>
<td>Event: Room assignment</td>
<td>All events should be assigned to a room</td>
</tr>
<tr>
<td>Event: Daily event time preference</td>
<td>Events should be held in time periods within the core teaching hours</td>
</tr>
<tr>
<td>Event: Daily event spread</td>
<td>Events should be spread throughout the day to manage congestion</td>
</tr>
<tr>
<td>Event: Room size robustness</td>
<td>Events should be held in a room with spare capacity, to allow for</td>
</tr>
<tr>
<td></td>
<td>unpredictability in enrolment</td>
</tr>
<tr>
<td>Event: Vacant room robustness</td>
<td>Events should be held in rooms of similar size, so any spare rooms</td>
</tr>
<tr>
<td></td>
<td>are as large and versatile as possible</td>
</tr>
<tr>
<td>Event: Room size comfort</td>
<td>Events should be held in a room which is not significantly too large</td>
</tr>
<tr>
<td></td>
<td>for the event</td>
</tr>
<tr>
<td>Event: Room soft attributes</td>
<td>Events should be held in a room which has all the requested soft attributes</td>
</tr>
<tr>
<td>Event: Building preference</td>
<td>Events should be held in a room which is close to the associated teaching department</td>
</tr>
<tr>
<td>Event: Soft time preassignments</td>
<td>Events with a preassigned time should be held at this time</td>
</tr>
<tr>
<td>Event: Soft room preassignments</td>
<td>Events with a preassigned room should be held in this room</td>
</tr>
<tr>
<td>Event: Long events room stability</td>
<td>Events which require more than one time period should be held in the same room</td>
</tr>
<tr>
<td>Course: Weekly event spread</td>
<td>Course events should be spread throughout the week</td>
</tr>
<tr>
<td>Course: Time stability</td>
<td>Events of a course on different days should be held at the same time each day</td>
</tr>
<tr>
<td>Course: Room stability</td>
<td>Events of a course should be held in the same room each day</td>
</tr>
<tr>
<td>Curriculum: Daily minimum hours</td>
<td>There should not be too few events from a curriculum in any day</td>
</tr>
<tr>
<td>Curriculum: Daily maximum hours</td>
<td>There should not be too many events from a curriculum in any day</td>
</tr>
<tr>
<td>Curriculum: Daily compactness</td>
<td>Curriculum events should be held within a maximum daily time span</td>
</tr>
<tr>
<td>Curriculum: Daily breaks</td>
<td>Curriculum events should not be taught in long unbroken blocks</td>
</tr>
<tr>
<td>Curriculum: Daily lunch breaks</td>
<td>Curriculum events should have a break around the lunch period</td>
</tr>
<tr>
<td>Curriculum: Spread lunch breaks</td>
<td>Not all curricula should have a lunch break at the same time</td>
</tr>
<tr>
<td>Curriculum: Travel time</td>
<td>Time-adjacent curriculum events should be held in geographically proximate rooms</td>
</tr>
</tbody>
</table>
2.2. QUALITY IN COURSE TIMETABLING

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Curriculum: Soft</td>
<td>The number and severity of time clashes between two courses in a soft curriculum should be minimised</td>
</tr>
<tr>
<td>Staff: Daily</td>
<td>There should not be too few events for a staff member in any day hours</td>
</tr>
<tr>
<td>Staff: Daily</td>
<td>There should not be too many events for a staff member in any day hours</td>
</tr>
<tr>
<td>Staff: Daily</td>
<td>Staff members teaching events should be held within a maximum compactness daily time span</td>
</tr>
<tr>
<td>Staff: Daily</td>
<td>Staff members teaching events should not be taught in long breaks unbroken blocks</td>
</tr>
<tr>
<td>Staff: Daily</td>
<td>Staff members teaching events should have a break around the lunch periods</td>
</tr>
<tr>
<td>Staff: Travel</td>
<td>Time-adjacent staff events should be held in geographically proximate rooms</td>
</tr>
<tr>
<td>Staff: Time</td>
<td>Staff preferences for the teaching time should be met</td>
</tr>
<tr>
<td>Staff: Room</td>
<td>Staff preferences for the teaching room should be met</td>
</tr>
<tr>
<td>Admin: Unused</td>
<td>Events should be held in only a subset of the available rooms to enable re-purposing of surplus rooms</td>
</tr>
<tr>
<td>Admin: Unused</td>
<td>Unused time periods for a room should be in a contiguous block, enabling the rental of the teaching space to a third party</td>
</tr>
<tr>
<td>General: Equity</td>
<td>Quality measures should be optimised equitably across courses, departments and faculties</td>
</tr>
</tbody>
</table>

Table 2.1: Quality Measures in University Course Timetabling

Although Table 2.1 is as comprehensive as possible, it is by no means exhaustive. Quality in timetabling is particularly subjective, and ultimately no single timetabling process can meaningfully consider every possible measure of quality. The most important focus in practical timetabling is to find a feasible solution which satisfies students and staff as much as possible.
In this thesis, we deal with practical timetabling data, and test our models using the specific set of priorities and regulations pertaining to that institution. However, in our description and formulation of the models, we intentionally maintain generality.

2.3 Literature Review

As demonstrated in Sections 2.1 and 2.2, university course timetabling is a broad field of study rather than a specific problem. There are significant variations in the types of problems tackled, and how the problem is formulated, modelled and solved. Course timetabling has also been given significant attention over a long period of time, resulting in a wealth of relevant literature.

However, it has historically been difficult to draw comparisons between different approaches, since not only were the problem sizes different, there was significant variation in which quality measures and constraints were used. From the perspective of benchmarking, the lack of standardisation has been a known issue for some time (Burke et al., 1998; Schaerf, 1999; Schaerf and Di Gaspero, 2007; McCollum, 2007; De Causmaecker and Berghe, 2012). Significant progress in this area is due to the PATAT (Practice and Theory of Automated Timetabling) conferences, and the three international timetabling competitions (ITC, 2002, 2007, 2011). Of particular relevance is track 3 of the second international competition (ITC2007) which focuses on curriculum-based university course timetabling (Di Gaspero et al., 2007). The instances from the ITC2007 remain the most widely used benchmarking instances, which we address in this thesis. See Section 3.3 for more detail on the nature of the problems. A summary on the ITC2007 was produced (McCollum et al., 2010) and a website has been constructed and maintained to provide a publicly accessible listing of all solutions (Bonutti et al., 2012).
Because there is no universal timetabling formulation, our coverage of relevant literature necessarily deals with a broad range of studies with different scopes, foci and methodologies. This will give the reader a solid context for the algorithms we develop in this thesis. To categorise the literature, in Table 2.2 we outline the primary differentiating factors which should be noted in consideration of any paper.

<table>
<thead>
<tr>
<th>Factor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scope of problem</td>
<td>Which subproblems are addressed (see Section 2.1)?</td>
</tr>
<tr>
<td>Quality measures</td>
<td>How is quality measured or defined (see Section 2.2)?</td>
</tr>
<tr>
<td>Methodology</td>
<td>What is the underlying solution methodology used in the proposed algorithm(s)? e.g. metaheuristics, mathematical programming</td>
</tr>
<tr>
<td>Results/Data</td>
<td>What type of data was used to generate the results? e.g. full-scale university, individual department, ITC2007 instances</td>
</tr>
<tr>
<td>Practical Implementation</td>
<td>Was the solution implemented in practice? Are implementation complexities considered or discussed?</td>
</tr>
<tr>
<td>Date</td>
<td>How recent is the approach? This has significant implications for older papers which had relatively limited access to computational resources.</td>
</tr>
</tbody>
</table>

Table 2.2: Differentiating Factors in Timetabling Literature

The nature of this field is such that survey papers are required relatively frequently, to keep pace with the quantity and diversity of research. We provide a comprehensive coverage of the state of research into the field of university course timetabling, with particular emphasis on modern optimisation-based approaches. We also believe several important optimisation-based papers have been overlooked which should be placed in the context of the field.
In Section 2.3.1, we reference and discuss previous surveys of the educational timetabling literature. We proceed to our review of the literature, firstly covering heuristic-based methods in Section 2.3.2, followed by optimisation-based methods in Section 2.3.3. In each category, we list the work, along with the defining features as per Table 2.2. We particularly focus on the last ten years, and the optimisation-based methods, which more directly relate to our intended research. Optimisation-based methods have also been particularly under-represented in surveys to date.

2.3.1 Previous Surveys

Carter and Laporte (1998) conduct the earliest survey into modern course timetabling for universities and schools. They also present a general introduction to the subproblems of course timetabling, and reference papers which apply to each. This generalisation remains useful, to establish later publications on one or more subproblems in the surrounding context. The authors particularly focus on papers which demonstrate the proposed algorithm on real data. Finally, they identify and discuss four general classes of timetabling algorithms: “global algorithms” (e.g. mathematical programming), “constructive heuristics”, “improvement heuristics”, and “interactive systems”. This also remains relevant to the recent literature where one or more of these methodologies is employed. We address global algorithms in Section 2.3.3, and both uses of heuristics in Section 2.3.2.

Schaerf (1999) provides a survey of papers from each of the areas of educational timetabling (see Figure 2.1). In each case, the problem is described initially as a “basic search problem”, where any feasible solution must be found, and then as an “optimisation problem” where quality is considered. The formulation for university course timetabling is concerned with just the core subproblems of time assignment and room assignment. Several references are provided for methods using each of graph theory/ network flow, integer programming, heuristics (e.g. rules, tabu search,
genetic algorithms), and constraint logic programming.

Burke and Petrovic (2002) provide an introduction to the methods for university timetabling developed at the ASAP research group (Automated Scheduling, Optimisation and Planning Group, 2014) at the University of Nottingham, which is a major contributor to the literature in the field. The research strongly emphasises metaheuristics, and covers the incorporation of multiple objectives, and the use of case-based reasoning to partially reuse past solutions. The same authors also publish a chapter (Petrovic and Burke, 2004) in a book on general scheduling (Leung, 2004), which references other methods including those based on mathematical programming. They also expand on their research into heuristic selection (which methods to employ in a given situation), using case-based reasoning, and hyper-heuristics.

Lewis (2008) conducts a survey which focuses on metaheuristic methods for university timetabling. Five core metaheuristic paradigms are identified: Evolutionary Algorithms, Ant Colony Optimisation, Tabu Search, Simulated Annealing and (Iterated) Local Search. Lewis also categorises the literature based on how it deals with satisfying both hard and soft constraints (i.e. feasibility and quality). “One-stage” algorithms handle both groups simultaneously, “two-stage” algorithms operate sequentially (i.e. find feasible solution and then improve), and “relaxation” algorithms omit the most difficult soft constraints initially, and change the search focus as the algorithm progresses. Lewis pays particular attention to the relationship of university timetabling to graph colouring, and how metaheuristics can utilise techniques from graph theory.

An excellent survey is provided by Qu et al. (2009) where they also begin with a review of previous surveys into educational timetabling. Although they focus only on examination timetabling, many of the techniques and practical revelations are ubiquitous throughout all of educational timetabling. Approaches based on graph
2.3. LITERATURE REVIEW

theory, constraint programming, heuristics (local search and population based), and decomposition are listed, in addition to a small number of techniques utilising multi-objective optimisation. Comprehensive numerical results are provided for several benchmark instances in examination timetabling. Of most relevance to wider timetabling are the insightful comments made on the future of research in this field. The authors identify that a greater focus should be made on increasing the generality of methodologies used, and on inter-disciplinary research.

Pillay (2014) surveys research into course timetabling as applied to schools, rather than universities. However, similar to the work of Qu et al. (2009), a particularly diverse set of approaches are outlined, with strong relevance beyond the immediate application. Comparison of such methods is mostly qualitative, as benchmark instances are still gaining traction within the high school timetabling community.

MirHassani and Habibi (2013) provide the most recent survey which focuses specifically on university course timetabling. The paper is recent enough to mention the growing body of optimisation-based literature (referred to as “Model based methods”) which has emerged particularly in the past 10 years. Several significant works which use integer programming are referenced, although the coverage could be more comprehensive. Heuristic methods are also covered, with the intention of updating the 2002 paper by Burke and Petrovic.

2.3.2 Heuristic-based Approaches

Hertz (1991) presents the first application of tabu search to university course timetabling, where the student sectioning, time assignment, and room assignment are addressed and solved for the Faculty of Economics of the University of Geneva.

Deris et al. (1997) use a branch-and-bound graph search to find solutions, using constraint-based reasoning to propagate constraints. This method is demonstrated on data from the Faculty of Computer Science and Information Systems of the
University of Technology, Malaysia.

Socha et al. (2002) are the first to use Ant colony optimisation, and test their algorithm on three randomly generated instances. These instances are later widely used, particularly by other nature-inspired approaches.

Burke et al. (2003) are the first to implement the great deluge algorithm for course timetabling problems. This is noted as being appropriate as the parameter settings are sufficiently simple for an inexperienced user to set a priority between speed of execution and solution quality. This is tested on instances from the 2002 International Timetabling Competition (ITC, 2002). Landa-Silva and Obit (2008) improve on the great deluge algorithm of Burke et al. (2003) by allowing the decay rate to be nonlinear, which adapts with the progress of the search. Landa-Silva and Obit (2009) extend this approach by embedding it within an evolutionary algorithm as an improvement after the mutation phase.

The doctoral thesis of Müller (2005) introduces the “iterative forward search” (IFS) metaheuristic, which is designed to solve large practical university timetabling problems. The approach is likened to a local search method which works with a feasible (although potentially incomplete) solution. Statistical methods are employed to prevent cycling (Müller et al., 2004), and a combination of hill climbing, great deluge and optionally simulated annealing are used. This hybrid approach has been used in practice for Purdue University in Indiana (Murray et al., 2007). This is one of the few examples of automated timetabling for a real-world sized university course timetabling problem subject to difficult modern quality measures.

With a practical focus, Müller et al. also address the minimal perturbation problem, first using a constraint satisfaction heuristic combined with a branch-and-bound process (Barták et al., 2004). This is continued by applying the IFS algorithm, which improves performance significantly (Müller et al., 2005).

These studies on practical timetabling culminate in one comprehensive paper
on the entire automated timetabling system (Rudová et al., 2011), which provides many valuable insights. The authors later address the problem of converting a curriculum-based timetabling model with complex course relationships into a model where student sections are used (Müller and Rudová, 2014). This is demonstrated for two faculties at Masaryk University, Czech Republic.

The IFS algorithm has additionally been demonstrated on the ITC2007 problems, winning two of three tracks, including Track 3 on curriculum-based course timetabling. Müller (2009) published a paper specifically on the application of their algorithm to the competition problems.

Abdullah et al. (2005) use a saturation degree heuristic to generate an initial solution, and a variable neighbourhood search to explore different local search neighbourhoods to improve this solution. In subsequent work, Abdullah et al. (2007b) expand the idea of a flexible neighbourhood through a *composite* neighbourhood structure. This allows the local search to move through multiple types of neighbourhoods simultaneously, significantly increasing the breadth of search. This is coupled with a randomised iterative improvement algorithm, meaning that worse moves are sometimes taken (in the spirit of tabu search). Abdullah et al. (2009) also use a saturation degree heuristic to generate an initial solution, followed by repeated applications of the great deluge algorithm and tabu search. These are all tested on the benchmarks from Socha et al. (2002).

Burke et al. (2006) describe a method known as case-based selection, where the most appropriate heuristic for a given problem (or case) is chosen from a set of course timetabling heuristics. This is achieved by observing the structure of the given problem, and comparing them to a database of already-solved problems with results on the performance of each heuristic. This method is extended in a subsequent work (Burke et al., 2007) where tabu search is used as a *hyper heuristic*, to select the most appropriate heuristic. The hyper heuristic is used throughout
solving the problem, and different heuristics may be chosen as events are scheduled. This method is tested on the benchmarks from Socha et al. (2002).

Kingston (2007) addresses “hierarchical” timetabling, where a master timetable (time assignment) is constructed from joining smaller timetables together via a set of rules. A layer tree data structure is proposed to represent such problems, where tree nodes designate how sub timetables may be combined with one another to maintain time constraints. Constraints on other resources (e.g. rooms, teachers) are also representable using auxiliary nodes.

McMullan (2007) proposes a modified great deluge algorithm where “re-heats” may be used, which allow worse solutions to be accepted and wider exploration to continue.

Abdullah et al. (2007a) develop a hybrid evolutionary algorithm, using local search. After the mutation phase of the population of solutions, a local search is applied to each member. They note that this approach has also been referred to as a memetic algorithm. This method is tested on the benchmarks from Socha et al. (2002).

Dammak et al. (2008) formulate an integer programme with variables indexed over all events, teachers, time periods, classrooms, and department sections. However, due to this extreme case of dimensionality, they propose two rules-based heuristic procedures which assign lecture events first, followed by tutorial events.

Turabieh et al. (2009) devise a population based metaheuristic using an “electromagnetism like mechanism” to attract or repel feasible timetables with respect to one another. This is coupled with a great deluge algorithm which decays the “force”. This method is tested on the benchmarks from Socha et al. (2002).

Geiger (2009) presents a construction heuristic and several local-search based improvement procedures, for the ITC2007 problems. In this work, Geiger compares the results of using a weighted sum scalarisation of quality measures, with using a
reference point based approach. In the latter, a decision maker is able to choose the values of each objective in a preferred solution. This is conceptually similar to goal programming. It is concluded that a reference based approach allows for more easily balanced outcomes, rather than choosing the coefficients (weights) in a weighted approach which can be difficult. Future research is proposed which would attempt to generate the entire Pareto front of solutions.

De Causmaecker et al. (2009) conduct a practical study for the KaHo Sint-Lieven School of Engineering. A tabu search algorithm is proposed for solving the weekly timetable problem, which is improved by a sequential evaluation of constraints i.e. the model is initially solved with only a subset of the constraints. However, this work is particularly noteworthy for its treatment of problems where each week is non-identical. Inspired by the work of Kingston (2005), a greedy procedure groups teaching events across the 12 weeks which are taught by the same teacher, forming a “pillar” across time. In this case, the local search neighbourhood deals with “horizontal” moves of the pillar to new rooms or time periods, and “vertical” moves where the pillar itself is rearranged. This algorithm is successfully applied in practice.

Bellio et al. (2012) propose a hybrid heuristic, which uses local search, simulated annealing and dynamic tabu search. Particular attention is given to determining an efficient search space, the definition of a neighbourhood, and the evaluation of solutions. This combination of heuristics is then embedded into a control strategy referred to as “token-ring search” from an earlier paper on combining types of neighbourhoods to improve local search (Di Gaspero and Schaerf, 2006). Instead of using moderate values for the parameters for each of the heuristics, the authors conduct a statistical analysis on a range of values for each parameter, to understand their impact on the overall algorithm. Using the R programming language, ANOVA tests are used to distinguish which parameters are particularly influential, so that regres-
sion models can be fitted. The result of the statistical tuning is a set of parameters which works well on all the test problems. These parameters are likely to work well on problems of similar size and structure to those in the test set.

Gunawan et al. (2012) present a complex hybrid method to solve the teacher assignment and time assignment problems. Integer programmes are used in conjunction with Lagrangian relaxation and a customised heuristic procedure to generate an initial feasible solution to the combined problem. This solution is then improved using a simulated annealing heuristic with an incorporated tabu search to focus the search. The authors note that using mathematical programming with Lagrangian relaxation to generate the initial solution, leads to better final solutions than if a heuristic had been used. The approach was tested on data from an unspecified university in Indonesia.

Within the context of high-school timetabling, Kingston (2012) studies the problems of finding a teacher assignment and a room assignment, after the time assignment is already solved. Three algorithms are proposed for teacher assignment, which find and improve a solution using rule-based heuristics. One algorithm is provided for room assignment, which also uses a rules approach to perform a basic assignment in the absence of complicating quality measures.

Kingston (2013b) also addresses the problem of how to repair an infeasible time assignment in the context of high school timetabling. An ejection chain heuristic method attempts to find a feasible solution while maintaining regularity, which is similar to the minimal perturbation problem.

More recent nature-inspired approaches have also been directly applied to the benchmarks of Socha et al. (2002), including harmony search (Al-Betar and Khader, 2012), and honey-bee mating (Sabar et al., 2012). Fong et al. (2014) use bee colony optimisation, hybridised with a great deluge algorithm.
2.3.3 Optimisation-based Approaches

Ferland and Roy (1985) solve the time assignment and room assignment problems separately and demonstrate an early interest in mathematical programming. Although each problem is modelled as a simple assignment problem, the large number of potential constraints (relative to computational resources) leads to the development of a quadratic assignment programme for each problem, which is solved by a simple heuristic algorithm. This method is successfully used at the Université de Montréal.

Mulvey (1982) formulates an assignment model to solve both the time and room assignment problems together. This model is approximated (and simplified) into a network model, which is solved as part of a heuristic procedure.

Glassey and Mizrahi (1986) propose an integer programming formulation for the room assignment problem with contiguous room stability, yet do not solve it due to the prohibitive number of variables (relative to available computational resources), and the possibility of non-integer solutions to the LP relaxation. Instead, they propose a simple heuristic procedure.

Gosselin and Truchon (1986) also approach the room assignment problem (with contiguous room stability) using an integer programming formulation, and aggregate the variables to reduce the problem size. When solving their model, they remark that the simplex method yielded integer solutions to the LP relaxation in every test case.

Dowsland (1990) deals with a time assignment problem where students have a wider range of course options, and so there will inevitably be student “disappointment” at the overlapping of some pairs of courses in any solution. This is a representation of curricula constraints which are treated as soft. Graph colouring, set partitioning and simulated annealing methods are proposed. However, the
problem is small enough for enumeration.

Badri (1996) employs integer goal programming to solve a binary problem of teacher assignment (referred to as faculty), and time assignment. These problems are treated sequentially, with basic objectives (e.g. staff and time preferences) treated in a lexicographic manner. Due to the very small problem size, the author and coworkers were later able to integrate the subproblems, and solve them together as a single integer goal programme (Badri et al., 1998).

Carter (1989) conducts the most advanced study into the room assignment problem, addressing the problem at the University of Waterloo, Canada. The contiguous room stability requirement is enforced using an iterative Lagrangian relaxation method. A wide range of quality measures are considered which are weighted and combined (scalarised) into a single objective function. The author also outlines the experience of satisfying staff and administration requirements while implementing this method in practice.

Carter and Tovey (1992) provide a comprehensive theoretical discussion of the complexity of different types of room assignment problems. Based on this insight, they provide a simple greedy algorithm and discuss its performance on each class of problems.

Carter (2001) contributes a detailed coverage of the algorithms used and experiences of implementing a course timetabling system at the University of Waterloo. A form of the student sectioning problem is solved first, using a graph-based algorithm specific to the case for Waterloo. The time assignment is then preceded by an important pre-processing stage to decompose the problem. Between each pair of sections, the expected number of student conflicts can be computed based on the number of students in both sections, and the preferred time periods for each section. Using graph techniques (colouring and matching) the time-assignment problem is decomposed so that each subproblem (featuring a set of sections) has minimal re-
relationships (with respect to clashes) to other subproblems. This allowed the time assignment of many independent clusters, as well as one large central problem. The authors originally intended to use a binary IP to perform each time assignment, however the computational difficulty of the problems (for 1985) led them to use a combination of heuristics. A greedy procedure originally assigns the course sections in decreasing order of how constrained they are. Once all are assigned, a local-search heuristic is used to swap the time periods of a pair of course sections if it will improve overall time assignment quality. Throughout the time assignment, room feasibility is maintained by a simple consideration of how many events requiring a room of each type are in each time period, and the number of available rooms. The specific room assignment is performed using a sequence of integer programmes, with each addressing a different time domain. Each integer programme uses a coefficient for quality between each event and room, but enforces more complex room stability constraints using Lagrangian relaxation, as covered in Carter (1989). This method was successfully implemented at the University of Waterloo for 15 years, satisfying staff and administration far better than the previous manual system.

Dimopoulou and Miliotis (2001) address the timetabling problem at the Athens University of Business and Economics, where subject groups are assigned to time groups using an integer programme. The room assignment is handled implicitly using homogenous room groups. An optimal solution is found quickly, but this is due to the use of a sole quality measure of a teacher’s preference coefficient for each subject group to time group assignment. In cases where teachers were assigned more than one subject group in a single time period, this would be amended manually.

Mirrazavi et al. (2003) apply integer goal programming to assign multiple ‘subjects’ together into rooms, and then apply a genetic algorithm to assign the times last. This method is demonstrated on data from the School of Computer Science and Mathematics at the University of Portsmouth, UK.
Daskalaki et al. (2004) published one of the first modern applications of integer programming to a university departments timetabling problem. Although the scale was limited to a single department with up to 92 courses, the authors focus solely on integer programming, and are able to solve the entire problem. They also conduct a review on the pre-2000 mathematical programming methods used for timetabling, and identify that due to advancements in mathematical programming techniques (Johnson et al., 2000), such methods are becoming more viable. Unfortunately, even for a small problem, they suffer dimensionality issues with nearly 20000 variables and constraints. The following year, two of the same authors published an efficient decomposition for the problem by solving for each day of the week separately (Daskalaki and Birbas, 2005). This greatly improved the tractability of the problem instances, while retaining as much as possible of the quality ensured by IP approaches.

Qualizza and Serafini (2005) use the decoupled approach of solving the time assignment problem before the room assignment. They note that this decomposition is convenient, and that it is not necessary to make specific event-to-room assignments when finding a time assignment, so that rooms can be aggregated into classroom types. An integer programme is presented where variables correspond to the weekly time assignment for a single course, and a branch-and-price strategy is devised to manage the large number of potential columns. A simple assignment model is given for the classroom assignment. In each case, a single preference coefficient is used for each event assignment, which cannot represent multiple and complex quality measures. An integer programme is presented to solve the room assignment problem and maximise course room stability, although they do not include results.

Avella and Vasil’Ev (2005) solve a university course timetabling problem consisting of time assignment and room assignment together as an integer programme using CPLEX. Although solving one large integer programme typically suffers dimension-
ality issues (as with Daskalaki et al. (2004)), the authors substantially improve performance through derivation of key cutting planes based on the known problem structure.

MirHassani (2006) solve a combined teacher, time and room assignment problem using a binary integer programme for the Shahrood University of Technology. A single scalarised objective is used to represent minimising curricula conflicts, and maximising time preferences. This model is small enough (1000 variables and constraints) to solve optimally using AIMMS.

Schimmelpfeng and Helber (2007) also propose an integer programme for a joint teacher-, time- and room- assignment timetabling problem, subject to a new range of soft quality measures. This model is solved, and the solution implemented at the School of Economics and Management at Hannover University in Germany. Although the problem is for only one department with 11 rooms, the dimensionality results in a large number of variables and constraints. Several quality measures are considered, which are represented as a single scalarised objective. Using their own school’s set of measures makes computational comparison difficult. However, their comprehensive study on the practical implementation in the School is a major contribution. They report that they were able to suitably impress the administration and staff (despite initial skepticism), with the vast majority of staff recommending the system be used in future and the administration investing funding for continued work.

Al-Yakoob and Sherali (2007) address the course timetabling problem at Kuwait University, which features unique timetabling considerations relating to gendered student sections. A spread of events across all time periods is also desired, for the purpose of minimising parking and traffic congestion. The time and room assignment are performed by solving an integer programme for each department, as the aggregated model is too large. This is managed by an *a priori* assignment of rooms.
to each department. The timetabling problem is continued in a companion paper (Al-Yakoob and Sherali, 2006), which assigns staff to the existing timetable, subject to teacher preferences, teaching load equity, and matching the gender of the teacher with the class section. This problem is addressed with an LP-based iterative heuristic where only a subset of variables are required to be integral, and are sequentially fixed in each iteration. The method of assigning staff is improved in a follow-up paper (Al-Yakoob and Sherali, 2013) where column generation is used to efficiently solve a linear programme with many possibilities for feasible class schedules for each staff member. This is embedded within a heuristic which recursively fixes a staff assignment and re-solves the LP, to ultimately find an integral solution.

Bakır and Aksop (2008) use a binary integer programming approach to the time and room assignment problem at the Department of Statistics at Gazi University, Turkey. The model is based on that of Daskalaki et al. (2004), and extended to include room stability (referred to as classroom consistency) as a hard constraint. The objective minimises a total dissatisfaction for groups of students and staff. The authors reduce the number of variables (originally to all times and rooms) by using the same approach as Qualizza and Serafini (2005) where rooms are aggregated into classroom types.

Burke et al. (2008a) propose an integer programme to solve the Udine Course Timetabling Problem (also adapted for the ITC2007), consisting of both time assignment and room assignment. They note that the soft constraints relating to penalising “patterns of classes and free periods in daily timetables” are particularly difficult to deal with as many auxiliary binary variables are required. They demonstrate how this model can be made computationally easier by using a pattern-based method to model this soft constraint. In this case, patterns of classes throughout the day are enumerated and represented as variables which can then be penalised in the objective function.
The same authors also produce a poster (Burke et al., 2008b), which provides an interesting perspective on mathematical programming methods. When timetabling consists solely of satisfying a set of hard constraints (i.e. a feasibility problem), it is equivalent to solving a graph colouring problem. However, in order to model soft constraints, many additional auxiliary variables and constraints are required, which significantly increases the difficulty. The authors conclude that timetabling for large real-world universities cannot presently be solved “optimally”, due to the scaling of the mathematical models, and the difficulty of modelling soft constraints. They also refer to the minimal perturbation problem as one of substantial practical importance.

The graph colouring aspect of timetabling is explored further by studying each of the ways timetabling can be encoded as a colouring problem and represented using integer programming (Burke et al., 2010b). By partitioning the graphs into “super-nodes”, a graph colouring problem can be represented as graph multi-colouring. This allows additional integer programming formulations for timetabling. The integer programming re-formulations result in improved performance in some cases, and allow opportunities for symmetry breaking.

Burke et al. (2010a) continue their work on mathematical programming in a paper which explores metaheuristic hybridisation, by solving several reduced MIPs. Their approach, called MEMOS (“Multiphase Exploitation of Multiple Objective-/Value-restricted Sub-models”), is based on the idea that using known structure, a large problem (the “monolithic formulation”) can be efficiently decomposed so that high quality lower and upper bounds are computable in a relatively short time.

The algorithm comprises initial “surface” searches where variables in the monolithic model are aggregated such that the problem is much easier to solve, yielding a solution which is feasible “on the surface” (although not necessarily feasible in the monolithic) and therefore provides a valid lower bound for the full problem. This so-
solution is then used in a “diving” stage whereby a neighbourhood around this solution is searched to find a solution which is feasible and of high quality in the monolithic formulation. The authors note that “each solver is, time-limit permitting, exact within the given search space, and together, the solvers provide heuristic solutions with global lower bounds”. Different control strategies are used, determining how to manage the surface and dive subproblems and which restrictions to use in the subproblems.

They further make the case for the importance of decompositions, due to the dynamic nature of real timetabling problems. This means that the model needs to be fast enough to re-solve frequently, and a change in quality measures may affect performance of a monolithic formulation unpredictably. The quality of solutions is not at high as those of Müller (the winner of the ITC2007), although lower bounds are found. The main contribution is a new methodology of a MIP-based hybrid metaheuristic.

Burke et al. (2012) approach their previously defined monolithic problem once again from a different angle. Rather than decomposing the problem, they focus on a specific branch-and-cut algorithm, by identifying some of the significant cutting planes. In deriving cuts from specific constraints in the model, and from the underlying graph colouring structure, they are able to significantly improve solve times.

van den Broek et al. (2009) solve an enrolment-based course timetabling problem, where the course sections are scheduled to consider each student’s individual timetable, so that desired courses can be taken simultaneously. Multiple objectives are managed using lexicographic optimisation.

