Noname manuscript No.
(will be inserted by the editor)

A Multi-Stage IP-Based Heuristic for Class Timetabling and Trainer Rostering

Oliver Czibula · Hanyu Gu · Aaron Russell · Yakov Zinder

Received: date / Accepted: date

Abstract We consider a timetabling and rostering problem involving periodic retraining of large numbers of employees at an Australian electricity distributor. This problem is different from traditional high school and university timetabling problems studied in the literature in several aspects. We propose a three-stage heuristic consisting of timetable generation, timetable improvement, and trainer rostering. Large-scale integer linear programming (ILP) models for both the timetabling and the rostering components are proposed, and several unique operational constraints are discussed. We show that this solution approach is able to produce good solutions in practically acceptable time.

Keywords Timetabling · Rostering · Integer Programming · Heuristic

O. Czibula
School of Mathematical Sciences
University of Technology Sydney
Tel.: +61-2-95142281
Fax: +61-2-95142260
E-mail: oliver.g.czibula@student.uts.edu.au

H. Gu
School of Mathematical Sciences
University of Technology Sydney
Tel.: +61-2-95142281
Fax: +61-2-95142260
E-mail: hanyu.gu@uts.edu.au

A. Russell
48-50 Holker Street
Silverwater, NSW 2128
Tel.: +61-2-87451569
Fax: +61-2-96486859
E-mail: ajrussell@ausgrid.com.au

Y. Zinder
School of Mathematical Sciences
University of Technology Sydney
Tel.: +61-2-95142279
Fax: +61-2-95142260
E-mail: yakov.zinder@uts.edu.au

1 Introduction

Ausgrid, Australia's largest electricity distributor, is responsible for building, repairing, and maintaining all electricity distribution equipment that service their operational area of about 22,275km^2. The voltages on Ausgrid's electricity network range from 230V to 132kV, and there is an extreme risk of electrocution if works are not performed carefully and with strict safeguards in place. Other hazards Ausgrid workers face include falling from heights, having objects dropped on them from above, working in confined spaces, and working in the presence of hazardous materials such as toxic gas, asbestos, or other harmful substances. Having such a hazardous working environment and supplying such an essential service to the population, it is among Ausgrid's highest priorities to deliver safety and technical training promptly and efficiently to all people where required, including employees, contractors, and to third parties, working on or near the electricity network as required by Australian industry law.

Most training delivered by Ausgrid has a limited validity period after which it is considered no longer valid unless "refreshed". Most courses have a validity period of 1 year, while others can last 3 or 5 years, and some have unlimited validity periods. Validity periods are subject to change as industry legislation is occasionally revised. If a worker does not successfully refresh training for a job role before it expires, they are not permitted to work on or near the electricity network in that role until they have successfully completed the required training.

Ausgrid delivers many different training courses, each of which is composed of one or more modules. Each student enrolled in a course must successfully complete all the modules in that course. Each module has a duration and a maximum number of students that it can accommodate. The modules of a course can be run in any order, however they must be run back-to-back with no gaps between the modules of a course. The only exception is lunch time, which is fixed to 12:00 to 12:30. The modules of courses that run for longer than a day must be arranged such that the modules do not overrun the length of any day. Courses run for a maximum of five days, and may not be interrupted by the weekend. If a course has a total duration of half a day or less, it may start first thing in the morning or immediately after lunch, otherwise if a course's duration is longer than half a day, it may only start first thing in the morning. Each course may be run an arbitrary number of times, perhaps several times in a single day, and each individual run is known as a course instance.

Ausgrid's operational area can be partitioned into a number of regions, each containing one or more training facilities, which we refer to as locations. Each location contains one or more rooms, which come in various sizes. Some rooms have a removable divider that allows them to be split into two separate, smaller rooms. Each module of each course instance is run in a room. Each room has a list of compatible modules and a maximum number of students it can accommodate. Since both modules and rooms have a limitation on the number of students, each module-room pair has a maximum number of students given by the minimum of these two values. Different modules of a course can be assigned to different rooms, but all the modules of a course instance must be assigned to a single location.

Modules are taught by trainers, each of whom has a location to which they are regularly assigned, and trainers may travel to other locations when required. All trainers work a standard 8 hour work day with a half hour lunch break. Trainers have other responsibilities aside from teaching, therefore the proportion of working time each trainer is assigned to teaching should be as close as possible to a value determined by the type of trainer.

Certain modules require shared, mobile resources which can be relocated from location to location. The total number of these modules that can run at any given time in any given location is limited by the quantity of the required equipment present.

Because training is delivered not only to Ausgrid employees, but also contractors and third parties, Ausgrid cannot schedule individual participants into classes in advance, as is the case in most high school and university timetabling. Instead, Ausgrid schedules classes to run at times and places where people are expected to need training, and those people are then booked into a suitable class at a time they are able to attend. Since most of the courses have a regular validity period, Ausgrid can anticipate how many people will require certain types of training in particular locations at particular times. Ausgrid should schedule at least enough capacity for each course in each region across the planning horizon to cater for the expected demand, wherever possible.

The robustness of the training plan is of great importance to Ausgrid. A robust timetable will not need to be significantly changed in the event of unexpected circumstances such as room or trainer unavailability. One feature of a timetable which detracts from its robustness is a room swap, which is where the class must move from one room to another throughout the course instance. In addition to being inconvenient and time consuming, if one of the rooms assigned to a course instance become unexpectedly unavailable and no substitute can be found, the course instance may need to be cancelled. Courses should be uniformly distributed throughout the planning horizon to maximise the likelihood that people will be able to find a class at a suitable time.

It is desirable for trainers not to have to travel excessively, as the further a trainer has to travel, the more cost is incurred by Ausgrid, and the greater the likelihood the trainer will be unexpectedly delayed. Additionally, for the same reasons outlined previously in regards to room swaps, having trainer swaps detracts from the robustness of the trainer roster as any unexpectedly unavailable trainer may jeopardise a course if no substitute can be found.

Certain rooms can be rented out to third parties if not in use for an extended period of time, generating some revenue for Ausgrid. Currently, due to administration costs, only a few rooms have the option to be rented out to third parties, however if the potential revenue generated can be shown to be substantial, Ausgrid may expand their room rental program.

Until recently, Ausgrid designed training timetables by hand, requiring an experienced person at least three days to produce the timetable for one month. Ausgrid currently uses a software tool to generate their training plans on a month-by-month basis. This software is able to rapidly generate feasible timetables and rosters, however it does not contain any optimisation functionality. Due to changing industry regulations related to safety and technical training, as well as long-term fluctuations of demand, Ausgrid needs a tool to manage and optimise their training plan that is capable of handling these changes in a flexible way.

Ausgrid's timetabling and rostering problem is a large-scale, multi-objective optimisation problem. It has many similarities to typical high school and university timetabling problems, but is distinguished from them in a number of important ways. The goal of the research in this paper is to investigate a flexible software tool incorporating mathematical optimisation techniques to help solve Ausgrid's timetabling and rostering problem. Due to the large-scale nature of the considered problem, we have so far been unable to produce an optimal solution in practically acceptable time. In this paper we propose a three-stage heuristic procedure consisting of an initial timetable generation stage, an iterative timetable improvement stage, and finally a trainer rostering stage. Integer programming (IP) models are developed for each stage, which address all the current practical requirements, and are also flexible to changes in requirements.

The remainder of the paper is organised as follows: Section 2 gives an outline of the current state of research in the area of academic timetabling. Section 3 describes the three-stage heuristic in detail. Section 4 describes the class timetabling ILP model, and Section 5 describes the trainer rostering ILP model. Section 6 discusses some important details about the implementation of the approach. Section 7 describes our computational experimentation and results. Finally, our conclusions are given in Section 8.

2 Literature Review

The literature review in this section gives a brief overview of the types of problems that have been solved in the field of academic timetabling. Comparisons with Ausgrid's problem are made, and highlights of important similarities and differences are presented.

Problems in the area of academic timetabling have attracted a great deal of research attention in recent decades [26]. In many cases the problem has been shown to be NP-hard [8], often by relating it to the graph colouring problem [18]. For the classroom assignment problem (CAP)—the problem of assigning n classes to a set of m classrooms, in such a way that each class is run exactly once and each room can be used at most once at any given time—Carter [8] proposes that there are three possible objectives: Feasibility, Satisfiability, and Optimisation. Feasibility asks whether there is any feasible solution given the constraints mentioned before, Satisfiability asks whether there is a feasible solution that puts each class into a satisfactory room, and Optimisation is the objective of minimising some linear cost function. Carter showed that the interval CAP satisfiability for even as

little as two time periods, as well as the feasibility of the non-interval CAP, are NP-complete.

Researchers in large-scale timetabling generally do not attempt to find optimal solutions to problems as they cannot be found in practically acceptable time. Instead, much of the recent research has been focused on approximation algorithms including metaheuristics [20][6], and decomposition methods such as Lagrangian relaxation [13] and column generation [27][25]. Some timetabling problems can be expressed as graph colouring problems [21], and there has been some research activity in using graph colouring heuristics to solve timetabling problems [24][22][29][5].

A recent trend has been to develop so-called "hybrid heuristics" that combine certain features of one heuristic with another, with the aim of improving performance by overcoming the heuristics' weaknesses. In [14], attempts were made to improve the convergence rate of Simulated Annealing (SA) by implementing the memory property of Tabu Search (TS) to solve a university course timetabling problem. The annealing rate in SA can have a dramatic influence over the performance of an SA implementation [31]. It is not uncommon for people to implement complex reheating rules to help the heuristic avoid being trapped in local minima prematurely. A novel hybrid heuristic was presented in [4], where the authors propose a Genetic Program (GP) to optimise the annealing schedule in SA. They presented the dedicated cooling schedules found by GP that converge fastest for particular problems, and they also provided the cooling schedule that converges fastest across all problems they tested.

Despite the difficulty in solving large-scale IP-based models, several researchers have had some success. Perhaps the earliest examples are [19] from 1969 and [1] from 1973. A more recent example of IP-based university course and examination timetabling is presented in [10], where the authors augmented an IP with a heuristic improvement stage. A high school timetabling problem is presented in [3], where the authors were able to solve the IP directly. An ILP was presented in [28], which had, amongst others, 99 teachers, 156 courses, and 181 teaching groups. The model had 35,611 rows, 91,767 integer variables, and 662,824 non-zeroes in the co-efficient matrix. An optimal solution was obtained in just 10 seconds using IBM ILOG CPLEX 9.1.2, or about 2 minutes with Coin-OR Branch-and-Cut.

The vast literature produced on university and high school timetabling does not address the problem considered in our paper. More specifically:

- The timetabling problem considered in this paper has different objectives to that of traditional university and high school class timetabling problems. Indeed, in our problem the focus is on satisfying demand, which fluctuates over the planning horizon, whereas in all publications of university and high school timetabling problems there is no fluctuation in demand, and typically the main focus is on the minimisation of teacher and student preference violations.
- Another objective of our problem is to strengthen the robustness of the timetable over long periods of time by levelling the utilisation of resources, whereas in university and high school timetabling problems, the goal is to construct a timetable for one or two weeks that repeats throughout the semester or year.

