

A Case Study of an Integer Programming Model for Instructor Assignments and Scheduling Problem

Eddie Cheng, Serge Kruk¹

Department of Mathematics and Statistics, Oakland University, Rochester, MI 48309, USA,
{echeng, kruk}@oakland.edu

We express an integer program formulation of the rostering problem academic departments face every semester. The goal is to model the real situation, with all the relevant exceptions and peculiarities and to optimize the instructor satisfaction level with their timetable. We introduce *Preference Sets* and *Preference Pairs* as the main abstraction on which the model is based. Abstraction that can be used to allow the subjective satisfaction level of instructors to be modeled satisfactorily. We describe an instance in some detail and comment on the surprisingly short solving time required by modern integer program solvers for a type of problem that, until recently, was considered too hard for this type of formulation. The Department of Mathematics and Statistics at Oakland University has adopted this formulation since 2006.

Keywords: Rostering, Case-study, Model.

1 Introduction

Timetabling course sections, students, rooms and instructors in higher education is an area that attracted much research. Every two years the Conference on the Practice and Theory of Automated Timetabling (www.asap.cs.nott.ac.uk/patat/patat-index.shtml) [6, 3, 5, 4, 8, 7] will see a number of papers on the subject. [2, 14] give recent research directions in this area and [13] argues very convincingly, by combing the publications of the last decades, that a large gap exists between the theory and the actual implementations used in institutions. Some recent case studies include [12, 10, 9, 11]. In such a timetabling problem, the emphasis is on scheduling courses to satisfy room constraints, sometimes some student constraints are included, but in almost all cases instructors' concern is treated only very partly if it is treated at all. For example, the authors of [9, 11] have an approach similar to ours, but only attempts at satisfying a request for a specific time period or course while satisfying other timetabling constraints. We will model many more types of requests since we are not concerned with room and time assignments. The authors of [1] model the faculty satisfaction problem in a manner similar to ours but their criteria are somewhat different and some of them are simpler.

At some universities, the individual unit, such as the Department of Mathematics and Statistics at Oakland University, has little or no control over the room assignment and has limited control over scheduling of section time. For example, one cannot change the section time of Calculus without considering section time of non-math sections such as engineering sections as most of the students in Calculus are engineering majors. Hence any timetabling project aimed at some form of global optimization will be interdepartmental. This, usually, means such project will not get started and timetabling is done piecemeal, often by hand.

In this setting, at Oakland University, the limited responsibility of Department of Mathematics and Statistics (DMS) is reduced to assigning courses to its faculty members. This task is traditionally performed by the associate chair manually. Due to a recent administrative realignment of

resources, the position of associate chair was eliminated. We approached the chair of the department and suggested that we cast the problem as an optimization problem where the objective is to maximize the “faculty happiness.” So this optimization task is to find teaching schedule for instructors. Given that the problem no longer involves timetabling of courses, a considerable amount of effort can be spent on maximizing such “faculty happiness.” This is now the primary focus rather than a secondary focus. We can now include many desires of the instructors into the model.

Somewhat more precisely, the problem is: given a number of courses with fixed credit values (typically 1 to 5), each with possibly multiple sections (1 to 10) meeting at set times of the week and given a number of instructors with fixed teaching loads in measured in number of credits (typically 0 to 9) and with subjective preferences for courses, for days of the week, for times of day, assign instructors to courses to maximize the satisfaction of the subjective preferences.

As in a typical operations research project, the first phase is to test our formulation; this is done by trial runs in assigning courses for Fall 2005 and Winter 2006. We then gather suggestions from faculty members to refine our formulation. It turns out that for most faculty members, their main concern is the teaching time as oppose to the actual courses. In particular, almost all prefer not to have a 5-day schedule so that they can devote at least one full day for research. Moreover, they like the teaching schedule to complement their service schedule. Finally, we implemented the new formulation in assigning courses for Fall 2006 and Winter 2007. This paper is not theoretical in nature, and no theoretical result is offered. This is a case study in using operations research to solve a real problem that has received little attention in the past. Moreover, the model we offer can be included into a more general timetabling model to allow some consideration of the satisfaction of instructors in addition to the usual time and space constraints.

2 Problem to solve

Every semester at Oakland University, each department is faced with the problem of assigning instructors to sections of courses, which we will, henceforth call *sections*. These sections have already assigned time-slots. Rooms will be later assigned by the registrar’s office to account for enrollment. The available timeslots are divided into two main categories. Monday-Wednesday-Friday schedule and Tuesday-Thursday schedule. The first meet only on Monday, Wednesday and Friday and the second on Tuesday and Thursday. Table 1 exemplifies some of the course offerings in Fall 2006.

MTH154	Section 1	MWF 1040 to 1147
MTH154	Section 2	MWF 1200 to 1307
MTH154	Section 3	MWF 1320 to 1427
MTH154	Section 4	TR 1730 to 1917
MTH155	Section 1	MWF 0800 to 0907
MTH155	Section 2	MWF 1040 to 1147
MTH155	Section 3	TR 1730 to 1917

Table 1: Course Offering

In this example, MTH 154 Calculus I has four sections and MTH 155 Calculus II has three sections. This set of all sections can be partitioned into subsets where each subset corresponds to a course. So the subset MTH 154 is of size 4 and the subset MTH 155 is of size 3. The complete list of timeslots is given in Table 2.