Al-Husain et al. (2011) build on the work of Badri (1996), by adding a third stage to the timetabling process, in which the classroom assignment problem is also solved. This uses integer goal programming to solve a problem from the College of
Lach and Lübbecke (2008) use a decoupling of the time and room assignment, applied to the more modern timetabling problems in the ITC2007. When tested on ITC2007 instances without considering any quality measures, the solve times are expectedly very low (under 1 second). The claim is made that all quality measures can be added without significantly worsening the solve time.

A follow up paper by Lach and Lübbecke (2012) covers their adaptation of the model to include the quality measures, and full computational results. Although finding high quality solutions, few of the problems were able to solve to optimality in both stages even with a generous time allotment. This is the first occurrence in the literature of integer programming approaches competing on the same datasets as heuristic approaches. The authors also indirectly refer to the minimal perturbation problem as a type of robustness where changes to the input cause minimal changes to the timetable.

Asín Achá and Nieuwenhuis (2012) develop an entirely new approach to solving modern course timetabling problems, based on satisfiability. From Karp (1972), it is well-known that finding a feasible solution to a binary integer programming problem is equivalent to solving a SAT (satisfiability) problem. However, handling soft constraints (with varying weights) requires solving a “Weighted-Partial-MaxSAT” problem, which is significantly more difficult. Asín Achá and Nieuwenhuis address the ITC2007 instances, and yield some interesting results. When a SAT problem is solved with soft constraints set to be hard (i.e. they must have zero penalty), optimal solutions of cost zero are found very quickly in the few cases where they exist. However, no useful information was gained in the remaining majority of cases. To build a more general algorithm, they encode the full problem as a Weighted-Partial-MaxSAT problem. This yields an optimal solution for approximately half the problems when given a substantial time allowance.
While interesting from a theoretical standpoint, the authors note that this method is of little practical use because it cannot provide a solution when terminated early. To tackle this problem, the authors propose using branch-and-bound for solving the Weighted-Partial-MaxSAT problem. Although the method performs poorly on many of the instances, the authors note their implementation of branch-and-bound is elementary and may have significant potential for improvement. This paper is an interesting proof of concept on using the well-studied class of satisfiability problems for course timetabling.

2.4 Discussion

The most evident conclusion of this literature review is that university course timetabling remains an open problem. Several survey papers (e.g. Lewis, 2008), have noted an increased interest in timetabling approaches using a particular method. It seems there is a trend of increased attention paid to university course timetabling problems in general. Furthermore, a particularly increased focus has been placed on developing a diverse range of algorithms, and the hybridisation of existing methods. A concerted effort has also been made to facilitate benchmarking between various algorithms. The 2007 International Timetabling Competition (ITC, 2007), and the subsequent benchmarking efforts (Bonutti et al., 2012, 2008) have served to demonstrate that success has been achieved through a wide variety of approaches.

Many authors have published on the application of a particular metaheuristic paradigm to a set of benchmarks in university timetabling (usually those generated by Socha et al., 2002), which tend to perform comparably to existing methods. Although these do contribute to the wealth of literature, the most valuable contributions come from authors publishing a string of material which is able to delve more deeply into the analysis and development of the proposed algorithms. This
is particularly the case where the authors are solving a practical problem, as significant detail must be given to cover the many subproblems involved in tackling a real-world problem. The most sophisticated timetabling system in the literature is arguably that of Rudová et al. (2011) which addresses large problems at Purdue University in Indiana and Masaryk University in the Czech Republic using the IFS heuristic.

Work on parameter-tuning for algorithms and metaheuristic hybridisation (e.g. Bellio et al., 2012) has shown good results in improving existing methods. However, it would be even more valuable if wider insights could be drawn from why particular parameters or combinations of heuristics will work for particular types of problems. This view is also expressed by Sörensen (2013) referring to the broader field of metaheuristics in general.

A particularly interesting direction in the field of timetabling research is that of hyper-heuristics. As the structure of university timetabling problems can differ substantially from one university to another, the concept of a set of algorithms (metaheuristics or exact algorithms) which can be employed as necessary may prove integral to the future of timetabling.

The optimisation literature is relatively more fragmented, with the majority of authors contributing one or two papers demonstrating an algorithm for part of the timetabling problem at their university. Carter (2001) offers the only comprehensive coverage of a full optimisation-based timetabling system. However, the algorithms are limited by the computational resources available in 1985. More recent applications of exact methods in a practical setting do exist, however most attempt to solve the combined time and room assignment problem using a single integer programme (e.g. Daskalaki et al., 2004; Daskalaki and Birbas, 2005; Avella and Vasil’Ev, 2005; Schimmelpfeng and Helber, 2007; Burke et al., 2008a) or as a single satisfiability problem (Asín Achá and Nieuwenhuis, 2012). Inevitably in these cases, the models
are either solved for a small university, or a single department at a larger university, and also with a simple set of quality measures. These are unsuitable for large universities where the majority of teaching space is shared between departments and faculties. Burke et al. (2008b) demonstrate the intractably large number of variables and constraints which are required for more complex or larger problems solved in this manner.

The university course timetabling problem can instead be considered as a time assignment problem followed by a classroom assignment problem (also known as “times first, rooms second”). This decomposition has been used by Carter (2001), Dimopoulou and Miliotis (2001), Qualizza and Serafini (2005), and Lach and Lübbecke (2012), although has not been comprehensively studied. The work of Lach and Lübbecke (2012) was conducted primarily on the ITC2007 problems, where it was demonstrated to generate solutions which are comparable to those from the best known heuristic methods. The research was additionally able to provide best bounds for the benchmark instances.

Inevitably this decomposed method sacrifices the exactness property of integer programming methods. Generating the timetable in two stages introduces the issues of maintaining the feasibility and quality i.e. the time assignment algorithm must exhibit forethought into the feasibility and quality attainable in the room assignment. In order to ensure a feasible room assignment exist, a set of simple constraints exists in the first stage which ensure that for each room size, the number of rooms this size (and larger) is at least the number of streams with this many students (or greater).

In our experience, this decomposition is commonly used in practice. Faculties or departments may prefer to generate a time assignment for their courses, and retain control over unique requirements and preferences. Kingston (2007) addresses this same concept in practical high school timetabling where a master timetable
is constructed from joining smaller timetables. By contrast, a room assignment typically cannot be performed by faculties as teaching space is shared, and therefore must be managed centrally. A similar system is described by Abdennadher et al. (2000).

Despite substantial progress in the development of timetabling algorithms, there remains a lack of coverage of approaches dealing with problems of the size and complexity seen in timetabling for large modern universities (McCollum, 2007). The ITC2007 problems in particular bear important differences and simplifications relative to problems seen in practice (see Section 3.3).

One area in which the literature is particularly lacking, is methods for solving the minimal perturbation problem. The importance of solving this problem has been mentioned in numerous works (e.g. Burke et al., 2008b). This problem has particularly broad application, as timetabling data is collected and revised prior to semester, changes to the planned timetable are inevitable. These changes are facilitated through solving a minimal perturbation problem which can inform the timetabler about the (minimal) level of disruption a proposed change would cause. Carter (2001) described the capacity for timetable representatives to make manual adjustments as the single most important part of their timetabling system.

We also need to address the topic of quality measures in timetabling (see Table 2.1 for examples). To solve practical problems, numerous quality measures must be considered, some of which are complex. However, the exact definition and measurement of these quality measures depends on the institution and is often flexible. We provide two examples here.

Firstly, many universities are concerned with the amount of time taken for students and staff to travel between two consecutive events. This is an important factor in timetabling, however, it lends itself to many (potentially) equally valid measurements. Is the aim to minimise the total summed travel time across all students and
staff, or should it be modelled so that “tardiness” beyond a certain allowable travel time is minimised? Should the weightings (or allowance) differ for students and staff? Should a long travel time be penalised per course affected, or weighted by the number of students in the course? Perhaps it is more appropriate to attempt to schedule all events in their home faculty buildings, thereby implicitly ensuring a low travel time for most students. Burke et al. (2012) discuss this idea in the context of whether timetablers should strive for room stability, or whether building preference is more appropriate.

As a second example, significant attention has been paid to measuring the compactness of daily events for a curriculum, which was used in the ITC2007. The definition used aims to penalise specific patterns of daily events so that daily events are essentially “clumped together” as much as possible, which may not necessarily be desirable. Measuring the desirability of a daily pattern of events to this exact specification is particularly difficult for IP approaches Burke et al. (2008a). Burke et al. (2012) develop a sophisticated cutting-plane algorithm to deal with the computational difficulties caused by modelling this objective measure, which is a valuable contribution. However, from a practical perspective, if the quality measure is extremely difficult to model, it may be re-defined flexibly while achieving the same intention. Assuming such a “clumping” is desirable, it could be similarly achieved by minimising the daily span of events, which is much easier to model in an integer programming framework.

The principle of practical quality versus theoretical quality is noted by Qualizza and Serafini (2005) as they deal with staff members’ personal preference coefficients. They note that preference coefficients are an intrinsically imperfect measure, and that achieving a solution within a few percent of optimal is acceptable.

Finally, we note that some important aspects of course timetabling have been relatively under-addressed. In particular, although all modern timetabling systems
are required to deal with multiple measures for solution quality, little use has been made of more advanced methods from multi-objective optimisation research. As a result, the tradeoffs between multiple objectives are not well understood.

This thesis explores the hypothesis that we can find good solutions to large practical university course timetabling problems. This is achievable using the natural decomposition into a time assignment and a room assignment, in conjunction with several efficient integer programming models within a broader heuristic framework.
This chapter introduces the course timetabling datasets which we use in this thesis to demonstrate the application and performance of our algorithms. The datasets derive from the University of Auckland (in Section 3.1), the Technical University of Denmark (in Section 3.2), and the University of Udine benchmark problems (in Section 3.3). We demonstrate how each dataset is unique in terms of which timetabling subproblems are addressed, how quality is measured, and the environment in which the problem is solved.

3.1 The University of Auckland

Processes for course timetabling at the University of Auckland have undergone significant changes since 2010, in an attempt to address the needs of a large modern university. In 2011, a new timetable construction process was implemented involving a new team of administrative staff, and utilising new timetabling software. This process involves a different method of data collection, and redefines the problem scope and choice of quality measures.
In this thesis, we refer to datasets from the University of Auckland from both 
2010 and 2013, which are respectively before and after the procedural changes to 
timetabling. In order to interpret the results from our algorithms in a practical 
context, this section explains each timetabling process and the associated data.

Section 3.1.1 explains general background information about the University and 
the role of each subproblem as part of the overall timetabling process. The remaining 
two sections, 3.1.2 and 3.1.3, outline the timetabling processes used in 2010 and 2013 
respectively.

3.1.1 Background

The academic calendar at the University of Auckland primarily consists of two 
twelve-week semesters. The first semester begins at the start of March, while the sec-
ond begins in mid-July. In each semester, a weekly teaching timetable is repeated 
which spans the fifty core teaching hours, from 08:00 to 18:00, Monday through 
Friday.

There is also a shorter six-week semester in January and February of each year, 
known as “summer school”. We have not addressed this semester in this thesis, as 
the problem is significantly less constrained and less difficult to solve.

We have similarly refined our focus to the main city campus of the University of 
Auckland, as the smaller campuses in Auckland are also less constrained, and are 
mostly logically separable from a timetabling perspective.

The problem of student sectioning (see Section 2.1) is relatively simple at the 
University of Auckland, and is addressed by faculties. The majority of courses use 
only one section for the lecture component, due to the low number of courses which 
are larger than the largest room. Sectioning is more common for the tutorial or lab 
component of a course, where smaller groups are desirable. Determining the size 
and number of sections is not complex, as these components can typically only be
taught in a small set of specialised rooms of a known size.

The problem of teacher assignment is similarly solved by faculties. As mentioned in Section 2.1, there are a limited number of staff who are qualified to teach any given course. The faculties are in a good position to decide the teacher assignment for their courses, as they understand each staff member’s teaching preferences and experience. Furthermore, other issues need to be considered such as particular contract requirements, vacation or research related travel etc., which would pose a problem for automated teacher assignment.

When the timetabling data is acquired from faculties, the time assignment and room assignment can begin, which constitute the general “timetable construction” for the University. This phase of course timetabling takes place approximately between July and October of the previous year. During this time, a feasible timetable is constructed for both semesters and tentative room assignments are made.

This is followed by the enrolment phase of timetabling which runs from November through to the second week of teaching in each semester. During this time, changes can occur to the underlying timetable data. These can include changes to the courses (e.g. late additions/removals), available staff (e.g. sickness, parental leave) or the number of enrolled students in a course. The enrolment data is particularly volatile, as the estimates used in the construction phase are replaced by actual enrolment figures. This introduces a challenge for the timetablers who may need to change existing room assignments in the case of unexpectedly high enrolment in any given course.

Although the timetabling construction initially schedules university courses, the enrolment phase of timetabling must also consider other university events. These include any events which occupy the common teaching space during semester, such as ad hoc room bookings, administration meetings and short courses.

When changes must be made to the timetable in order to accommodate changes
in the data, minimising the disruption to the overall timetable is important. This process is performed manually and relies on the knowledge of timetablers.

3.1.2 2005 – 2010 Timetable Construction Process

Constructing a timetable requires finding a time and room assignment for all events of courses. Table 3.1 shows statistics relating to the size of the course timetabling problem at the city campus in 2010. For the purposes of Table 3.1, an “event” refers to a class meeting of any length (i.e. including long events).

<table>
<thead>
<tr>
<th></th>
<th>2010 Semester 1</th>
<th>2010 Semester 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Faculties</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Departments</td>
<td>73</td>
<td>73</td>
</tr>
<tr>
<td>Courses</td>
<td>985</td>
<td>911</td>
</tr>
<tr>
<td>Events</td>
<td>1965</td>
<td>1866</td>
</tr>
<tr>
<td>Total event hours</td>
<td>2383</td>
<td>2231</td>
</tr>
<tr>
<td>1 hour events</td>
<td>1561</td>
<td>1516</td>
</tr>
<tr>
<td>2 hour events</td>
<td>390</td>
<td>335</td>
</tr>
<tr>
<td>3 hour events</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>Rooms</td>
<td>72</td>
<td>72</td>
</tr>
</tbody>
</table>

Table 3.1: UoA 2010 Dataset Problem Size

Initially each faculty will generate its own time assignment for each event to meet its particular requirements. This includes finding a high quality solution for its students and staff, while respecting non-overlapping requirements for courses within a curriculum, and courses which must be taught by a common lecturer. Typically this is not a major task, as it can be managed by making incremental changes to the timetable from the previous year. Guidelines are provided to faculties which help to achieve a good spread of events throughout the day and week. For example, a minimum of 15% of events must be scheduled on Friday. Other rules also exist,
such as ‘two-hour long events must start on an even hour’ and ‘long events of three or more hours are accepted only by arrangement’, because they are seen to disrupt the ability to place regular one hour events into rooms.

The involvement of faculty in the time assignment process ensures a high quality solution. Faculties and individual departments each have particular requirements which are not necessarily easy to formalise or generalise. We provide examples relevant to the Engineering Faculty, which teaches relatively structured programmes taught in four “parts” over four years. Courses from Part 1 are taken by all engineering students, and typically require more than one section per course (due to the high total enrolment). However, courses from Parts 2–4 are more specialised and are typically only offered in only one section per semester. This set of circumstances leads to the following preferred scheduling practices:

- Lectures for Part 1 and Part 3 are held in the morning time periods, while Part 2 and Part 4 are held in afternoon time periods. This accommodates “split-part” students who need to take courses from two different parts.

- Two independent sections of two Part 1 courses are “paired” together into the same time periods, to avoid students deliberately attending the wrong section, and causing more students to attend than the room’s capacity. For example, two courses (“121” and “131”) are both taught twice on a given morning, and each must occupy an 8am time period, which is seen as undesirable. This rule schedules each of their non-8am events into the same time period. Table 3.2a shows an example where this rule is not applied (since “121 B” and “131 B” are held at different times). In this case, many students would attend both the later events, despite being enrolled in an 8am event. Table 3.2b demonstrates a pairing where this is structurally not possible.

- If teaching at 8am is necessary, one department has a preference for “sharing
3.1. THE UNIVERSITY OF AUCKLAND

<table>
<thead>
<tr>
<th>Mon/Wed/Fri</th>
<th>8am</th>
<th>9am</th>
<th>10am</th>
<th>11am</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) Unpaired courses</td>
<td>121 A</td>
<td>121 B</td>
<td>131 A</td>
<td>131 B</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mon/Wed/Fri</th>
<th>8am</th>
<th>9am</th>
<th>10am</th>
<th>11am</th>
</tr>
</thead>
<tbody>
<tr>
<td>(b) Paired courses</td>
<td>121 A</td>
<td>121 B</td>
<td>131 A</td>
<td>131 B</td>
</tr>
</tbody>
</table>

Table 3.2: Unpaired vs Paired Courses

<table>
<thead>
<tr>
<th>Mon</th>
<th>Wed</th>
<th>Fri</th>
</tr>
</thead>
<tbody>
<tr>
<td>8am</td>
<td>221</td>
<td>221</td>
</tr>
<tr>
<td>9am</td>
<td>231</td>
<td>231</td>
</tr>
<tr>
<td>10am</td>
<td>241</td>
<td>241</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mon</th>
<th>Wed</th>
<th>Fri</th>
</tr>
</thead>
<tbody>
<tr>
<td>8am</td>
<td>221</td>
<td>241</td>
</tr>
<tr>
<td>9am</td>
<td>231</td>
<td>221</td>
</tr>
<tr>
<td>10am</td>
<td>241</td>
<td>231</td>
</tr>
</tbody>
</table>

Table 3.3: Time Stability vs Time Equity

the burden" equitably across all morning courses. This is achieved by assigning each morning course to one 8am time period per week, rather than assigning one course to 8am for all its events. The latter was more commonly applied in the faculty (as it honoured time stability) and a different course may be scheduled at 8am in the following year. Tables 3.3a and 3.3b demonstrate the preference for equity and stability respectively.

The Lecture Theatre Management Unit (LTMU) are responsible for the assignment and booking of rooms, both ad hoc and for timetabled teaching. The faculty timetables are collated by the LTMU who attempt to find a high quality feasible room assignment. This is achieved by solving a series of integer programmes for a prescribed set of quality measures. If no feasible room assignment exists, the
timetable must be modified such that one can be found. At a meeting chaired by the LTMU, specific conflicts are addressed and faculties adjust their timetables and/or requested room sizes and attributes. This process repeats until a feasible room assignment can be found. An overview of this process is shown in Figure 3.1. Each integer programme is represented using a blue rectangle, for the objective specified.

3.1.3 2011 – 2013 Timetable Construction Process

Statistics relating to the size of the timetabling problem in 2013 are given in Table 3.4. These figures are notably larger than those in Table 3.1 for 2010. A small factor in this increase is due to the general growth of the University, as new course offerings open and new facilities become available. However, this alone does not account for the sizeable discrepancy. For the 2010 data, many course components and rooms were not included in the central timetabling process. Instead, these rooms were considered “faculty rooms” rather than “pool rooms”, and were managed by faculties. For example, many courses feature a tutorial and/or lab component which the faculty would assign to these rooms. As a part of the new timetabling process, all rooms in the university are part of the central system.

The Timetable Services Office (TSO) perform the timetable construction in 2013, consisting of both a time and room assignment. Faculty representatives work with TSO staff to input course related data into a centralised system. This data is more complex than the 2010 data, as it includes constraints and quality measures relating to the time assignment (e.g. relating to the curricula). Typically the TSO staff will begin by timetabling a small block of courses, which may all be from one particular academic programme. Once a time and room assignment is determined for this block of courses, another block is processed. At this stage, “problems” such as curricula violations are permitted, which are addressed later by manual adjustments.
3.1. THE UNIVERSITY OF AUCKLAND

Figure 3.1: 2010 UoA Timetabling Process
In some cases, faculties will still provide a starting time assignment for some of their courses. This is required when there are many constraining curricula over a small set of courses, such as Part 1 Engineering.

Timetabling software is utilised to visualise the constraints and basic quality measures which relate to making an individual event-to-time or room assignment. A simple automated scheduler may be used, which conducts a similar sequential process as a human timetabler, although the solutions require substantial manual adjustment. Ultimately, the timetable construction remains a predominantly manual process.

For the most part, quality is considered as each course is scheduled, in the context of its own assignment. General guidelines for quality are manually considered, such as the favourability of time periods in the middle of the day, and the preference for rooms local to the home faculty.
3.1.4 Discussion

The most fundamental difference between the timetabling processes used in 2010 and 2013 is the method of decomposition used. In 2010, each event is first assigned to a time period (by faculties), followed by iterations of an automated room assignment and manual perturbation. In 2013, each event is simultaneously assigned to a time period and room. However, due to the limited software capabilities, assignments can only be made for small groups of events at a time. Therefore, this method can be considered a decomposition into individual timetabling problems where each problem is constrained to avoid conflicts with the previous assignments, and to utilise the unused rooms in each time period.

It is well-known in the academic timetabling community, that it is extremely challenging to provide a generic software product which can perform optimisation in timetabling. Existing packaged software is predominantly a tool to visualise the constraints associated with making an individual assignment in a manual process. This may lead to a generally poor utilisation of space, and difficulty in finding a feasible solution. Even state-of-the-art algorithms struggle to assign a time and room simultaneously while considering complex measures of quality such as room stability. Due to these limitations, the timetable construction in 2013 takes a minimum of two working weeks of effort from TSO staff and faculty administrators.

However, such software inherently excels at allowing fine control over the timetable. With access to comprehensive and suitably presented timetabling data, skilled timetablers at the TSO are able to efficiently address real-world complications. These include consideration of numerous guidelines and exceptions, which cannot be accurately modelled in an automated algorithm. Through experience, the timetabling staff have substantial knowledge about the University from a practical perspective.

The approach we use in this thesis aims to incorporate the positive elements
from each of these timetabling processes. Utilising the faculty input in generation of
the time assignment (as with the 2010 approach), allows many of their requests to
be satisfied. An automated room assignment algorithm is also beneficial to ensure a
high quality solution. Finally, an algorithmic approach to the minimal perturbation
problem plays an important role in both the construction and enrolment phases of
timetabling. At each of these stages, the experience of timetablers is essential. How-
ever, rather than making a large number of manual assignments, they are involved
in utilising the algorithms, evaluating the solutions, and adjusting where necessary.
This different paradigm is particularly relevant for the maintenance of a timetable.
The minimal perturbation problem is able to assist in deciding how to deal with any
arbitrary change in the data or unforeseen circumstance.

3.2 The Technical University of Denmark

In this thesis, we refer to datasets from the Technical University of Denmark (DTU)
from two semesters, Autumn 2011 and Spring 2012. This enables us to demonstrate
the versatility and effectiveness of our algorithms on an additional practical problem.

Section 3.2.1 explains general background information about the University and
explains the role of each subproblem as part of the overall timetabling process.
Section 3.2.2 outlines the timetabling processes used in the corresponding 2011 –
2012 period.

3.2.1 Background

The academic calendar at the Technical University of Denmark primarily consists
of two thirteen-week semesters. The spring semester runs from February to May,
while the autumn semester runs from September to December. In each semester,
a weekly teaching timetable is repeated which involves four daily time periods, on
Monday through Friday. Each day consists of a morning session (from 8am to 12pm) and an afternoon session (from 1pm to 5pm). Events may be four hours in duration and start at 8am or 1pm, or they may be two hours in duration start at 8am, 10am, 1pm, or 3pm. As a result, we effectively deal with four daily time periods (each two hours in duration), for a total of 20 weekly time periods.

As with most universities, the teacher assignment for courses is conducted by faculties, and is assumed to be a fixed input to the timetabling problem. However, the time assignment at DTU is also simple, due to a highly structured nature. Faculties assign their courses to either one or two sessions per week, depending on whether they are full-load or half-load courses.

Within each four-hour session, most courses at DTU teach a two-hour lecture event, followed by a two-hour tutorial event. These components are typically held in different rooms, with a break in-between. Sectioning is rarely applied to lectures. However, tutorials may be sectioned into two or three independent and concurrent events. A smaller number of courses will teach a four-hour event known as a “class”. A class requires the same room for the entire four-hour session, and will only be taught in one section.

This leaves the sectioning and room assignment problems to the timetabling administration, AUS.

3.2.2 2011 – 2012 Timetable Construction Process

Although the administration must solve a sectioning and room assignment problem, the former is not typically difficult. As with the University of Auckland, only a small number of courses are split into sections, and only a certain number of courses are candidates for sectioning based on the staff requirements.

The room assignment problem is the most complex part of timetabling at DTU, involving several quality measures and constraints which are specific to the situation.
faced by DTU. Each room has a student capacity, and is also categorised as one of:
an auditorium, a study room, a study area.

As with other practical timetabling problems, it may not be possible to assign a
room to every event, for the given time assignment. In this case, the first priority
of the room assignment is to find a feasible room for as many events as possible.
However, in the case of DTU, a partial room assignment may be considered as a
feasible solution to the problem. This is because there are additional rooms which
are not included in the room assignment process, and are only used when necessary.
The availability of these rooms is not fully known to AUS, as they are managed by
individual departments.

As a second objective, AUS would like the unassigned events to be as small as
possible, in terms of the number of students. This makes the assignment of these
events from department rooms more likely to be feasible. For a similar reason, some
courses may be designated as mandatory to assign in any feasible partial assignment,
as they are known to be unsuitable for the smaller department rooms.

Additionally, each event type (lecture, tutorial or class) is most suitable for a
particular type of room. Lectures may be taught in an auditorium or study room,
tutorials may be taught in a study room or a study area, and classes must be taught
in a study room. Although the timetabling guidelines from AUS imply that these
requirements are hard constraints, they are broken (when necessary) in practice. In
many cases, these constraints are enforced automatically due to the relative sizes of
each room type. For example, lectures are often too large to be held in any study
room or study area, so must be held in an auditorium.

To maximise the quality of event-to-room assignments, the most important pri-
ority is a preference for events to be taught in the same region of the campus, or
“quadrant”, as the faculty offices.

Finally, we list two additional constraints which apply uniquely to DTU. Firstly,
for lectures which are taught in multiple sections (i.e. multiple simultaneous events), each of these events must be in the same building. This is because the lecturer will teach to all sections simultaneously via an interactive live broadcast from one room. This constraint upholds a technical requirement in this process. Secondly, if a two-hour lecture event is held in a study room (instead of an auditorium), and it is followed by a two-hour tutorial event, the tutorial must be in the same study room.

Similarly to the TSO at the University of Auckland, AUS at DTU use timetabling software to assist in making room assignments, although the process remains mostly manual.

3.2.3 Discussion

The timetabling problem at DTU provides an interesting comparison to the problem at the University of Auckland. For the time assignment, DTU uses a highly structured specification for when to teach events. This method may be suitable for courses at DTU, as it exclusively offers academic programmes in engineering and scientific disciplines. This is contrasted to the UoA, which offers programmes from diverse fields of study, and requires a flexible time assignment algorithm to cater for a diverse range of course structures.

However, the room assignment problem is relatively similar at each university. The primary difference is that DTU consider a partial room assignment to be feasible, as unassigned events can be taught in department-controlled rooms. This differs from the University of Auckland where all available rooms are included in the system, and a partial room assignment must be resolved, typically by perturbing the time assignment.

DTU also emphasise the importance of the room preference quality measure in generating a room assignment. This is used to favour the placement of events into buildings close to the relevant teaching department. Due to the large physical size
of the campus (see scaled map in Figure 3.2), travelling across the campus during a session (i.e. 10am and 3pm) may take up to 20 minutes, which would unacceptably cut into teaching time.

In 2012, the DTU timetabling problem was addressed in a masters thesis by Bærentsen (2012), proposing an integer programme to solve the room assignment and student sectioning. Bagger et al. (2014) revisited this problem, proposing a new integer programme as part of a heuristic algorithm (referred to as a matheuristic).

The proposed algorithms demonstrate substantial potential for use in practice. The current approach of manual timetabling (for the room assignment and sectioning) by AUS takes 3 to 5 workdays to find a feasible solution.

3.3 University of Udine Benchmarks

The University of Udine benchmark instances consist of 21 course timetabling datasets originally developed for track 3 of the second international competition (ITC2007). These instances are based on practical course timetabling data, and are widely used for benchmarking recent methods in the literature.

The competition used a “formulation” referred to as UD2, which specifies a set of weighted quality measures, and a set of hard constraints. With the intention of diversifying the scope of the benchmarking problems, several of the competition organisers released additional problem formulations in a subsequent paper (Bonutti et al., 2012). The full set of 5 specifications, UD1–UD5 is intended to address a broader range of potential applications. This work is accompanied by a “problem management system” website, which provides a public listing of all solutions Bonutti et al. (2008), and provides tools for benchmarking and validating solutions.

In this section, we briefly introduce the 21 datasets and the 2 most relevant formulations, UD2 and UD5. For additional details, we refer to the competition website.
Solving the Udine problems requires solving a time and room assignment problem for a given dataset specifying information on the courses, rooms, time periods, curricula etc. Table 3.5 lists important characteristics about the size of each dataset. After the dataset name, the next four columns of 3.5 list the number of courses, events,
rooms and time periods, which give an indication of the size of the problem. The time periods are spread across 5 or 6 days, e.g. 30 periods in comp01 represents 5 days with 6 periods each.

The sixth column represents the “utilisation” of all rooms in all time periods. This is calculated as the number of events divided by the total number of available time periods for all rooms. When all rooms are available in all time periods, the denominator expression is the number of rooms multiplied by the number of time periods. The utilisation gives an idea of the congestion or space for flexibility. However, this figure should be treated as approximate, as it only provides an overview of all rooms which may be misleading. For example, a particular subset of rooms (e.g. large lecture theatres) may be very highly utilised (all rooms occupied all day), while the overall utilisation figure may be lower, if there is an abundance of smaller rooms.

Finally columns 7 and 8 show the “size” of courses, in terms of the number of associated events. The average “size” is calculated as the number of events divided by the number of courses. These figures have implications for both the practicality of the data, and the difficulty of solving the problem.

Based on these 21 datasets, the formulation specifies which potential constraints should be applied, and whether they are hard constraints or have an associated penalty for violation i.e. quality measures. For all formulations, hard constraints ensure that all events are assigned to a time period and room. The time assignments must not cause any curricula or teacher conflict. The room assignment must not cause any room conflict i.e. more than one event assigned to a room (in any period).