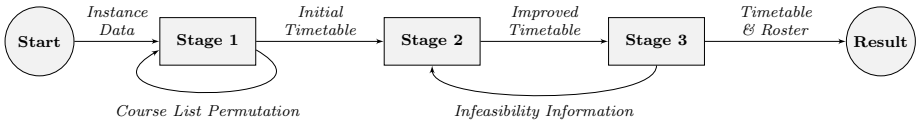

Fig. 1 A high-level view of the three-stage approach.

- The timetabling problem considered in our paper is intended for a commercial organisation, where the ability to rent out available rooms is important. The consideration of such opportunity for renting out rooms is not found in university and high school timetabling.
- University and high school timetabling problems require the allocation of a given set of classes to locations and times, whereas in our problem, the number of classes and their sizes is part of the decisions which are to be made.
- In our problem the timetable should satisfy rigorous restrictions on how courses, which are composed of several modules, are scheduled. This is in contrast to university timetabling, where there are no comparable restrictions on the time and location for the assignment of different activities, such as lectures, computer laboratories, tutorials, etc.
- The problem considered in this paper requires decisions on the movement of scarce mobile resources between different locations, which is very unusual in university and high school timetabling.

The above-mentioned differences make the mathematical models commonly used in university and high school timetabling unapplicable to our problem, which causes the necessity to develop new mathematical models together with new optimisation procedures that address the specific features and objectives of the problem on hand.

Being new and interesting from a mathematical point of view, the considered problem is also important from a practical perspective. Indeed, training and retraining of employees is an important and costly component of the business for many organisations. For example, according to a 2013 report from the Minerals Council of Australia, the vast majority of mining training in 2012 was conducted by the Australian mining industry, with over $1.1 billion spent. The report states that the training expenditure is almost 5.5% of total payroll, and that almost 98% of training expenditure is industry-funded with only 2% coming from government subsidies. [23]

The areas where research on timetabling for training and retraining can be of significant benefit also include other utility companies, forestry companies, oil and gas companies, construction companies, etc., to mention a few.

3 Optimisation Procedure

In order to improve tractability and simplify the modelling of Ausgrid's problem, we have divided it into two related sub-problems: a class timetabling problem

and a trainer rostering problem. The former is the problem of allocating modules to rooms forming classes, where the objectives are that the unsatisfied student demand is minimised, room swaps are minimised, and the potential to rent out rooms is maximised. The latter is the problem of, subject to an existing timetable of classes, allocating trainers to modules where the objective is to minimise trainer travel and minimise trainer swaps. Aggregated trainer numbers, as opposed to individual trainers, are included for capacity purposes only, in order to improve the likelihood that the timetable will allow for a feasible trainer roster. The combined solution to both sub-problems results in a complete timetable and roster.

To have a flexible tool that is able to solve these complex sub-problems, we have developed two IP models: one for the class timetabling sub-problem and one for the trainer rostering sub-problem. IP is flexible in the sense that one can simply add, remove, or substitute constraints to modify the model in various ways. Decomposing a larger, integrated problem into multiple sub-problems, such as Ausgrid's training problem into the timetabling and rostering sub-problems, often results in substantially faster optimisation, although usually at the cost of solution quality. In our case, even after decomposing the problem into two sub-problems, the timetabling IP model was still too large to be solved in practically acceptable time given real world data for organisations such as Ausgrid.

In order to produce a complete, usable timetable and roster for Ausgrid in practically acceptable time, we propose a three stage heuristic approach (see Figure 1). The first two stages tackle the class timetabling sub-problem, while the third stage tackles the trainer rostering sub-problem. The first stage produces an initial feasible class timetable, the second stage attempts to improve the class timetable, and the third stage allocates individual trainers, subject to the timetable produced in the second stage, to form a roster.

3.1 Stage 1: Initial timetable construction

The considered class timetabling problem is NP-hard [30][8]. We choose to construct the initial class timetable using the class timetabling IP model rather than using metaheuristics, as each generated solution is guaranteed to be optimal with respect to the model being solved, even though the constructed initial timetable as a whole is unlikely to be optimal.

The first stage constructs an initial feasible timetable incrementally. This is accomplished by successively solving the class timetabling IP model for one course instance at a time. Each time a course instance is scheduled, the rooms and other resources it occupies become unavailable for subsequent course instances for its duration. As more course instances are present in the partial schedule, more rooms become unavailable at certain times, and more resources and trainers are consumed in particular locations at particular times. In each subsequent iteration, variables corresponding to the occupied rooms, resources, and trainers in the partial solution are fixed when creating the IP model. As less rooms, resources, and trainers become available in each subsequent iteration, the resulting models have, in general, fewer decision variables.

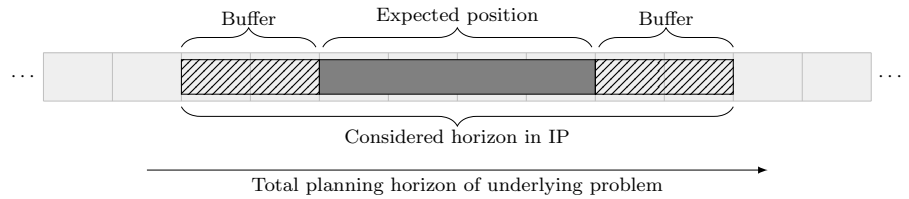

Fig. 2 A buffer added to either end of the reduced planning horizon.

The order of the list that determines the sequence in which course instances are to be scheduled can be arbitrary. Scheduling courses one by one using different sequences likely results in different timetables with different objective values. It is reasonable to assume that if the most demanding courses—those which have longer durations, with many modules, and very specific room and resource requirements—are scheduled first, and the least demanding courses are scheduled last, then a better quality initial timetable can be constructed. Based on similar ideas in [7] and [17], a scheduling complexity value should be estimated for each course, and course instances should be scheduled in decreasing order of these values. If the courses are scheduled in descending order of their estimated scheduling complexity, we can reasonably expect there to be a greater likelihood that an initial timetable can be constructed in the first stage that minimises unsatisfied demand than if the course instances were scheduled in a very different order. The course scheduling complexity value can be estimated by considering:

- The total duration of the course;
- The number of unique mobile resources required by the modules of the course, and the duration for which they are needed; and
- The room specificity of the modules in the course, i.e. how few rooms the modules are compatible with, and how many unique, specific rooms are required.

Several initial timetables can be constructed by considering the above-mentioned order with minor perturbations, such as swapping two course instances in the list. The timetable with the best objective value can be used in subsequent stages.

To further reduce the size of the IP model, we can add additional restrictions based on the course instance being solved. We can determine when each course instance should run in order for them to be uniformly distributed along the timeline. Then, when allocating a single course instance, we can consider a much shorter planning horizon centred around this time instead of considering the whole planning horizon. Since we cannot guarantee that the course can be scheduled at exactly the times we want, the restricted planning horizon may need to be extended to allow some freedom in scheduling (see Figure 2). The shorter the considered planning window, the more control we have over the position of the course and the smaller the IP model will be, however a feasible solution may not exist. The buffer in the planning horizon can start out small, and be incrementally increased if no solution can be found.

When scheduling course instances one by one, it is also possible to consider only a single region in the IP model. Looking at the demand of a course, together with

the state of the current partial solution, we decide in which region the next course instance should be placed, and include only that region in the IP model. Moreover, any room incompatible with all of the course's modules should also be excluded from the IP model.

The final IP for each iteration of the first stage becomes substantially smaller and can be typically solved in a few seconds. Given data from Ausgrid, it contains one course instance with 1 to 5 modules; a planning horizon of between 24 and 144 time periods; and one region with a 1 to 4 locations, each with 1 to 15 rooms.

3.2 Stage 2: Timetable improvement

The first stage constructs an initial feasible timetable that it is unlikely to be optimal, since poor decisions made at the early stages of the process can have a compounding effect on the remaining allocations. The second stage attempts to improve the timetable using a Large Neighbourhood Search (LNS) heuristic. As with the first stage, the second stage makes use of the timetabling IP model to produce new solutions.

In each iteration, the components of the incumbent solution whose change may improve the objective value are identified. For example, suppose one course instance affects the objective value because it contains some room swaps. Perhaps these room swaps were necessary at the time the initial timetable was constructed as there was no other way to fit it in, but can now be eliminated. To attempt to improve the timetable in this regard, the identified course instance, together with at least one other course instance, should be removed from the timetable and the class timetabling IP model should be re-solved for the removed course instances, subject to the remaining timetable. The more course instances are considered simultaneously, the better the outcome is likely to be, however the computational effort required grows rapidly in the number of selected instances.

See Figure 3 for an illustration of a poorly constructed timetable, where three courses are scheduled one by one, in order. Due to the limited compatibility between modules and rooms, and poor decisions early in the process, the modules of Course 3 have been assigned to three separate rooms. During stage 2, the instance for Course 3 is identified as impacting the objective value due to the room swaps. Solving the timetabling IP model for only Course 3 alone cannot produce a better solution as it is already in the optimal position subject to Courses 1 and 2. If the timetabling IP model is solved for Courses 1, 2, and 3 simultaneously, then the timetable shown in Figure 4 can be produced, eliminating the room swap cost of Course 3 entirely.

In each iteration, the only variables present in the IP are those pertaining to the selected course instances; all other variables are fixed to a constant value determined by the state of the remainder of the timetable. The "large neighbourhood" in the LNS heuristic is the set of all feasible solutions to the IP model. Stage 2 continues until any stopping criterion has been reached, such as a time limit or failure to improve the timetable after a certain number of iterations.

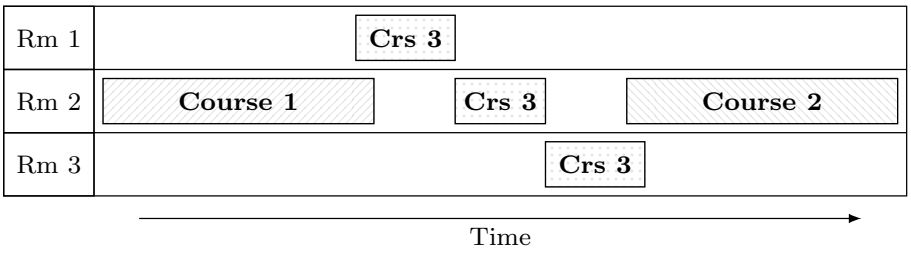

Fig. 3 An example of a timetable with unnecessary room swaps.