Quality in the time assignment is measured by the following desirable traits:

- The spread of events from a course on different days throughout the week
- The compactness or clumping of events from a curriculum on a given day
<table>
<thead>
<tr>
<th>Name</th>
<th>Courses</th>
<th>Events</th>
<th>Rooms</th>
<th>Periods</th>
<th>Utilisation(%)</th>
<th>Average</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>comp01</td>
<td>30</td>
<td>160</td>
<td>6</td>
<td>30</td>
<td>88.9</td>
<td>5.33</td>
<td>8</td>
</tr>
<tr>
<td>comp02</td>
<td>82</td>
<td>283</td>
<td>16</td>
<td>25</td>
<td>70.8</td>
<td>3.45</td>
<td>7</td>
</tr>
<tr>
<td>comp03</td>
<td>72</td>
<td>251</td>
<td>16</td>
<td>25</td>
<td>62.8</td>
<td>3.49</td>
<td>7</td>
</tr>
<tr>
<td>comp04</td>
<td>79</td>
<td>286</td>
<td>18</td>
<td>25</td>
<td>63.6</td>
<td>3.62</td>
<td>7</td>
</tr>
<tr>
<td>comp05</td>
<td>54</td>
<td>152</td>
<td>9</td>
<td>36</td>
<td>46.9</td>
<td>2.81</td>
<td>3</td>
</tr>
<tr>
<td>comp06</td>
<td>108</td>
<td>361</td>
<td>18</td>
<td>25</td>
<td>80.2</td>
<td>3.34</td>
<td>7</td>
</tr>
<tr>
<td>comp07</td>
<td>131</td>
<td>434</td>
<td>20</td>
<td>25</td>
<td>86.8</td>
<td>3.31</td>
<td>7</td>
</tr>
<tr>
<td>comp08</td>
<td>86</td>
<td>324</td>
<td>18</td>
<td>25</td>
<td>72.0</td>
<td>3.77</td>
<td>7</td>
</tr>
<tr>
<td>comp09</td>
<td>76</td>
<td>279</td>
<td>18</td>
<td>25</td>
<td>62.0</td>
<td>3.67</td>
<td>7</td>
</tr>
<tr>
<td>comp10</td>
<td>115</td>
<td>370</td>
<td>18</td>
<td>25</td>
<td>82.2</td>
<td>3.22</td>
<td>7</td>
</tr>
<tr>
<td>comp11</td>
<td>30</td>
<td>162</td>
<td>5</td>
<td>45</td>
<td>72.0</td>
<td>5.40</td>
<td>9</td>
</tr>
<tr>
<td>comp12</td>
<td>88</td>
<td>218</td>
<td>11</td>
<td>36</td>
<td>55.1</td>
<td>2.48</td>
<td>3</td>
</tr>
<tr>
<td>comp13</td>
<td>82</td>
<td>308</td>
<td>19</td>
<td>25</td>
<td>64.8</td>
<td>3.76</td>
<td>7</td>
</tr>
<tr>
<td>comp14</td>
<td>85</td>
<td>275</td>
<td>17</td>
<td>25</td>
<td>64.7</td>
<td>3.24</td>
<td>6</td>
</tr>
<tr>
<td>comp15</td>
<td>72</td>
<td>251</td>
<td>16</td>
<td>25</td>
<td>62.8</td>
<td>3.49</td>
<td>7</td>
</tr>
<tr>
<td>comp16</td>
<td>108</td>
<td>366</td>
<td>20</td>
<td>25</td>
<td>73.2</td>
<td>3.39</td>
<td>7</td>
</tr>
<tr>
<td>comp17</td>
<td>99</td>
<td>339</td>
<td>17</td>
<td>25</td>
<td>79.8</td>
<td>3.42</td>
<td>7</td>
</tr>
<tr>
<td>comp18</td>
<td>47</td>
<td>138</td>
<td>9</td>
<td>36</td>
<td>42.6</td>
<td>2.94</td>
<td>3</td>
</tr>
<tr>
<td>comp19</td>
<td>74</td>
<td>277</td>
<td>16</td>
<td>25</td>
<td>69.2</td>
<td>3.74</td>
<td>7</td>
</tr>
<tr>
<td>comp20</td>
<td>121</td>
<td>390</td>
<td>19</td>
<td>25</td>
<td>82.1</td>
<td>3.22</td>
<td>7</td>
</tr>
<tr>
<td>comp21</td>
<td>94</td>
<td>327</td>
<td>18</td>
<td>25</td>
<td>72.7</td>
<td>3.48</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 3.5: Udine Benchmark Dataset Characteristics

- The spread of events from a curriculum on different days throughout the week

For the UD2 formulation (as used in the ITC2007), quality in the room assignment is measured by the course room stability i.e. minimising the number of unique rooms each by each course.

For the UD5 formulation, room assignment quality is measured by the travel distance for consecutive events of a curriculum. If two events from the same curriculum are held in rooms from different buildings, a penalty is incurred.

Finally, for all formulations, a measure of room assignment quality is the minimisation of the amount that rooms are assigned to over capacity events. Although
each room may only be occupied by one event at any time, any event may be held in any room regardless of the number of students.

3.3.2 Discussion

We address potential shortcomings of the way the ITC2007 problems are designed, in terms of the problem structure and the way quality is quantified. Although the problems have been derived from real data at the University of Udine in Italy, we find some features to be unusual. We are particularly interested in how these widely-used benchmark instances relate to practical problems.

Firstly, we find that courses in the ITC2007 problems frequently have an extremely high number of events. Most problems have several courses with up to 7 events. In our experience, it is uncommon for a course to need to teach more than 4 events in the same week, which desire to be in the same room. For example, at the University of Auckland, normal-sized courses typically have 3 lectures and 1 tutorial per week. However, the lecture and tutorial components are usually treated as separate courses when modelling, as they are structurally different. For example, the lectures may be taught in one section in a large lecture theatre, whereas the tutorials are taught in many sections in smaller rooms. As a result, quality measures such as time stability and room stability do not logically apply between a lecture and tutorial.

The utilisation of university resources is another factor which appears to be abnormally high in the ITC2007 problems. This naturally makes the problems more difficult, as the algorithms operate with less flexibility for placements of events. Studies into the utilisation of teaching space at universities (Beyrouthy et al., 2007) suggest that rooms are occupied 50% of the time on average, rather than the 60%-80% (see Table 3.5), which is typical for the ITC2007 problems.

We also find that the scale of ITC2007 problems covers small to medium size
problems, but does not cater to problems faced by large institutions. The largest
ITC2007 problem (comp07) features 131 courses with 434 events and 20 rooms,
which is significantly smaller than the problem faced by the University of Auckland,
as shown in Tables 3.1 and 3.4.

As far as quality measures are concerned, we find that using a soft limit for
room capacity (which features in all five specifications), is less realistic than a hard
limit. The majority of rooms will have a certain number of fixed seats which cannot
easily be increased, providing a natural hard limit. In the case of the University of
Auckland, the number of students cannot legally exceed the number of seats. The
“soft” undesirability of a near-full room can be modelled as an event-based solution
quality measure similar to spare seat robustness.

For the UD5 specification, the quantification of the travel distance penalty is also
unusual. A penalty is applied when consecutive events from the same curriculum are
held in different buildings. However, the penalty is applied for each curriculum the
events feature in. Pairs of courses may exist together in more than one curriculum,
which means the penalty for a particular set of events is multiplied by the number
of curricula they both appear in. This weighting is arbitrary, particularly because
the problems include redundant curricula which are dominated by other curricula
i.e. they feature a subset of the courses of another curriculum. These dominated
curricula have no effect on any constraint or quality measure, except to alter the
quantification of the travel distance penalty. Potentially, the travel distance penalty
could be weighted by the number of students influenced, or the distance between
buildings (for a problem with more detailed data).

Finally, we would like to discuss the specific choices of quality measures. It
is acknowledged by the competition organisers (Bonutti et al., 2012), and many
researchers in the field, that there is no universal measure of timetabling quality.
Not only do different rankings of importance of commonly-desired timetable fea-
tured exist, but there can even be contradicting views of whether a given feature is desirable. For example, two ITC2007 quality measures relating to curriculum compactness (which favour a “bunching” of events) may be considered undesirable by some timetablers, who prefer a wider spread of events throughout the day. Furthermore, even for the same set of priorities there may be many equally valid ways to define or quantify a quality measure in practice. As mentioned by Burke et al. (2010a), if there are many similar rooms in one building or location, it may be more important to teach all events of a course in one of these rooms rather than measuring stability with respect to a specific room. We agree, and demonstrate in Chapter 5 that optimising room preference (an event-based measure) is significantly easier than optimising course room stability (which is pattern-based). In our approach, the first priority for a feasible timetable is maximising room preference, which can be solved efficiently. Course room stability is then improved, but only without reducing the total room preference. It is also likely that maximising room preference will implicitly minimise students’ travel distance, since courses within a curriculum are typically taught by the same department or faculty.

We use this example of measuring course room stability to demonstrate how different quantifications of quality may be equivalent from a practical perspective, yet differ substantially in difficulty when solving the problem with a particular approach. This is consistent with the sentiment of the ITC2007 competition organisers, that although an algorithm outperforms another on a certain set of benchmarks, this does not imply that it is a superior algorithm in general (McCollum et al., 2010).

Recent work by Smith-Miles (2014) develops methods for the generation of benchmark datasets for timetabling, which ensure a good representation of practical problems. Initially the key “features” of datasets which affect algorithmic performance are identified e.g. room utilisation. Using principal component analysis, the features of a range of practical problems can be used to generate datasets which
resemble this structure (and the range of structure within practical timetabling). Specific knowledge of the features associated with each dataset also allows us to compare the relative advantages of different algorithms on different types of problems. The results of this study formalise our concerns about the ITC2007 datasets. The methodology is explained in the context of more general graph colouring and examination timetabling in Smith-Miles et al. (2014).
In this chapter, we propose a method for finding a time assignment for a set of university courses. Although our preferred timetabling philosophy involves using a faculty-generated time assignment, this method has been developed for situations where a starting time assignment is not available. This may be because a faculty is new, has been re-structured, or does not have the resources or desire to provide a time assignment.

When this problem is solved in practice, it is anticipated that solutions will be subject to modification, in collaboration with each individual faculty. This allows faculties to incorporate their own requirements and preferences relating to the time assignment for their courses. Such requirements may be difficult to elicit or formalise a priori, although not necessarily difficult to model. Faculty preferences may include teaching one course before another during the week, honouring staff teaching-time preferences, or granting exceptions to regular timetabling constraints. An example of a constraint exception may be if a single teacher is capable of teaching two events at once, such as supervising two nearby tutorials.

In this chapter, we generate a starting time assignment for the 2013 data from
the University of Auckland (see Section 3.1.3). This is implemented for all faculties together, rather than individually. We incorporate the constraints and quality measures which are considered by the Timetable Services Office (TSO), but are unable to tailor each faculty time assignment without extensive collaborative work. The primary purpose of this chapter is to find a feasible time assignment on a modern timetabling dataset, which we also use in later chapters to demonstrate the room assignment and minimal perturbation algorithms.

Section 4.1 gives an introduction to concepts and tradeoffs inherent in generating a time assignment, and how they can be considered. Section 4.2 discusses the theory and practice of how to ensure a near-feasible room assignment will exist for the generated time assignment. Section 4.3 outlines the general integer programming model used for time assignment in this thesis. Section 4.4 presents the computational results of solving this model for different quality measures and room feasibility constraints. Finally, Section 4.5 provides a discussion of the proposed method, including potential improvements to the algorithmic performance, and its application to generating tailored faculty-specific time assignments.

4.1 Introduction

As part of course timetabling, the primary focus of solving a time assignment problem is to find a feasible time period for every event. Fundamental constraints include ensuring that there are no conflicts between courses from any curriculum, or courses taught by the same teacher. Other equally important constraints will typically also exist, relating to the time patterns of events from a common course or curriculum. For example, it is commonly required that a course which consists of a number of lectures each week will have each lecture on a different day of the week. An obvious exception is when the course requires teaching to be in the form of long events
(e.g. two-hour lectures), in which case the time contiguity of events must be upheld.

Quality within the time assignment may be measured in many ways, some of which have been discussed in Section 2.2. Each quality measure can be modelled as a hard constraint, or as an objective to be optimised.

A common quality consideration in the time assignment is a notion of “regularity” across the week. This may be achieved by maximising the time stability for each course, where all course events are ideally taught at the same time of day. As previously demonstrated in Tables 3.3a and 3.3b, ensuring time stability may conflict with time equity, if certain time periods are known to be undesirable.

A more general form of time stability is referred to as course compactness, which measures the span in the time of day for a course’s events across the week. Perfect course compactness occurs when all events are taught at the same time of day (on different weekdays), which also corresponds to strict time stability. However, the allowance for imperfect compactness is typically important, as it may not be possible to achieve strict time stability for all courses. An example is provided in Table 4.1a, where 5 courses each consist of 3 lecture events per week. If these courses are constrained to enforce strict time stability, and are not permitted to overlap in time with one another (e.g. due to being part of a curriculum), they must be spread across 5 hours. This is potentially a poor time assignment, as it may cause infeasibility for a more constrained problem. For example, another course of 3 events cannot be placed, despite 10 unused time periods. When this requirement is relaxed, we can use the time assignment shown in Table 4.1b. Although courses ‘B’ and ‘D’ do not use the same time of day for all events, the placement of their events can still be considered “compact” (e.g. within 2 hours). Most importantly, this time assignment allows us improved flexibility to achieve a much more compact utilisation of the available teaching hours. Events from other courses may be compactly placed in the 12pm and 2pm time periods as indicated.
4.1. INTRODUCTION

However, the notion of compactness so far has only been applied at the level of an individual course. A broader approach to compactness which is commonly used, is to address the curriculum compactness, where events of all courses from a curriculum are used to measure the compactness. This allows us to consider the regularity of event starting times across the week, and also the number of hours that students are required to be at the university campus.

The compactness of a curriculum is measured by the span of starting times for all weekly events in courses from this curriculum. A span of 0 hours would correspond to all events starting at the same time of day. This is not usually possible, as it would require a maximum of five events across all curriculum courses. A typical curriculum may include 3 courses, each with 3 events. A span of 4 hours corresponds to the earliest event starting 4 hours earlier than the latest (not necessarily on the same day). For example, if all events start at 8am, 9am, 10am, 11am or 12pm. This can be considered to be an acceptable level of compactness. A span of 9 hours corresponds to a curriculum with events starting at 8am and 5pm.

Revisiting the previous example from Table 4.1, the relaxation of strict time stability also made possible an improvement in the curriculum compactness. Table 4.2 demonstrates the same comparison as Table 4.1, except it also avoids the potentially undesirable situation of a two-day gap in teaching for any course. An example is shown for course “B” in Table 4.1, which is taught on Monday, Thursday and Friday.
4.2. ROOM ASSIGNMENT FEASIBILITY

<table>
<thead>
<tr>
<th>Time</th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thu</th>
<th>Fri</th>
</tr>
</thead>
<tbody>
<tr>
<td>10am</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11am</td>
<td></td>
<td></td>
<td>B</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>12pm</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1pm</td>
<td>D</td>
<td>D</td>
<td></td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>2pm</td>
<td></td>
<td></td>
<td>E</td>
<td>E</td>
<td>E</td>
</tr>
</tbody>
</table>

(a) Strict time stability

<table>
<thead>
<tr>
<th>Time</th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thu</th>
<th>Fri</th>
</tr>
</thead>
<tbody>
<tr>
<td>10am</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>11am</td>
<td>C</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>C</td>
</tr>
<tr>
<td>12pm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1pm</td>
<td>D</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2pm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(b) Course compactness

Table 4.2: Limitations of Strict Time Stability (reduced day gaps)

The final quality measure we mention is the concept of time preference, where each course (or associated teacher) may have certain preferences for time periods which are more or less favourable. As depicted in the examples from Section 3.1.2, teaching at any 8am time period is considered undesirable. This is also commonly the case for teaching at 5pm, particularly on Friday. Although the use of these “outermost” time periods may be necessary from the perspective of finding a feasible time assignment (and a room assignment), as a quality measure we can prefer these periods are avoided.

4.2 Room Assignment Feasibility

Because we solve the time assignment problem before the room assignment problem (rather than simultaneously), we are interested in finding a time assignment which has a feasible or near-feasible room assignment. The degree of room assignment feasibility (measured by the number of events optimally assignable to a feasible room) can be treated as a quality measure within the time assignment. A lesser degree of room assignment feasibility will ultimately require greater perturbation to the time assignment, causing a degradation to the quality measures used in the generation of the time assignment.

To a certain extent, the existing constraints in the time assignment which ensure conflict-free curricula and teacher assignments already enforce a spread of events.
However, in practice these constraints can still permit overloading of certain time periods and result in a significantly infeasible room assignment.

The difficulty of guaranteeing a feasible room assignment for a time assignment problem depends on whether the problem involves long events, and the complexity or diversity of available rooms. The simplest case occurs when there are no long events, and all rooms are homogenous. In the absence of long events, the room-feasibility of each time period may be addressed independently. As a result, for each time period, room feasibility can be ensured with a constraint requiring that the number of assigned events does not exceed the number of available rooms.

Adding complexity, in much of the timetabling literature, the feasibility of an event-to-room assignment is dependent on the size of the room (i.e. number of seats) being at least equal to the size of the event (i.e. number of students). When this complexity is introduced, it is useful to graphically represent the feasibility relationship between all events and rooms for a given time period. Figure 4.1a provides a simple example where sets of events and rooms are represented as circular and rectangular nodes respectively. The arcs establish the relationship that all rooms which are descendants of an event, are feasible for that event. Each node may represent multiple events or rooms which are of the same size. Furthermore, we may merge any event nodes which have a common set of room descendants and any room nodes which have a common set of event ancestors. The merged graph is shown in Figure 4.1b. To ensure room feasibility, we add a constraint for each merged set of events in each time period, so that the number of events of at least this size does not exceed the number of available rooms of at least this size. This set of events and rooms is called a room group.

For even more complex practical problems, the feasibility of an event-to-room assignment will also depend on the attributes of a room. A room attribute is any characteristic (or feature) of a room which affects its feasibility for any event. For
4.2. ROOM ASSIGNMENT FEASIBILITY

(a) Before merging

(b) After merging

Figure 4.1: Room and Event Size Hierarchy
example, an event may require a room with a piano, a demonstration bench, or one which is located within a certain region of the campus. A room is only feasible for an event if it is at least equal in size, and possesses a superset of the event’s required attributes. Figure 4.2 provides a simple example to represent a more general feasibility relationship. When considering an arbitrary set of room attributes, it is much more difficult to ensure room feasibility. Lach and Lübbecke (2008) proved in Lemma 6 that the number of constraints required can be exponential in the number of attributes. Therefore, for practical problems with a large set of different attributes, it may not be possible to add the full set of room group constraints for each time period to ensure room feasibility.

![Figure 4.2: Room and Event Size and Attribute Set Hierarchy](image)

The final complication for room assignment feasibility is that of long events,
which cause interdependence between time periods. Instead of treating each time
period separately, the strict room stability requirement on a long event cannot be
simply enforced by adding constraints to the time assignment. Determining whether
a feasible room assignment exists, in the presence of long events, is an NP-hard
problem, as proven by Carter and Tovey (1992). In Section 5.3, we expand on this
result and its implications.

The problem faced by the University of Auckland includes both a complex set
of room attributes and room stability requirements on long events. However, al-
though it is not possible to guarantee a feasible room assignment when solving the
time assignment, this is not necessarily required. As with faculty-generated time
assignments, we expect to need to perturb the time assignment to find room fea-
sibility. Furthermore, the dynamic nature of timetabling data can cause feasibility
in the room assignment to be lost (and then repaired) continually as unpredictable
changes to the data occur.

Therefore, for the purposes of generating a starting time assignment, our fo-
cus must be on satisfying time assignment constraints while providing for a “near-
feasible” room assignment. As previously mentioned, the constraints on curricula
and teacher conflicts provide a degree of spread in the time assignment, which we
supplement by choosing a subset of available room group constraints.

4.3 Mathematical Model

Based on the concepts introduced in Sections 4.1 and 4.2, we present an integer
programming formulation for the time assignment problem. Let the variable $x_{et}$
take the value 1 if event $e \in E$ is taught in time period $t \in T_e$. All notation is
defined in the List of Notation (pg. xvii–xviii). The integer programme we solve is
composed of either (4.1) or (4.2) for an objective function, and subject to constraints
Objective (4.1) models the time preference, where each event has a coefficient expressing its preference for that time period. Objective (4.2) models the compactness for a subset of curricula, where we minimise the span between the earliest and latest hour for all events across the week of each curriculum.

Constraints (4.3) ensure that each event is assigned to a feasible time period. Constraints (4.4) ensure that no more than one event (or one long event) from
each course is assigned on any day. Constraints (4.5) ensure that no more than one event from each curriculum is assigned in any time period. Constraints (4.6) ensure that no more than one event taught by each teacher is assigned in any time period. Constraints (4.7) apply the room feasibility group constraints, ensuring that no more events with a particular size and set of attributes are assigned in any time period than there are rooms with those attributes. Constraints (4.8) enforce the time contiguity requirement on the constituent events of a long event. Finally, constraints (4.9) and (4.10) ensure that the curriculum hour variables are correctly tied to the assignment variables.

4.4 Results for UoA 2013

To test this model and explore the impact of different quality measures and room group feasibility constraints, we run several computational tests. In Section 4.4.1, we place an emphasis on finding a high quality feasible time assignment, whilst in Section 4.4.2 we include the additional room feasibility constraints.

All tests in this section are run using Gurobi 5.6 on 64-bit Ubuntu 14.04, with a quad-core 3.5GHz processor (Intel i5-4690). All parameters are set to their defaults, except the “MIP Focus” which is set to 1, to focus the search of finding feasible solutions rather than proving optimality (Gurobi Optimization, Inc., 2015). The time limit is set at 3600 seconds.

4.4.1 Time Assignment Quality

In this section, we test the model (4.3)–(4.12) with the omission of the room group constraints (4.7). These enable us to focus on finding a high-quality feasible time assignment.

We initially test with no objective function (i.e. with the goal of feasibility),
followed by applying each of the two objective functions from Section 4.3, time preference (4.1) and curriculum compactness (4.2). Finally, we test the model with both quality measures, by optimising the curriculum compactness followed by the time preference. This is an application of lexicographic optimisation, where a constraint is introduced to the latter time preference model to maintain the quality of the curriculum compactness measure.

The time preference quality is measured as 1 for events starting at 9am to 4pm, and 0 for events starting at 8am and 5pm. The measurement of curriculum compactness is based on a subset of curricula; those which contain courses all from one faculty, and involve at least 200 students for two or more of the courses. This subset represents the most important main study pathways from each faculty.

Tables 4.3a and 4.3b present the results of solving 4 time assignment problems for each of Semester 1 and 2 of 2013 at the University of Auckland. Column 1 specifies which quality measure has been applied for each time assignment problem. Columns 2, 3 and 4 give the results of solving this model, in terms of each objective function value and the solve time, with a maximum of 3600 seconds. Each model involves approximately 150,000 variables and 140,000 constraints. Note that the time preference (TP) objective is modelled as a maximisation, whereas the curriculum compactness objective is modelled as a minimisation (of the span of events). Objective values in brackets signify that this quality measure has not been optimised in this model, and is computed at its incidental value.

The results on both semesters’ datasets are interpreted together, as they demonstrate the same trends. Firstly, we consider the impact of different quality measures. It is demonstrated to be easy to find a feasible solution to the time assignment problem (i.e. which assigns all events), in less than 10 seconds. Maximising the time preference increases the difficulty. However, the solve times remain very manageable, at less than one minute. The optimal values for the time preference correspond
4.4. RESULTS FOR UOA 2013

<table>
<thead>
<tr>
<th>Quality Measure</th>
<th>Objective</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>— / feasibility</td>
<td>(2851)</td>
<td>(1267)</td>
</tr>
<tr>
<td>Time Preference, TP</td>
<td>3480</td>
<td>(927)</td>
</tr>
<tr>
<td>Curr. Compactness, CC</td>
<td>(2687)</td>
<td>313</td>
</tr>
<tr>
<td>CC $\Rightarrow$ TP</td>
<td>3467</td>
<td>313</td>
</tr>
</tbody>
</table>

(a) Semester 1

<table>
<thead>
<tr>
<th>Quality Measure</th>
<th>Objective</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>— / feasibility</td>
<td>(3215)</td>
<td>(911)</td>
</tr>
<tr>
<td>Time Preference, TP</td>
<td>3950</td>
<td>(697)</td>
</tr>
<tr>
<td>Curr. Compactness, CC</td>
<td>(3093)</td>
<td>323</td>
</tr>
<tr>
<td>CC $\Rightarrow$ TP</td>
<td>3093</td>
<td>323</td>
</tr>
</tbody>
</table>

(b) Semester 2

Table 4.3: UoA 2013 Basic Time Assignment Results

to all events assigned to the central 80% of time periods. In the absence of other quality measures or room constraints, this demonstrates it is possible to assign all events to a subset of time periods without causing a curriculum conflict.

The curriculum compactness quality measure is notably much more computationally difficult. Even with a reduced subset of curricula, we reach the time limit when solving these problems. However, the feasible (yet suboptimal) solutions produced have substantially improved curriculum compactness relative to when this quality measure is not considered.

To understand whether the curriculum compactness is sufficiently high in the final incumbent solution, we look at the average span in start time for events in each curriculum. For the first semester, 173 curricula are in the compactness subset, resulting in an average span of 1.8 hours (313/173) in start time for the models which address this quality measure. For the second semester with 120 curricula in
this subset, there is an average span of 2.7 hours (323/120) in start time. These are both considered to be of high quality by the University.

4.4.2 Room Assignment Feasibility

In this section, we test the model (4.3)–(4.12) with the inclusion of the room group constraints 4.7. This allows us to test the impact of these constraints on the quality measures, solve time, and room assignment feasibility. To apply these constraints, in each time period, one room group is generated for each merged event node. The objectives used are the same as in Section 4.4.1.

Tables 4.4a and 4.4b present the results of solving the time assignment problems from Tables 4.3a and 4.3b with the additional room group constraints. The results from Tables 4.3a and 4.3b are also included, to assist with comparison.

Columns 1 and 2 specify, for each time assignment problem, which quality measure has been used and the number of room group constraints that have been applied. Columns 3 and 4 give the results of solving this model, in terms of each objective function value and the solve time, with a maximum of 3600 seconds. Finally, column 5 specifies the number of events which can find a feasible room for this time assignment, expressed as an absolute number and as a percentage of all events. This quantification of room assignment feasibility is not easily calculated from the output of the time assignment, and is computed separately using techniques from Chapter 5.

Initially, we compare the effect of the room group constraints on the room feasibility. For each of the quality measures in both semesters, we can observe that the inclusion of room group constraints improves the room feasibility. The effect is particularly significant in the case of time preference. This is achieved without any cost to the optimal time preference objective, which may initially seem surprising. However, this is due to the flexibility provided in the room group constraints. Al-
### 4.4. RESULTS FOR UOA 2013

<table>
<thead>
<tr>
<th>Quality Measure</th>
<th>Room Group Constraints</th>
<th>Objective</th>
<th>Time (s)</th>
<th>Room Assignment</th>
<th>Feasibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>— / feasibility</td>
<td>0 (2851)</td>
<td>(1267)</td>
<td>8.1</td>
<td>4382</td>
<td>(97.6%)</td>
</tr>
<tr>
<td>— / feasibility</td>
<td>488 (2847)</td>
<td>(1252)</td>
<td>27.5</td>
<td>4474</td>
<td>(99.6%)</td>
</tr>
<tr>
<td>Time Preference, TP</td>
<td>0 3480 (927)</td>
<td>10.4</td>
<td>3890</td>
<td>(86.6%)</td>
<td></td>
</tr>
<tr>
<td>Time Preference, TP</td>
<td>488 3480 (929)</td>
<td>60.8</td>
<td>4334</td>
<td>(96.5%)</td>
<td></td>
</tr>
<tr>
<td>Curr. Compactness, CC</td>
<td>0 (2687) 313 3600.0</td>
<td>4184</td>
<td>(93.2%)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Curr. Compactness, CC</td>
<td>488 (2691) 326 3600.0</td>
<td>4450</td>
<td>(99.1%)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CC ⟷ TP</td>
<td>0 3467 313 3600.0</td>
<td>4042</td>
<td>(90.0%)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CC ⟷ TP</td>
<td>488 3464 326 3600.0</td>
<td>4457</td>
<td>(99.3%)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(a) Semester 1

<table>
<thead>
<tr>
<th>Quality Measure</th>
<th>Room Group Constraints</th>
<th>Objective</th>
<th>Time (s)</th>
<th>Room Assignment</th>
<th>Feasibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>— / feasibility</td>
<td>0 (3215)</td>
<td>(911)</td>
<td>8.2</td>
<td>5143</td>
<td>(96.6%)</td>
</tr>
<tr>
<td>— / feasibility</td>
<td>502 (3207)</td>
<td>(918)</td>
<td>30.8</td>
<td>5264</td>
<td>(98.9%)</td>
</tr>
<tr>
<td>Time Preference, TP</td>
<td>0 3950 (697)</td>
<td>40.9</td>
<td>4568</td>
<td>(85.8%)</td>
<td></td>
</tr>
<tr>
<td>Time Preference, TP</td>
<td>502 3950 (717)</td>
<td>272.4</td>
<td>5085</td>
<td>(95.5%)</td>
<td></td>
</tr>
<tr>
<td>Curr. Compactness, CC</td>
<td>0 (3093) 323 3600.0</td>
<td>4879</td>
<td>(91.7%)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Curr. Compactness, CC</td>
<td>502 (2978) 436 3600.0</td>
<td>5157</td>
<td>(96.9%)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CC ⟷ TP</td>
<td>0 3093 323 3600.0</td>
<td>4879</td>
<td>(91.7%)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CC ⟷ TP</td>
<td>502 3913 435 3600.0</td>
<td>5212</td>
<td>(97.9%)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(b) Semester 2

Table 4.4: UoA 2013 Full Time Assignment Results

though it is not likely to be possible to find a time and room assignment which only uses the central time periods, the room group constraints are more permissive than finding a feasible room assignment.

For the curriculum compactness, the addition of room group constraints appears to slightly reduce the quality of the best solution found. One may expect these constraints to increase the spreading of events in each day and thus decrease the compactness. However, we can only observe the best solution after 3600 seconds, which is not necessarily precise with the optimal value.
Observing the final model for each semester, the solution generated is of high quality for both the time preference and curriculum compactness, and features a high degree of room feasibility. As a result, we can be satisfied that this solution fulfils the objective of this chapter.

4.5 Discussion

The results in this chapter demonstrate that it is possible to generate a feasible time assignment for all faculties of a large university, while considering quality measures. Although some quality measures are computationally difficult to address, it is important to place these findings in the context of our broader timetabling methodology, and the problem in practice.

While some universities use a fixed faculty-generated time assignment, and others involve no faculty involvement (other than the collection of course data), our timetabling philosophy favours a cooperative approach to the time assignment problem, so that complex faculty requirements can be addressed. As a result, in practice we favour solving each faculty time assignment problem separately.

For the purposes of this work, we have dealt with all faculties simultaneously, in a monolithic formulation. As a result, finding a feasible solution to all timetabling constraints is significant in itself. First and foremost, timetabling is a problem of feasibility, and quality measures are achieved only where possible. For this reason, the starting time assignments we have generated are considered sufficient.

An interesting comparison can be made in the room assignment feasibility between our generated time assignments from 2013, and those manually generated from 2010. Our time assignment solutions allow between 95% and 99% room feasibility (when room group constraints are used), which are suitably high to be repaired without major disruption (see Chapter 6). However, the manually generated start-
ing time assignments for both semesters of 2010 allowed 99% room feasibility. This is likely to be because the time assignment had been “rolled-forward” from the timetable implemented the previous year, for which a complete room assignment existed.

In a practical implementation, addressing each faculty time assignment independently raises issues when the time assignments are combined. For example, conflicts may arise relating to inter-faculty staff and inter-faculty curricula. For the University of Auckland, there are a sufficiently low number of inter-faculty activities that these conflicts can be addressed manually, as was done in 2010. However, we would be able to use optimisation algorithms to solve this problem, either by resolving individual faculty time assignments (with conflicting time periods removed), or by using an optimisation algorithm to minimally perturb each time assignment (see Chapter 6). There is also scope for more advanced methods such as Dantzig-Wolfe decomposition, where each constraint block corresponds to a faculty’s internal requirements, and inter-faculty staff and curricula form the coupling constraints.

Another issue which arises when each faculty is independently addressed is the room assignment feasibility. The room group constraints used in this chapter rely on considering events from all faculties fitting into all campus rooms. Each faculty will use many rooms which are also required by other faculties, such as large lecture theatres. One alternative method is to apply constraints in each time period so that a maximum percentage of each faculty’s events are taught in this time period. This ensures a spread of events, which indirectly improves room feasibility.