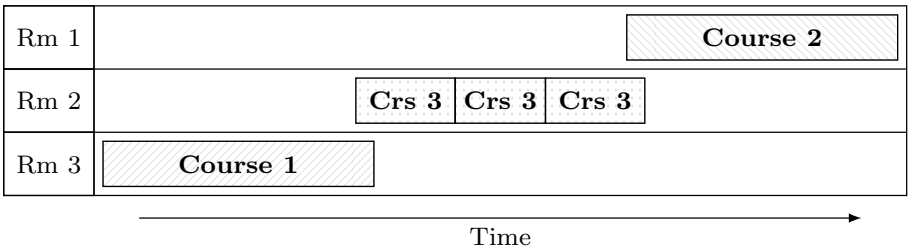

Fig. 4 Unnecessary room swaps in Figure 3 avoided.

Stage 2 attempts to improve situations where there are:

1: courses where not all demand has been satisfied,
2: course instances with many room swaps, and
3: course instances occupying rooms that could otherwise be rented out.

We identify several course instances to be rescheduled, which are relevant to one another with respect to opportunity to make improvements. In the case of 1, the instances of a selected course whose demand has not been satisfied in a given region and time period should be paired with other course instances that have an overlapping set of compatible rooms with the selected instances. If necessary, additional instances can be created for the courses with insufficient capacity. In the case of 2, a course instance which contains room swaps should be paired with one or more other course instances in the same region and time period, and that have an overlapping set of compatible rooms with the selected instance. In the case of 3, a set of selected course instances that occupy rooms that could otherwise be rented out should be paired with one or more other course instances in the same region and time period, and that have an overlapping set of compatible rooms with the selected instances, and that are not currently occupying the rentable rooms. It is important to balance the need to make improvements by selecting as many course instances as possible, with the need to keep size of the IP model manageable by selecting as few courses as possible. The procedure in Stage 2 is heuristic, and therefore cannot guarantee an optimal solution. The algorithm is also vulnerable to becoming stuck in a local minimum if the selected neighbourhood is not sufficiently large.

Dorneles et. al. [11] solve a Brazilian high school timetabling problem with a similar IP-based "fix-and-optimize" heuristic that is conceptually very similar

to the Stage 2 (improvement) approach discussed here. In their implementation, they select one of three possible neighbourhood types: classes, teachers, or days. They explore the set of neighbourhoods by means of a Variable Neighbourhood Descent (VND) method, in which they limit the size of the neighbourhood by some predefined parameter, and the neighbourhoods are selected randomly. The sizes of their sub-problems are sufficiently small to enable repeated solution by a general-purpose MIP solver in acceptable time. Our Stage 2 approach is similar in that we also solve sub-problems, defined by a subset of all variables of the whole timetabling problem, with respect to the remaining class timetable. In contrast, our neighbourhoods are selected by analysing the incumbent timetable and identifying which scheduled course instances negatively affect the objective value. Of the neighbourhood types discussed in [11], the 'classes' neighbourhood type is the only type applicable in our case, as our timetabling sub-problem only includes aggregate trainers, and the length of courses relative to time granularity make the 'days' neighbourhood computationally impractical. The size of our neighbourhoods is adaptive, based on analysis of which other course instances are most relevant with respect to opportunity to make improvements, to the course instances being improved in a particular iteration.

3.3 Stage 3: Rostering

In the third stage, the rostering IP model is used to allocate specific trainers to each module on the timetable. While the IP for the timetabling sub-problem cannot be solved directly for Ausgrid data due to the size of the model, the IP model for the rostering sub-problem, subject to the produced timetable, is considerably smaller and can be solved to optimality, typically in under a minute.

Research in rostering has spanned several decades and has steadily been gaining research attention. The workforce rostering problem can be formulated and solved in many different ways including Artificial Intelligence, Constraint Programming, Metaheuristics, and Mathematical Programming approaches [12]. A commonly used model for scheduling and rostering comes from Dantzig's set covering formulation [9][12]. We found that the network model in [2] provides an efficient solution approach for our rostering problem, which is represented as a one minimum cost flow networks for each trainer, together with some non-network side constraints. The objective of the rostering IP model is to minimise the total flow cost. The structures of the networks are given by the timetable for which the roster is being generated. Flow along a particular arc on a particular network corresponds to a particular trainer being allocated to a particular module. Since the locations of trainers and the locations of scheduled modules are known, it is simple to determine the distance each trainer will need to travel in order to teach a given set of modules. Moreover, determining the number of trainer swaps is as simple as counting the incidences where the ith trainer differs from the $(i+1)$th trainer for a given course instance. Flow along each arc on each network therefore represents some amount of travel distance and the possibility of a trainer swap. The flow cost of each arc is given by a linear combination of the travel cost and trainer swap cost. While worker pay is often a high priority in many other rostering problems, we do not consider it in our rostering sub-problem as Ausgrid trainers are paid a fixed

salary regardless of what modules they teach. The case of considering workforce pay may be a consideration in future research.

If the produced timetable has no feasible solution with respect to trainer rostering, the reason must be determined and the problem should be corrected in the timetable. Due to the nature of the rostering sub-problem, infeasibility always arises when there are insufficient compatible trainers to teach the modules on the timetable. If there are n trainers capable of teaching a particular set modules, however there are $m > n$ of those modules running at a particular time period in a particular location, then there clearly can be no feasible roster. In this case, we return to stage two and solve the timetabling model with the additional constraint that, at all times, the total number of times modules requiring this type of trainer running at any given time must not exceed n. If this approach continues to fail to allow a feasible roster can be found, some course instances may need to be removed from the timetable resulting in some unmet course demand.

4 Timetabling Model

4.1 Time Discretisation

In the timetabling model, time is discretised into half-hour periods. Periods that are not available for training are not considered, which include lunch times and any time outside working hours. Periods are grouped into coarser intervals called days, each of which contain exactly 15 periods. Multiple days are grouped together into time windows, which represent longer durations such as a week or a month. In practice, we generally consider a window to span from the first to the last working day of a calendar month. Finally, periods are grouped together into rental windows. A rental window represents a set of consecutive periods in which rooms may be rented out to a third party. Rental windows range from being half a day to several days long. Rental windows may overlap with one another, such as a morning, an afternoon, and a whole day rental window on a given day. If a rentable room is unoccupied for one or more rental windows, it may be rented out to a third party, potentially bringing in additional revenue to Ausgrid.

4.2 Input Data Set-up

Some pre-processing on the input data can reduce the size of the resulting IP models. For the purpose of the class timetabling model, we fix the number of instances of each course to a practical estimation. While in practice each individual module can exist in many different courses, for the purposes of the IP model we require each module to be part of exactly one course instance - we refer to these as module instances. Each module in each course must therefore be duplicated such that each course instance has a set of unique modules.

Each location may have several rooms, however some rooms may be regarded as identical, i.e. they have the same list of compatible modules, the same capacity, the same rental income, etc. In this case, we can group identical rooms into room

types, where each room type represents a set of rooms in a given location that are functionally identical. Rooms in different locations are never grouped together, and each compound room—room with removable dividers—has its own, unique room type.

Compound rooms can be modelled using a set of mutually exclusive room pairs. Suppose room C can be split into two smaller rooms A and B. Rooms A and B can be used simultaneously, however the use of A is mutually exclusive with that of C, as is the the use of B with that of C.

4.3 List of Symbols

Sets of primary objects:

P	The indexed set of periods.
D	The indexed set of days.
Ω	The indexed set of time windows.
Λ	The indexed set of rental windows.
L	The set of locations.
Ξ	The set of regions.
$\hat{M}$	The indexed set of module instances.
C	The set of courses.
R	The set of room types.
T	The set of resource types.
$S^{(i)}$	The i^{th} element of any indexed set S.

Sets of derived objects:

I_c	The indexed set of instances for course $c \in C$.
$B_{c,i}$	The indexed set of modules instances for course $c \in C$ instance $i \in I_c$.
P_d	The indexed set of periods in day $d \in D$.
$\bar{P}_\omega$	The indexed set of periods in time window $\omega \in \Omega$.
$\tilde{P}_\lambda$	The indexed set of periods in rental window $\lambda \in \Lambda$.
$\hat{P}_c$	The set of periods in which course $c \in C$ may start.
$\bar{L}_\xi$	The set of locations in region $\xi \in \Xi$.
$\tilde{R}$	The set of mutually exclusive room pairs.
$\tilde{R}_l$	The set of room types in location $l \in L$.
R_m	The set of room types suitable for module $m \in \hat{M}$.

Primary decision variables:

$X_{m,r,p}$	$\in \{0,1\}$	1 if module $m \in \hat{M}$ runs in a room of type $r \in R$ starting at period $p \in P$, or 0 otherwise.
$Y_{c,i,p}$	$\in \{0,1\}$	1 if course $c \in C$ instance $i \in I_c$ starts at period $p \in P$, or 0 otherwise.
$\hat{Y}_{c,i,l}$	$\in \{0,1\}$	1 if course $c \in C$ instance $i \in I_c$ runs in location $l \in L$, or 0 otherwise.

$\psi_{t,l,k,p}$	$\in \mathbb{Z}^+$	The quantity of resource $t \in T$ moving from location $l \in L$ to location $k \in L$ (l and k may be the same), starting at period $p \in P$.
$\hat{\psi}_{t,l,k,d}$	$\in \mathbb{Z}^+$	The quantity of resource $t \in T$ moving from location $l \in L$ to location $k \in L$ (l and k may be the same), overnight at the end of day $d \in D$.
$\phi_{l,d}$	$\in \mathbb{Z}^+$	The number of trainers assigned to location $l \in L$ on day $d \in D$.

Auxiliary variables:

$\hat{X}_{m,r,p}$	$\in \{0,1\}$	1 if module $m \in \hat{M}$ runs in a room of type $r \in R$ during period $p \in P$, or 0 otherwise.
$\bar{Y}_{c,i}$	$\in \{0,1\}$	1 if course $c \in C$ instance $i \in I_c$ runs, or 0 otherwise.
$\tilde{Y}_{c,i,\omega,\xi}$	$\in \mathbb{Z}^+$	The number of students expected to sit in course $c \in C$ instance $i \in I_c$ during time window $\omega \in \Omega$ in region $\xi \in \Xi$.
$u_{c,\omega,\xi}$	$\in \mathbb{Z}^+$	The number of students not accommodated for course $c \in C$ during window $\omega \in \Omega$ in region $\xi \in \Xi$.
t_m	$\in \{0,1\}$	The new room flag for module $m \in \hat{M}$. If the room type for this module is that same type of room as for the previous module, if applicable, then $t_m = 0$, otherwise $t_m = 1$.
$\rho_{r,\lambda}$	$\in \mathbb{Z}^+$	The number of rooms of type $r \in R$ occupied during rental window $\lambda \in \Lambda$.
Z_i	$\in \mathbb{R}$	The ith goal term in the objective function.

Constants:

σ_t	$\in \mathbb{Z}^{++}$	The quantity available of resource $t \in T$.
$\delta_{t,l,k}$	$\in \mathbb{Z}^+$	The time required (in periods) for a unit of resource $t \in T$ to move from location $l \in L$ to location $k \in L$.
$\theta_{l,d}$	$\in \mathbb{Z}^+$	The number of trainers normally allocated to location $l \in L$ on day $d \in D$.
$\theta^{\max}$	$\in \mathbb{Z}^+$	The maximum number of additional trainers permitted to any location on any given day.
$\theta^{\min}$	$\in \mathbb{Z}^+$	The maximum number of subtracted trainers permitted from any location on any given day.
$Q_{r,p}$	$\in \mathbb{Z}^+$	The quantity of room type $r \in R$ available at period $p \in P$, or 0 otherwise.
$d_{r,\lambda}$	$\in \mathbb{R}^+$	The expected revenue from renting out a unit of room type $r \in R$ during rental window $\lambda \in \Lambda$.
l_c	$\in \mathbb{Z}^{++}$	The length (in periods) of course $c \in C$.
π_c	$\in \mathbb{Z}^{++}$	The length (in periods) of the rolling time window used to compute the minimum and maximum number of times a course $c \in C$ should be run.
π_c^+	$\in \mathbb{Z}^{++}$	The maximum number of times a course $c \in C$ should be run in any given time window of length π_c.
π_c^-	$\in \mathbb{Z}^+$	The minimum number of times a course $c \in C$ should be run in any given time window of length π_c.

$s_{c,\omega,\xi}$	$\in \mathbb{Z}^+$	The demand (in students) for course $c \in C$ during window $\omega \in \Omega$ in region $\xi \in \Xi$.
α_i	$\in \mathbb{R}^+$	The coefficient of the ith goal in the objective function.
u_m	$\in \mathbb{Z}^{++}$	The maximum number of students module $m \in \hat{M}$ can hold.
v_r	$\in \mathbb{Z}^{++}$	The maximum number of students room type $r \in R$ can hold.

4.4 Core Timetabling Constraints

The following constraints express the core requirements of the timetabling problem and are likely to appear in many similar timetabling problems:

$$\hat{X}_{m,r,p} = \sum_{q=0}^{d_m-1} X_{m,r,(p-q)} \quad \forall m \in \hat{M}, r \in \hat{R}_m, p \in P \quad (1)$$

$$\sum_{m \in \hat{M}} \hat{X}_{m,r,p} \leq Q_{r,p} \quad \forall r \in R, p \in P \quad (2)$$

$$\sum_{m \in \hat{M}} \hat{X}_{m,\tilde{r}_1,p} + \sum_{m \in \hat{M}} \hat{X}_{m,\tilde{r}_2,p} \leq 1 \quad \forall \{\tilde{r}_1, \tilde{r}_2\} \in \tilde{R}, p \in P \quad (3)$$

$$\sum_{r \in R} \sum_{p \in P} X_{m,r,p} = \bar{Y}_{c,i} \quad \forall c \in C, i \in I_c, m \in B_{c,i} \quad (4)$$

$$\bar{Y}_{c,i} = \sum_{p \in \hat{P}_c} Y_{c,i,p} \quad \forall c \in C, i \in I_c \quad (5)$$

The auxiliary variables $\hat{X}_{m,r,p}$ are set up from $X_{m,r,p}$ according to (1). The constraints (2) express the requirement that rooms should not be double-booked, however since identical rooms within a single location are aggregated together, the right-hand-side is given by the quantities of the aggregated rooms. The constraints (3) also express the requirement that splittable rooms should not be double-booked, however since splittable rooms are never aggregated together, the right-hand-side remains 1. The constraints (4) ensure that each module of a course is run exactly once if the course is run, or not at all. The expressions (5) set up the $\bar{Y}_{c,i}$ variable, which is a sum over all periods of $Y_{c,i,p}$, and also imply $\sum_{p \in \hat{P}_c} Y_{c,i,p} \leq 1$ for all course instances.

4.5 Characteristic Constraints

The remaining constraints express the operational requirements that are rarely found in traditional timetabling problems.

4.5.1 Module Positioning Constraints

$$\sum_{m \in B_{c,i}} \sum_{r \in R_m} \hat{X}_{m,r,p} = \sum_{q=0}^{l_c - 1} Y_{c,i,(p-q)} \quad \forall c \in C, i \in I_c, p \in P \tag{6}$$

$$\sum_{m \in B_{c,i}} \sum_{r \in \tilde{R}_l} \sum_{p \in P} X_{m,r,p} = |B_{c,i}| \times \hat{Y}_{c,i,l} \quad \forall c \in C, i \in I_c, l \in L \tag{7}$$

The constraints (6) expresses the requirement that the modules for a course run back-to-back, and (7) expresses the requirement that all the modules for a course must be run in exactly one location.

4.5.2 Capacity Constraints

The following constraints determine the capacity of each course instance in each time window and region based on the values of the X and Y variables:

$$\tilde{Y}_{c,i,\omega,\xi} \leq \mathscr{M}_{c,\xi} \sum_{p \in \bar{P}_\omega} Y_{c,i,p} \quad \forall c \in C, i \in I_c, \omega \in \Omega, \xi \in \Xi \tag{8}$$

$$\tilde{Y}_{c,i,\omega,\xi} \leq \mathscr{M}_{c,\xi} \sum_{l \in \bar{L}_\xi} \hat{Y}_{c,i,l} \quad \forall c \in C, i \in I_c, \omega \in \Omega, \xi \in \Xi \tag{9}$$

$$\begin{aligned} \tilde{Y}_{c,i,\omega,\xi} \leq \mathscr{M}_{c,\xi}(1 - \sum_{p \in \bar{P}_\omega} X_{m,r,p}) + \min\{u_m, v_r\} \\ \forall c \in C, i \in I_c, \omega \in \Omega, \xi \in \Xi, m \in B_{c,i}, l \in \bar{L}_\xi, r \in \tilde{R}_l \end{aligned} \tag{10}$$

The constraints (8)-(10) set up the $\tilde{Y}_{c,i,\omega,\xi}$ variables given some sufficiently large constant $\mathscr{M}_{c,\xi}$. For best IP performance, the value of $\mathscr{M}_{c,\xi}$ should be chosen to be as small as possible, which is the largest course capacity for the course c in the rooms available in region ξ.

4.6 Trainer Movement Constraints

Trainers are considered in a generalised, aggregated way for capacity purposes only. Nevertheless, we permit the quantity of these generalised trainers to change per location per day to give a coarse representation of trainer movements. Each trainer has a location where they are normally based, however they may be required to travel to other locations. The total quantity of trainers at location $l \in L$ on day $d \in D$, by default, is given by the constant $\theta_{l,d}$.

$$\phi_{l,d} \leq \theta_{l,d} + \theta^{\max} \quad \forall l \in L, d \in D \tag{11}$$

$$\phi_{l,d} \geq \theta_{l,d} - \theta^{\min} \quad \forall l \in L, d \in D \tag{12}$$

$$\sum_{l \in L} \phi_{l,d} = \sum_{l \in L} \theta_{l,d} \quad \forall d \in D \tag{13}$$

$$\sum_{m \in \hat{M}} \sum_{r \in \tilde{R}_l} \hat{X}_{m,r,p} \leq \phi_{l,d} \quad \forall l \in L, d \in D, p \in P_d \tag{14}$$

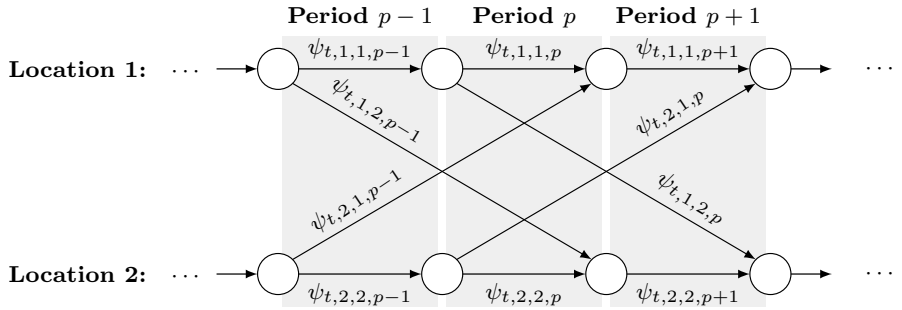

Fig. 5 A sample flow network for some resource t about period p with 2 locations.

The constraints (11) and (12) establish the minimum and maximum number of trainers permitted to be at a given location on a given day, and the constraints (13) ensure that the total number of trainers allocated to each location is equal to the total number of trainers expected to be working company-wide on that day. The constraints (14) express the requirement that, at any given time, the total number of modules run in a location concurrently must not exceed the number of generalised trainers we have chosen to allocate there.

4.7 Resource Movement Constraints

A network formulation can be used to represent the flow of resources between locations across time. Resources, in this context, refer to shared, mobile pieces of equipment that are required for teaching particular modules. For each type of resource for each day, we construct a flow network with the nodes arranged in a rectangular lattice (See Figure 5). The horizontal axis represents time, and the vertical axis represents the various locations. Each node represents the end points of a time period at a given location. Adjacent nodes are connected by directed arcs horizontally and pointing forward in time, with the flow along those arcs representing the quantity of the resource available at a particular location at a particular time. Nodes are also connected between different locations by directed arcs in such a way that the time interval from the source node to the destination node is given by the time required to move the resource from the source location to the destination.

We permit resources to move from any location to any other other location overnight at no cost, therefore the initial condition for each resource network for each day is simply that the sum across all locations must equal the available quantity of that resource.

If $l = k$, the variables $\psi_{t,l,k,p}$ represents the quantity of resource $t \in T$ available at location $l \in L$ during time period $p \in P$. If $l \neq k$, the variable represents the quantity of the resource moving from location $l \in L$ to location $k \in L$ starting its journey at $p \in P$. Since the transport time of resource $t \in T$ from $l \in L$ to $k \in K$ is given by $\delta_{t,l,k}$, the arc represented by $\psi_{t,l,k,p}$ will be connected to the node that represents the start of period $p + \delta_{t,l,k}$.

The flow balance equations for the networks are expressed as follows:

$$\sum_{l \in L} \sum_{k \in L} \psi_{t,l,k,P_d^{(1)}} = \sigma_t \qquad \forall t \in T, d \in D \tag{15}$$

$$\sum_{k \in L} \psi_{t,k,l,(p-\delta_{t,k,l})} = \sum_{k \in L} \psi_{t,l,k,p} \quad \forall t \in T, l \in L, d \in D, p \in (P_d \setminus P_d^{(1)}) \tag{16}$$

We ensure that the number of times resources are used is limited by the number available at the time:

$$\sum_{m \in \hat{M}_t} \sum_{r \in \tilde{R}_l} \hat{X}_{m,r,p} \leq \psi_{t,l,l,p} \quad \forall l \in L, t \in T, p \in P \tag{17}$$

4.8 Spreading Constraints

For each course, we have a defined minimum and maximum number of instances that may be run in any arbitrary set of consecutive periods of a predetermined length:

$$\sum_{i \in I_c} \sum_{q=0}^{\pi_c} Y_{c,i,p+q} \geq \pi_c^- \quad \forall c \in C, p \in \hat{P}_c \tag{18}$$

$$\sum_{i \in I_c} \sum_{q=0}^{\pi_c} Y_{c,i,p+q} \leq \pi_c^+ \quad \forall c \in C, p \in \hat{P}_c \tag{19}$$

The constraints (18) and (19) establish the minimum and maximum number of instances, respectively, that must be run across all regions for each course. Selection of the π_c and the π_c^- and π_c^+ constants is made given the problem data.