Finally, because it is not always possible to solve our time assignment model to optimality (e.g. when dealing with curriculum compactness), there is substantial research potential in addressing this problem. Although the metaheuristic literature features algorithms which generate a time assignment (see Section 2.3.2), we have not encountered any which deal exclusively with the time assignment. This
may be because it is simpler to write a metaheuristic to address the time and room assignment simultaneously, rather than to write separate algorithms for room assignment and any perturbation required. As shown in subsequent chapters, we have an implementation of these latter algorithms, which can in fact be solved with exact methods. Therefore, although there is significant scope in pursuing more complex optimisation algorithms for the time assignment problem, we would be interested to see the results of an advanced metaheuristic approach.
The Classroom Assignment Problem

In this chapter, we present an integer programming method for finding a high quality room assignment for a set of university courses. This method is designed to be versatile in terms of modelling power, and to maintain tractability on large instances encountered in practice.

The classroom assignment problem is a component of nearly every timetabling system presented in the literature, although it is most commonly solved together with the time assignment problem. We address the room assignment problem separately, which is most appropriate for the complex multi-objective problems at the University of Auckland and the Technical University of Denmark. We also test our method on the Udine benchmark instances, which provide a comparison between our method and others in the literature.

Through this work, we are able to expand upon previous results into the difficulty of classroom assignment problems in general. Although most variants of the classroom assignment problem found in practice are NP-hard, we demonstrate why many instances can be solved efficiently.

In Section 5.1, we provide a general introduction to the concepts of modelling
classroom assignment problems, and present the theoretical foundations of these problems using graph theory. In Section 5.2, we introduce a simple example of a classroom assignment problem, outline our a general integer programming model, and define some common quality measures. In Section 5.3, we provide an insight into the matrix structure of the integer programme and demonstrate how fractions can arise in the linear programming relaxation. We also interpret these insights using graph theory, and apply this knowledge to help identify which room assignment problems will be easier or more difficult to solve. In Section 5.4, we present the results of our method on data from the University of Auckland in 2010 and 2013. In Section 5.5, we adapt our integer programme for the problem at the Technical University of Denmark, and present results. In Section 5.6, we present results on two formulations of the Udine benchmark problems (UD2 and UD5) and discuss our method in the context of other heuristic approaches.

5.1 Introduction

The classroom assignment problem attempts to find a feasible assignment for a set of events to a set of rooms. For a room to be considered feasible for an event, it must be of sufficient size, and feature the requested set of attributes. It is also required that each room may only be occupied by a maximum of one event in each time period. This constraint is notably not present for examination timetabling, where events from multiple courses may be permitted to share a room.

When solving a room assignment problem, it is anticipated that there may not exist a complete assignment which provides a room for all events. In this case, we typically wish to find the best partial room assignment.

One situation where we may be limited to finding a partial room assignment occurs when the time assignment includes overly congested time periods. Due to
a fixed time assignment, unassigned events in congested time periods cannot be
assigned to available rooms in less congested time periods. For timetabling at the
University of Auckland, a partial room assignment is considered infeasible, which
we address by solving a minimal perturbation problem (see Chapter 6). A similar
problem in the time assignment problem (i.e. a partial time assignment) is less likely
to occur, as it relies on less predetermined data (only the curricula and teachers).

Another situation where it is not possible to assign all events to a room can occur
when only a subset of available rooms are included in the room assignment process.
As introduced in Section 3.2, this is an intentional part of the room assignment
process at DTU, and is addressed by manually assigning the unassigned events to
the additional rooms. This problem does not apply to the time assignment, where
all feasible time periods are included in the timetabling process, whether they are
desirable or undesirable.

Quality in the room assignment is discussed in general in Section 2.2, and for
specific datasets in Chapter 3. Most research in the timetabling literature places
little focus on room assignment quality, as a result of the problem being solved
together with the time assignment. When room assignment quality is addressed,
it is limited to a single measure of quality. Here we explain how different types of
quality measures affect the modelling of the room assignment problem.

The most elementary formulation of the room assignment problem addresses a
single, simple measure of quality, where a cost is defined for each possible event-to-
room assignment. There must also be no contiguous room stability requirement on
long events. This formulation is used by Ferland and Roy (1985), and Al-Husain
et al. (2011).

For this simple formulation, room assignment for each time period can be inde-
dependently modelled as finding a maximum weighted bipartite matching between the
set of events and rooms. A simple example for 3 events and 3 rooms is shown in
Figure 5.1a, where each node represents an event or room, and each arc represents an event-to-room assignment with an associated cost. The same problem may instead be modelled as finding a maximum weighted independent set in the graph of all feasible event-to-room assignments, for each time period. In this case, each node is associated with the cost of a feasible assignment, and each arc precludes a solution from including both nodes together. Figure 5.1b depicts this graph, where each group of nodes corresponds to the possible assignments for each event, and forms a clique (shown with solid arcs), ensuring this event can only be assigned to one room. Similarly, each room may only be used by one event, and all nodes for each room also form a clique (shown with dashed arcs). It is noted that the independent set graph in Figure 5.1b forms the “line graph” of the bipartite matching graph in Figure 5.1a (Schrijver, 2003).

The problem becomes more complex if we introduce the additional constraint of contiguous room stability on long events. This can be represented using a more general maximum weighted independent set problem which includes the interdependencies between adjacent time periods. An example problem is shown in Figure 5.2. In this example, events $e_2$ and $e_5$, from adjacent time periods $t_1$ and $t_2$, form a long event. These events require the same room, so they are represented together in each node. This means that finding a maximum weighted independent set on this graph (a room assignment solution for this formulation) cannot be addressed one time period at a time. However, the dependencies are limited to blocks of adjacent time periods on the same day. This formulation is used by Glassey and Mizrach (1986), Gosselin and Truchon (1986) and Carter (1989). The work of Carter (1989) is particularly noteworthy as the author proposes several room assignment quality measures. However, these are ultimately scalarised to a single objective function when solving the problem, due to the available computational resources at the time of development.
5.1. INTRODUCTION

Further complexity is introduced when considering a quality measure which causes interdependencies between any subset of time periods, rather than just a contiguous block of time periods. The most common example, course room stability, minimises the number of different rooms used by each course. This problem can again be modelled as a maximum weighted independent set problem, as shown for an example problem in Figure 5.3. In this example, events $e_3$ and $e_9$ from non-adjacent time periods $t_1$ and $t_n$, are both part of the same course. It is desirable for these events to be taught in the same room, which requires a common set of nodes to represent this situation. However, the original nodes remain, as these are still feasible assignments, and will be used if it is not possible to offer a stable room to all
course events. Other events in this example are part of courses which include events in other time periods. This is represented by dashed arcs which extend to connect to events from many other time periods. Modelling the course room stability generates a substantial number of new nodes, and a large amount of connectedness between time periods. These factors further complicate the maximum weighted independent set problem, making it more difficult to solve.

As part of a broader work, Qualizza and Serafini (2005) propose an integer programme to solve this problem, although they do not include results. Lach and Lübbecke (2012) also propose an integer programme which models course room stability as part of their solution to the problems posed in the 2007 International Timetabling Competition (Di Gaspero et al., 2007) (see Section 3.3).
Finally, as introduced in Sections 3.1 and 3.2, practical problems often require consideration of multiple measures of quality. These are not easily represented using a well-known graph theoretical approach. Although multiple objectives may be scalarised into a single objective function, this approach has limitations. In the remainder of this chapter, lexicographic optimisation is used to address quality measures which feature a strict hierarchy of importance. Although conceptually simple, this approach is applicable to practical problems. However, more complex methods involving quality tradeoffs are demonstrated and discussed in Chapter 7.
5.2 Mathematical Model

In this section, we introduce a small example of a room assignment problem, and demonstrate how this can be modelled as a maximum set packing problem (Nemhauser and Wolsey, 1988). To solve this problem, we propose an integer programming-based approach, which provides a certainty of the feasibility (or infeasibility) of the room assignment and of the solution quality.

To handle different measures for quality, our model is solved sequentially for a prescribed series of solution quality measures. The quality with respect to each measure is preserved in subsequent solutions using an explicit constraint. In the terminology of multi-objective optimisation (Ehrgott, 2005), this is a lexicographic optimisation algorithm, which is guaranteed to find a Pareto optimal solution i.e. no quality measure can be improved without reducing the quality of at least one other measure.

As it may not be possible to find a room for all events, our approach will find an efficient partial room assignment which makes the best possible use of the available rooms. For the University of Auckland case, it will also identify specifically which time periods are over-booked and which sizes (and types) of rooms are in shortage in each period. This information is important when timetablers decide how to modify the timetable, and the related analytics may also be of use to other administrative parties to understand the bottlenecks in the system. For the Technical University of Denmark, our approach will find the best quality partial room assignment, for their prescribed quality measures.

5.2.1 Example Problem

Table 5.1 presents the fundamental data on the courses and events for an example problem. Precise definitions for the terminology and notation used in column
headers is provided in Section 5.2.2.

<table>
<thead>
<tr>
<th>Course (c)</th>
<th>Size (\textit{size}_c)</th>
<th>Room Attributes (\textit{att}_c)</th>
<th>Events (e)</th>
<th>Time Period (\textit{t}_e)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(c_1)</td>
<td>125</td>
<td>_</td>
<td>(e_1)</td>
<td>(t_1)</td>
</tr>
<tr>
<td>(c_2)</td>
<td>60</td>
<td>Demonstration Bench</td>
<td>(e_1)</td>
<td>(t_1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(e_2)</td>
<td>(t_2)</td>
</tr>
<tr>
<td>(c_3)</td>
<td>60</td>
<td>_</td>
<td>(e_1)</td>
<td>(t_1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(e_2)</td>
<td>(t_2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(e_3)</td>
<td>(t_3)</td>
</tr>
<tr>
<td>(c_4)</td>
<td>60</td>
<td>Demonstration Bench</td>
<td>(e_1)</td>
<td>(t_2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(e_2)</td>
<td>(t_3)</td>
</tr>
</tbody>
</table>

Table 5.1: Course and Event Data

Table 5.2 presents the data on which rooms are available for this example. Each room has a size (i.e. the maximum student capacity), a set of room attributes, and a set of time periods when this room may be used.

<table>
<thead>
<tr>
<th>Room (r)</th>
<th>Size (\textit{size}_r)</th>
<th>Room Attributes (\textit{att}_r)</th>
<th>Available Time Periods (\textit{T}_r)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(r_1)</td>
<td>150</td>
<td>_</td>
<td>(t_1, t_2, t_3)</td>
</tr>
<tr>
<td>(r_2)</td>
<td>75</td>
<td>Demonstration Bench</td>
<td>(t_1, t_2, t_3)</td>
</tr>
<tr>
<td>(r_3)</td>
<td>75</td>
<td>Demonstration Bench</td>
<td>(t_1, t_2, t_3)</td>
</tr>
</tbody>
</table>

Table 5.2: Room Data

A simple model for the room assignment problem uses variables corresponding to a feasible event-to-room assignment. However, a more general approach models the assignment of a set of events, or pattern, to a feasible room.

Processing the data from Tables 5.1 and 5.2, we can generate the core problem data for Example 1 in Table 5.3. For each course, we show the time period in which each course event is taught, the feasible rooms for these events (determined by the room size and attributes), and the course patterns (all possible subsets of course events).
5.2. MATHEMATICAL MODEL

Example 1. A small classroom assignment problem

<table>
<thead>
<tr>
<th>Course (c)</th>
<th>Pattern (p)</th>
<th>Room (r)</th>
</tr>
</thead>
<tbody>
<tr>
<td>c₁</td>
<td>{e₁}</td>
<td>r₁</td>
</tr>
<tr>
<td>c₂</td>
<td>{e₁, e₂}</td>
<td>r₂, r₃</td>
</tr>
<tr>
<td>c₃</td>
<td>{e₁, e₂, e₃}</td>
<td>r₁, r₂, r₃</td>
</tr>
<tr>
<td>c₄</td>
<td>{e₁, e₂}</td>
<td>r₂, r₃</td>
</tr>
</tbody>
</table>

Table 5.4: A Feasible Solution

Table 5.4 gives a feasible solution to this problem, where patterns are assigned to feasible rooms, and the pattern-to-room assignments ensure that each room is used at most once in each time period. If our objective is to maximise the number of events assigned, this solution is clearly optimal, with all events assigned. If we also want to minimise the number of different rooms used by each course, the solution can be improved by assigning pattern \{e₂, e₃\} of course c₃ to room r₁.

5.2.2 Notation

To model the room assignment problem, we are required to formalise the definition of specific terminology and notation.

A meeting pattern \( p \) is defined to be a subset of events for a given course that will
be assigned the same room. For course $c$, let $P_c$ denote the set of all its patterns, the power set of course events. Let $\text{length}_c$ and $\text{length}_p$ denote the number of events in a course and pattern respectively. As a power set, $P_c$ will feature $2^{\text{length}_c} - 1$ elements, which potentially could be large. However, in practice, the number of events per course is usually quite small (for example, averaging between 2 and 3 at the University of Auckland). Let $P$ denote the set of all patterns, i.e. $P = \bigcup_{c \in C} P_c$.

Note that while each pattern $p$ uniquely identifies a set of events, an event is usually in more than one pattern. This is evident in Example 1, where Table 5.3 shows the events in each pattern for all courses. Let $P_e$ denote the set of all patterns which contain event $e$, i.e. $P_e = \{p \in P : e \in p\}$.

Let $R$ denote the set of rooms in the pool of common teaching space, where $\text{size}_r$ and $\text{att}_r$ correspond to the room size and set of room attributes for a room $r$. $R_c$ represents the set of rooms which are suitable for events of course $c$, i.e. $R_c = \{r \in R : \text{size}_r \geq \text{size}_c, \text{att}_r \supseteq \text{att}_c\}$. Using this definition, the course and room data from Tables 5.1 and 5.2 respectively can be processed to generate $R_c$ for each course in Table 5.3. A pattern $p$ of course $c$ will have the set of feasible rooms for this pattern $R_p$, as a subset of $R_c$. For the rooms within $R_c$, a course’s preference for a particular room is given by some preference function $\text{Pref}(c, r)$. This is usually used to place courses into buildings as close as possible to their teaching department, but may be used for any measure of preference e.g. more modern rooms.

Let $A$ denote the set of all room attributes, i.e. $A = \bigcup_{r \in R} \text{att}_r$. In addition to physical room attributes, this set may contain abstract auxiliary attributes to assist with modelling. For example, a room may possess the attribute of being within a given maximum geographical distance from a particular teaching department. Abstract room attributes may also be used if it is undesirable for a course to be assigned to a room with a particular room attribute. In the general case, this can be modelled by generating a complementary room attribute which corresponds to
not possessing’ the undesirable attribute. The set of rooms is thus partitioned by those with the original undesirable attribute, and those with the complementary attribute. In many cases, partitions of the set of rooms already exist, such as when rooms are designated as one of several types. In this case, requesting a room with one attribute automatically precludes being assigned a room with the other attributes.

From the previous chapter, $T$ denotes the set of all usable time periods in the timetabling domain, which are of a common duration (often one hour) and are non-overlapping. For practical problems, we also introduce $T_r$ to denote the set of time periods for which room $r$ is available for teaching. Due to other prescheduled events, every room may have its unique set of available time periods. Each event $e \in E$ occurs during a prescribed time period $t_e$ given by the timetable. For each pattern $p \in P$, let $T_p$ denote the set of time periods this pattern features in, i.e. $T_p = \{t_e : e \in p\}$.

As with the time assignment model in Section 4.3, long events require one event $e \in E$ for each time period they are held in. If a long event requires the same room for its entire duration, contiguous room stability, this is enforced by pruning the set of patterns for this course, $P_c$, to only include patterns which contain all or none of these events. All events of a pattern are assigned to the same room, which enforces the contiguous room stability requirement.

Finally, let $P_{rt}$ denote the set of all patterns which include an event in time period $t$, and for which room $r$ is suitable, i.e. $P_{rt} = \{p \in P : r \in R_p, t \in T_p\}$.

### 5.2.3 Integer Programming Formulation

Using the notation defined in Section 5.2.2, we present an integer programming formulation of a pattern-based set packing model for room assignment. In this formulation, the binary variables $x_{pr}$ are indexed by feasible pattern-to-room assignments. Specifically, let the variable $x_{pr}$ take the value 1 if pattern $p \in P$ is to be
5.2. MATHEMATICAL MODEL

held in room \( r \in R_p \). For a given objective function \( w \) (representing some measure of solution quality), an optimal assignment of patterns to rooms can be determined by solving the following integer programme (5.1)–(5.5).

\[
\begin{align*}
\text{maximise} & \quad \sum_{p \in P} \sum_{r \in R_p} w_{pr} x_{pr} \\
\text{subject to} & \quad \sum_{p \in P_{rt}} x_{pr} \leq 1, \quad r \in R, \ t \in T_r \\
& \quad \sum_{p \in P_e} \sum_{r \in R_p} x_{pr} \leq 1, \quad e \in E \\
& \quad \sum_{p \in P_c} \sum_{r \in R_p} x_{pr} \leq 1, \quad c \in C, \ r \in R_c \\
& \quad x_{pr} \in \{0, 1\}, \quad p \in P, \ r \in R_p
\end{align*}
\]

Constraints (5.2) ensure that at most one event is assigned to each room in each period, while constraints (5.3) ensure that at most one room is assigned for each event. Constraints (5.3) do not need to be met with equality, because it is not assumed that a feasible room assignment for all events will exist. Constraints (5.4) ensure that each course uses at most one pattern per room, i.e. all events from a course that are assigned to a room, should be part of the same (maximal) pattern.

In this chapter, the model is solved in a hierarchical or lexicographic manner i.e. successively for a prescribed series of solution quality measures. This means that one model is solved for each of the different objectives, where each objective function appears as a hard constraint in subsequent optimisations. The particular objectives used and their lexicographic ordering will depend on the needs of a particular institution. For example, given the objective functions and their values \((w_l^t, w_0^t)\), \(l \in \{1, \ldots, k - 1\}\), the \(k\)th integer programme would include constraints (5.6). The effect of treating these constraints as elastic is considered later in Section 7.4.
\[ \sum_{p \in P} \sum_{r \in R_p} w_{pr}^l x_{pr} \geq w_0^l, \quad l \in \{1, \ldots, k - 1\}. \]  

(5.6)

The model is referred to as pattern-based because \( P \) contains all patterns of events for each course. However, depending on the objective function \( w \), we can formulate the model with a restricted set of patterns \( \bar{P} \subseteq P \) without losing modelling power.

If \( \bar{P} \) is restricted to only the patterns which correspond to a single event, i.e. \( \bar{P} = E \), then the event-based model is obtained. This can be used for any measure of solution quality which relates to the suitability of a room for a particular event i.e. event-based measures. This is in contrast to pattern-based measures which relate to the suitability of a room for any set of course events (see Section 5.2.4).

For an event-based model, if we consider the additional requirement of contiguous room stability, then for each long event we must include the pattern of all constituent events together, and remove the single-event pattern for each of the long events. Each event is part of only one pattern, however patterns may contain more than one event. This is no longer a purely event-based model, which has implications for its complexity and computational difficulty, as explained in Section 5.3. For purely event-based models, and those which require contiguous room stability, we must omit constraints (5.4) which are only valid when an event can be part of more than one pattern.

If \( \bar{P} \) is restricted to only those patterns corresponding to a complete course, i.e. \( \bar{P} = C \), then the course-based model is obtained. Note that any feasible solution to the course-based model requires that each course uses the same room for all events, which is not usually feasible in practice. Constraints (5.4) should again be omitted, as they are redundant in this case.
5.2.4 Measures of Solution Quality

Measures of solution quality for this model can be either event- or pattern-based. We define several common quality measures which are event-based, and course room stability which is pattern-based. If we need to optimise or constrain a pattern-based measure (as in constraint (5.6)), a pattern-based model is required. Event-based measures, however, can apply to either an event- or pattern-based model. Note that each event-based objective coefficient includes the term \( length_p \), which provides linear scaling for when \( p \) contains more than one event.

Several measures of solution quality are described below, and defined in (5.7)–(5.13) for the coefficients \( w_{pr} \) in the objective function (5.1).

**Event hours (EH)** Maximise the total number of events assigned a room over all events. If it is known that a feasible room assignment exists, this is equal to the total number of events in \( E \), and this quality measure can be omitted. Furthermore, in this case, an explicit lexicographic constraint (5.6) is not required in subsequent iterations, because constraints (5.3) can be treated as equalities which has the same effect.

\[
w_{pr} = length_p, \quad p \in P, \ r \in R_p
\]  

(5.7)

**Seated student hours (SH)** Maximise the total number of hours spent by students in all events assigned a room i.e. events are weighted by their number of students. This is only used when it is not possible to assign a room for all events, and we wish to prioritise events of large courses to be assigned.

\[
w_{pr} = length_p \times size_c, \quad c \in C, \ p \in P_c, \ r \in R_p
\]  

(5.8)
Seat utilisation (SU)  Maximise the total ratio of the number of students to the room size over all events assigned a room. This is only used when it is not possible to assign a room for all events, and we wish to prioritise a close fitting of events into rooms.

\[ w_{pr} = \text{length}_p \times \frac{\text{size}_c}{\text{size}_r}, \quad c \in C, \ p \in P_c, \ r \in R_p \]  

(5.9)

Room preference (RP)  Maximise the total course-to-room preference over all events assigned a room. This may be a teacher’s preference, or it may be used to teach courses close to the relevant teaching department’s offices, as at the University of Auckland. Preferences are determined at the department-to-building level (i.e. all courses from each department have the same preference for all rooms from each building) and may take the value -1, 0 or 1 to indicate undesirability, indifference, or preference.

\[ w_{pr} = \text{length}_p \times \text{Pref}(c, r), \quad c \in C, \ p \in P_c, \ r \in R_p \]  

(5.10)

Room type suitability (RT)  Maximise the total number of events assigned a room of a suitable type. This is applied for the DTU problem where it is feasible for events to be assigned to an “unsuitable” room. The suitability may take the value 1 or 0, following a set of simple rules presented in Section 3.2.2.

\[ w_{pr} = \text{length}_p \times \text{Suit}(c, r), \quad c \in C, \ p \in P_c, \ r \in R_p \]  

(5.11)

Course room stability (RS)  Minimise the total number of different rooms, assigned to each course, over all courses. The disruption to the room stability of a course by one of its patterns is given by \((\text{length}_c - \text{length}_p)/\text{length}_c\). In a feasible room assignment, the sum of these fractions by patterns of a course will sum to the
number of additional rooms used, relative to the target ‘1 room per course’. For example, a course with 3 events could use just 1 pattern (all events in the same room), 2 patterns (2 events in the same room, 1 in a different room), or 3 patterns (each event in a different room). Using the disruption formula, the first case with 1 pattern causes a disruption of zero since no additional rooms are used. The second case will cause a disruption of 1/3 for the larger pattern, and 2/3 for the smaller pattern, summing to 1 additional room. The 3 patterns of the final case disrupt stability by 2/3 each, summing to 2 additional rooms.

\[ w_{pr} = -\frac{\text{length}_c - \text{length}_p}{\text{length}_c}, \quad c \in C, \ p \in P_c, \ r \in R_p \quad (5.12) \]

**Spare seat robustness (SR)** Maximise the total robustness of the room assignment to changes in each course’s enrolment size, \( size_c \). The room assignment is typically decided prior to student enrolment, so \( size_c \) is necessarily an estimate of the number of students who will enrol. Therefore, a room which is close in size to the expected enrolment of an assigned course may be considered non-robust to variability in the enrolment size. In practice, the enrolment variability is likely to be different for each course. For example, the enrolment for an entry level course (or one with few pre-requisites) may be less predictable than enrolment for an advanced course on a structured study pathway. An example of a general robustness function is given below, where the room utilisation \((size_c/size_r)\) is considered sufficiently robust when below \( \alpha \), and non-robust when above \( \beta \).

\[ w_{pr} = \text{length}_p \times \begin{cases} 1 & \frac{size_c}{size_r} < \alpha \\ \left( \frac{\beta - \frac{size_c}{size_r}}{\beta - \alpha} \right) & \alpha \leq \frac{size_c}{size_r} < \beta \\ 0 & \beta \leq \frac{size_c}{size_r} \end{cases} \quad c \in C, \ p \in P_c, \ r \in R_p \quad (5.13) \]
In this chapter, we use the parameters of 0.7 for $\alpha$ and 0.9 for $\beta$, giving the robustness function shown in Figure 5.4.

![Figure 5.4: Robustness Function](image)

### 5.3 Computational Difficulty

The computational complexity of the room assignment problem is addressed by Carter and Tovey (1992). In this section, we review and expand upon their findings, through an insight into the structure of the mathematical programmes.

#### 5.3.1 Event-based Problems

The simplest class of room assignment problems are those which can be formulated with an event-based model ($\bar{P} = E$). This requires that long events which span multiple time periods do not need the same room for each period (i.e. no contiguous room stability requirement). It is also assumed that we are not measuring course room stability. Carter and Tovey refer to this as the ‘1-period’ problem because there are no interdependencies between periods, and each period may be solved separately. For any objective function (5.1), the constraint matrix defined by (5.2)–(5.3) (since (5.4) is invalid) of this problem is known to be totally unimodular.
Therefore, event-based models can be solved in polynomial time using an assignment problem algorithm (e.g. the Hungarian algorithm), or solving the event-based linear programme.

5.3.2 Event-based Problems with Contiguous Room Stability

A more practically useful class of problems are those which enforce contiguous room stability on long events. Carter and Tovey (1992) refer to this as the ‘interval problem’ and prove it is NP-hard to find a feasible solution even when the problem is limited to just two time periods. As introduced in Section 5.2.2, modelling contiguous room stability means we can no longer use a purely event-based model, because patterns are required to place the constituent events of a long event into the same room. This alters the matrix structure, such that fractions can occur i.e. the LP relaxation is no longer guaranteed to be naturally integer. The smallest example of this was presented by Carter and Tovey, shown here as Example 2. For each course \( c \) in Table 5.5, events are shown in their respective time periods, and the feasible rooms for this course are given as \( R_c \). For our formulation defined by (5.1)–(5.5), the constraint matrix for this problem is shown in Figure 5.5. This example happens to also be a course-based problem (one pattern corresponding to all a course’s events), so variables are generated for each course for each feasible room. Constraints (5.2) are identified by the period and room, and constraints (5.3) are identified by the course they apply to.

**Example 2.** A minimal event-based problem with contiguous room stability requirements featuring fractionating 7-order 2-cycles

Solving the IP (with constraint matrix shown in Figure 5.5) to maximise the number of event hours, subject to the contiguous room stability requirements, will find an optimal solution which assigns 7 (out of 8) events. However, the LP relax-
5.3. COMPUTATIONAL DIFFICULTY

<table>
<thead>
<tr>
<th>c</th>
<th>t₁</th>
<th>t₂</th>
<th>Rc</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>e₁</td>
<td>r₁, r₂</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>e₁</td>
<td>e₂</td>
<td>r₂, r₃</td>
</tr>
<tr>
<td>C</td>
<td>e₁</td>
<td></td>
<td>r₃, r₄</td>
</tr>
<tr>
<td>D</td>
<td>e₁</td>
<td>e₂</td>
<td>r₁, r₄</td>
</tr>
<tr>
<td>E</td>
<td>e₁</td>
<td></td>
<td>r₁, r₃</td>
</tr>
<tr>
<td>F</td>
<td>e₁</td>
<td></td>
<td>r₂, r₄</td>
</tr>
</tbody>
</table>

Table 5.5: Time Periods and Feasible Rooms

ation is able to assign all 8 events, with each variable taking the value of 0.5.

Early work into the properties of binary matrices (Berge, 1972; Ryan and Falkner, 1988) shows that odd-order 2-cycles (submatrices with row and column sums equal to 2) within a binary constraint matrix permit fractional solutions to occur. Conversely, if no such cycles exist, the matrix is said to be balanced and the problem will be naturally integer (i.e. solvable as a linear programme). The rows and columns in the Example 2 constraint matrix (Figure 5.5) have been ordered to show the odd-order 2-cycles which cause the observed fractional LP solution and corresponding integrality gap. A 7-order cycle can be formed by starting at any of the ‘B’ or ‘D’ columns (4 in total), and connecting to each right-adjacent variable until the other column from this course is encountered (treating the right-most column as connected to the left-most). The cycle which starts at Dr₁ is shaded.

This cycle may also be understood through the maximum weighted independent set formulation of this problem, as introduced in Section 5.1. Variables in our integer programme (i.e. pattern-to-room assignments) are represented as nodes, and constraints are represented using cliques (i.e. so that only one variable in the set of clique variable nodes can take the value 1). In this case, each constraint involves only two variables, so cliques are formed by a single arc. Constraints (5.2) are represented with horizontally dashed arcs, while constraints (5.3) are represented with vertically solid arcs. This formulation is depicted in Figure 5.6, with the shaded odd-order
5.3. COMPUTATIONAL DIFFICULTY

![Set Packing Constraint Matrix](image)

Figure 5.5: Set Packing Constraint Matrix

cycle from Figure 5.5 shown in red. This graph helps to demonstrate how the variables from courses ‘B’ and ‘D’ provide the necessary inter-period linkage.

In order for this type of cycle (Figures 5.5 and 5.6) to cause an integrality gap, with respect to the *event hours* objective, a very specific structure must be present. Each of the six possible combinations of two rooms (out of the four rooms in total) must be the only feasible rooms \( R_c \) for each of the six courses. Furthermore, there must be no overlap in feasible rooms between the courses featuring in \( t_1 \) only, those in \( t_2 \) only and those in both periods. This is clearly observed in Figure 5.6.

Recall that course events can typically be held in any room which provides *at least* enough seats for the course size, and possesses *at least* the requested room attributes. Therefore, the set of feasible rooms for a course will be a superset of those for any larger course and for any course requiring additional attributes. This nested set relationship makes it unlikely in practice that so many different combinations of rooms will occur as the set of feasible rooms for different courses. Furthermore, this
structure must occur in consecutive time periods ($t_1$ and $t_2$), rather than any two time periods. Any alteration to the feasible rooms or time periods for each course will close the integrality gap and potentially break the cycle structure in the matrix. For example, if room $r_1$ was removed from the set of feasible rooms for course $A$, this would break the cycle, and the optimal LP solution would have an objective value of 7, the same as the optimal IP solution. Conversely, if room $r_3$ was added to the set of feasible rooms for course $A$ (as well as existing rooms 1 and 2), an IP solution would exist at the LP objective value of 8, again closing the integrality gap.

It is also possible to construct higher order cycles by either extending this cycle through more courses within the 2 time periods, or by extending across more contiguous time periods. However, these rely on even greater specific structure to be present in the problem.

When the fractional solutions corresponding to odd-order cycles do not cause an integrality gap, they are not precluded from appearing in a solution to the LP. However, they are less likely to be found by an IP solver. This was confirmed in our tests on data from the University of Auckland (for the event-based model with contiguous
room stability) for all objectives listed in Section 5.2.4. Solving the LP relaxation returns a solution with a very small number of fractional variables (typically corresponding to 1 or 2 sets of cycles shown in Figure 5.5), with no integrality gap (for any objective measure). Interestingly, if we solve the IP with Gurobi (Gurobi Optimization, Inc., 2015), a proprietary solver, an optimal solution is found at the root node even with all presolve, cuts and heuristics disabled. This suggests that Gurobi is performing additional ‘integerising’ LP iterations when it is solving the LP relaxation of an IP.

Although our problem from the University of Auckland is clearly not naturally integer, the fractions which arise are very limited in number, and do not cause an integrality gap. Without using an IP solver, we were able to find an optimal IP solution by adding small perturbations to the objective coefficients of the patterns representing long events. We were also able to use a cutting plane approach to find an optimal IP solution, by applying any violated odd-hole inequalities (which correspond to the odd-order cycles, Schrijver, 2003) at the optimal LP solution, and then re-solving the LP. The results from using the Gurobi IP solver, and from using these LP-based methods, demonstrate that although this optimisation problem is NP-hard, the structure of our practical problems is such that any fractions which arise can be easily handled.

We believe that the improbability of encountering cycles of the nature shown in Figure 5.5, also explains why the earlier work of Gosselin and Truchon (1986) reported naturally integer LPs. Both our tests and theirs were performed on real data, and branch-and-bound has not been required. Therefore, it seems likely that practical event-based problems with contiguous room stability requirements can be solved very efficiently.
5.3.3 Pattern-based Problems

The most difficult class of room assignment problems are those which require a pattern-based model \((\vec{P} = P)\), because they consider a pattern-based quality measure such as course room stability. We address course room stability as a quality measure to be maximised, rather than a hard constraint where all events of a course must be held in the same room. Carter and Tovey (1992) address only the latter case, which they refer to as the ‘non-interval problem’. In the context of our formulation, the non-interval problem is represented by a course-based model, a special case of the pattern-based model, which is unlikely to have a feasible solution for practical problems. To model course room stability as a quality measure, for each course we must generate a pattern for each subset of course events (as per Section 5.2.2), for each room. For courses with only two events, the three patterns generated per room correspond to the first event only (‘pattern 1’), the second event only (‘pattern 2’), and both events (‘pattern 3’).

A minimal example of a difficult pattern-based problem is shown as Example 3. For this problem (specification in Table 5.6), it is not possible to offer a stable room to all courses. For our formulation defined by (5.1)–(5.5), the fractionating part of the constraint matrix is shown in Figure 5.7. Note that only constraints (5.2) and variables which relate to ‘pattern 3’ (both events) for each course are shown. The additional 12 variables correspond to each of the six events assigned to each of the two rooms.

**Example 3.** A minimal pattern-based problem featuring fractionating 3-order 2-cycles

Solving the IP (with partial constraint matrix shown in Figure 5.7) to maximise the course room stability, will find an optimal solution with quality of -1 (a penalty of 1). However, the LP relaxation is able to find an optimal solution with quality
5.3. COMPUTATIONAL DIFFICULTY

<table>
<thead>
<tr>
<th>c</th>
<th>t₁</th>
<th>t₂</th>
<th>t₃</th>
<th>Rₑ</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>ₑ₁</td>
<td>ₑ₂</td>
<td>r₁, r₂</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>ₑ₁</td>
<td>ₑ₂</td>
<td>r₁, r₂</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>ₑ₁</td>
<td>ₑ₂</td>
<td>r₁, r₂</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.6: Time Periods and Feasible Rooms

\[
\begin{bmatrix}
A_{p₃r₁} & B_{p₃r₁} & C_{p₃r₁} & A_{p₃r₂} & B_{p₃r₂} & C_{p₃r₂} & \ldots \\
1 & 1 & 1 & 1 & 1 & 1 & \ldots \\
1 & 1 & 1 & 1 & 1 & 1 & \ldots \\
\end{bmatrix} \leq \begin{bmatrix}
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
\ldots \\
\end{bmatrix}
\]

Figure 5.7: Set Packing Partial Constraint Matrix

of 0 (no penalty), with each ‘pattern 3’ variable (those shown in Figure 5.7) taking the value 0.5, and other variables taking the value 0.

Here, integer solutions incur a penalty because they require at least one of the courses to use the undesirable ‘pattern 1’ and ‘pattern 2’ variables, which assign two events from the same course to different rooms. The constraint matrix in Figure 5.7 shows how the desirable ‘pattern 3’ variables of each course can form odd-order 2-cycles (as shaded). Note that the cycle is entirely contained within constraints (5.2), meaning that two variables need only be connected by both representing patterns occupying the same room at the same time. This is unlike the cycles in Example 2 which also involve constraints (5.3).

We also observe the maximum weighted independent set formulation of this problem, in Figure 5.8. The shaded odd-order cycle from Figure 5.7 is shown in red, with the additional 12 variables omitted for clarity. This graph helps to demonstrate the relatively simple structure responsible for this type of cycle.
The requirements for this type of cycle (Figures 5.7 and 5.8) to exist are that there must be three courses which share two common feasible rooms and each course must feature in a different two of the three time periods. To cause an integrality gap with respect to the course room stability objective, there must also be a relative shortage of available feasible rooms for these courses, in the particular time periods. This could be due to a generally high utilisation rate over all rooms, or because particular sizes and types of rooms are in shortage. Clearly, if a third room was introduced into Example 3 which was feasible for even one of the courses, the same cycles would exist, yet there would no longer be an integrality gap.

Unlike Example 2, in this case, there is no specific room feasibility requirement, and the cycles can be formed over any three time periods. As a result, there are many more opportunities for such cycles to occur. When these cycles are part of a larger problem, note that the courses do not need to have the same number of events as one another, because the pattern-based model allows any subset of events to be independent (room-wise) from the rest of a course’s events. As a result, courses with a large number of events are a major contributor to this type of fractionality, as they introduce many patterns which span different time periods. Larger cycles with this basic structure can also exist, e.g. using 5 courses and 5 time periods instead of 3.

We solved the LP relaxation for the pattern-based model optimising course room
stability on data from the University of Auckland and the ITC2007. The LP solutions were typically much more fractional than those for the event-based model with contiguous room stability, and most problems with a non-zero penalty at the optimal IP solution had an integrality gap. A cutting plane approach using odd-hole and clique inequalities (mentioned for Example 2) was much less effective due to many opportunities for the fractions to re-occur without reducing the gap. Consequently, most practical pattern-based problems require the use of a sophisticated IP solver (utilising techniques such as presolve, cuts, heuristics and branching), as covered in our main results in Section 5.4.

5.3.4 Lexicographic Optimisation Constraints

So far we have considered the difficulty of room assignment problems defined by (5.1)–(5.5). It remains to address the effect of adding lexicographic optimisation constraints (5.6). When solving a purely event-based model, recall that the constraint matrix (defined by (5.2)–(5.3), since (5.4) is invalid) is totally unimodular. As a consequence, the polytope defined by the constraints has integer extreme points. If we solve this model to optimality for an event-based objective measure, the solution must lie on a facet of the polytope, which itself must have integer extreme points. Therefore, if we add a lexicographic constraint (5.6) to an event-based model, the new feasible region is this face, which must remain naturally integer. Although the new constraint matrix may no longer be totally unimodular (due to the elements of constraint (5.6)), it will retain the naturally integer property for any number of constraints added through this process. The LP relaxation may be slightly more difficult to solve for each lexicographic constraint added, however no integer programming is required, so the solve time should be acceptable for all practical problems.

For event-based models with contiguous room stability requirements, and for pattern-based models, we have demonstrated that fractional extreme points exist on
the polytope. Adding a lexicographic constraint will only limit the feasible region
to a facet of this polytope, which may still include these extreme points. Therefore, adding lexicographic constraints will not (necessarily) make the problem easier or remove fractional solutions. However, these two models differ in the quantity and nature of fractional solutions which appear for practical instances. These can typically still be solved in an acceptable time due to the limited fractionating structures in the LP relaxation for event-based models with contiguous room stability requirements.

As shown by Example 3, fractionating structures form more readily in pattern-based models, which can cause them to be significantly more difficult to solve than event-based models. Also, in a lexicographic ordering of objectives, once a pattern-based measure is used, all subsequent iterations will require a pattern-based model. Due to the fractionating potential, the solve time for different pattern-based problems can vary substantially, as demonstrated in Section 5.4.

5.4 Results for UoA Problems

All computational results in this section are run using Gurobi 5.6 on 64-bit Ubuntu 14.04, with a quad-core 3.5GHz processor (Intel i5-4690). To exploit the well-studied structure of set packing problems (Avella and Vasil’Ev, 2005), only zero-half and clique cuts are generated, with both set to “aggressive” generation (Gurobi Optimization, Inc., 2015). The time limit is set at 3600 seconds.

5.4.1 UoA 2010

To validate our method we process the University of Auckland’s timetabling data from Semesters 1 and 2 in 2010. We first test on ‘starting’ time assignments which have been generated by faculties, and for which a feasible room assignment is unlikely
to exist. We also test on a ‘final’ time assignments which has been manually modified (see Section 3.1.2) such that a feasible room assignment is known to exist. The specific quality measures chosen are those shown in the flowchart from Figure 3.1, and contiguous room stability is required on long events.

<table>
<thead>
<tr>
<th>Semester 1</th>
<th>Objectives/Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event Hours (total 2400)</td>
<td>EH 2374 * 2374 = 2374 = 2374 =</td>
</tr>
<tr>
<td>Seated Student Hours</td>
<td>SH 251841 * 253864 = 253864 =</td>
</tr>
<tr>
<td>Seat Utilisation</td>
<td>SU 1775.8 * 2077.6 = 2077.6 =</td>
</tr>
<tr>
<td>Room Preference</td>
<td>RP 538 * 712 = 839 *</td>
</tr>
<tr>
<td>Solve Time (s)</td>
<td>0.5 1.5 1.7 3.9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Semester 2</th>
<th>Objectives/Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event Hours (total 2234)</td>
<td>EH 2211 * 2211 = 2211 =</td>
</tr>
<tr>
<td>Seated Student Hours</td>
<td>SH 235610 * 237579 = 237579 =</td>
</tr>
<tr>
<td>Seat Utilisation</td>
<td>SU 1568.0 * 1940.9 = 1940.9 =</td>
</tr>
<tr>
<td>Room Preference</td>
<td>RP 451 * 613 = 727 *</td>
</tr>
<tr>
<td>Solve Time (s)</td>
<td>0.5 1.3 1.3 4.7</td>
</tr>
</tbody>
</table>

Table 5.7: UoA 2010 Starting Room Assignment Results

Table 5.7 shows the results of our method on the starting time assignment from each semester. Each column lists the results of solving one iteration of the lexicographic algorithm i.e. solving an integer programme (5.1)–(5.5) maximising the objective marked with an asterisk, subject to any lexicographic constraints (5.6) marked with an equality sign. The first iteration maximises the event hours but is unable to find a room for all events, demonstrating that this is an infeasible time assignment. It is noted that the number of events for the starting time assignment in each semester differs from the final time assignment. This reflects unrelated changes to planned courses which occur in the interim period, such as while the manual perturbations were taking place.

The solve time for each integer programme is notably low because these are all event-based solution quality measures. As explained in Section 5.3, event-based
models with contiguous room stability requirements have near-integral LP relaxations. After solving all four models we have a partial room assignment which is Pareto efficient with respect to this set of objectives. This will provide useful information to assist in perturbing the time assignment to achieve feasibility in the room assignment.

Table 5.8: UoA 2010 Final Room Assignment Results

Table 5.8 shows the results of our method on final feasible time assignments, yielding a feasible room assignment without any further need for perturbing the time assignment. This is shown by the fact that the first iteration is able to find a room for all events in the respective semester. The first two iterations notably have a short solve time, while the latter two iterations take a considerably longer time. This is because optimising the course room stability uses a pattern-based model which requires the use of integer programming techniques (presolve, cuts, heuristics and branching) to find and confirm an optimal solution.

In theory, many iterations of a lexicographic ("optimise-and-fix") algorithm will eventually tightly constrain the problem. However, in this case we see that significant gains continue to be made to later quality measures, and the solve times remain
manageable.

5.4.2 UoA 2013

To validate our method on recent data, we process the University of Auckland’s timetabling data from Semesters 1 and 2 in 2013. As introduced in Section 3.1.3, the timetabling process used in 2013 does not provide a starting time assignment, and features no use of optimisation. Consequently, the starting time assignment we use for these results has been generated using the techniques from Chapter 4. Specifically, we use the highest quality time assignments generated, corresponding to the final row in Tables 4.4a and 4.4b for Semesters 1 and 2 respectively.

As with the 2010 results, we use the quality measures shown in the flowchart from Figure 3.1, and contiguous room stability is required on long events.

<table>
<thead>
<tr>
<th>Semester 1</th>
<th>Objectives/Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event Hours (total 4490)</td>
<td>EH 4457 * 4457 = 4457 = 4457 =</td>
</tr>
<tr>
<td>Seated Student Hours</td>
<td>SH 283761 287281 * 287281 = 287281 =</td>
</tr>
<tr>
<td>Seat Utilisation</td>
<td>SU 3310.6 3316.0 4009.9 * 4009.9 =</td>
</tr>
<tr>
<td>Room Preference</td>
<td>RP 1519 1535 1614 1828 *</td>
</tr>
<tr>
<td>Solve Time (s)</td>
<td>2.8 4.3 5.2 16.4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Semester 2</th>
<th>Objectives/Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event Hours (total 5322)</td>
<td>EH 5212 * 5212 = 5212 = 5212 =</td>
</tr>
<tr>
<td>Seated Student Hours</td>
<td>SH 310731 318591 * 318591 = 318591 =</td>
</tr>
<tr>
<td>Seat Utilisation</td>
<td>SU 3908.8 3945.6 4550.2 * 4550.2 =</td>
</tr>
<tr>
<td>Room Preference</td>
<td>RP 2009 2002 2073 2357 *</td>
</tr>
<tr>
<td>Solve Time (s)</td>
<td>4.1 6.0 8.3 20.4</td>
</tr>
</tbody>
</table>

Table 5.9: UoA 2013 Starting Room Assignment Results

Table 5.9 shows the results of our method on the starting time assignment from each semester. These results can be interpreted similarly to those in Table 5.7, as they feature the same set of event-based quality measures, and associated rapid
solve times. Based on this time assignment and partial room assignment for each semester, we solve a minimal perturbation problem, as explained in Chapter 6. This uses information from the partial room assignment to assist with perturbing the time assignment, so that a complete room assignment can be found. The results for the 2013 UoA problems in Table 5.9 are provided in Section 6.6.

<table>
<thead>
<tr>
<th>Semester 1</th>
<th>EH</th>
<th>RP</th>
<th>RS</th>
<th>SR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event Hours (total 4490)</td>
<td>4490 *</td>
<td>4490 =</td>
<td>4490 =</td>
<td>4490 =</td>
</tr>
<tr>
<td>Room Preference</td>
<td>1527</td>
<td>2496 *</td>
<td>2496 =</td>
<td>2496 =</td>
</tr>
<tr>
<td>Course Room Stability</td>
<td>1527</td>
<td>2496 *</td>
<td>2496 =</td>
<td>2496 =</td>
</tr>
<tr>
<td>Spare Seat Robustness</td>
<td>2172.0</td>
<td>2141.3</td>
<td>2185.1</td>
<td>3266.3 *</td>
</tr>
<tr>
<td>Solve Time (s)</td>
<td>2.7</td>
<td>1.5</td>
<td>11.6</td>
<td>17.4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Semester 2</th>
<th>EH</th>
<th>RP</th>
<th>RS</th>
<th>SR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event Hours (total 5322)</td>
<td>5322 *</td>
<td>5322 =</td>
<td>5322 =</td>
<td>5322 =</td>
</tr>
<tr>
<td>Room Preference</td>
<td>2033</td>
<td>3199 *</td>
<td>3199 =</td>
<td>3199 =</td>
</tr>
<tr>
<td>Course Room Stability</td>
<td>2033</td>
<td>3199 *</td>
<td>3199 =</td>
<td>3199 =</td>
</tr>
<tr>
<td>Spare Seat Robustness</td>
<td>2574.7</td>
<td>2662.8</td>
<td>2790.7</td>
<td>3573.3 *</td>
</tr>
<tr>
<td>Solve Time (s)</td>
<td>3.9</td>
<td>2.7</td>
<td>222.2</td>
<td>3600.0</td>
</tr>
</tbody>
</table>

Table 5.10: UoA 2013 Timetable Room Assignment Results

Table 5.10 shows the results of our room assignment method on a time assignment which has been minimally perturbed from the time assignment. It is observed in the first iteration that we are able to find a room for all events in each respective semester. The first two iterations notably have a short solve time, as they are event-based problems with contiguous stability. Although each iteration of this lexicographic room assignment algorithm will typically take a longer solve time than the previous iteration, it is observed that this is not true for the first two iterations. This is most likely because this problem has a complete room assignment, which allows the event hours objective to be represented by representing constraints (5.3) at equality, rather than requiring a new lexicographic constraint of the form in constraints (5.6).
The latter two iterations take a longer time to solve, due to the use of a pattern-based model, which is substantially more evident in the case of Semester 2. The final iteration which maximises the spare seat robustness for Semester 2 terminates at the time limit. However, it is very close to confirming an optimal solution (bounded at 3881.7, or a 0.15% gap).

Although it takes over an hour to solve the final iteration, we note that a feasible room assignment with optimal room preference can be found in less than 10 seconds. In less than 5 minutes of cumulative solve time, we are also able to find a (Pareto) optimal value for the course room stability, which greatly reduces the penalty from over 600 to less than 20. Recalling that these problems are much larger than those from 2010, these results satisfactorily validate our algorithm as suitable for use in a practical timetabling process from a quality and time perspective.

5.4.3 Discussion

In this section, we have demonstrated that this algorithm is suitable for practical problems. It is also notable that the difficult and time consuming quality measures are only used when generating a “final” room assignment, i.e. once a complete room assignment has been found. This means that verifying the existence of a complete room assignment can be performed very quickly (less than 5 seconds), and finding a high quality solution with respect to event-based quality measures can also be achieved quickly (less than 30 seconds). This is important in practice, as timetablers may need to frequently re-solve the room assignment model to verify feasibility as additional data is received, or for analytics purposes.

In Chapter 7, we also provide a comparison of these solutions to those implemented at the University of Auckland for the same time period. The room assignment quality comparison is made alongside the comparison for the time assignment.
5.5 Results for DTU Problems

To validate our method on another practical problem, we process the Technical University of Denmark’s timetabling data for the Autumn 2011 and Spring 2012 semesters. As detailed in Section 3.2, the timetabling process used at DTU encompasses important differences from the process at UoA. In particular, DTU requires staff at the timetabling administration (AUS) to solve a student sectioning problem as well as a room assignment problem. The sectioning problem is not necessarily difficult, with only a small number of courses sectioned. However, it cannot be easily incorporated into the room assignment model presented in this chapter. For this reason, we use the sectioning of courses which has been implemented by DTU for the relevant time period.

Differences also exist within the room assignment problem, which we examine in Section 5.5.1, and outline how they may be represented using our formulation. We present results of this adapted method in Section 5.5.2 and provide discussion in Section 5.5.3.

5.5.1 Adaptation of the Model

As introduced in Section 3.2.1, we can model an event in the DTU timetable as representing a two-hour block of teaching. A four-hour block of teaching is therefore represented as two events which form a long event, and thus require contiguous room stability. This reduction enables the number of weekly time periods to be halved, to 20 in total.

At DTU, contiguous room stability is important for long events. However, there is no constraint or quality measure (such as course room stability) which causes interdependencies between events in different sessions. As a result, we represent this problem using an event-based model with contiguous room stability, which is
also subject to a small number of additional constraints as outlined in this section.

The first difference in modelling the DTU problem is that DTU considers the assignment of some events to be “mandatory” in any feasible solution. This can be simply incorporated into our model, as the associated constraints within (5.3) can be met with equality.

Recall from Section 3.2.1 that some lectures are taught in multiple sections. For lectures which are sectioned into several events, this set of events is referred to as a section group. At DTU, all events in each section group must be taught in the same building, which is enforced by constraints (5.14). For this constraint, all events in each section group are required to use the same building as the first event in this section group. Additional notation is required, where $LEC\_SECT\_GRP$ represents the set of all lecture section groups, and $sect\_grp[1]$ represents the first event in a section group. Note that the set $P_e$ is used to refer to a single pattern because each event can be in only one pattern in this type of model (event-based with contiguous stability).

\[
\sum_{r \in Bldg} x(P_e)_r - \sum_{r \in Bldg} x(P_{sect\_grp[1]})_r = 0 \quad \text{bldg} \in BLDG,
\]

\[
\text{sect\_grp} \in LEC\_SECT\_GRP,
\]

\[
e \in (sect\_grp \setminus sect\_grp[1]) \quad (5.14)
\]

If a two-hour lecture is followed by a two-hour tutorial, and the lecture uses a study room, the tutorial must also use the same study room. In the case of a tutorial taught in multiple sections, one of these sections must use the study room. This requirement is enforced using constraints (5.15) where $P_{lec}$ represents the subset of all patterns ($P_{lec} \subset P$) consisting of a two-hour lecture event taught at the start of a session (i.e. 8am or 1pm). Let $P_{tut\_lec}$ denote the set of patterns which consist of two-hour tutorial event taught directly after lecture event $p_{lec}$. Finally, let $R_{sr}$ denote the subset of rooms ($R_{sr} \subset R$) which correspond to study rooms.
\[ x_{p_{lec}r} - \sum_{p_{tut} \in P_{tut} \setminus P_{lec}} x_{p_{tut}r} \leq 0 \quad p_{lec} \in P_{lec}, \]
\[ r \in (R_{p_{lec}} \cap R_{sr}) \]  
(5.15)

5.5.2 DTU 2011–2012

The quality measures used at DTU do not require adaptations to the room assignment model, and can be simply modelled as coefficients for each pattern-to-room assignment. The first two quality measures we use for the DTU problem are maximising the event hours, and the seated student hours which are equivalent in definition to those at the UoA, as defined in Section 5.2.4. The third quality measure used is room type suitability, which applies specifically to DTU, and is also defined in Section 5.2.4. Maximising the room type suitability represents the desire for events to be assigned into the particular room type which best suits the associated event type. The final quality measure we consider is the room preference, which represents the desire for events to be taught in a close geographic proximity to the faculty offices. The measurement of room preference for DTU differs from the UoA case, as the available DTU data only considers 0 and 1 values as preference coefficients, and not -1.

Table 5.11 shows the results of our adapted room assignment method on two datasets from DTU. For the Autumn 2011 dataset, maximising the event hours reveals that we are able to find a room for all events. In this case, we are not required to maximise the seated student hours, as this is necessarily optimal. For the Spring 2012 dataset, we are able to find a room for almost every event, with only 3 unassigned events. Considering the latter two quality measures, we observe that both make a substantial improvement as part of the lexicographic process, particularly in the room preference. This is because the room type suitability is
5.5. RESULTS FOR DTU PROBLEMS

<table>
<thead>
<tr>
<th>Objectives/Iterations</th>
<th>Autumn 2011</th>
<th>Spring 2012</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>EH</td>
<td>SH</td>
</tr>
<tr>
<td>Event Hours (total 1103)</td>
<td>1103</td>
<td>-</td>
</tr>
<tr>
<td>Seated Student Hours</td>
<td>48829</td>
<td>-</td>
</tr>
<tr>
<td>Room Type Suitability</td>
<td>825</td>
<td>-</td>
</tr>
<tr>
<td>Room Preference</td>
<td>302</td>
<td>-</td>
</tr>
<tr>
<td>Solve Time (s)</td>
<td>0.3</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 5.11: DTU Room Assignment Results

necessarily high in any feasible solution. Lectures will typically be assigned into auditoriums as required by the number of students, and tutorials will typically be assigned into study rooms or study areas, as they comprise the majority of feasible rooms. The final important observation is that the problems solve very quickly, in less than 1 second. This is because we are able to represent the problem using a modified event-based model with contiguous stability, which is efficiently solvable.

<table>
<thead>
<tr>
<th>Objectives/Iterations</th>
<th>Autumn 2011 IP Implemented</th>
<th>Spring 2012 IP Implemented</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event Hours</td>
<td>1103</td>
<td>1004</td>
</tr>
<tr>
<td>Seated Student Hours</td>
<td>48829</td>
<td>47149</td>
</tr>
<tr>
<td>Room Type Suitability</td>
<td>1069</td>
<td>975</td>
</tr>
<tr>
<td>Room Preference</td>
<td>934</td>
<td>821</td>
</tr>
</tbody>
</table>

Table 5.12: DTU Room Assignment Quality Comparison

In Table 5.12, we provide a comparison between our solutions and those implemented at DTU for the same time period. We are able to find a room for more events than the manual approach, while addressing the most important quality measures. As a caveat to the latter measurement, we note that the measurement of quality is
dependent on the number of events assigned. Our capability to assign more events improves our ability to generate a solution with a greater number of high quality event-to-room assignments. As with timetabling in general, the greatest emphasis should be placed on the number of events assigned (i.e. feasibility).

5.5.3 Discussion

The most evident result of this section is that our method is versatile. To address the room assignment problem at DTU, quality measures are simply modelled, and two constraint classes are incorporated which do not adversely affect the solve time. Due to our use of integer programming to obtain an optimal solution, it is expected that the solution quality is higher than when using a solely manual approach.

Existing studies by Bærentsen (2012) and Bagger et al. (2014) are similarly able to validate their algorithms, although they also solve the student sectioning problem. Due to a different weighting and choice of quality measures, we are unable to meaningfully compare our solutions. We observe in the results of Bærentsen (2012) and Bagger et al. (2014), that allowing a variable size of events (as required to determine sectioning) makes the problem substantially more difficult. In the work of Bærentsen (2012) where an exact method is used, optimal solutions are not found after 10 hours of solve time.

The difficulty of solving a combined sectioning and room assignment problem suggests it may be worthwhile to address only the room assignment algorithmically. Determining the sectioning manually (which is currently the process) is not necessarily difficult, as only a small proportion of courses require sectioning. Furthermore, the sectioning is likely to be predictable based on what was successfully implemented from the previous year. Timetabling staff are in a good position to manually determine the number and size of sections, with knowledge of the available rooms in the university. As with all practical timetabling, the process is likely to involve a
degree of “trial-and-error”, as faculty must approve of a particular sectioning from a staffing perspective. An advantage of the room assignment model in this section is the feasibility of different sectioning solutions be evaluated rapidly.

Comparing our solutions to those implemented by DTU, it is important to note that we address the practical problem in an academic environment. This means we are not necessarily exposed to the complete set of complexities as faced by timetabling staff, such as changes in the data, and many potential quality considerations which were not included in the provided data.

For one example, we consider the modelling of room suitability. In this section, we have treated the room suitability as a quality measure, where a penalty is incurred for breaking the rule of assigning a suitable room. However, in Table 5.12 we can observe that the DTU solution only breaks this rule a small number of times. It is possible that the room suitability requirement will only be breakable for a subset of events, while other events require a suitable room as a hard constraint. For our solutions, 1069 and 975 events (respectively for each semester) are assigned in suitable rooms.

Another observation about the DTU implemented solutions is that they are of high quality in terms of the room preference, with over two thirds of events in a preferred room. Our results have shown that when we are not specifically optimising for room preference, less than a third of events will be taught in a preferable room. This reinforces the notion that timetablers in general possess substantial knowledge and experience with the particular problem at their university.

Finally, we note that it may be desirable to include the omitted faculty rooms in the initial room assignment. For the model in this section, it is simple to include a quality measure which penalises (but permits) the assignment to any particular rooms. Although faculty approval would be required for each such assignment, a greater overall quality (including less reliance on these rooms) may be achieved by
this approach.

5.6 Results for Udine Benchmarks

5.6.1 Udine UD2

As previously stated, the main focus of our work is on practical problems which feature many room assignment solution quality measures. However, we also address instances the UD2 problems (from Track 3 of the 2007 International Timetabling Competition), as these are widely used in the literature as benchmarks. We introduce these problems in Section 3.3.

It is firstly noted that benchmarking our work directly against ITC entrants is not possible, due to the fact that we focus solely on the room assignment. However, we are interested in finding a room assignment for timetables from the ITC2007 entries, to test the performance of the room assignment model on a diverse set of instances. All timetables were retrieved from the publicly accessible listing at http://tabu.diegm.uniud.it/ctt (Bonutti et al., 2008), where our final solutions have also been uploaded.

To address the ITC2007 problems, we first solve for the UD2 specification (as used in the competition) which treats course room stability as the only room assignment solution quality measure. To solve the room assignment, we have used the timetables from Tomáš Müller’s heuristically-generated solutions, which were the overall winner of the ITC2007 (Müller, 2009) and are available for the full set of ITC2007 problems.

Our results for all ITC2007 problems except comp01 are shown in Table 5.13. Columns 2 and 3 give information on the room utilisation and average connectedness for each problem, which we interpret below. Column 4 gives the quality (penalty) from the time assignment solution quality measures of Müller’s solution, which is a
### Table 5.13: Results on Müller’s UD2 (ITC2007) Time Assignments

<table>
<thead>
<tr>
<th>Problem</th>
<th>Müller’s Results</th>
<th>Our Room Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Name</td>
<td>Util</td>
</tr>
<tr>
<td>comp02</td>
<td>0.71</td>
<td>0.64</td>
</tr>
<tr>
<td>comp03</td>
<td>0.63</td>
<td>0.58</td>
</tr>
<tr>
<td>comp04</td>
<td>0.64</td>
<td>0.71</td>
</tr>
<tr>
<td>comp05</td>
<td>0.47</td>
<td>0.17</td>
</tr>
<tr>
<td>comp06</td>
<td>0.8</td>
<td>0.76</td>
</tr>
<tr>
<td>comp07</td>
<td>0.87</td>
<td>0.75</td>
</tr>
<tr>
<td>comp08</td>
<td>0.72</td>
<td>0.74</td>
</tr>
<tr>
<td>comp09</td>
<td>0.62</td>
<td>0.65</td>
</tr>
<tr>
<td>comp10</td>
<td>0.82</td>
<td>0.70</td>
</tr>
<tr>
<td>comp11</td>
<td>0.72</td>
<td>0.33</td>
</tr>
<tr>
<td>comp12</td>
<td>0.55</td>
<td>0.17</td>
</tr>
<tr>
<td>comp13</td>
<td>0.65</td>
<td>0.75</td>
</tr>
<tr>
<td>comp14</td>
<td>0.65</td>
<td>0.59</td>
</tr>
<tr>
<td>comp15</td>
<td>0.63</td>
<td>0.53</td>
</tr>
<tr>
<td>comp16</td>
<td>0.73</td>
<td>0.75</td>
</tr>
<tr>
<td>comp17</td>
<td>0.8</td>
<td>0.71</td>
</tr>
<tr>
<td>comp18</td>
<td>0.43</td>
<td>0.13</td>
</tr>
<tr>
<td>comp19</td>
<td>0.69</td>
<td>0.68</td>
</tr>
<tr>
<td>comp20</td>
<td>0.82</td>
<td>0.71</td>
</tr>
<tr>
<td>comp21</td>
<td>0.73</td>
<td>0.69</td>
</tr>
</tbody>
</table>

Summary of several weighted penalty factors defined in Bonutti et al. (2012). Column 5 gives the penalty from room assignment solution quality measures from Müller’s solution, which is equivalent to the course room stability penalty (for the UD2 specification). We can compare this to our IP optimal course room stability penalty in column 6. Column 7 gives the objective value of the LP relaxation, where an ‘I’ represents an integral LP relaxation. Column 8 gives the number of nodes which are explored in the solve process, and column 9 gives the solve time to optimality.

We do not include a result for comp01, because our approach does not model a “soft” room size. When constrained to original room sizes, the comp01 room assignment problem is infeasible for any time assignment, as noted by Asín Achá.
and Nieuwenhuis (2012). In our method, an infeasible room assignment (for a given time assignment) is confirmed when the optimal room assignment maximising the event hours is not able to assign all events to a room.