4.9 Objective Function

Being a large-scale industrial problem, there are many potential objectives we can consider. In this paper, we consider three objectives in a weighted linear function:

- Minimise the number of expected students not accommodated;
- Maximise the rental revenue; and
- Minimise the number of room swaps in the timetable.

The first objective, denoted by Z_1, is to minimise the number of students not accommodated:

$$\sum_{i \in I_c} \tilde{Y}_{c,i,\omega,\xi} + u_{c,\omega,\xi} = s_{c,\omega,\xi} \quad \forall c \in C, \omega \in \Omega, \xi \in \Xi \tag{20}$$

$$u_{c,\omega,\xi} \geq 0 \qquad \forall c \in C, \omega \in \Omega, \xi \in \Xi \tag{21}$$

$$\sum_{c \in C} \sum_{\omega \in \Omega} \sum_{\xi \in \Xi} u_{c,\omega,\xi} = Z_1 \tag{22}$$

The second objective, denoted by Z_2, is to maximise the rental revenue:

$$\sum_{m \in \hat{M}} \hat{X}_{m,r,p} \leq \rho_{r,\lambda} \quad \forall r \in R, \lambda \in \Lambda, p \in \tilde{P}_\lambda \tag{23}$$

$$Z_2 = \sum_{r \in R} \sum_{\lambda \in \Lambda} [-d_{r,\lambda} \times (\hat{Q}_{r,\lambda} - \rho_{r,\lambda})] \tag{24}$$

where $\hat{Q}_{r,\lambda}$ is the smallest value of $Q_{r,p}$, $\forall p \in \tilde{P}_\lambda$ for each $\lambda \in \Lambda$.

The last objective, denoted by Z_3, is to minimise the number of room swaps across all courses:

$$X_{m,r,p} - \sum_{n \in B_{c,i}, m \neq n} \hat{X}_{n,r,(p-1)} \leq t_m \qquad \forall c \in C, i \in I_c, m \in B_{c,i}, r \in R_m, p \in P \tag{25}$$

$$Z_3 = \sum_{m \in \hat{M}} t_m \tag{26}$$

The objective function is a weighted linear sum of the the individual objectives:

$$\text{minimise:} \quad Z = \alpha_1 Z_1 + \alpha_2 Z_2 + \alpha_3 Z_3 \tag{27}$$

with weights α_1, α_2, and α_3.

In reality, Ausgrid's timetabling problem has an ordered bi-criteria objective: the goal is to maximise the potential room rental revenue and minimise the number of room swaps (Z_2 and Z_3) over the set of timetables that minimise the number of unsatisfied students (Z_1). This can be achieved by solving the timetabling sub-problem for the primary objective of minimising Z_1, and then solving the timetabling sub-problem again for the secondary objectives of minimising over Z_2 and Z_3, but with the additional constraint that Z_1 must assume the optimal value determined previously. A simpler way of achieving the same result is to give a sufficiently large coefficient to Z_1 with respect to the coefficients of Z_2 and Z_3.

5 Rostering Model

Given a solution to the class timetabling problem from Section 4, the rostering model describes the problem of allocating specific trainers to modules resulting in a complete timetable and roster. A minimum cost network flow approach, together with some side constraints, can be utilised to give a simple and convenient representation of the rostering problem.

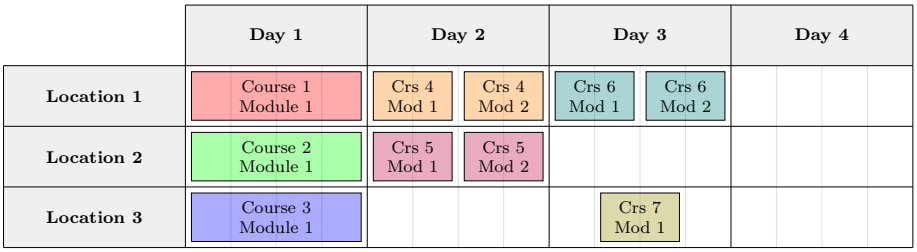

Fig. 6 A sample timetable, simplified for viewing in this format, showing 4 days, 3 locations, and 7 courses each with 1 or 2 modules.

Given a timetable, a flow network is constructed for each trainer. There are two different types of nodes, and four different types of arcs in the networks: home nodes, activity nodes, commencement arcs, transition arcs, return arcs, and bypass arcs. Home nodes represent the location where the trainer starts and ends their work day. Activity nodes represent specific modules that can be taught by the trainer. Commencement arcs—representing the trainer commencing training for a particular day—originate from home nodes and end at activity nodes. Transition arcs—representing trainer finishing teaching one module and then teaching another, with or without a break—originate from activity nodes and end at activity nodes. Return arcs—representing the trainer completing their training for a particular day—originate from activity nodes and end at home nodes. Bypass arcs—representing a particular day when a trainer will not deliver any training—originate at home nodes and end at home nodes.

As an example, Figure 6 shows a simplified view of a timetable with 7 courses, and the corresponding flow network is shown in Figure 7. In Figure 7, flow along any of the arcs from (4:1)→(5:2), (H2)→(4:2), (H2)→(5:2), (5:1)→(4:2), or (H3)→(6:2) indicate the presence of a trainer swap. Care should be taken not to double-count trainer swaps. The convention we use is to count only instances where the incoming trainer differs from the preceding trainer, not when the outgoing trainer is different from the proceeding trainer. For our incoming-only convention, we do count flow along the arcs (4:1)→(H3), (5:1)→(H3), or (6:1)→(H4) when determining the number of trainer swaps.

Flow costs on the arcs of the network are determined by two factors. The first factor is determined by the distance the trainer needs to travel for the allocation, including travel to the first module taught in a day, travel from the last module taught in a day, and also travel from module to module. The second factor is the trainer swap cost. A trainer swap happens if the arc starts with an activity node which is not the last module of a course instance, but ends with a home node or an activity node from a different course instance.

We introduce the following symbols for the rostering model:

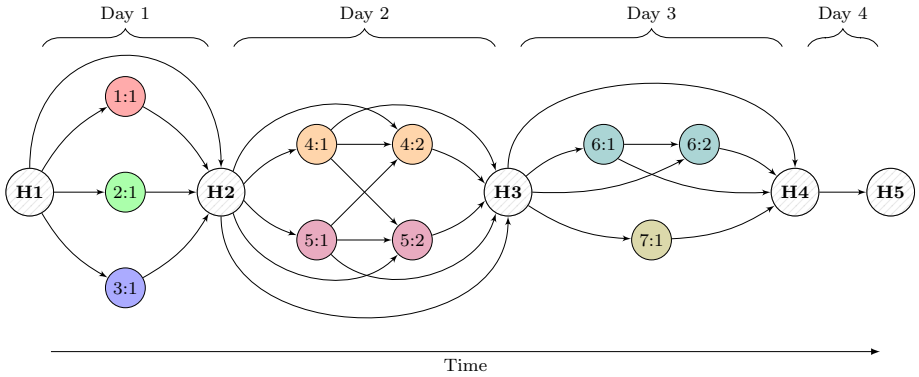

Fig. 7 The flow network corresponding to the sample timetable shown in Figure 6. (Home nodes are hatched, and activity nodes are solid)

Sets of objects:

D	The indexed set of days.
$\hat{M}$	The indexed set of module instances.
M_d	The set of modules that run within day $d \in D$
T	The set of trainers.
T_m	The set of trainers capable of teach module $m \in \hat{M}$.
$pred(m)$	The set of predecessors of module $m \in \hat{M}$.
$succ(m)$	The set of successors of module $m \in \hat{M}$.

Variables:

$\bar{\psi}_{\tau,m}$	$\in \{0,1\}$	1 if trainer τ teaches module m as their first module on that day, or 0 otherwise.
$\psi_{\tau,m,n}$	$\in \{0,1\}$	1 if trainer τ teaches module m followed by module n, or 0 otherwise.
$\tilde{\psi}_{\tau,m}$	$\in \{0,1\}$	1 if trainer τ teaches module m as their last module on that day, or 0 otherwise.
$\hat{\psi}_{\tau,d}$	$\in \{0,1\}$	1 if trainer τ doesn't teach any modules on day d, or 0 otherwise.
$X_{m,\tau}$	$\in \{0,1\}$	1 if trainer τ teaches module m, or 0 otherwise.

The flow balance equations for the network are as follows:

$$\hat{\psi}_{\tau,d} + \sum_{m \in M_d} \bar{\psi}_{\tau,m} = 1 \qquad \forall \tau \in \mathrm{T}, d \in D \tag{28}$$

$$\bar{\psi}_{\tau,m} + \sum_{n \in pred(m)} \psi_{\tau,n,m} = \sum_{n \in succ(m)} \psi_{\tau,m,n} + \tilde{\psi}_{\tau,m} \quad \forall \tau \in \mathrm{T}, m \in \hat{M} \tag{29}$$

where (28) ensures that, at the start of each day, the trainer either teaches one or more modules or does not teach any modules; and (29) conserves flow throughout the day. Since the flow for each day is implicitly conserved by (28) and (29), we do not require any additional equations to balance the flow from day to day.

Any integral flow is always a feasible schedule for a single trainer, i.e. the network is constructed in such a way the trainer can never be required to be in two places at once, nor can they be required to teach a module they are not capable of teaching.

The individual trainer networks on their own cannot guarantee a feasible solution to the rostering problem, as multiple trainers could be allocated to the same module, or modules may be left with no trainer at all. We introduce some side constraints that integrate the many trainer networks into a single IP model.

$$\bar{\psi}_{\tau,m} + \sum_{n \in pred(m)} \psi_{\tau,n,m} = X_{m,\tau} \qquad \forall m \in \hat{M}, \tau \in \mathrm{T} \tag{30}$$

$$\sum_{\tau \in \mathrm{T}_m} X_{m,\tau} = 1 \qquad \forall m \in \hat{M} \tag{31}$$

where (30) sets up the auxiliary variable $X_{m,t}$, which is 1 if trainer t teaches module m or 0 otherwise, and (31) ensures that every scheduled module is taught be exactly one trainer.

In order to ensure fairness, we wish to avoid, wherever possible, the situation where one trainer is scheduled to train much more or less than their peers.

$$U_\tau^- \leq \sum_{m \in \hat{M}} (w_m \times X_{m,\tau}) \leq U_\tau^+ \quad \forall \tau \in \mathrm{T} \tag{32}$$

where U_t^- and U_t^+ are the minimum and maximum number of periods, respectively, that we permit trainer $t \in T$ to teach.