Note that three of the problems had integral LP relaxations, and are very rapid (<0.5s) to solve. Another fourteen of the problems do not have integral LP relaxations, but find an optimal IP solution at the root node (i.e. without branching). These problems do contain odd-order cycle induced fractions. However, Gurobi is able to find an integer solution relatively quickly (<10s) using cuts and/or heuristics. Only one problem, comp10, uses branching to find an optimal solution when there is no integrality gap.

The remaining two problems, comp07 and comp20, are the only cases of odd-order cycles causing an integrality gap, as demonstrated in Example 3 from Section 5.3. In these cases, there are many ways for the cycles to re-occur (with the same objective value) after branching or cuts are applied. For comp20, the solver was able to prove no integer solution could exist at the LP objective value, while comp07 required a substantially longer time. The aggressive cut generation parameter used with Gurobi results in significant computational work expended on attempting to improve the lower bound to confirm optimality. However, it should be noted that a good (or even optimal) solution can be found more quickly than the time required to confirm optimality, particularly when parameters are chosen for this purpose.

Focussing on the most difficult problems in our study (as measured by solve time), it is interesting to observe the correlations to the room utilisation, and the average connectedness. In this case, utilisation is measured as the total number of events divided by the total number of available time periods for all rooms. The connectedness quantifies the extent of the interdependencies between time periods, as introduced in Section 5.1. A time period is considered to be connected to another time period if there exists a course with an event in both time periods. The
connectedness of each time period is measured as the number of periods to which it connects, divided by to the total number of other periods. The ‘Conn’ figures shown in Table 5.13 represent the average connectedness for all time periods.

Figure 5.9 demonstrates the positive correlation between the time taken to solve each UD2 problem, and each of these factors. The time axis is scaled logarithmically to represent the variation in solve times. Each set of points shows the relationship between this factor and the solve time, for all 20 problems (points). Although positive correlations are clear for the utilisation (blue crosses) and connectedness (green triangles), a closer correlation is shown using red stars which is simply calculated as the product of the other two terms. We use this product to represent the effects of both factors in one measurement.

The observation that the room utilisation and connectedness are strongly correlated to the difficulty is consistent with our theoretical results from Section 5.3. Our theoretical results firstly demonstrate how problems with a high room utilisation are more likely to exploit the odd order cycles and cause an integrality gap between the IP and LP relaxation. We also demonstrate that the extent to which time periods are “linked” together, creates opportunities for odd-order cycles to occur. Therefore, we expect that the product of these terms (as shown in Figure 5.9) will be a good predictor of the difficulty for a given problem. This observation may be useful in practice (e.g. for online optimisation), as the prediction is a simple calculation made from the data, and may be used to choose an algorithm or set of parameters.

As a final experiment on these problems, we observe that the online listing provides the best solutions and best bounds found for the UD2 problems from any method, with no restrictions on solve time (Bonutti et al., 2008). The majority of best known solutions incur no room stability penalty, and we are able to generate an equivalent room assignment quickly. However, the previously best known solution to comp21 by Moritz Mühenthaler incurred a timetable penalty of 74, and a room
assignment penalty of 1 (for a total penalty of 75). For this timetable, our model was able to find a room assignment with 0 penalty after 9.9 seconds of solve time, yielding a new best solution with a total penalty of 74. The lower bound of 74, which was provided by Gerald Lach, confirms our result is an optimal solution to comp21. This result is encouraging in terms of validating the co-utilisation of both heuristic methods (as used by Mühlenthaler) and optimisation methods for difficult problems.
5.6. RESULTS FOR UDINE BENCHMARKS

5.6.2 Udine Extended UD5

Although the UD2 specification was used for the ITC2007, the follow-up work by Bonutti et al. (2012) introduces three other specifications which have received significantly less attention in the literature. Here we address UD5, as it includes a room assignment solution quality measure, travel distance, which relates to the physical distance which students within a curriculum must travel between consecutive events.

In this specification, a penalty is incurred where two events from the same curriculum are contiguous in time, and are taught in rooms from different buildings. This represents a group of students being required to travel an undesirably large distance in the time between contiguous events. This quality measure cannot be directly represented using our pattern-based formulation, as it relates to pairs of events which are possibly from different courses (unlike patterns which are sets of events from a single course), and measures the stability of which building is assigned, rather than stability in being assigned to a specific room.

To model this quality measure, we introduce a set of auxiliary variables where $y_{e_1, e_2, \text{bldg}}$ takes the value 1 if the curriculum-contiguous pair of events $e_1$ and $e_2$ are both assigned to a room in building $\text{bldg} \in \text{BLDG}$. The set of all such event-pairs is given as $\text{CU}_\text{CONT}_\text{PAIRS}$. In a complete room assignment, maximising the number of event-pairs which use rooms from the same building (i.e. building stability) is clearly equivalent to minimising the travel distance penalty for event-pairs which use rooms from different buildings.

Starting from an event-based model, we apply the objective function given as (5.16) to maximise the building stability. We also introduce constraints (5.17) and (5.18) which simply “tie” the auxiliary variables to the event-to-room variables of the underlying event-based formulation.

$$\text{maximise} \sum_{e_1, e_2 \in \text{CU}_\text{CONT}_\text{PAIRS}} \sum_{\text{bldg} \in \text{BLDG}} y_{e_1, e_2, \text{bldg}} \quad (5.16)$$
To test our room assignment model adapted for the UD5 specification, we use the heuristically-generated time assignments generated by Andrea Shaerf, a prominent researcher in this field (Bellio et al., 2012). At the time of writing, these constitute the only complete set of solutions for the UD5 specification which are available on the public listing of solutions.

\[
y_{e_1,2 \text{bldg}} - \sum_{r \in R_{\text{bldg}}} x_{p_{e_1} r} \leq 0 \quad e_{1,2} \in \text{CU}_\text{CONTPAIRS},
\]

\[
bldg \in \text{BLDG}
\] (5.17)

\[
y_{e_1,2 \text{bldg}} - \sum_{r \in R_{\text{bldg}}} x_{p_{e_2} r} \leq 0 \quad e_{1,2} \in \text{CU}_\text{CONTPAIRS},
\]

\[
bldg \in \text{BLDG}
\] (5.18)

Table 5.14: Results on Shaerf’s UD5 Time Assignments

<table>
<thead>
<tr>
<th>Problem</th>
<th>Shaerf’s Results</th>
<th>Our Room Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Timetable</td>
<td>Room</td>
</tr>
<tr>
<td>comp02</td>
<td>128</td>
<td>42</td>
</tr>
<tr>
<td>comp03</td>
<td>163</td>
<td>28</td>
</tr>
<tr>
<td>comp04</td>
<td>82</td>
<td>8</td>
</tr>
<tr>
<td>comp05</td>
<td>606</td>
<td>89</td>
</tr>
<tr>
<td>comp06</td>
<td>112</td>
<td>36</td>
</tr>
<tr>
<td>comp07</td>
<td>61</td>
<td>36</td>
</tr>
<tr>
<td>comp08</td>
<td>77</td>
<td>6</td>
</tr>
<tr>
<td>comp09</td>
<td>164</td>
<td>12</td>
</tr>
<tr>
<td>comp10</td>
<td>62</td>
<td>74</td>
</tr>
<tr>
<td>comp11</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>comp13</td>
<td>153</td>
<td>16</td>
</tr>
<tr>
<td>comp14</td>
<td>93</td>
<td>32</td>
</tr>
<tr>
<td>comp15</td>
<td>168</td>
<td>52</td>
</tr>
<tr>
<td>comp16</td>
<td>99</td>
<td>30</td>
</tr>
<tr>
<td>comp17</td>
<td>145</td>
<td>40</td>
</tr>
<tr>
<td>comp18</td>
<td>122</td>
<td>26</td>
</tr>
<tr>
<td>comp19</td>
<td>135</td>
<td>18</td>
</tr>
<tr>
<td>comp21</td>
<td>174</td>
<td>14</td>
</tr>
</tbody>
</table>

Table 5.14 shows the results of solving our adapted model, presented with the
same column headings as for Table 5.13. In column 4, we present both the travel distance penalty which may be compared to the penalty in Schaerf’s solutions (column 3), and also the objective value from (5.16) which is provided in brackets. Columns 5 and 6 show the number of nodes used when solving the IP, and the total time required.

Comparing the quality of our solutions (column 4) to those of Schaerf (column 3), we are able to improve in every case where a penalty is incurred, often by a substantial margin. As with the tests for the UD2 specification, we have used the original room size limits rather than the modified limits from Shaerf’s solutions. Inevitably this causes some problems (comp12 and comp20) to be infeasible, as expanding the room size is required for a complete room assignment to exist for the given time assignment.

Finally, columns 5 and 6 show that although our adapted event-based model is no longer naturally integer, all models solve very rapidly.

5.6.3 Discussion

Although we have only addressed the room assignment component of the Udine benchmark problems, our results on these problems offer valuable insights. Firstly, the results for the UD2 specification have empirically demonstrated the difficulty of solving a pattern-based model for a diverse range of problem instances. The findings have supported our theoretical insights into which problems are likely to be fractional and difficult to solve as integer programmes.

The most important result from this section is that optimisation methods offer substantial promise when incorporated with heuristics. Many heuristics operate on the basis of finding a solution and improving it until the allotted time expires. At a minimum we have demonstrated that our integer programme can be incorporated after running a heuristic, to improve a part of the overall solution. Finding a new
“best-known” solution to *comp21* is a particularly evident example of this. However, more broadly there are substantial opportunities for incorporating this type of method within a heuristic, where it forms part of the solution process.
In this chapter, we present a general integer programming-based method for solving the minimal perturbation problem in university course timetabling. This problem involves the time assignment and room assignment simultaneously, which requires us to localise the solution space as much as possible. This is achieved by utilising the exactness of integer programming within a broader heuristic framework.

Minimal perturbation problems may arise in many situations within the scope of practical timetabling. Most commonly the problem is recognised as part of the enrolment phase of timetabling, where unexpected changes to course enrolments or other timetabling data may occur. Adjusting an existing timetable is typically significantly preferable to generating a new timetable, as such changes have a negative impact on students and staff. Furthermore, as demonstrated in this thesis, generating a timetable requires substantial time and effort from administrators and teaching staff.

In addition to developing an automated method of solving this problem, this chapter explores the broader applications of minimal perturbation problems within the construction phase of timetabling. We demonstrate the application of this
method using examples based on data from the University of Auckland in 2010 and 2013.

In Section 6.1, we introduce the minimal perturbation problem and outline situations where it can arise. In Section 6.2, we present the expanding neighbourhood heuristic algorithm to solve this problem. In Section 6.3, we define two versatile event-based integer programmes which are used in the expanding neighbourhood algorithm to optimally re-assign events to time period and rooms. In Section 6.4, we define a course-based integer programme which can be used to model disruption of a course structure. In Section 6.5, we discuss how to define the starting neighbourhood and rules for expansion in order to improve the possibility of finding a high quality solution quickly. In Sections 6.6 and 6.7, we present computational results of this method, on problems from the construction and enrolment phases of timetabling respectively. Finally, in Section 6.8, we discuss ways to use this algorithm to address an even wider range of quality considerations, and consider the implementation of this algorithm in practice.

The algorithms and results presented in this chapter have been published as a full paper in the PATAT 2014 conference proceedings (Phillips et al., 2014).

6.1 Introduction

A feasible solution to the university course timetabling problem specifies a time period and room for every course event, without violating any hard constraints. The minimal perturbation problem arises when an existing timetable contains hard constraint violations, or infeasibilities, which need to be resolved. The objective is to resolve all infeasibilities while minimising the disruption or perturbation to the remainder of the timetable.

The minimal perturbation problem is first comprehensively addressed in the
context of general dynamic scheduling (El Sakkout et al., 1998). El Sakkout and Wallace (2000) propose an algorithm based on constraint programming techniques, which leverages the efficiency of linear programming to solve part of the problem.

Recent work on the minimal perturbation problem has been in the broader context of general constraint satisfaction problems (CSPs). Zivan et al. (2011) develop a branch-and-bound-based tree search in “difference-space”, where nodes represent the set of variables perturbed. Fukunaga (2013) develops an improved search method in “commitment-space” where nodes represent the commitment of a variable to a value.

Within university timetabling, the minimal perturbation problem has been addressed using metaheuristics in the work of Barták et al. (2004), Müller et al. (2005), Rudová et al. (2011) and Kingston (2013b), which we introduce in Section 2.3.2.

University course timetabling is widely accepted to be a dynamic problem in practice, where data may continually change throughout construction and implementation of a timetable (McCollum, 2007; Kingston, 2013a). For complex timetabling at large universities, minimal perturbation problems can arise in each stage of the timetabling process.

The early construction phase of timetabling occurs when most of the timetabling data has been gathered, and construction of a timetable is starting. Most time or room assignments are considered to be tentative, and may be changed relatively freely. At this stage, almost any changes to the data are possible. Examples include new or removed courses, changes to staff employment status and room availabilities. Some infeasibilities may not need to be resolved until the data is more complete.

The late construction phase occurs when the timetable is close to being finalised for publication. This stage is the most similar to the curriculum-based timetabling problem addressed in the literature. As introduced in Section 3.1, time assignments at the University of Auckland are determined in close collaboration with faculties,
to satisfy their specific and complex requirements. In this case, changing the time period for an event would be disruptive, whereas the room assignment may be more freely perturbed.

Major changes to the data are less likely at this stage, and all infeasibilities should be resolved. We note that infeasibilities may also arise due to the method of constructing a timetable, rather than solely due to changes in the data. For example, if faculties choose their own time assignments independently (often “rolled-forward” from the previous year with changes), this can produce a timetable for which there is no feasible room assignment. Although this application of the minimal perturbation problem has not been addressed in university course timetabling, Ásgeirsson (2012) develops heuristics for a conceptually similar problem within staff scheduling.

The enrolment phase of timetabling begins once the timetable is published, and students have started to select courses. This phase extends into the semester, and a further distinction can be made on whether the semester has started. Once enrolments have begun, it can be disruptive to change either the time period or the room assignment for an event. However, the former is often particularly disruptive, as students and staff may have external obligations affecting their personal timetable.

During this phase, many potential changes to the data can cause an infeasibility. The most common example occurs when a course receives an unexpectedly high enrolment, such that the existing room assignment is no longer suitable. At the University of Auckland, it is a legal requirement that there may not be an excess or overflow of students in a room. Although not all events will be attended by every enrolled student (e.g. sickness, retention, recorded lectures), the first events of a semester are typically well attended.

We also consider an example where the availability of one or more rooms is lost. A room may become temporarily unavailable for reasons such as damage to the premises or equipment, or if there is nearby construction work. Alternatively, a
room may become permanently unavailable, for example, if it is repurposed from
teaching space to office space. At the University of Auckland, repurposing may be
conducted if a particular type of room is under-utilised. For example, if there are
several similar tutorial rooms, each of which are occupied in less than 40% of time
periods.

We finally note that unexpected changes to the data are not necessarily due to
unpredictability in real-world circumstances, and may instead be due to errors in the
data. A large quantity of data is required to represent all aspects of the timetabling
problem, and data is frequently subject to change between timetabling semesters.
Ideally, all data is known and corrected in the construction phase of timetabling.
However, it is possible for residual errors or omissions to remain undetected until
the enrolment phase. For example, a data error may be a misreporting of room
attributes as facilities are added, upgraded or discontinued. A less obvious data
error may occur if a faculty has omitted a curriculum which relates to a new course.

As previously mentioned, infeasibilities can arise as a result of any violated hard
constraint in either the time or room assignment. However, rather than considering
an infeasibility as the violation of a particular type of constraint, it is useful to
generalise each infeasibility as one or more unassigned events. For example, if an
event is no longer suitable for its assigned room, it is treated as an unassigned
event (as opposed to infeasibly assigned to the room). Similarly, if a curriculum is
introduced which causes a conflict between two events in the same time period, one
(or both) are unassigned.

By this process, a timetabling solution with various violated constraints may be
represented by two sets of events; those which are feasibly assigned to a time period
and room, and those which are not assigned. Additionally, each unassigned event
has a preferred time period which is typically at the time it was previously assigned.
When the minimal perturbation problem is solved, perturbations for all events are
calculated with respect to the current (or preferred) time period.

For practical minimal perturbation problems, we can have reasonable confidence that it is possible to find a feasible solution without major perturbation to the existing timetable structure. Whether the infeasibilities arise from rolling forward an old timetable with changes, or if there are unexpected changes to enrolment, it is likely that the infeasible timetable will be “close” to feasibility, i.e. only a small number of events will need to change time period or room. Furthermore, because rooms are utilised in approximately 50% of available time periods (Beyrouthy et al., 2007), usually there are many solutions, or ways to restore feasibility.

6.2 Expanding Neighbourhood Algorithm

Solving the minimal perturbation problem requires assigning both a time period and a room for each event, rather than addressing these problems separately. However, building a model with variables indexed over all events, time periods and rooms could easily result in millions of binary variables (Burke et al., 2008b), which would be intractable. As a result, we would like to build a model which resolves each infeasibility in as small a neighbourhood as possible. The neighbourhood around an infeasibility is defined by a restricted set of events which can be moved, and subsets of time periods and rooms to which events can be moved. All events outside this neighbourhood are fixed to their existing time and room assignment.

Clearly we do not have a priori knowledge of the minimum neighbourhood size required to resolve a given infeasibility. Therefore, we propose an expanding neighbourhood algorithm which addresses each infeasibility sequentially. In each iteration, we choose a time period with unassigned events to focus on. Around this time period, we generate a small neighbourhood, which defines a restricted set of possibilities for how events can be reassigned. For example, the neighbourhood may
be defined to only consider events of similar size to the unassigned events, and only to/from the time periods one hour before or after their current time period. Within this neighbourhood, an integer programme (IP) is solved to maximise the number of neighbourhood events which can be assigned to a suitable time period and room. If it is not possible to increase the number of assigned events inside the neighbourhood, we are required to expand the size of the neighbourhood. The neighbourhood is continually expanded until we are able to assign more events inside the neighbourhood than were previously assigned. At this stage, we can re-solve the neighbourhood IP to find a solution which minimises the disruption caused by assigning this number of events.

This process constitutes one iteration of the algorithm, resulting in a decrease in the number of unassigned events. The algorithm continues to iterate until all events are assigned. Within each iteration, note that we stop expanding the neighbourhood once the number of assigned events can be improved, rather than only when all neighbourhood events are assigned. This means we may use more than one iteration to resolve the infeasibilities in a given time period. This algorithm is presented as Algorithm 1.

Each iteration of Algorithm 1 requires the solution of at least 2 IPs. These include the first IP which maximises the number of assignable events in the initial neighbourhood, and the final IP which minimises the disruption in the final neighbourhood. An additional IP must also be solved each time the neighbourhood is expanded. Although a large number of iterations and IPs may be required, each IP model will be relatively small, due to the optimistic methodology of starting with a small neighbourhood and only increasing the model size when necessary.

The use of an exact method is well-suited to the minimal perturbation problem. In contrast to other methods (e.g. manual or heuristic), the major advantage of incorporating integer programming is that it provides certainty of whether we are
Algorithm 1 Expanding Neighbourhood Algorithm

1: while true do
2: \[ t \leftarrow \text{GetInfeasibleTimePeriod}(\text{Timetable}) \]
3: if \( t \) does not exist then
4: terminate successfully
5: end if
6: \[ N \leftarrow \text{GenerateInitialNeighbourhood}(t) \]
7: searching \( \leftarrow \) true
8: while searching true do
9: \[ IP \leftarrow \text{BuildNeighbourhoodIP}(N) \]
10: \[ \text{EventsAssignable} \leftarrow IP.\text{Solve}(\text{obj: MaxEventHours}) \]
11: if \( |\text{EventsAssignable}| > |N.\text{EventsAssigned}| \) then
12: \[ \text{EventsAssignable} \leftarrow IP.\text{Solve}(\text{obj: MinDisruption}) \]
13: \[ \text{Timetable}.\text{Update}(%\text{EventsAssignable}) \]
14: searching \( \leftarrow \) false
15: else
16: \( N.\text{Expand}() \)
17: end if
18: end while
19: end while

required to expand the neighbourhood. If the maximum number of assignable events is equal to the current number of assigned events, we have certainty that a given neighbourhood is of insufficient size. This statement cannot be made if manual or heuristic methods are used. Furthermore, because the size of each neighbourhood is kept small, our method is very fast. This is a notable advantage over a manual approach.

In the following sections we explore the application of this algorithm to the minimal perturbation problem as it exists within course timetabling. The definition of the starting neighbourhood, and the process of expansion, should each be dependent on the nature of the given infeasibility. The neighbourhood definition should not only uphold the constraints of the time and room assignment problems, but also be tailored to include the variables which are likely to resolve the infeasibility.
6.3 Event-based Neighbourhood Model

Solving the minimal perturbation problem using Algorithm 1 requires the solution of a number of integer programmes. For each neighbourhood considered in each iteration, we solve an integer programme to maximise the number of events which can be assigned to a time period and room. Once the number of assigned events can be increased, we solve an integer programme to determine the optimal way to assign additional events while minimising the disruption to the remainder of the timetable.

A simplified representation of a neighbourhood is a subset of the original timetabling problem, where we consider a subset of events $E_N \subseteq E$, time periods $T_N \subseteq T$, and rooms $R_N \subseteq R$. As detailed for the time assignment problem, only a subset of time periods are suitable for a given event $e$, i.e. $T_e \subseteq T_N$. Similarly for the room assignment, $R_{et} \subseteq R_N$, as not every room is suitable for every event, and not every room is available in every time period. Therefore, the precise representation of a neighbourhood is given by the set of variables as indexed over all event-time-room assignments for $e \in E_N$, $t \in T_e$, and $r \in R_{et}$.

When solving a minimal perturbation problem, we must consider the effect of the fixed events (i.e. $e \in E \setminus E_N$) on the set of suitable time periods and rooms for neighbourhood events. Many explicit constraints in the time assignment and room assignment models which relate to fixed events can be represented implicitly in the minimal perturbation model. For example, the time assignment formulation requires that courses teach a maximum of one lecture event on any given day. If a fixed event from course $c$ is taught on day $d$, any neighbourhood events of this course may not be moved to this day i.e. $T_e \cap T_d = \emptyset \ \forall e \in (c \cap E_N)$. Similarly, we represent the effect of fixed events on the curriculum and teacher constraints. For example, if an event from curriculum $curr$ is fixed in a time period within the neighbourhood, no events from courses in this curriculum can be moved to this time.
period. If the minimal perturbation problem is solved during the enrolment phase of timetabling, we note the set of curricula may be different from the curricula used to construct the timetable. It is important that no enrolled student has a timetable which becomes infeasible after the minimal perturbation problem is solved.

We also consider the complication of long events which lie partially in the neighbourhood, i.e. one or more constituent events are fixed, and one or more are in the neighbourhood. This situation is the simplest if we are not concerned with contiguous room stability. In this case, the constituent events which are in the neighbourhood are permitted to change room, but not time period. If we also wish to enforce contiguous room stability, it is not possible to perturb only a single constituent event without perturbing the full long event. In this situation, we can either expand the neighbourhood so that the long event is entirely included, or fix all parts of the long event. In our implementation, the decision to expand the neighbourhood is only made if this long event is unassigned i.e. is part of the infeasibility which needs to be resolved.

Once these constraints are implicitly satisfied in the neighbourhood sets, the neighbourhood model is only required to enforce constraints which relate to the assignment of neighbourhood events relative to each other. Using notation defined in this section and in the List of Notation (pg. xvii–xviii), we present an integer programming formulation of an event-based neighbourhood perturbation model. In this formulation, the binary variables $x_{etr}$ take the value 1 if event $e \in E_N$ is to be held at time $t \in T_e$ in room $r \in R_{et}$. Solving the following integer programme (6.1)–(6.7) will determine the maximum number of neighbourhood events which can be assigned to a time period and room.

$$\text{maximise } \sum_{e \in E_N} \sum_{t \in T_e} \sum_{r \in R_{et}} x_{etr}$$ (6.1)
subject to $\sum_{e \in E_{tr}} x_{etr} \leq 1 \quad t \in T_N, r \in R_t$ \hfill (6.2)

$\sum_{t \in T_e} \sum_{r \in R_{et}} x_{etr} \leq 1 \quad e \in E_N$ \hfill (6.3)

$\sum_{e \in (e \cap E_F) \cap (T_e \cap T_d)} \sum_{t \in T_e} \sum_{r \in R_{et}} x_{etr} \leq 1 \quad c \in C_N, d \in D_N$ \hfill (6.4)

$\sum_{e \in E_{curr}} \sum_{r \in R_{et}} x_{etr} \leq 1 \quad curr \in CU_N, t \in T_N$ \hfill (6.5)

$x_{etr} - x_{(e-1)(t-1)r} = 0 \quad e \in (E_N \setminus E_F), t \in T_e, r \in R_{et}$ \hfill (6.6)

$x_{etr} \in \{0, 1\} \quad e \in E_N, t \in T_e, r \in R_{et}$ \hfill (6.7)

The objective function (6.1) maximises the total number of events which are assigned to a time period and room. Constraints (6.2) ensure that each available room in each time period is occupied by a maximum of one event, while constraints (6.3) ensure that each event is assigned to at most one room in any time period. Constraints (6.4) ensure that two events from the same course cannot be assigned to any time period on the same day. Only the first event $e \in E_F$ in any long event is included in each constraint, because long events are represented using more than one individual event $e$. Constraints (6.5) ensure that two events from the same curriculum cannot be assigned to the same time period. Lastly, constraints (6.6) enforce strict time contiguity and room stability on the constituent events of a long event.

If room stability is not required for long events, constraints (6.6) can be altered so that each constraint is summed over all suitable rooms (rather than applied as one constraint per room) allowing the assigned room to change between individual event-hours.

Once we have increased the number of assigned events, we wish to minimise the disruption caused by assigning this number of events. For a weighting of penalties $v_{etr}$, we solve the following modified integer programme (6.8)–(6.9), which includes
(6.2)–(6.7).

\[
\text{minimise} \quad \sum_{e \in E_N} \sum_{t \in T_e} \sum_{r \in R_{ut}} v_{etr} * x_{etr} \quad (6.8)
\]

subject to: (6.2)–(6.7)

\[
\sum_{e \in E_N} \sum_{t \in T_e} \sum_{r \in R_{ut}} x_{etr} = |EventsAssignable| \quad (6.9)
\]

The objective (6.8) minimises the total timetable disruption between the proposed timetable solution, and the initial (infeasible) timetable. Each assignment variable is multiplied by a disruption coefficient. The disruption penalties for an event can vary depending on the number of time periods moved, whether the room changes, and how this relates to any fixed events from this course. With sufficient available data, precise disruption penalties can be specified for each perturbation. Constraints (6.9) are introduced to ensure the maximum number of events are assigned.

### 6.4 Course-based Neighbourhood Model

Although the event-based formulation is versatile at modelling the disruption for perturbing each event, it is not able to model the time stability for a course. The time stability quality measure favours scheduling all weekly events from a course at the same time of day. We define \( C_{\text{stab}} \in C \) as the subset of courses that are concerned with time stability. We define \( C_{N\text{stab}} \) as the set of courses which are concerned with time stability and also include an event within the neighbourhood \( N \), i.e. \( C_{N\text{stab}} = C_N \cap C_{\text{stab}} \). Additionally, \( H_N \in H \) is the set of hours from all neighbourhood time periods, \( h_t \) is the hour of the day for time period \( t \), and \( c_e \) is the course with which event \( e \) is associated (i.e. \( e \in c \)). Let the variable \( y_{eh} \) take the
value 1 if any event of course $c$ is held in hour $h$ in the timetable.

Building on the event-based formulation for minimising disruption, we propose the following course-based integer programme.

\[
\text{minimise} \quad (6.8) + \sum_{c \in C_{\text{stab}}} \sum_{h \in H_N} w_{ch} \cdot y_{ch} \\
\text{subject to:} \quad (6.2) - (6.7), (6.9)
\]

\[
x_{etr} - y_{eh} \leq 0 \quad e \in E_N, t \in T_e, r \in R_{et} \\
y_{ch} \in \{0, 1\} \quad c \in C_{\text{stab}}, h \in H_N
\]  

(6.10) (6.11) (6.12)

The objective function (6.10) consists of the event-based disruption (6.8) and an expression to penalise each course for each unique hour of the day it uses for any of its events. Clearly, each course must use a minimum of one unique hour, which is not counted when reporting the penalty to time stability. Constraints (6.11) appropriately tie the values of the $y_{ch}$ variables to the $x_{etr}$ variables.

This course-based IP may be used in place of the event-based IP in Algorithm 1 (on line 12). Although no additional modifications to the algorithm are required, this approach benefits substantially from a tailored neighbourhood definition. In order to minimise the time stability, the neighbourhood should clearly be chosen so that many such potential perturbations can be made.

6.5 Defining the Neighbourhood

As introduced in the previous sections, our method defines a neighbourhood around each infeasibility, instead of formulating a monolithic IP with all events, time periods and rooms. To maximise the effectiveness of our algorithm, each neighbourhood (as defined by $e \in E_N$, $t \in T_e$, and $r \in R_{et}$) is tailored to address the particular
infeasibility which we are attempting to resolve. This section explains how we define the starting neighbourhood and rules for neighbourhood expansion, so that we prioritise favourable perturbation variables.

In addition to focussing the search on variables which correspond to a low disruption, we also consider which variables are the most likely to resolve a given infeasibility. Prioritising such variables reduces the number of expansions required, and allows the neighbourhood to remain as small as possible. This in turn corresponds to smaller integer programmes and a shorter overall solve time.

In Section 6.5.1, we address which time periods to include in the neighbourhood. The set of time periods largely determines the disruptions associated with the perturbation variables, as time perturbations are the most disruptive. In Section 6.5.2, we address which rooms to include in the neighbourhood, which can be focussed to resolve the given infeasibility.

With a definition for the set of time periods and rooms to include in the neighbourhood, the set of events is simply determined. In addition to the unassigned events (which comprise the infeasibility), we consider the events currently assigned to the time period and room which we are introducing to the neighbourhood.

6.5.1 Neighbourhood Time Periods

Each unassigned event has a desired time period, so it is logical to expand the neighbourhood around this time period. The starting neighbourhood will consist only of variables which make a small perturbation from the current timetable, i.e. those which allow movements within and around this time period. As the neighbourhood expands, events from more distant time periods (relative to the infeasibility) are considered, and we permit larger movements of individual events.

When using an event-based model of disruption, we apply a disruption penalty of 1 for each hour of the day moved and 2 for each day moved. There is also a
small penalty ($\epsilon$) for changing room within the same time period. The disruption coefficients provide a simple way to determine the order in which additional time periods are included in the neighbourhood.

For a course-based model, the disruption is computed as the sum of event-based and course-based disruptions (as specified by (6.10)). In this work, we apply a penalty of 5 for each disruption to time stability (i.e. an additional hour). This penalty must be sufficiently large to offset the event-based penalty of moving several events from a course.

In Table 6.1, we provide an example, where infeasibility can be resolved by perturbing one event by two hours. However, this solution (shown in Table 6.1a) results in a course-based disruption of 5, in addition to the event-based disruption of 2 (for a total of 7). The solution in Table 6.1b is able to maintain time stability by moving all 3 events of this course by 2 hours, for a total disruption of 6. This example demonstrates a situation where it is less disruptive to move additional events, to avoid the large course-based penalty. However, if this course consisted of 4 events, or if a perturbation of 3 hours was required, it would be less disruptive to only move the single event (for these chosen penalty coefficients).