The objective of the rostering problem is to minimise the flow cost all networks simultaneously.

$$\begin{aligned} \min \sum_{\tau \in \mathrm{T}} \sum_{m \in \hat{M}} [c_1(\tau, m) \times \tilde{\psi}_{\tau,m}] + \sum_{\tau \in \mathrm{T}} \sum_{m \in \hat{M}} \sum_{n \in \hat{M}} [c_2(\tau, m, n) \times \psi_{\tau,m,n}] + \\ \sum_{\tau \in \mathrm{T}} \sum_{m \in \hat{M}} [c_3(\tau, m) \times \bar{\psi}_{\tau,m}] + \sum_{\tau \in \mathrm{T}} \sum_{d \in D} [c_4(\tau, d) \times \hat{\psi}_{\tau,d}] \end{aligned} \tag{33}$$

where $c_1(\cdot)$, $c_2(\cdot)$, $c_3(\cdot)$, and $c_4(\cdot)$ give the flow costs of the commencement, transition, return, and bypass arcs, respectively, where the flow costs are characterised by any applicable trainer travel costs and trainer swap costs.

6 Implementation

As a result of the back-to-back restriction when scheduling the modules of a course instance, there is an implied set of feasible start times for each module instance. These can be determined by permuting the order of the modules of the course, and determining the start time of each module relative to the course start time. Clearly, if no module permutation can allow for certain modules to start at certain times, then those variables can be eliminated from the model.

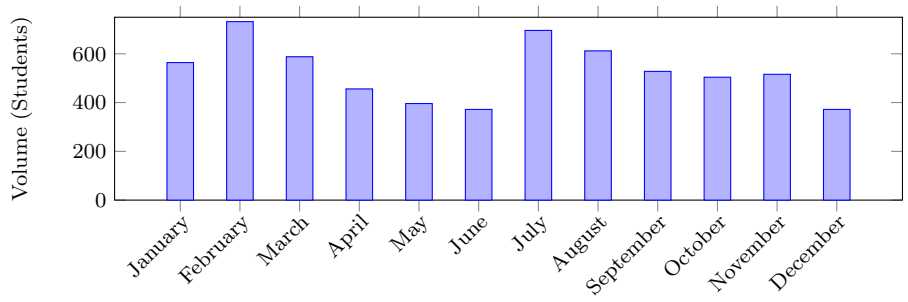

Fig. 8 Typical yearly training volume at Ausgrid.

The solution space of the timetabling model exhibits some symmetry, as, for a given timetable, the indices of the instances of any course can be permuted without affecting the objective value. There are $n!$ ways of indexing the n instances of a single course across an existing timetable. We can eliminate many symmetric solutions by introducing the following constraints:

$$\sum_{q=0}^{p} Y_{c,i,q} \geq Y_{c,(i+1),p} \quad \forall c \in C, i \in I_c, p \in P_c \tag{34}$$

which ensures that, for any given course, instance i must be run in order to run instance $i+1$, and also that instance i must be run no later than instance $i+1$.

In Stage 2 (improvement), since the solution at iteration i is feasible at iteration $i+1$, it can be provided to the IP solver as a "warm start" or "advanced start" to begin the search procedure [16].

7 Computational Experiments

Ausgrid's supplied both current and historical data related to class timetabling and trainer rostering. The overwhelming majority of their training volume comes from the eight most frequently run courses, which have between one and four modules and needed up to 26 instances each for a monthly timetable. There were five regions and 15 locations, and room counts ranged from one to eight per location. There were 21 trainers in 11 of the 15 locations, and each trainer could teach between 8 and 14 modules. There were eight working hours per day, split into half-hour periods. There were three rental windows per day, one in the morning, one in the afternoon, and one for the whole day. There was one demand window in each month that encompassed the entire planning horizon.

Training volume at Ausgrid fluctuates from month to month, following a yearly pattern. Since most courses are valid for 12 months, students who complete a course in a particular month are likely to request the same course again around the same time the following year. Figure 8 shows typical training volume by month. February—the busiest month—is usually about twice as busy as June or December—the least busy months.

Test Case	Cols	Rows	Test Case	Cols	Rows	Test Case	Cols	Rows
(L|10|2)	13261	18565	(M|10|2)	14427	20730	(H|10|2)	16267	23869
(L|10|3)	29215	31393	(M|10|3)	33099	37108	(H|10|3)	36605	42592
(L|10|4)	56253	49907	(M|10|4)	61749	57005	(H|10|4)	69639	66068
(L|10|5)	83383	64823	(M|10|5)	93693	77022	(H|10|5)	106155	91076
(L|15|2)	23246	36118	(M|15|2)	26284	41816	(H|15|2)	29976	48737
(L|15|3)	52356	58944	(M|15|3)	60013	70093	(H|15|3)	70050	84574
(L|15|4)	103785	93173	(M|15|4)	121753	113339	(H|15|4)	134357	127404
(L|15|5)	147920	124199	(M|15|5)	178637	159168	(H|15|5)	200594	185017
(L|20|2)	35802	56074	(M|20|2)	41623	66586	(H|20|2)	47713	78115
(L|20|3)	83827	100481	(M|20|3)	96509	120324	(H|20|3)	115221	148142
(L|20|4)	153137	150021	(M|20|4)	178934	180809	(H|20|4)	206623	214455
(L|20|5)	253259	223389	(M|20|5)	299697	275786	(H|20|5)	347315	327884
(L|25|2)	52191	82204	(M|25|2)	60888	97115	(H|25|2)	72637	117327
(L|25|3)	121575	151682	(M|25|3)	149025	193513	(H|25|3)	174001	229970
(L|25|4)	225070	226831	(M|25|4)	273907	282633	(H|25|4)	321000	338276
(L|25|5)	352849	322377	(M|25|5)	421138	399011	(H|25|5)	507011	493472

Table 1 The number of variables (Cols) and constraints (Rows) for the timetabling IP model for each test case.

We generated a series of 48 test cases with similar properties to the supplied data. All test cases and solution files can be downloaded from `https://goo.gl/zUWYV2`. The test cases had planning horizons ranging between two and five weeks, between two and five regions each with two locations per region, and between two and four rooms per location. The courses were generated to be between one and five days long, with up to 15 modules per course. Each of the test cases target one of three timetable densities: Low (`L`), Medium (`M`), and High (`H`). The course demand values in the `L`, `M`, and `H` test cases were tuned so that, if all course instances are run at 80% of their student capacity, the resulting timetable will be $\frac{1}{3}$, $\frac{1}{2}$, and $\frac{2}{3}$ full, respectively, where a full timetable is one where every room is occupied at every time slot. For brevity, we refer to each test case by their target timetable density, number of days, and number of regions. For example, (L|10|2) is the smallest test case with low target timetable density, 10 days and 2 regions, whereas (H|25|5) is the largest test case with high target timetable density, 25 days and 5 regions.

The smallest test case, (L|10|2), had 13261 variables and 18565 constraints for the complete timetabling sub-problem, and the largest test case (H|25|5) had 507011 variables and 493472 constraints. The number of variables (columns) and constraints (rows) for each of the test cases can be seen in Table 1.

We used IBM ILOG CPLEX 12.5.0.0 [15] 64-bit on an Intel i7-4790K quad-core 4.00Ghz system with 16GB of RAM, running Windows 7 Professional. Our code was written in C# 4.0, and interacted with CPLEX using the IBM ILOG Concert API. We used default CPLEX settings, except we increased the maximum allowed memory usage to the total amount of free physical memory.

We chose the weights in the class timetabling objective function to be $\alpha_1 = 10^6$, $\alpha_2 = 5$, and $\alpha_3 = 1$, based on empirical testing and Ausgrid's inspection of the produced timetables. For the rostering objective, we gave each km travelled an objective weighting of 1, and each trainer swap an objective weighting of 5.

Test Case	S1	S2	Test Case	S1	S2	Test Case	S1	S2
(L|10|2)	13	12	(M|10|2)	22	12	(H|10|2)	43	13
(L|10|3)	44	28	(M|10|3)	103	27	(H|10|3)	168	9550
(L|10|4)	195	11634	(M|10|4)	287	12227	(H|10|4)	481	13075
(L|10|5)	420	14549	(M|10|5)	571	15660	(H|10|5)	1015	16999
(L|15|2)	45	25	(M|15|2)	102	31	(H|15|2)	167	34
(L|15|3)	146	62	(M|15|3)	239	86	(H|15|3)	554	83
(L|15|4)	565	19226	(M|15|4)	1116	18734	(H|15|4)	1714	20107
(L|15|5)	1146	21492	(M|15|5)	2347	24786	(H|15|5)	3986	27158
(L|20|2)	103	49	(M|20|2)	158	282	(H|20|2)	339	66
(L|20|3)	443	129	(M|20|3)	589	165	(H|20|3)	3779	18129
(L|20|4)	1413	820	(M|20|4)	3964	28955	(H|20|4)	6377	28961
(L|20|5)	6084	36161	(M|20|5)	8558	42273	(H|20|5)	19297	71159
(L|25|2)	232	94	(M|25|2)	331	109	(H|25|2)	805	107
(L|25|3)	888	273	(M|25|3)	1916	907	(H|25|3)	7410	24196
(L|25|4)	3006	1492	(M|25|4)	7962	38773	(H|25|4)	17354	43881
(L|25|5)	10768	48409	(M|25|5)	12577	54548	(H|25|5)	43749	60956

Table 2 The amount of time, in seconds, the algorithm spent in Stage 1 (S1) and Stage 2 (S2) for each test case.

Test Case	$Z_1^{(1)}$	$Z_2^{(1)}$	$Z_3^{(1)}$	$Z_1^{(2)}$	$Z_2^{(2)}$	$Z_3^{(2)}$
(L|10|2)	0	-8131	42	0	-9455	42
(L|10|3)	0	-11773	54	0	-13991	56
(L|10|4)	9	-14398	105	9	-16184	92
(L|10|5)	3	-15568	112	3	-21403	111
(L|15|2)	0	-11327	65	0	-13632	66
(L|15|3)	0	-16389	98	0	-22053	87
(L|15|4)	0	-21551	146	0	-25996	156
(L|15|5)	34	-27153	156	12	-24446	166
(L|20|2)	0	-15804	91	0	-20507	89
(L|20|3)	0	-22585	139	0	-28911	125
(L|20|4)	0	-34912	160	0	-40988	158
(L|20|5)	0	-43984	257	0	-45091	251
(L|25|2)	0	-16621	115	0	-20060	106
(L|25|3)	0	-33200	135	0	-39603	143
(L|25|4)	0	-38811	250	0	-54119	207
(L|25|5)	0	-50316	274	0	-62890	277

Table 3 The objective value components for stages 1 and 2 for the test cases with low target timetable density.