When using a course-based model of disruption, it is not useful to simply expand the neighbourhood around a central time period, as is suitable for event-based models. In order to generate variables which avoid a time stability disruption, when the neighbourhood expands into a new time period, the neighbourhood should expand into all time periods in the week at the same hour. The solution in Table 6.1b clearly can only be found if the neighbourhood includes all “10am” time periods across the week.

Finally, we note that in all iterations of Algorithm 1, the perturbation penalties are calculated relative to the starting timetable. This means that an event which is perturbed in an early iteration may have its perturbation penalty reduced (or
6.5. DEFINING THE NEIGHBOURHOOD

(a) A 2-hour perturbation, with loss of time stability

(b) Three 2-hour perturbations, without loss of time stability

Table 6.1: Course-based Perturbation Example

removed) in a later iteration.

6.5.2 Neighbourhood Rooms

When considering a set of neighbourhood time periods around an infeasibility, we consider which rooms from each time period to include. We are most likely to increase the number of assigned events by considering rooms which are approximately the same size as the unassigned events. Including these rooms in the neighbourhood creates opportunities for unassigned events to find a suitable room in a “nearby” time period. However, due to a generally high room utilisation, such rooms are typically occupied by other events. Due to the further complicating factors of staff and curricula (which constrain each event differently), solutions to minimal perturbation problems typically involve a series of perturbations (see Section 6.7).

As introduced in Section 4.2, the feasibility of an event-to-room assignment de-
depends on the size and attributes of the room. Abstractly, the most useful rooms to include in the neighbourhood are those which are approximately the same size and feature the set of attributes as required the unassigned events.

To formalise the method in which rooms are considered for inclusion in the neighbourhood, we use the hierarchy of room and event attribute sets and sizes, as depicted in Figure 4.2 from Section 4.2. This representation allows us to identify which rooms feature the combination of attributes and size that is most likely to resolve a given infeasibility. For an unassigned event, in each neighbourhood time period we initially include the closest superior rooms in the attribute set graph. By definition, this identifies the rooms which “fit” this event the best (in terms of size and attributes). Subsequently, we need to include rooms both from the superior and inferior sides of the unassigned event. The total number of rooms included in the neighbourhood is set as a proportion of all possible rooms.

For example, if the unassigned events require large tutorial rooms, including superior rooms in the neighbourhood ensures that we consider rooms which are larger and suitable for tutorials. This will resolve the infeasibility if such rooms are vacant in nearby time periods. However, we also include inferior rooms (such as smaller tutorial rooms, or larger non-tutorial rooms). If a large tutorial room is occupied by an event which can instead be taught in an inferior room, these inferior rooms must be in the neighbourhood to allow this movement. The inferior and superior rooms are identified from the hierarchy graph, and are added using a greedy breadth-first-search until the required proportion of rooms is met.

For a given neighbourhood, the proportion of rooms added in the infeasible time period may be chosen as greater than the proportion of rooms used in the “distant” time periods. In this work, 50% of all inferior and superior rooms may be considered for an unassigned event in the infeasible time period. In the most distant time period, this proportion is reduced to 20% (with a linear relationship between, based on the
distance penalty). As the neighbourhood expands in time, the proportion of rooms included in existing time periods is allowed to increase, to expand the search.

The effectiveness of this method to identify the most important rooms can be significantly affected by the quality of the (partial) room assignment. When the minimal perturbation problem arises as part of the construction phase of timetabling (e.g. as depicted in Figure 3.1), the room assignment algorithm can ensure a high quality partial room assignment. In this thesis, we use a lexicographic algorithm (see Chapter 5) for room assignment which is Pareto optimal with respect to the event hours, seated student hours, seat utilisation and room preference.

Maximising the event hours in the room assignment ensures that the smallest possible number of events remain unassigned to a room. Therefore, in order to find a suitable room for unassigned events (without causing other events to be unassigned), perturbations to the time assignment are necessarily required.

Maximising the number of seated student hours in the room assignment ensures that any unassigned events will be as small as possible. As a result, if we observe that large events remain unassigned, we can infer a shortage of large rooms in the associated time periods. This would not necessarily be true if we had only maximised the event hours. Without maximising the seated student hours, the existence of unassigned large events could be due to a general lack of rooms of any size.

The room assignment process also maximises the seat utilisation, where it is favourable to assign events to rooms which are closely matched in size. This optimisation is important, particularly for time periods which do not contain an unassigned event themselves, but are adjacent or near to time periods with unassigned events. In the case of a complete room assignment for a particular time period, the previous optimisations (of event hours and seated student hours) would permit assigning small events in larger rooms than necessary, provided it is still possible to assign all events. Maximising the seat utilisation will result in the largest (most flexible)
rooms remaining vacant.

The room assignment process may be further altered to include one or more quality measures addressing the room attributes. For example, prioritising the assignment of events which require many attributes, or assigning events into rooms with a good “fit” in terms of attributes. These are analogous to maximising seated student hours, and seat utilisation respectively. However, for our datasets the attributes are less important than the room sizes, because rooms with a similar size (but different attribute sets) are typically “close” on the hierarchy and will be added to the neighbourhood early in the solution process.

For minimal perturbation problems where we are not able to re-assign rooms using a room assignment algorithm, such as in the enrolment phase, we cannot make the previous inferences about the cause of infeasibility. In this situation it may be possible to resolve the infeasibility by perturbing very few events (or even no events), such as when an unassigned event can be simply assigned to a suitable vacant room in the same time period.

6.6 Results for Construction Phase

To demonstrate our algorithm, we present results on minimal perturbation problems from the construction phase of timetabling. We use the datasets from Semesters 1 and 2 at the University of Auckland in 2010 and 2013.

Each problem is initially addressed using an event-based model of disruption. For the 2010 dataset we also test a course-based model of disruption, as this is a faculty-generated time assignment where time stability is valued for many courses.

The disruption penalties and expansion rules are given in Section 6.5. It is important to note that we demonstrate one particular implementation of the expanding neighbourhood algorithm. As with all timetabling algorithms in this thesis, the spe-
cific implementation details (e.g. choice of expansion rules) should be customised to the relevant problem. For example, based on an understanding of the specific priorities and bottlenecks of a particular university system, it may be more appropriate to readily expand the neighbourhood into many new time periods (allowing large movements in time for individual events), but only consider a small subset of potential rooms.

All computational results in this section are run using Gurobi 5.6 on 64-bit Ubuntu 14.04, with a quad-core 3.5GHz processor (Intel i5-4690).

6.6.1 UoA 2010

As introduced in Section 3.1.2, the timetabling process at the University of Auckland in 2010 involved each faculty generating a time assignment for their own courses. The individual faculty time assignments can be collated into a time assignment for the full university, which is a ‘starting’ time assignment as it is likely to require perturbation. In Table 5.7 from Section 5.4.1 we present the results of the room assignment process on this ‘starting’ time assignment, which generates a high quality partial room assignment.

In this section we solve the minimal perturbation problem to find a suitable time period and room for the unassigned events in each semester.

Event-based Model

Solving the minimal perturbation problem using an event-based IP formulation, within the expanding neighbourhood algorithm (Algorithm 1) gives the results shown in Table 6.2. The first group of rows gives a summary of the overall process, listing the total number of events assigned (which were previously unassigned), the number of iterations of the expanding neighbourhood algorithm required, the total number of IPs solved, and the total time taken. The next group of rows lists the
event-based perturbations applied to the timetable over the entire process. For each perturbation type the number of events perturbed is stated, and the total disruption (weighted for each type of perturbation) is given. We also list the course-based perturbations applied, although these are not measured or penalised by this model. These perturbations refer to the number of extra hours used by events of each course which desires time stability. Finally, the last group of rows gives an indication of the size of the integer programmes, by listing information on the largest (as measured by the number of variables) IP solved.

The results in Table 6.2 demonstrate that it is possible to find a feasible time and room assignment for all events within a short solve time. As stated in Section 6.1, this is our expectation, as the starting timetable is already close to feasibility.

The total amount of disruption to the timetable also appears acceptable. Only a small number of events are required to change time period, and the perturbations are relatively minor (i.e. events remain close to their original time period).

The small size of the largest integer programme demonstrates the importance of focussing the neighbourhood (from Section 6.5). The number of events and time periods in the neighbourhood is significantly smaller than the total number of events and time periods. During the development of this method, we observed significantly larger neighbourhoods before finding a feasible solution, which resulted in a poorer performance.

The short solve time can be partly attributed to the small neighbourhoods which correspond to a low number of variables and constraints in the integer programmes. However, we also note that these problems benefit from an integerising structure in the IP, which is similar to that of assignment problems (or bipartite matching). Optimal integer solutions are typically found near (or at) the optimal LP solution.
Summary

<table>
<thead>
<tr>
<th></th>
<th>Semester 1</th>
<th>Semester 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assigned Events</td>
<td>26</td>
<td>23</td>
</tr>
<tr>
<td>Iterations of Algorithm 1</td>
<td>17</td>
<td>10</td>
</tr>
<tr>
<td>Neighbourhood IPs Solved</td>
<td>40</td>
<td>25</td>
</tr>
<tr>
<td>Total Solve Time (s)</td>
<td>0.9</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Event-based Perturbations

<table>
<thead>
<tr>
<th></th>
<th>Semester 1</th>
<th>Semester 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change of Room</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>0 days, 1 hour</td>
<td>23</td>
<td>24</td>
</tr>
<tr>
<td>0 days, 2 hours</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>0 days, 3 hours</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Total Penalty</td>
<td>45</td>
<td>42</td>
</tr>
</tbody>
</table>

Course-based Perturbations

<table>
<thead>
<tr>
<th></th>
<th>Semester 1</th>
<th>Semester 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 extra hour</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2 extra hours</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Total Penalty</td>
<td>15</td>
<td>25</td>
</tr>
</tbody>
</table>

Largest IP

<table>
<thead>
<tr>
<th></th>
<th>Semester 1</th>
<th>Semester 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Events $</td>
<td>E</td>
<td>$</td>
</tr>
<tr>
<td>Time Periods $</td>
<td>T</td>
<td>$</td>
</tr>
<tr>
<td>Rooms $</td>
<td>R</td>
<td>$</td>
</tr>
<tr>
<td>Variables</td>
<td>4977</td>
<td>9131</td>
</tr>
<tr>
<td>Constraints</td>
<td>1323</td>
<td>1752</td>
</tr>
<tr>
<td>Solve Time (s)</td>
<td>0.1</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Table 6.2: UoA 2010 Event-based Timetable Construction Results

Course-based Model

Solving the same problem as the previous section using a course-based IP formulation, gives the results shown in Table 6.3. This table uses the same row headings as Table 6.2.

The results in Table 6.3 demonstrate that it is possible to consider a more complex objective involving auxiliary variables, and still maintain relatively short solve times.
6.6. RESULTS FOR CONSTRUCTION PHASE

<table>
<thead>
<tr>
<th>Summary</th>
<th>Semester 1</th>
<th>Semester 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assigned Events</td>
<td>26</td>
<td>23</td>
</tr>
<tr>
<td>Iterations of Algorithm 1</td>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td>Neighbourhood IPs Solved</td>
<td>24</td>
<td>16</td>
</tr>
<tr>
<td>Total Solve Time (s)</td>
<td>7.0</td>
<td>5.0</td>
</tr>
</tbody>
</table>

| Event-based Perturbations                    |            |            |
| Change of Room                              | 10         | 18         |
| 0 days, 1 hour                              | 18         | 27         |
| 0 days, 2 hours                             | 8          | 2          |
| 0 days, 3 hours                             | 0          | 1          |
| 1 day, 0 hours                              | 8          | 8          |
| 1 day, 1 hour                               | 3          | 0          |
| Total Penalty                               | 59         | 50         |

| Course-based Perturbations                  |            |            |
| Total Penalty                               | 0          | 0          |

| Largest IP                                  |            |            |
| Events $|E_N|$                                   | 345        | 381        |
| Time Periods $|T_N|$                                | 15         | 15         |
| Rooms $|R_N|$                                   | 66         | 61         |
| Variables                                   | 48201      | 68263      |
| Constraints                                 | 28037      | 37813      |
| Solve Time (s)                              | 1.7        | 2.7        |

Table 6.3: UoA 2010 Course-based Timetable Construction Results

The total amount of disruption to the timetable is similar to when the event-based model is used, except it consists of an increased event-based disruption and no course-based disruption. We specifically observe an increase in the number of “lateral” perturbations of 1 day and 0 hours, as these avoid incurring a penalty to the time stability.

The size of largest neighbourhood is significantly greater for these problems than the event-based problems. As explained in Section 6.5.1, this is due to the require-
ment of considering time periods across the full week, rather than a smaller number centred around a particular time period.

The course-based IPs require a significantly longer solve time than in the event-based case. In addition to an increased number of variables and constraints, this is due to the increased opportunities for fractionality, and a corresponding integrality gap.

6.6.2 UoA 2013

As introduced in Section 3.1.3, the 2013 data at the University of Auckland requires us to generate both a time and room assignment. The time assignment results are given in Section 4.4.2, and a partial room assignment is found in Section 5.4.2. Following on from the partial room assignment, we must solve a minimal perturbation problem to resolve the infeasibility of unassignable events.

Event-based Perturbation

Solving the minimal perturbation problem using an event-based IP formulation, within the expanding neighbourhood algorithm (Algorithm 1) gives the results shown in Table 6.4. This table uses the same row headings as Table 6.2.

The problems addressed in Table 6.4 are notably larger (in terms of the number of unassigned events) than those faced in the 2010 dataset. In particular, for the Semester 2 problem, many unassigned events require a large number of iterations of Algorithm 1, many IPs solved, and a greater overall solve time. The size of the largest neighbourhood is also greater than the neighbourhoods for the 2010 event-based results. This can be explained by the nature of the 2010 data, which originates from a “rolled-forward” timetable which is close to feasibility. When many events are unassigned in a small number of neighbouring time periods, several neighbourhood expansions may be required. However, consistent with the philosophy of our method,
6.6. RESULTS FOR CONSTRUCTION PHASE

<table>
<thead>
<tr>
<th>Summary</th>
<th>Semester 1</th>
<th>Semester 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assigned Events</td>
<td>33</td>
<td>110</td>
</tr>
<tr>
<td>Iterations of Algorithm 1</td>
<td>17</td>
<td>37</td>
</tr>
<tr>
<td>Neighbourhood IPs Solved</td>
<td>37</td>
<td>114</td>
</tr>
<tr>
<td>Total Solve Time (s)</td>
<td>5.2</td>
<td>33.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Event-based Perturbations</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Change of Room</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>0 days, 1 hour</td>
<td>35</td>
<td>100</td>
</tr>
<tr>
<td>0 days, 2 hours</td>
<td>16</td>
<td>25</td>
</tr>
<tr>
<td>0 days, 3 hours</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>0 days, 4 hours</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>1 day, 0 hours</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>1 day, 1 hour</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Total Penalty</td>
<td>70</td>
<td>198</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Largest IP</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Events $</td>
<td>E_N</td>
<td>$</td>
</tr>
<tr>
<td>Time Periods $</td>
<td>T_N</td>
<td>$</td>
</tr>
<tr>
<td>Rooms $</td>
<td>R_N</td>
<td>$</td>
</tr>
<tr>
<td>Variables</td>
<td>11999</td>
<td>63971</td>
</tr>
<tr>
<td>Constraints</td>
<td>18692</td>
<td>37813</td>
</tr>
<tr>
<td>Solve Time (s)</td>
<td>0.2</td>
<td>0.9</td>
</tr>
</tbody>
</table>

Table 6.4: UoA 2013 Event-based Timetable Construction Results

the largest neighbourhood remains at a manageable size with a solve time of less than 1 second.

As with the previous results, the majority of perturbations correspond to an event moving by 1 hour. However, in the case of the most difficult data from Semester 2, a small number of events are perturbed by 4 hours. Depending on details of the specific events involved, this may be acceptable or it may be considered too great a perturbation. For example, if this event is part of a curriculum taught as a morning or afternoon programme, a perturbation of 4 hours may either remain within the existing time bounds of this curriculum, or constitute a substantial 4 hour extension.
Recall that for this problem, the perturbations are made to a time assignment which we generated in Section 4.4.2. Therefore, we are able to directly assess the disruption caused by our perturbations, by looking at the change in the time assignment quality measures. The initial (pre-perturbation) and final (post-perturbation) values for each quality measure are given in Table 6.5. Although each objective worsens (as expected), the final solution can still be considered to be of high quality.

Ultimately, these construction-phase results (along with those from 2010) can be considered promising, as a feasible time and room assignment is found for all events.

### 6.7 Results for Enrolment Phase

To demonstrate our algorithm on another type of minimal perturbation problem, we present results from the enrolment phase of the timetabling process. We analyse two scenarios which can cause infeasibility in an existing timetable, based on the datasets from Semester 2 at the University of Auckland in 2010 and 2013. The same perturbation penalties and solve parameters are used as in Section 6.6.

#### 6.7.1 UoA 2010 Over-enrolment

This example is used to analyse the problem arising when courses are subject to an unexpectedly high enrolment, such that the existing room assignment is no longer feasible. Table 6.6 defines such a situation for two introductory courses which are
6.7. RESULTS FOR ENROLMENT PHASE

<table>
<thead>
<tr>
<th>Course Name</th>
<th>Time Periods</th>
<th>Planned Enrolment</th>
<th>Revised Enrolment</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOCIO 100</td>
<td>Mon 12pm, Thu 2pm</td>
<td>320</td>
<td>500</td>
</tr>
<tr>
<td>LAW 121G</td>
<td>Mon 12pm, Wed 12pm, Fri 12pm</td>
<td>269</td>
<td>500</td>
</tr>
</tbody>
</table>

Table 6.6: Scenario Changes to Course Enrolments

scheduled during “peak” time periods. These particular courses are susceptible to an unpredictable enrolment, as they are available to new-entrant students, and may be taken as an elective by students from many academic programmes.

This situation is modelled as 5 unassigned events, because the existing room assignments are no longer valid. If this situation had arisen within the construction phase of timetabling, we could initially re-solve the room assignment algorithm. This can improve the partial room assignment, such as assigning one of these events and leaving a smaller event unassigned. However, this is typically not suitable for an enrolment phase problem, as it may result in significant perturbations to the room assignment. Therefore, we address this situation solely as a minimal perturbation problem.

**Event-based Perturbation**

We first solve the problem of unassigned events using an event-based IP formulation. This problem is sufficiently small (in terms of the number of unassigned events) for the solution to be presented visually, as shown in Table 6.7. The 5 unassigned events (i.e. the events of “SOCIO100” and “LAW121”) are bolded, and perturbations are demonstrated using arcs. Each perturbation affecting the time assignment is shown using a bold arc. When the arc from one event points to another event, this represents the former event “displacing” the latter event, in terms of occupying its assigned room. This must be accompanied by a perturbation of the latter event to...
find a new suitable room. A simple case is shown on Wednesday, between “LAW121” and “PROP344”. In this case, there is a vacant room at 11am which is suitable for “PROP344”, but not for “LAW121”. If the vacant room were suitable for the unassigned event, a room perturbation would not have been required in the optimal solution. A more complex chain of perturbations is shown on Monday, originating from “LAW121”.

In the perturbations on Friday, note that the event from “LAW121” does not change time period, but causes “STATS108” to move to an hour earlier instead. This type of manoeuvre commonly occurs in solutions to minimal perturbation problems, and is understood through the set of suitable time periods for each event. Some events are relatively inflexible in potential movements (due to curricula, staffing and other requirements), and a feasible solution may involve moving another event which has greater flexibility.

This solution corresponds to a total event-based disruption of 6, comprised of 6 1-hour perturbations. Solving the 12 IPs required for this problem was very rapid,
as each involved less than 1500 variables, and terminated in less than one second.

**Course-based Perturbation**

In the previous solution (Table 6.7), three courses (“LAW121”, “CHEMM121”, and “STATS108”) incurred a disruption to time stability. To address the situation where time stability is important, we solve the minimal perturbation problem using a course-based model of disruption.

The solution to this problem is presented visually in Table 6.8, where the impact of modelling time stability is evident. For example, all 3 events from “LAW121” are reassigned to one hour later in the day to maintain the structure. Also, to preserve time stability for “HIST102” which moves from 11am on Monday, an event on Wednesday similarly moves an hour later. This solution shows the lateral perturbation of an event from “COMLA201”, which moves from 1pm on Friday to 1pm on Thursday, so that time stability is unaffected. Finally, an interesting pair of perturbations occur between events from “POLIT113” and “ANTHR106” where a simple swap of assigned room occurs at 12pm on Thursday. This is explained by observing that “ANTHR106” consists of a long event, which requires the same room for the events at 12pm and at 1pm. Therefore, this room swap is ultimately part of the chain of perturbations used to assign the Friday event from “LAW121”.

This solution corresponds to a total event-based disruption of 9, comprised of 7 1-hour perturbations, and 1 1-day perturbation. Although this is a greater penalty than in the event-based solution, we are now able to resolve the infeasibilities with no disruption to the time stability. Solving the IPs required for this problem was again rapid, with less than 5000 variables in the largest case, and cumulative solve time of less than one second.
6.7. RESULTS FOR ENROLMENT PHASE

<table>
<thead>
<tr>
<th>Monday</th>
<th>Tuesday</th>
<th>Wednesday</th>
<th>Thursday</th>
<th>Friday</th>
</tr>
</thead>
<tbody>
<tr>
<td>10am</td>
<td>HIST102</td>
<td>CHEMM121</td>
<td>HIST102</td>
<td></td>
</tr>
<tr>
<td>11am</td>
<td>ACCTG102</td>
<td>HIST364</td>
<td>POLIT113</td>
<td>ANTHR106</td>
</tr>
<tr>
<td>12pm</td>
<td>SOCIO100</td>
<td>LAW121</td>
<td>LAW121</td>
<td></td>
</tr>
<tr>
<td>1pm</td>
<td>LAW121</td>
<td>ACCTG221</td>
<td>ANTHR106</td>
<td></td>
</tr>
<tr>
<td>2pm</td>
<td>SOCIO100</td>
<td></td>
<td>PSYCH203</td>
<td></td>
</tr>
<tr>
<td>3pm</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6.8: Course-based Solution

6.7.2 UoA 2013 Loss of Room Availability

This example analyses the problem arising when a room becomes unavailable, after it has been used in the timetabling process and has events currently assigned. Although the total number of available rooms is decreased, if the unavailable room is relatively common (i.e. there are equivalent or superior rooms in the hierarchy graph), this problem can typically be resolved. The unassigned events are necessarily spread across many different time periods, so it is likely that each infeasibility can be resolved locally.

For this example we remove two rooms, “206-201” and “206-220”, which are shared by events from many departments, and are originally scheduled to host 52 events in total. These rooms possess a standard set of lecture room attributes
6.7. RESULTS FOR ENROLMENT PHASE

<table>
<thead>
<tr>
<th>Summary</th>
<th>Semester 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assigned Events</td>
<td>52</td>
</tr>
<tr>
<td>Iterations of Algorithm 1</td>
<td>18</td>
</tr>
<tr>
<td>Neighbourhood IPs Solved</td>
<td>37</td>
</tr>
<tr>
<td>Total Solve Time (s)</td>
<td>5.92</td>
</tr>
</tbody>
</table>

| Event-based Perturbations                    |            |
| Change of Room                              | 54         |
| 0 days, 1 hour                              | 33         |
| 0 days, 2 hours                             | 4          |
| 0 days, 3 hours                             | 1          |
| Total Penalty                               | 44         |

| Largest IP                                   |            |
| Events $|E_N|$                                      | 296        |
| Time Periods $|T_N|$                                    | 4          |
| Rooms $|R_N|$                                      | 215        |
| Variables                                    | 24237      |
| Constraints                                  | 13600      |
| Solve Time (s)                               | 0.3        |

Table 6.9: UoA 2013 Loss of Room Availability Results

(e.g. two data projectors, tiered seating), and can seat 44 and 107 students respectively. Solving this problem using an event-based model gives the results shown in Table 6.9.

Similar to the results from Section 6.6, Table 6.9 demonstrates that it is possible to find a feasible time and room assignment for all events within a short solve time. However, the total amount of disruption penalty can be considered to be particularly low for this number of unassigned events. This is because some events are able to be assigned to a suitable room without perturbing the time assignment, i.e. suitable vacant rooms exist in some of the time periods.

The results in this section demonstrate both the broad application of minimal
perturbation problems, and the effectiveness of our proposed algorithm. The visual representations of even the simple solutions also suggest it would difficult for a human timetabler to identify such a set of perturbations quickly.

6.8 Discussion

The results in this chapter demonstrate that the expanding neighbourhood algorithm is well-suited to solving minimal perturbation problems. By leveraging the exactness of integer programming, we are able to obtain high quality solutions within a short solve time. This fulfils the most important requirements for the use of this method in a practical setting. To conclude, we discuss potential improvements in the way this method can be implemented, which improve the modelling of real-world factors and improve the quality of solution in practice.

By definition, the minimal perturbation problem considers a high quality solution to be one with a small amount of perturbation from the original timetable. This corresponds to the implicit assumption that the existing timetable is of high quality or that a perturbation causes a disruption. This is true in the case of problems in the enrolment phase, or during the construction phase when a starting time assignment has been provided by faculties. However, in this thesis we also deal with construction phase problems where we have generated the time assignment algorithmically. In this case, since we have knowledge of the quality measures which are used to generate the existing time assignment, we have been able to measure the disruption to each quality measure directly. Although not conducted in this work, we could have represented the quality for each event-time assignment in the minimal perturbation model as the time preference directly. More advanced quality measures such as curriculum compactness can also be considered, although they notably relate to a large number of events which are unlikely to all be in the neighbourhood
simultaneously. In this case, the disruption is calculated based on the locations of the fixed events, e.g. perturbations which extend the existing boundary hours of the curriculum (i.e. the latest and earliest hours) should be penalised.

Another issue which applies to problems in the construction phase is whether we should consider room assignment quality measures. The minimal perturbation algorithm presented in this chapter solely attempts to find a feasible room assignment, without regard for the quality. However, we could also use this algorithm even after a feasible solution is found, to improve the quality such as the room preference. In this case, we could consider a series of small neighbourhoods, centred around time periods with poor room preference, and attempt to find an optimal re-assignment which improves the room preference while minimally perturbing the time assignment. This concept is less applicable for enrolment phase problems, which incur a loss of quality for any adjustment to current timetable.

A further quality consideration relates to the fairness or equity of disruptions caused for each party (e.g. faculties). In the construction phase, a desirable solution may be one where the total disruption is spread equally between multiple parties. Although our method addresses each infeasibility separately, this could be represented throughout the solve process by applying a greater cost of disruption for parties which have already been disproportionately disrupted.

In the enrolment phase, a “fair” apportionment of the disruption may be substantially different, if one party is culpable for an unexpected change. For example, if a faculty wishes to introduce additional tutorial events to the timetable, it may be appropriate to restrict timetable changes to other events from this faculty. This itself can form a neighbourhood definition where we place a large penalty on moving events from other faculties. These less desirable perturbations will only be added to the neighbourhood if the infeasibility cannot be resolved with the original faculty’s events. In our experience with manual perturbations, this occurs in practice
where faculties may make manual perturbations providing they do not affect the assignments of another faculty.

A final comment on the disruptiveness of each feasible perturbation, involves a consideration of the number of perturbations made. In our approach, two one-hour perturbations are equivalent in disruption to one two-hour perturbation. However, due to the nature of the expanding neighbourhood algorithm, the former is more likely to be part of a solution. This often corresponds to an equitable solution, whereas it is arguably more favourable from a practical utilitarian perspective to make one two-hour perturbation. A perturbation of one hour causes a certain inconvenience, but it is possible that a two hour perturbation is less than “twice as inconvenient”. From the perspective of timetabling staff, it may also be helpful to limit the number of affected events, due to the manual work involved in approval. This is an implementation detail, and potentially the different solutions should be provided to the timetabler (as options in a decision support system), using this external information.

It is also important to consider the situation when the minimal perturbation problem is infeasible. This may occur if an essential room in the university becomes unavailable, such as one with specialist equipment, or the largest lecture theatre. If it is not possible to reassign all events by perturbing the time and room assignments, timetablers and faculty must assess changes to the data itself. For example, whether a course can be re-sectioned into smaller events, or whether this equipment can be moved to another room. In each case, the minimal perturbation algorithm from this chapter can be utilised to explore whether a proposal can feasibly reassign all events, and the total level of disruption.

We recall that we are not only looking for a feasible solution; we are also aiming to minimise the disruption caused by our perturbations. Although the latter is our objective function within each neighbourhood IP, feasible solutions may be found
which cause significant disruption to the timetable. In this case, it may be possible to expand the neighbourhood further, and find a solution with a lower disruption. In our early tests, this method marginally improve results. However, it became less effective as we conducted further research into defining the neighbourhood efficiently, which identifies the low disruption perturbations which are likely to be feasible. In the case of a problem where it is feasible to explore the larger neighbourhood size, or in a less understood problem domain, the method may still be desirable.

Finally, we revisit the assumption that all events will have a preferred time period, which perturbations for that event are based upon. In the case of events from a new course (which was not considered in the construction process), there may be no existing time period. In this situation, one resolution is to simply assign a feasible time for each event which avoids any curriculum or teacher conflicts. This can be solved as a regular minimal perturbation problem for each event, potentially with a low penalty for perturbing these events due to their arbitrary placement. A more complex resolution is to use a style of neighbourhood where we consider many time periods simultaneously, but only a very small subset of available rooms. All new events are included in the neighbourhood (although there is no perturbation penalty), and neighbourhood expansions consist of increasing the number of rooms considered. This method minimises the disruption to the rest of the timetable, since the new course is required to be flexible.
In this chapter, we examine the place of automated timetabling algorithms as part of a complete timetabling system. An important component of each timetabling system deals with the collection and representation of data. This includes detailed information about all courses, staff, rooms etc. within the university, as well as the choice and weighting of quality measures. Another component of a timetabling system deals with the provision of reports and analytics to university administrators, such as information on the utilisation of rooms. The complexity of insights offered from these auxiliary outputs to the timetabling process is dependent on the capability of the algorithms.

In Section 7.1, we provide a brief overview of the components in a typical timetabling system. In Section 7.2, we compare the solution generated in this thesis for the University of Auckland 2013 data, with the implemented solution. We also discuss the important caveats of such a comparison which arise from data complications and the general limitations of timetabling data. In Section 7.3, we discuss the use of the algorithms developed in this thesis to conduct scenario analysis for mid-term and long-term planning. In Section 7.4, we experiment with multi-objective
optimisation to show the tradeoffs between quality measures, and to model new measures of quality such as equity.

7.1 Introduction

In the process of developing automated timetabling algorithms, it is important to consider the context of the broader timetabling system, and the relevance of timetabling to university administration. We depict the components of a general timetabling system in Figure 7.1, which is adapted from Figure 1 in Dimopoulou and Miliotis (2001). The timetabling staff (and sometimes faculty representatives) interact with timetabling software, via a user interface, to perform three main functions. Firstly, the software facilitates the maintenance of databases, which store a wide range of requisite information. Secondly, functionality is provided for algorithm-assisted timetabling, which may be simple or complex. Thirdly, the software provides the ability to view components of the timetable in isolation (e.g. the occupancy of a particular room), and evaluate the quality of the timetable. This final component is also used to generate reports for senior administrative staff, for example the room utilisation in each time period across the week, or the fulfilment of staff preferences.

A small number of papers in the literature address the practical issues of implementing a timetabling system. Schimmelpfeng and Helber (2007) provide a strong coverage of their experience, particularly on the difficulties using data collected from teaching staff in the context of their specific problem. Carter (2001) and Fizzano and Swanson (2000) also discuss the use of their methods in practice, and comment on the value that university administrators were able to derive from scenario analysis using their algorithms.