The time the algorithm spent in Stage 1 (construction) and Stage 2 (improvement) is shown in Table 2. During timetable construction in the first stage, no time limit is imposed. During Stage 2, there are two termination criteria: the LNS algorithm terminates if it is unable to improve the solution after a number of iterations or if a time limit is reached. The time limit was chosen to be $1000 \cdot |D| \cdot |\Xi| \cdot \kappa$ seconds and the number of iterations at which to terminate if no improvement can be made was chosen to be $10 \cdot |D| \cdot |\Xi| \cdot \kappa$, where, for a given test case, D is the set of included days (as defined in Section 4), Ξ is the set of included regions (as defined in Section 4), and κ is $\frac{1}{3}$, $\frac{1}{2}$, and $\frac{2}{3}$ for test cases with low, medium, and high target density, respectively. Due to the time required to solve each individual IP model, the the total time sometimes exceeds the time limit slightly.

Test Case	$Z_1^{(1)}$	$Z_2^{(1)}$	$Z_3^{(1)}$	$Z_1^{(2)}$	$Z_2^{(2)}$	$Z_3^{(2)}$
(M|10|2)	0	-6477	54	0	-7682	49
(M|10|3)	0	-6715	83	0	-10070	72
(M|10|4)	14	-9884	122	14	-12894	116
(M|10|5)	5	-11353	140	4	-13897	114
(M|15|2)	0	-6937	84	0	-9989	93
(M|15|3)	0	-16667	89	0	-20669	90
(M|15|4)	35	-17520	168	15	-22039	173
(M|15|5)	18	-21380	182	18	-22456	174
(M|20|2)	0	-15756	89	0	-19439	91
(M|20|3)	0	-16910	163	0	-27363	129
(M|20|4)	6	-23372	218	6	-24548	208
(M|20|5)	8	-33400	312	0	-35073	298
(M|25|2)	0	-15625	121	0	-18673	111
(M|25|3)	0	-21981	183	0	-28373	185
(M|25|4)	2	-37006	295	2	-38862	281
(M|25|5)	4	-53925	318	4	-56626	303

Table 4 The objective value components for stages 1 and 2 for the test cases with medium target timetable density.

The solution quality results for Stage 1 (construction) and Stage 2 (improvement) for the test cases with low target timetable density are given in Table 3. Stage 1 was able to construct timetables that satisfied all demand for 13 of the 16 cases. Of the three cases where not all demand could be satisfied, Stage 2 was only able to improve one of them with respect to the number of unsatisfied students, bringing the number of unsatisfied students from 34 down to 12 for that test case. Due to the room compatibility, course spreading, resource movement, and trainer capacity constraints, not all test cases have feasible timetables where all student demand can be satisfied. Stage 2 was able to improve the rental revenue objective by about 18% on average, with a maximum improvement of about 39%. Stage 2 was able to reduce the number of room swaps by about 4 swaps per test case on average with a maximum improvement of 43 eliminated room swaps, or about 3% on average with a maximum improvement of 20%.

The solution quality results for Stage 1 (construction) and Stage 2 (improvement) for the test cases with medium target timetable density are given in Table 4. Stage 1 was able to construct timetables that satisfied all demand for 8 of the 16 cases. Of the eight cases where not all demand could be satisfied, Stage 2 was able to improve three of them with respect to the number of unsatisfied students, bringing the total number of unsatisfied students from 48 down to 19 for those three cases. Stage 2 was able to improve the rental revenue objective by about 23% on average, with a maximum improvement of about 62%. Stage 2 was able to reduce the number of room swaps by about 8 swaps per test case on average, with a maximum improvement of 34 eliminated room swaps, or about 6% on average with a maximum improvement of 26%.

The solution quality results for Stage 1 (construction) and Stage 2 (improvement) for the test cases with high target timetable density are given in Table 5. Stage 1 was able to construct timetables that satisfied all demand for 9 of the 16 cases. Of the seven cases where not all demand could be satisfied, Stage 2 was able

Test Case	$Z_1^{(1)}$	$Z_2^{(1)}$	$Z_3^{(1)}$	$Z_1^{(2)}$	$Z_2^{(2)}$	$Z_3^{(2)}$
(H|10|2)	0	-4516	64	0	-5939	64
(H|10|3)	8	-5700	94	2	-8110	81
(H|10|4)	29	-8914	124	17	-10856	126
(H|10|5)	34	-9460	155	5	-10587	141
(H|15|2)	0	-5539	93	0	-9338	98
(H|15|3)	0	-7822	131	0	-15269	127
(H|15|4)	49	-20583	191	49	-21617	182
(H|15|5)	40	-20922	233	40	-21976	222
(H|20|2)	0	-8298	136	0	-12972	125
(H|20|3)	0	-9299	209	0	-18822	190
(H|20|4)	23	-19805	254	23	-20818	242
(H|20|5)	0	-21448	314	0	-21448	314
(H|25|2)	0	-8529	161	0	-13369	168
(H|25|3)	0	-17871	210	0	-24004	231
(H|25|4)	0	-29399	350	0	-30873	334
(H|25|5)	26	-36185	393	26	-37999	375

Table 5 The objective value components for stages 1 and 2 for the test cases with high target timetable density.

Test Case	S3	Test Case	S3	Test Case	S3
(L|10|2)	0.1	(M|10|2)	0.1	(H|10|2)	0.3
(L|10|3)	0.2	(M|10|3)	0.3	(H|10|3)	0.5
(L|10|4)	6.2	(M|10|4)	8.6	(H|10|4)	10
(L|10|5)	18	(M|10|5)	16	(H|10|5)	21
(L|15|2)	0.1	(M|15|2)	0.2	(H|15|2)	0.4
(L|15|3)	0.4	(M|15|3)	0.5	(H|15|3)	0.5
(L|15|4)	25	(M|15|4)	27	(H|15|4)	27
(L|15|5)	22	(M|15|5)	31	(H|15|5)	38
(L|20|2)	0.3	(M|20|2)	1.6	(H|20|2)	0.3
(L|20|3)	0.4	(M|20|3)	0.5	(H|20|3)	4.7
(L|20|4)	10	(M|20|4)	12	(H|20|4)	15
(L|20|5)	25	(M|20|5)	29	(H|20|5)	41
(L|25|2)	0.4	(M|25|2)	0.5	(H|25|2)	0.6
(L|25|3)	0.4	(M|25|3)	4.3	(H|25|3)	11
(L|25|4)	10	(M|25|4)	17	(H|25|4)	22
(L|25|5)	29	(M|25|5)	33	(H|25|5)	39

Table 6 The amount of time, in seconds, the algorithm spent in Stage 3 (S3) for each test case.

to improve three of them with respect to the number of unsatisfied students, bringing the total number of unsatisfied students from 71 down to 24 for those three cases. Stage 2 was able to improve the rental revenue objective by about 34% on average, with a maximum improvement of about 202%. Stage 2 was able to reduce the number of room swaps by about 6 swaps per test case on average, with a maximum improvement of 19 eliminated room swaps, or about 3% on average with a maximum improvement of 16%.

The amount of time, in seconds, the algorithm spent in Stage 3 (rostering) for each test case is shown in Table 6. The test case that required the longest time to produce an optimal roster required only 41 seconds, and the largest test case required only 39 seconds to produce an optimal roster.

Test Case	Cols	Rows	Test Case	Cols	Rows	Test Case	Cols	Rows
(L|10|2)	2533	2085	(M|10|2)	2756	2287	(H|10|2)	3495	2854
(L|10|3)	4154	3238	(M|10|3)	4998	3848	(H|10|3)	5862	4538
(L|10|4)	10086	8015	(M|10|4)	12159	9464	(H|10|4)	13337	10285
(L|10|5)	13284	9945	(M|10|5)	14941	11187	(H|10|5)	17969	13391
(L|15|2)	3385	2706	(M|15|2)	4559	3603	(H|15|2)	4808	3786
(L|15|3)	6513	5107	(M|15|3)	6766	5264	(H|15|3)	9093	7064
(L|15|4)	17939	13850	(M|15|4)	19070	14675	(H|15|4)	19283	14743
(L|15|5)	20955	15562	(M|15|5)	22425	16507	(H|15|5)	27700	20106
(L|20|2)	4874	3902	(M|20|2)	4929	3925	(H|20|2)	9771	5352
(L|20|3)	9164	6996	(M|20|3)	9478	7207	(H|20|3)	13400	10000
(L|20|4)	18323	13831	(M|20|4)	24231	18227	(H|20|4)	27456	20529
(L|20|5)	32205	23638	(M|20|5)	37657	27487	(H|20|5)	40842	29757
(L|25|2)	5983	4811	(M|25|2)	6164	4948	(H|25|2)	9101	7200
(L|25|3)	10178	7619	(M|25|3)	13269	9852	(H|25|3)	16155	11881
(L|25|4)	23550	17618	(M|25|4)	31560	23417	(H|25|4)	37222	27483
(L|25|5)	34816	25222	(M|25|5)	37397	26970	(H|25|5)	46211	33073

Table 7 The number of variables (Cols) and constraints (Rows) for the roster IP model, subject to the final generated timetable, for each test case.

Test Case	Travel	Swaps	Test Case	Travel	Swaps	Test Case	Travel	Swaps
(L|10|2)	209.7	1	(M|10|2)	0.0	0	(H|10|2)	419.3	1
(L|10|3)	1045.7	1	(M|10|3)	378.6	0	(H|10|3)	1723.1	1
(L|10|4)	233.0	0	(M|10|4)	490.6	0	(H|10|4)	388.3	1
(L|10|5)	322.3	1	(M|10|5)	177.6	0	(H|10|5)	391.2	2
(L|15|2)	1512.6	1	(M|15|2)	1629.0	1	(H|15|2)	1803.5	1
(L|15|3)	266.1	0	(M|15|3)	105.6	0	(H|15|3)	853.1	1
(L|15|4)	1505.6	0	(M|15|4)	1887.1	1	(H|15|4)	1887.1	1
(L|15|5)	3343.1	0	(M|15|5)	4193.0	0	(H|15|5)	4172.1	0
(L|20|2)	20.1	1	(M|20|2)	20.1	1	(H|20|2)	20.1	1
(L|20|3)	0.0	0	(M|20|3)	0.0	0	(H|20|3)	0.0	0
(L|20|4)	22.3	2	(M|20|4)	0.0	1	(H|20|4)	0.0	1
(L|20|5)	269.2	1	(M|20|5)	669.1	0	(H|20|5)	1056.2	0
(L|25|2)	1569.2	7	(M|25|2)	1024.9	3	(H|25|2)	3546.1	4
(L|25|3)	0.0	1	(M|25|3)	0.0	0	(H|25|3)	0.0	0
(L|25|4)	1399.4	1	(M|25|4)	2896.9	1	(H|25|4)	3665.0	1
(L|25|5)	0.0	1	(M|25|5)	0.0	4	(H|25|5)	50.9	3

Table 8 The total trainer travel distance and number of trainer swaps for each roster produced.