Despite these cases, a comment frequently made about the timetabling literature
is that too few papers address practical issues (McCollum, 2007). This thesis has so far focussed on the development of generalised timetabling algorithms, with reference to the modelling of different types of data. In the remainder of this chapter, we revisit the complexities of timetabling data, and pay particular attention to the use of our algorithms for the generation of reports and analytics.

7.2 Benchmarking for UoA 2013

In this section, we provide a comparison of our generated solutions to those which were implemented in practice, for the University of Auckland 2013 problems. We also address the caveats of making such a comparison, and discuss the related issues surrounding the collection and interpretation of timetabling data.
7.2.1 Solution Comparison

Table 7.1 shows the quality of our timetabling solutions from the approach presented in this thesis (“IP”), compared to the solutions implemented in practice. Quality measures from both the time and room assignment problems are shown to give an overview. Note that all quality measures are expressed in a maximisation sense (i.e. larger is better), except for the curriculum compactness.

<table>
<thead>
<tr>
<th></th>
<th>Semester 1</th>
<th></th>
<th>Semester 2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IP</td>
<td>Implemented</td>
<td>IP</td>
<td>Implemented</td>
</tr>
<tr>
<td>Event Hours</td>
<td>4490</td>
<td>4490</td>
<td>5322</td>
<td>5322</td>
</tr>
<tr>
<td>Time Preference</td>
<td>3458</td>
<td>3179</td>
<td>3875</td>
<td>3656</td>
</tr>
<tr>
<td>Curriculum Compactness</td>
<td>408</td>
<td>933</td>
<td>478</td>
<td>592</td>
</tr>
<tr>
<td>Room Preference</td>
<td>2496</td>
<td>1923</td>
<td>3199</td>
<td>2512</td>
</tr>
<tr>
<td>Course Room Stability</td>
<td>-49</td>
<td>-395</td>
<td>-114</td>
<td>-495</td>
</tr>
<tr>
<td>Spare Seat Robustness</td>
<td>3266.3</td>
<td>2710.4</td>
<td>3573.3</td>
<td>2983.1</td>
</tr>
</tbody>
</table>

Table 7.1: UoA 2013 Solution Quality Comparison

As expected, our approach performs well, and is able to attain a more favourable value for each quality measure than the implemented solution. However, as with the comparisons to DTU (in Sections 5.5.2–5.5.3), we acknowledge the caveats associated with conducting experiments in an academic environment. The context of interpreting each specified quality measure is covered in this section, while more general data issues are discussed in Section 7.2.2.

The implemented solution is of high quality with respect to the time preference and room preference, which are known to be priorities for the TSO. However, we know from discussions with the TSO that they would like to improve the course room stability, which is difficult with the current system. As detailed in Section 5.3, it is also a particularly difficult quality measure to optimise (manually or computationally) as it relates to patterns rather than individual events. Therefore,
the significant improvement in course room stability is considered to be a particular strong-point of our approach. It is also worth noting that the measurement of course room stability does not depend on preference coefficients, which may introduce an arbitrary component to measuring quality.

The remaining quality measures of curriculum compactness and spare seat robustness are less important in this comparison. The curriculum compactness is partially addressed by the time preference, which avoids the first and last periods of the day. The curriculum compactness quality measure is generally considered as less important at the University of Auckland relative to much of the timetabling literature. The spare seat robustness quality measure formed an important part of the timetabling process from 2010. However, because the 2013 timetabling process involves significantly more manual intervention, robustness may have been considered more abstractly. For example, consider a particular event-to-room assignment which appears not to be robust by our calculation. It is possible that the relevant course is known to be associated with little variability in enrolment, or that a degree of robustness is already considered in the faculty’s enrolment estimate. No such detailed data exists in the timetabling system, so comparisons based on the robustness measure are less definitive.

Finally, an important comparison which is not shown in Table 7.1, is the time taken to generate each respective solution. It is difficult to precisely estimate the time required to generate the implemented solutions, as the data collection is conducted simultaneously. The majority of data collection and construction takes place in an intensive two week period as a full-time collaboration with representatives from each faculty. This is followed by substantial additional work for the TSO in the following weeks which resolves problems in the proposed timetable. By contrast, the method detailed in this thesis is able to generate a feasible solution to the complete timetabling problem in less than 5 minutes. This includes solving IPs to
optimality for basic (yet important) measures of quality such as time preference and room preference. A solution which requires more complex quality measures such as curriculum compactness and course room stability as used in this thesis, requires less than 4 hours.

The conclusions of our comparison for the University of Auckland problems are similar to those made for the room assignment problem at DTU. We have validated the quality of our solutions, and demonstrated that the problem can be solved within a relatively short timeframe.

7.2.2 Data Issues

To meaningfully represent a complex and diverse university system, a large amount of high quality data is required. However, it is inevitably not feasible to obtain precise data for how every conceivable quality measure is valued by every university stakeholder. As a result, the quality measures which are used should be chosen to provide the best representation which can be modelled. Similarly, for some types of timetabling constraints, we operate in the knowledge that these may be deliberately violated (or waived) in order to be able to assign all events.

An example from the implemented solutions for the University of Auckland exists where curriculum conflicts have been permitted, i.e. two events from a curriculum are taught in the same time period. However, this only occurs within “less important” curricula, such as those which apply to choices of electives, rather than core sets of courses. The specific data on the importance of curricula could be used to model curricula violations as a quality measure, however this data is not formalised. It is likely that our solutions enforce some constraints for conflict-free curricula, where it would be better to violate the constraint and achieve a higher value for another quality measure. A similar concept also applies to staff constraints, where conflicts are allowed in particular circumstances, as addressed by De Causmaecker et al.
Another example of constraints which are commonly waived, are the room size and room attribute requests made for each course. In some instances, a faculty may use such requests to pick a preferred room for their courses, which ultimately circumvents the room assignment process. This room may be surplus to the size or attribute requirements, and be better utilised by events of another course. A similar situation is when the estimate for the number of students includes some robustness “leeway”. This has been observed in data from the early construction phase of timetabling, where courses have requested room types which do not exist, such as very large tutorial rooms.

Clearly, data validation is an essential component of the data collection process when the TSO interact with faculty administrators. However, these observations lead to the question of which other preferences and constraints exist but are not represented.

Acquisition of the necessary data for the University of Auckland 2013 dataset has been a particularly complex process. Specifically, we have been required to extract data from a complex database system, which is part of a third-party commercial timetabling package. Because the data is presently represented in the generalised structure of third party software, it requires re-interpretation in the context of the University of Auckland to be meaningfully used. Furthermore, the technique used by the software to generate a timetable has a significant impact on which data is represented. Many quality decisions are made in the mind of the timetabler because much of the timetable is manually constructed. If such quality considerations are not formally represented in the data, we are unable to incorporate them in our models.

Although extracting data from the University’s timetabling system has been sufficient to validate the algorithms in this thesis, it is clearly more favourable to access the data through direct collaboration with experienced timetablers and the
7.3 Scenario Analysis

A significant benefit of using an automated algorithm for timetabling is the reduction in the number of “man-hours” required to find a solution. In addition to improving the generation of a timetable for the university, this efficiency affords the capability of performing a wide variety of scenario analyses.

Broadly, any potential change in the data can be considered as a “scenario”, and evaluated for its effect on the feasibility or quality of the solutions. In Chapter 6, we have demonstrated the use of the minimal perturbation problem to address scenarios which exist in the enrolment-phase of timetabling. However, this is only one component, which exists in the context of repairing infeasibility for an already published timetable (e.g. coping with the loss of a room). In this section, we discuss the use of timetabling construction algorithms to evaluate scenarios in the context of mid-term and long-term planning.

7.3.1 Mid-term Planning

Scenario analysis may be useful to evaluate proposals in mid-term planning, which apply to the upcoming year. Before a timetable is constructed, we can evaluate the importance of the use of a particular group of rooms. This may be useful if construction work in the university requires that certain rooms are unoccupied for an entire timetabling semester. We can actually evaluate these scenarios to determine not just the feasibility of such a proposal, but the impact on each quality measure and whether it is within the bounds of an “acceptable” inconvenience.

A similar scenario involves altering which weekly time periods are used for teaching. If the final time period of each day is considered undesirable, we can evaluate...
how the removal of this time period would affect generating a solution. In the general case, this proposal would increase the congestion for rooms (potentially to the point of infeasibility), and would cause greater conflicts in curricula, which reduces the students’ choice of courses. The opposite proposal can also be considered, where new time periods are introduced to address a shortage of large lecture theatres. Although changing the available time periods is not common in course timetabling, it is much more accepted in the closely related problem of examination timetabling.

In addition to scenarios which affect the university’s core timetabling resources (time periods and rooms), mid-term planning decisions can include alterations to the timetabling constraints and quality measures. For example, it would be possible to relax the contiguous room stability requirement on long events (or the constraints (5.14) and (5.15) used at DTU), to observe the degree of improvement in quality measures. Alternatively, any constraint can be reformulated as a quality measure. As mentioned in Section 7.2.2, this particularly applies to modelling curriculum conflicts which are treated as “soft” constraints in practice.

7.3.2 Long-term Planning

Scenario analysis is also useful in the context of long-term planning. An obvious example is where new construction projects, and/or discontinuation of old facilities, can be considered from a timetabling perspective. Deciding on the room types which will add the most value to the university requires an understanding of the current bottlenecks in the timetabling system (and how this is expected to change in the long-term). One source of this information is the utilisation figure for each room type in the university. For this purpose, we look at the room hierarchy (i.e. Figure 4.2) which identifies the utilisation for types of rooms, rather than individual rooms. As reported in the literature (Beyrouthy et al., 2007), the average room utilisation is between 60% and 80% (over all time periods). However, it is well-known that the
largest lecture theatres are utilised to a much greater degree, in the 90% range.

Rather than experimenting with many proposals for which types of rooms should be added or removed, a more algorithmic approach can be used. For a time assignment which has a feasible room assignment (i.e. all events can be assigned to a room), we can solve a modified room assignment model (5.1)–(5.5) to find the minimum number of rooms (perhaps out of a set of candidates) which are required. This is modelled by introducing a set of auxiliary variables where \( y_r \) takes the value 1 if room \( r \in R \) is used in the solution. A set of cost coefficients \( c_r \) for each room may also be used where rooms are priced differently. The objective function given as (7.1) is used in the room assignment model from Section 5.2.3 instead of (5.1). Finally, the constraints (7.2) are introduced, which “tie” the auxiliary variables to the event-to-room variables of the underlying event-based formulation.

\[
\begin{align*}
\text{minimise} \quad & \sum_{r \in R} c_r * y_r \\
\quad & x_{pr} - y_r \leq 0 \\
\quad & p \in P, r \in R_p
\end{align*}
\]

Solving this model determines which rooms can be feasibly removed (perhaps from a candidate set of rooms). To model proposed new facilities, dummy rooms may also be added, with a particular cost for their use. To explore the quality implications of using a subset of available rooms, this can be part of the lexicographic optimisation process. More complexity can also be added, such as constraints on the \( y_r \) variables requiring a set of rooms (e.g. an entire building) to be considered together.

In each example given, it is clear that conducting scenario analysis requires human decision makers (planners and timetablers) to interact in an iterative fashion with the proposed algorithms. The mid-term planning examples simply utilise the capabilities of the timetabling construction algorithms proposed in this thesis. However, evaluating more sophisticated long-term planning scenarios ultimately requires
the involvement of operations research practitioners and the development of more customised algorithms.

7.4 Multi-Objective Analysis

Practical course timetabling requires the consideration of many objectives to address the requirements of multiple university stakeholders. This section provides a discussion of methods which can be used to deal with multiple objectives. The multi-objective optimisation terminology is defined in context, however, we refer to Ehrgott (2005) for precise mathematical definitions.

7.4.1 Lexicographic Optimisation

Lexicographic optimisation involves using a strict ordering of objective functions in decreasing order of importance. Each objective is maximised in turn, with previous objectives fixed at their optimal value. This is appropriate when one objective has absolute priority over another.

This principle has been applied multiple times throughout this thesis. Most important is the timetabling philosophy that assigning as many events as possible (which is abstractly referred to as feasibility) is an absolute priority over finding a high quality solution.

Lexicographic optimisation is also applied between quality measures. For example, at the University of Auckland it is more important that events are taught close to their home faculty (room preference) than it is for the events of a course to be held in a stable room. This principle is applied in the results for Chapter 5. However, less certain applications of a lexicographic ordering are also used, which are considered in Section 7.4.3.
7.4.2 Max-ordering Optimisation

Max-ordering optimisation involves the maximisation of the lowest (worst) value from a set of objectives. In the context of achieving a fair allocation of resources to multiple parties, this is commonly referred to as an application of Rawls’ Second Principle of Justice (Rawls, 2009).

This principle can be also applied iteratively, where the quality for the next worst-off party is maximised, and so on lexicographically for all parties. This is known as max-min fairness, or lexicographic max-ordering optimality in the general case.

For course timetabling problems, it is clearly desirable to achieve a level of fairness when allocating shared university resources (access to rooms) between parties (faculties). However, in the timetabling literature only Mühlenthaler and Wanka (2014) address this concept, using a simulated annealed metaheuristic for max-min fairness.

In this thesis, each quality measure has been calculated using the utilitarian principle where quality is summed across all parties (as standard in timetabling literature). To experiment with timetabling fairness, we consider the fairness of the course room stability quality for each faculty. For example, Table 7.1 (from Section 7.2) shows a total course room stability penalty of 114 for our Semester 2 solution. However, this does not consider how the penalty is spread across faculties.

Because some faculties teach more courses than others, an equitable distribution of the total penalty relates to a faculty’s “average penalty per course”. Approximately, this can be considered as the percentage of courses from a faculty which experience a disruption to course room stability. Based on this objective measure for each faculty, we solve a max-ordering optimisation problem, followed by a utilitarian optimisation problem. This maximises the quality for the worst-off faculty, and then
sets this as the minimum for all parties while maximising the total quality. This is easier to solve computationally than finding a max-min fairness solution, while still ensuring a degree of relative fairness. Specifically, in a solution generated using this method, each faculty is either the worst-off itself (but at its highest possible quality), or has quality at least as high as the worst-off faculty. 

<table>
<thead>
<tr>
<th></th>
<th>Semester 1</th>
<th></th>
<th>Semester 2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fair First</td>
<td>Utilitarian</td>
<td>Fair First</td>
<td>Utilitarian</td>
</tr>
<tr>
<td>Max Stability Disruption (%)</td>
<td>4.15</td>
<td>(4.32)</td>
<td>5.92</td>
<td>(9.09)</td>
</tr>
<tr>
<td>Total Room Stability</td>
<td>-49</td>
<td>-49</td>
<td>-119</td>
<td>-114</td>
</tr>
</tbody>
</table>

Table 7.2: UoA 2013 Course Room Stability Fairness

Table 7.2 shows the results using the “Fair First” process as described, compared to the pure “Utilitarian” method. For each method, the first row shows the percentage of courses which experience a disruption for the worst-off faculty. The second row shows the total course room stability disruption summed over courses from all faculties. In the utilitarian case, the percentage of courses is shown in brackets since it is not explicitly measured or optimised. All models are solved after the room preference quality measure has been optimised, as part of the usual lexicographic process from Chapter 5.

For the Semester 1 case, minimising the disruption for the worst-off faculty first had a small positive effect, lowering to 4.15% from 4.32%. This is achieved without any cost to the total room stability. For the Semester 2 case, a substantial improvement is observed for the worst-off faculty. However, a small cost to the total stability is incurred. In practice, the timetablers could consider whether the size of the improvement of the worst-off utility is sufficient to justify a particular decrease in total quality, i.e. to consider the “price of fairness” (Bertsimas et al., 2011).

Finally, we note why the course room stability quality measure was chosen for this multi-objective analysis. In the case of course room stability, different feasible
room assignments exist which cause each faculty to potentially have a low or high disruption. However, without adjusting the data, this is not necessarily true for the room preference quality measure. Due to the fixed physical location of each faculty relative to the available rooms, a faculty may necessarily have the lowest room preference in any feasible solution.

7.4.3 Weighted Sum Scalarisation

A more general approach than lexicographic optimisation for handling multiple objective functions is a weighted sum scalarisation. This defines the relative importance of each objective, rather than applying a strict dominance. By solving multiple problems with a different choice of weightings, we are able to explore the tradeoffs between improving each objective at the expense of the others. Almost universally in the timetabling literature, a single set of weightings is chosen for several quality measures, and this weighted sum expression is used as the objective function.

In this section, we use a weighted sum scalarisation search to explore the quality tradeoff between two objectives (i.e. a bi-objective problem). The process used to find a set of efficient (Pareto optimal) solutions is given as Algorithm 2. Note that this method will find extreme supported efficient solutions, but it will not necessarily find all efficient solutions.

Initially, lexicographic optimisation is used for each ordering of objectives, which will find the two nondominated “endpoints”. These points form an interval in objective-space, which we use to search for additional nondominated points (and corresponding efficient solutions). Objective weightings (denoted by $\lambda$) are chosen so that any extreme supported nondominated point in this interval will be found in preference to either of the endpoints. When a new point is found, we add the two new search intervals to the back of the queue of intervals. Searching each interval in order of the queue ensures that the we find a good “spread” of points, if there
is not enough time (or need) to search exhaustively. In this section, we terminate Algorithm 2 with a maximum of 20 solutions.

**Algorithm 2 Bi-objective Weighted Sum Search Algorithm**

1. `solutionSet ← set()`
2. `intervalQueue ← queue()
3. `maxSolns ← 20`
4. `IP ← BuildRoomAssignmentIP()`
5. `leftSoln ← IP.LexicographicSolve(obj: z_1, z_2)`
6. `rightSoln ← IP.LexicographicSolve(obj: z_2, z_1)`
7. `intervalQueue.push([leftSoln, rightSoln])`
8. **while** `intervalQueue.length() > 0 and solutionSet.length() < maxSolns **do**`
9. `[l, r] ← intervalQueue.pop()`
10. `λ ← ComputeIntervalLambda([l, r])`
11. `newSoln ← IP.Solve(obj: λz_1 + (1 − λ)z_2)`
12. `solutionSet.add(newSoln)`
13. **if** `ComputeIntervalLambda([l, newSoln]) ≠ λ` **then**
14. `intervalQueue.push([l, newSoln])`
15. `intervalQueue.push([newSoln, r])`
16. **end if**
17. **end while**

To choose the pairs of quality measures to apply this algorithm to, we note that the most useful results are generated when two partially conflicting quality measures are considered. We must also be willing to compromise the quality of one for a gain in the quality of the other i.e. there is no strict dominance. For this reason, we do not consider the room preference versus the course room stability.

An interesting comparison to make is the room preference versus the spare seat robustness at the University of Auckland. We conduct this comparison in the absence of the course room stability quality measure, which (in practice) should be optimised after a solution from this frontier set is chosen. These problems are near-naturally integer, and solutions can be found quickly, because both quality measures are event-based. Figure 7.2 shows the sets of nondominated points for each semester.
of 2013.

Another comparison is made between the course room stability and the spare seat robustness. In this case, we have already optimised and fixed the room preference, which is required for its lexicographic precedence to the spare seat robustness. These problems are much more difficult because course room stability is pattern-based, and took several hours to complete in the Semester 2 case. Figure 7.3 shows the sets of nondominated points for each semester of 2013.

The generation of Pareto-optimal frontiers is useful to understand the tradeoffs within timetabling. When timetablers interact with faculty and administrative staff, this visual representation can be simply understood without technical knowledge. Furthermore, if dealing with faculties where the exact priorities are unclear (i.e. the preferred weighting between two objectives is not known), providing multiple solutions with different emphases may help to elicit true quality weightings and improve satisfaction with the solution.

Modelling these tradeoffs can also be a “staging area” for what to include in the core algorithm. Multi-objective optimisation allows us to understand the scope of how much a quality measure can be improved, and how this might detract from other measures of quality. Alternatively, we may observe that one quality measure is highly compatible with another measure and does not need to be separately represented. In the case of strongly conflicting quality measures, we are able to identify the most important priority decisions which need to be made.

In the scope of multi-objective optimisation methods, a weighted sum scalarisation search is considered to be relatively simple. More advanced methods are capable of finding a greater number of solutions, such as unsupported efficient solutions. For example, the epsilon constraint method optimises one quality measure while applying a minimum quality on the other. By repeatedly moving the minimum quality across the range of possible quality values, unsupported efficient solutions can also
Figure 7.2: Supported Nondominated Points (SR vs RP) UoA 2013
Figure 7.3: Supported Nondominated Points (SR vs RS) UoA 2013
be found. However, the IP models used in this method are often very difficult to solve (Ehrgott, 2005). A more advanced method is the elastic constraint method which hybridises the weighted sum scalarisation search and the epsilon constraint method which can be more efficient (Ehrgott and Ryan, 2002). For the purposes of this example, we note that we have been able to generate an acceptable number of supported nondominated points for all four problems, with no substantial “gaps” between the quality of two solutions. As a result, more advanced methods have not been required.
8.1 Key Results

This thesis has presented three mathematical programming-based algorithms for practical university course timetabling. These algorithms address the time assignment problem, classroom assignment problem, and minimal perturbation problem, and each provides novel results and insights. However, the significant overarching conclusion of this thesis is that mathematical programming is a viable technique for use on real-world problems. Our method is able to construct a high quality solution within a suitable timeframe for use in practice. In addition to empirically demonstrating our method at the University of Auckland, we have developed each algorithm in a generalised context which can be applied to a wide range of problems. These include not only other university course timetabling problems, but more broadly within educational timetabling.

For the time assignment problem, we have shown that integer programming can be used to quickly find an optimal solution, with respect to the time preference of each event. However, we also demonstrate the complexity of defining and optimising
more complex quality measures, such as curriculum compactness. For this reason, we conclude that faculty involvement in generating a time assignment is highly desirable.

We also have explored the difficulty of maintaining feasibility in the room assignment. For a particular set of rooms, the number of constraints required (for guaranteed feasibility) scales exponentially in the number of room attributes. Although this is intractable for practical problems (with complex arbitrary room attributes), we provide numerical results showing the high degree of feasibility which can be achieved by selecting a small subset of these constraints.

The classroom assignment problem is particularly important, as a ubiquitous component of practical timetabling. For this problem, we have contributed theoretical insights on the features of practical problems which affect the difficulty to solve. Specifically, when quality is defined using an event-based measure (e.g. room preference), an optimal solution can typically be found quickly, despite NP-hardness.

However, when quality is defined using a pattern-based measure such as course room stability, the problem can be much more difficult. The patterns of events connect multiple time periods together, which create many opportunities for fractions to occur in the LP relaxation. This makes the IP more difficult to solve. We empirically demonstrate this consequence of high inter-period connectivity, and high room utilisation. Using this insight, we are able to a priori identify the difficult instances.

To address multiple quality measures, we demonstrate the use of a lexicographic optimisation algorithm, which enforces a strict priority ordering. We include comprehensive experimental results for this method. The main tests for this method are conducted on our motivating example problems at the University of Auckland. However, we are also able to adapt the room assignment model to solve the timetabling problem at the Technical University of Denmark. This result validates the versatility of our method for diverse practical problems.
Lastly for the room assignment results, we solve the Udine benchmark instances which provide an additional academic validation of our method. Our results show that we are able to quickly improve on solutions generated by existing state-of-the-art metaheuristics. This is an important result which suggests there is significant potential for an integration of methodologies, known as “matheuristics”.

For the minimal perturbation problem, we propose the first mathematical programming-based method in the literature. The minimal perturbation problem is essentially a search for feasibility within a relatively small set of possibilities, and is an excellent application for mathematical programming. The expanding neighbourhood algorithm is shown to be efficient and versatile.

Within practical course timetabling, we have explored the situations where the minimal perturbation problem can arise. Initially in the construction phase of timetabling, our algorithm can be used to find the closest feasible timetable to one which is desirable but infeasible. This can occur when a time assignment has been based on faculty requests without consideration for the room assignment.

In the enrolment phase of timetabling, our algorithm can be used to repair unexpected changes to an existing timetable. This is relevant to every practical timetabling system, and has been previously noted as underaddressed in timetabling literature.

We also experiment with different types of disruption caused when a timetable is perturbed. Using a re-defined neighbourhood (and set of expansion rules), the expanding neighbourhood algorithm can be used for complex definitions of disruption relating to course or curriculum structure, rather than individual events.

The final general result of this research relates to the analytics capabilities of our algorithms. In the simplest case, we have discussed the potential for conducting analytics using our algorithms with modified data, to model different scenarios. This has significant application in mid-term and long-term planning, where a uni-
versity can evaluate the impact of changes, such as expanding the facilities. This capability has been mentioned previously in the literature, however, it has not been demonstrated or extended. As an example, we also show how to adapt the room assignment model to minimise the total cost of rooms.

We have also explored the analytics relating to quality tradeoffs inherent to multi-objective optimisation. This includes the equity consideration of ensuring timetabling quality across different faculties. We provide results which show how an improved level of fairness can be achieved, at a small cost to the overall quality. Finally, we depict the tradeoff involved when considering multiple quality measures in a timetable. Using multi-objective methods, we can generate a Pareto-optimal frontier of solutions, which has not been previously conducted in the literature. This capability has substantial practical value for many university stakeholders who can visualise the timetabling tradeoffs, and be involved in the choice between the many possible feasible timetables.

8.2 Future Work

For each of the technical chapters of this thesis we have concluded with a discussion of results and proposed future work for the relevant algorithm. Although the algorithms are able to generate a high quality timetable within an acceptable timeframe, there are clearly many opportunities for improvement. In this section we provide a general overview for how timetabling algorithms can be made more efficient, and we consider which other auxiliary problems within this field can be addressed. Due to the size of modern universities, and the complexity of measuring quality, it seems likely that timetabling research will be an active field for many years into the future.

To improve the performance of exact methods, there are many advanced techniques which have not yet been tested. For example, decomposition methods such
as branch-and-price remain to be comprehensively explored. Some promising work has been conducted on effective cutting plane generation for timetabling problems, however this requires greater research. There are also opportunities to customise the individual components of the IP solver, such as the branching decisions and embedded heuristics. It is well-known that modern IP solvers utilise many heuristics within the solve process.

The metaheuristic community is similarly advancing the performance of timetabling algorithms, primarily through greater customisation to the problem domain. A particularly interesting direction is that of hyper-heuristics, where a heuristic is used to determine which algorithm to employ. This methodology provides a promising opportunity for the incorporation of exact methods. Our research demonstrates that many problems within timetabling can be solved optimally within a short timeframe. Hyper-heuristics can also be employed where many algorithms address the same problem, finding and improving a solution iteratively. The results in this thesis on the Udine benchmark instances provide validation of this concept. Using the room assignment algorithm, we are able to substantially improve existing heuristically-generated solutions.

Finally, future timetabling research is likely to address the many problems which arise in practical timetabling that are auxiliary to the core timetabling process. These include problems examined in this thesis such as the minimal perturbation problem, and the provision of analytics to timetablers. Room planning problems in particular show substantial potential to advance the field. The capability to understand which sizes and types of rooms are needed has demonstrable financial value to university administrators.
8.3 Discussion

In this final section, we provide general thoughts on the potential for benchmarking in practical timetabling, and discuss ideas for improving the implementation of a timetabling system.

The objective of creating a practical timetabling system has been achieved in this thesis. However, we have an academic interest in how the system compares to metaheuristic approaches on the same dataset. The most suitable comparison would be against the timetabling system of Rudová et al. (2011), which is based on the iterative forward search metaheuristic (Müller, 2005). However, in order to conduct such an experiment fairly, significant work would be required to implement the heuristic correctly and adapt it to suit the given problem. There are also multiple ways to measure quality which may be equally valid in a practical context. A pre-defined set of such modelling choices is likely to favour one methodology. Therefore, we conclude that conducting a fair comparison of two timetabling systems is a complex task.

The Udine benchmark instances have served a strong purpose in advancing timetabling algorithms through comparability. In this thesis we have used the Udine benchmark instances where possible. However, in Section 3.3.2 we have also made the case that these problems are not necessarily representative of practical timetabling. Notably, most literature addresses either the Udine benchmark instances, or the authors’ own specific practical problem. A binary decision between practically-oriented research and comparable research is clearly not ideal.

To attempt “middle-ground”, we propose that a new set of instances could be generated which allow much greater flexibility. Specifically, a large set of quality measures could be presented, with comprehensive data provided to allow for quantification. A practical timetabling algorithm could then be tested on any subset of
the quality measures which are applicable for the intended practical problem.

For example, the avoidance of curricula conflicts in the time assignment is near-universal. However, while sometimes viewed as a hard constraint, conflict avoidance is often a quality measure in practical timetabling. For curricula which only relate to a small number of inter-faculty students, a conflict may be undesirable, yet acceptable.

The precise data required for these instances would not need to be based on data from a particular university, but should be of a realistic structure. Due to the diversity in real-world timetabling problems, many instances would be required to include problems with different sizes and features.

A flexible set of benchmark instances would also facilitate the future research we propose relating to understanding tradeoffs between quality measures. It would additionally be a new insight to understand how the different representations of abstract quality measures (such as curriculum compactness) affect the computational difficulty of timetabling problems.

The notable drawback of such a set of instances, is that algorithms which address different sets of quality measures cannot be directly compared. However, if many timetabling algorithms are presently only tested on private data instances, the comparability is arguably improved. Furthermore, this benchmarking concept does not preclude the issuance of a precise specification of quality measures. Such a specification is logically required if direct comparisons must be made, such as for a timetabling competition.

Finally, we discuss the current paradigm of centralised timetabling, and how algorithms may enable improved timetabling processes in the future. The current timetabling process seeks to receive all data from faculties “up-front”, for each quality measure. Once the problem is solved, the faculties are presented with a draft timetable upon which they may request manual changes. The manual adjustments
are conducted in an iterative process, which is intended to take as little time as possible. There are several drawbacks to this system. Firstly, it may be too demanding on faculties to provide the requested detail of data, particularly if there are many quality measures. Secondly, faculties are required to conform to a pre-decided set of quality measures, and are not able to express priorities between quality measures. Thirdly, faculties may find it difficult to numerically express the relative quality tradeoffs, without the context of a practical example. This method ultimately excludes faculties from the decision making process. In the literature, practical accounts of implementing a timetabling system cite many of these difficulties.

As an alternative paradigm, there may be potential in an approach which is designed around iterative faculty input. Initially, we only require the faculty to give the simplest preference data to model quality, and only the constraints which are known to be truly “hard” are considered. Then we solve this simplified problem, which can be performed quickly using our timetabling algorithms. Faculties are then invited to examine the draft timetable for their courses, and are able to communicate their requested changes and associated priorities. This will include new hard constraints as well as a range of requested quality improvements ranging from most to least important. With this new data for all faculties, we can re-solve the timetabling problem, which will ideally honour most requests. This process may iterate several times until eventually all significant issues will be removed, and we are left with a small number of unsatisfied preferences. At this stage, it will be clear to timetablers and faculties that honouring the remaining preferences (such as the avoidance of teaching at 8am), is not possible without re-introducing more serious preference violations (such as a moderately important curriculum conflict). Although there are drawbacks to any method, this alternative addresses many issues with the existing paradigm. Achieving a high degree of quality in practice requires accurate data from faculties. This approach allows each faculty to decide on their
most important quality considerations (which differ between faculties), and engages
them directly in defining the quality tradeoffs.

To conclude, this thesis makes a significant contribution to developing and val-
validating practical algorithms which make use of a new methodology. However, con-
ducting a full-scale implementation is clearly the most important future work for our
timetabling system. Using the capabilities of our algorithms we can be optimistic
about overcoming the data and implementation challenges, and further developing
this field.