The number of variables (columns), and constraints (rows) for the rostering IP model for each test case, subject to the final generated timetable is given in Table 7. The rostering IP model, subject to the final generated timetable has, on average, about 13% as many variables and about 9% as many constraints as the complete timetabling IP model for problem described the test case. Moreover, the complete timetabling IP model has about 16% more constraints than variables, whereas the rostering IP model has about 23% fewer constraints than variables. The dominant network flow structure of the rostering model, together with these substantial differences may explain, in part, why the rostering model is computationally tractable even for the largest of the test cases, while the timetabling model remains intractable for all but the smallest test cases.

The total distance travelled by trainers, in kilometres, and the total number of trainer swaps present in each of the produced rosters is shown in Table 8. Rosters

TestCase	Z1	Z2	Z3	DT	TS	Time	T-Ratio	TT-Opt	TT-Heur	R-Opt	R-Heur
(L|10|2)	0	-9455	42	209.7	1	48	0.52	-47233	-47233	1049.3	1049.3
(L|10|3)	0	-13991	56	1045.7	1	98	0.73	-69899	-69899	5229.4	5229.4
(L|15|2)	0	-13660	67	1454.5	1	171	0.41	-68233	-68094	7273.3	7564.2
(L|15|3)	0	-22053	87	371.7	0	597	0.35	-110178	-110178	1858.5	1330.4
(L|20|2)	0	-20576	89	20.1	1	1460	0.10	-102791	-102446	101.7	101.7
(L|25|2)	0	-20060	106	1390.3	5	8173	0.04	-100194	-100194	6956.3	7852.8
(M|10|2)	0	-7682	48	0.0	1	862	0.04	-38362	-38361	1.0	0.0
(M|10|3)	0	-10070	72	677.4	0	349	0.37	-50278	-50278	3387.0	1893.1
(M|15|2)	0	-10265	90	1745.3	1	1974	0.07	-51235	-49852	8727.7	8146.0
(M|15|3)	0	-20669	90	0.0	0	2865	0.11	-103255	-103255	0.0	528.1
(M|20|2)	0	-19439	90	20.1	1	1594	0.28	-97105	-97104	101.7	101.7
(H|10|2)	0	-5939	64	629.0	0	177	0.32	-29631	-29631	3145.0	2097.7
(H|10|3)	2	-8416	82	1643.2	1	35168	0.28	1958002	1959531	8217.0	8616.4
(H|15|2)	0	-9373	99	1745.3	1	909	0.22	-46766	-46592	8727.7	9018.6
(H|15|3)	0	-15270	124	743.4	0	81161	0.01	-76226	-76218	3716.9	4266.4
(H|20|2)	0	-13041	127	20.1	1	41349	0.01	-65078	-64735	101.7	101.7

Table 9 Results related to the optimal solutions for some of the smaller test cases.

which had zero trainer travel and swaps were found in 23 of the 48 test cases. There was an average of about 18km travelled and about 0.02 trainer swaps per location per day, which is consistent with Ausgrid's previous training rosters.

CPLEX was able to find optimal solutions to the full timetabling models described in Section 4 and rostering models described in Section 5 to 16 of the 48 test cases, and the results are compared with those from the three-stage heuristic in Table 9. The columns from left to right are the test case name, the optimal values for Z_1, Z_2, and Z_3 in the timetabling sub-problem, the optimal values for trainer travel distance (DT) and trainer swaps (TS) in the rostering sub-problem subject to the optimal timetable, the time required to find the optimal solution (Time), in seconds, the ratio of total time required by our heuristic to total time required for CPLEX to find the optimal solution (T-Ratio), the optimal objective value for the timetabling problem (TT-Opt), the objective value for the timetabling sub-problem our heuristic found (TT-Heur), and the optimal (R-Opt) and heuristic (R-Heur) values for the rostering sub-problem, subject to the optimal and heuristic timetables, respectively. Our heuristic was able to find the optimal timetable for 7 of the 16 cases. Since the rostering sub-problem is based on the produced timetable, different timetables yield a different set of feasible rosters and optimal solutions. The optimal roster, subject to the optimal timetable, had a worse objective value in five cases, had the same objective value in five cases, and a better objective value in six test cases.

Ausgrid does frequently engage in long-term strategic planning, where the purchase and sale of organisational resources must be investigated, specific trainer specialisations must be determined, and the structure of courses must be adjusted to meet the changing needs of the organisation and the requirements of changing safety laws. For long-term strategic planning purposes, having high quality timetables and rosters is of paramount importance to the organisation and the time requirements of the three-stage heuristic are within acceptable limits. It is a matter of ongoing research to further improve the process and reduce total computation time where possible.

8 Conclusions

In this paper we studied an academic timetabling and rostering problem involving periodic retraining of large numbers of employees at Ausgrid, an Australian electricity distributor. We developed an IP model to solve the timetabling of course instances, and an IP model to solve the rostering of trainers to an existing class timetable. Both models were developed so that they can deal with all the practical requirements in a flexible manner given the changing nature of the organisation and industry laws.

Due to the size of the problem, given Ausgrid's data, it was not possible to solve to optimality in practically acceptable time. A three-stage heuristic framework has been presented, which consists of an initial timetable generation stage, an iterative timetable improvement stage, and a trainer rostering stage. All three stages utilise the IP models that were developed to produce solutions. The computational results show that the proposed approach is able to generate solutions for the problem sizes typical for training at Ausgrid, and that the approach is effective for both operational and strategic planning purposes. The proposed LNS algorithm is easily generalisable to many other class timetabling problems from other institutions.

In the few cases where we were able to solve the timetabling sub-problem to optimality, the optimal rosters occasionally yielded poorer solutions compared with the optimal rosters to timetables produced by our heuristic. Even for different timetables with identical objective values, the optimal rosters can have very different objective values. This reinforces that solving each sub-problem to optimality does not always lead to a global optimum, which suggests an integrated model is worth investigating.

Future research can investigate different modelling techniques and related solution approaches to further reduce the required computation time. It can be investigated whether replacement of the first stage with a different heuristic or metaheuristic approach has a significant impact on solution time or quality. Others can adapt the introduced techniques to other problems and determine whether they are successful in other problem domains.

References

1. Akkoyunlu, E.A.: A linear algorithm for computing the optimum university timetable. The Computer Journal **16**(4), 347–350 (1973)
2. Balakrishnan, N., Wong, R.T.: A network model for the rotating workforce scheduling problem. Networks **20**(1), 25–42 (1990)
3. Birbas, T., Daskalaki, S., Housos, E.: Course and teacher scheduling in hellenic high schools. In: 4th Balkan Conference on Operational Research, Thessaloniki, Greece (1997)
4. Bölte, A., Thonemann, U.W.: Optimizing simulated annealing schedules with genetic programming. European Journal of Operational Research **92**(2), 402–416 (1996)
5. Burke, E., Elliman, D., Weare, R.: A university timetabling system based on graph colouring and constraint manipulation. Journal of research on computing in education **27**(1), 1–18 (1994)
6. Burke, E.K., Petrovic, S.: Recent research directions in automated timetabling. European Journal of Operational Research **140**(2), 266–280 (2002)

7. Carter, M.W., Laporte, G., Lee, S.Y.: Examination timetabling: Algorithmic strategies and applications. Journal of the Operational Research Society pp. 373–383 (1996)
8. Carter, M.W., Tovey, C.A.: When is the classroom assignment problem hard? Operations Research **40**(1-supplement-1), S28–S39 (1992)
9. Dantzig, G.B.: Letter to the editor-a comment on edie's traffic delays at toll booths. Journal of the Operations Research Society of America **2**(3), 339–341 (1954)
10. Dimopoulou, M., Miliotis, P.: Implementation of a university course and examination timetabling system. European Journal of Operational Research **130**(1), 202–213 (2001)
11. Dorneles, Á.P., de Araújo, O.C., Buriol, L.S.: A fix-and-optimize heuristic for the high school timetabling problem. Computers & Operations Research **52**, 29–38 (2014)
12. Ernst, A.T., Jiang, H., Krishnamoorthy, M., Sier, D.: Staff scheduling and rostering: A review of applications, methods and models. European journal of operational research **153**(1), 3–27 (2004)
13. Fischetti, M., Widmayer, P.: Towards solving very large scale train timetabling problems by lagrangian relaxation. In: 8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS08), vol. 9 (2008)
14. Gunawan, A., Ng, K., Poh, K.: A hybrid algorithm for the university course timetabling problem. In: Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling (2008)
15. IBM: IBM CPLEX Optimizer. `http://www.ibm.com/software/commerce/optimization/cplex-optimizer/` (2014)
16. IBM: IBM User's Manual for CPLEX. `ftp://public.dhe.ibm.com/software/websphere/ilog/docs/optimization/cplex/ps\_usrmancplex.pdf` (2015)
17. Kahar, M.M., Kendall, G.: The examination timetabling problem at universiti malaysia pahang: Comparison of a constructive heuristic with an existing software solution. European journal of operational research **207**(2), 557–565 (2010)
18. Karp, R.M.: Reducibility among combinatorial problems. Springer (1972)
19. Lawrie, N.L.: An integer linear programming model of a school timetabling problem. The Computer Journal **12**(4), 307–316 (1969)
20. Lewis, R.: A survey of metaheuristic-based techniques for university timetabling problems. OR spectrum **30**(1), 167–190 (2008)
21. Marx, D.: Graph colouring problems and their applications in scheduling. Electrical Engineering **48**(1-2), 11–16 (2004)
22. Mehta, N.K.: The application of a graph coloring method to an examination scheduling problem. Interfaces **11**(5), 57–65 (1981)
23. Minerals Council of Australia: Training and Education Activity in the Minerals Sector. `http://www.minerals.org.au/file_upload/files/reports/Final_Report_Minerals_Council_2013.pdf` (2013)
24. Neufeld, G., Tartar, J.: Graph coloring conditions for the existence of solutions to the timetable problem. Communications of the ACM **17**(8), 450–453 (1974)
25. Papoutsis, K., Valouxis, C., Housos, E.: A column generation approach for the timetabling problem of greek high schools. Journal of the Operational Research Society **54**(3), 230–238 (2003)
26. Pillay, N.: A survey of school timetabling research. Annals of Operations Research **218**(1), 261–293 (2014)
27. Qualizza, A., Serafini, P.: A column generation scheme for faculty timetabling. In: Practice and theory of automated timetabling V, pp. 161–173. Springer (2005)
28. Schimmelpfeng, K., Helber, S.: Application of a real-world university-course timetabling model solved by integer programming. Or Spectrum **29**(4), 783–803 (2007)
29. Ülker, Ö., Özcan, E., Korkmaz, E.E.: Linear linkage encoding in grouping problems: applications on graph coloring and timetabling. In: Practice and Theory of Automated Timetabling VI, pp. 347–363. Springer (2007)
30. de Werra, D.: An introduction to timetabling. European Journal of Operational Research **19**(2), 151–162 (1985)
31. Yao, X.: A new simulated annealing algorithm. International Journal of Computer Mathematics **56**(3-4), 161–168 (1995)