1	MWF	0800	0907	2	MWF	0920	1027	3	MWF	1040	1147
4	MWF	1200	1307	5	MWF	1320	1427	6	MWF	1440	1547
7	MW	1530	1717	8	TR	0800	0947	9	TR	1000	1147
10	TR	1300	1447	11	TR	1530	1717	12	MW	1730	1917
13	TR	1730	1917	14	MW	1930	2117	15	TR	1930	2117

Table 2: Timeslots

In our department, we have about twenty-eight full-time faculty members. However, due to sabbatical leaves, usually about twenty-five are active in any given semester. A number of sections are covered by adjunct faculty members and graduate students. Each regular member typically teaches two courses per semester. The traditional procedure is for each faculty member to send the associate chair a list of his/her favourite courses and a list of courses he/she want to stay away from. The faculty member also indicates his/her time-of-day and day-of-week preferences. For example “only afternoons” or “only Monday-Wednesday-Friday” sections. Historically, a number of variations on these preferences were expressed. From these the associate chair constructs a schedule by assigning instructors to sections by hand. Often such an assignment is obtained from modifying the teaching schedule from the year before.

Our job was to automate this process and improve upon it. Part of the work involved developing the database manager and access code to enter the preferences. We will not discuss this aspect of the project here but will rather concentrate on the operations research aspect of the work, the development of an integer linear programming model to automate the assignment. To improve upon this process, the objective function is to maximize faculty *satisfaction* (a fuzzy concept to be defined later). Specifically, we address the following issues that we deem deficient in the traditional process.

1. There is neither ordinal nor cardinal relationship between a course preference and a time-slot preference.
2. The relationship in the list of favourite courses for an instructor is only ordinal and not cardinal.
3. Although modifying the teaching schedule from the year before produces a good schedule, some instructors are pigeonholed into a couple of courses.

3 Integer linear model

3.1 Preferences and input data

The input is a list of sections (a section, is an instance of a course, meeting at specific times of the week, with a value of a certain number of credits) and a list of instructors (each with a list of preferences) and a charge of a certain number of credits to teach.

Each instructor indicates his preferences via weights attached to sections in various direct and indirect ways. In the most direct manner, preferences provided by each instructor include a weight for each course. Note that there may be more than one section of a given course. The instructor, in this first list, is indicating his preference for subject matter, not time of the day or day of the week.

We model this first list of preference as a weight $wc_i^c \in \mathbb{R}$, for each section c , for instructor i . We allocate the weight given to a course to each of its sections. Note that preferences can be positive,

indicating an attraction for the given section or negative, indicating a repulsion. The absence of a preference for section c will imply $wc_i^c = 0$, a “don’t care” indication.

To model time preferences of instructor, we introduce the concept of *preference sets*, S_j for $j \in \{0, \dots, N_{PS}\}$ in the model, where N_{PS} is the number of preference sets. Those are subsets of all sections related by time. Typical instances of these preference sets are enumerated in the Table 3. For example, all the sections that have a meeting at 8 in the morning are in S_0 . So MTH 154 Section 1 is in S_1 and S_4 only. Instructor i indicates a preference for each S_j and the weight $ws_i^j \in \mathbb{R}$ is distributed to every section in S_j .

j	S_j
0	0800h sections
1	Morning sections (0800h – 1159h)
2	Afternoon sections (1200h – 1729h)
3	Night sections (1730h – 2100h)
4	Monday-Wednesday-Friday sections
5	Tuesday-Thursday sections
6	Friday sections
7	1930h sections

Table 3: Preference sets.

To model more complex requests, we introduce *preference pairs*, P_j for $j \in \{0, \dots, N_{PP}\}$ sets of pairs of sections, which we illustrate in the Table 4. (Again, N_{PP} denotes the number of preference pairs.) For example, the pair (MTH 154 Section 4, MTH 155 Section 1) is in S_1 and the pair (MTH 155 Section 1, MTH 155 Section 3) is in S_0 . Instructor i indicates a preference for each P_i and its weight ($wp_i^j \in \mathbb{R}$) is distributed to every element of P_j .

j	P_j
0	Morning and night sections on same day
1	Night section of one day followed by morning section of the next day
2	Monday and Tuesday sections
3	Consecutive sections
4	Three consecutive timeslots
5	A class then free timeslot, then a class

Table 4: Preference pairs.

Notice that the three categories of user weights, the course preference weights, the set preference weights and the pairs preference weights must satisfy the following for each instructor $i \in I$,

$$\sum_{c \in C} |wc_i^c| + \sum_{j=0}^{N_{PS}} |ws_i^j| + \sum_{j=0}^{N_{PP}} |wp_i^j| = 1.$$

This has a normalizing effect on the weights of all instructors. (The users are not required to satisfy this constraint manually when entering their preferences, of course. We simply scale the weights as we generate the integer program.)

3.2 Decision variables

The main index sets are C, I and T . The complete list of sections, C is partitioned into k subsets C_1, \dots, C_k each representing the sections of a given course. The set of instructors is I and the set of possible timeslots is T .

The decision variable is

$$x_{ic} = \begin{cases} 1, & \text{if instructor } i \in I \text{ is in section } c \in C \\ 0, & \text{otherwise.} \end{cases}$$

To simplify some expressions, we introduce two indicator variables. This is done at no computational cost since they are simple sums of the decision variables. From (1a), we see that z_c , when non-zero, indicates whether section c is assigned to any instructor. From (1b), we see that y_{it} indicates whether an instructor is busy at a given time.

$$z_c := \sum_{i \in I} x_{ic} \quad \forall c \in C \quad (1a)$$

$$y_{it} := \sum_{c \in X_t} x_{ic} \quad \forall i \in I, \forall t \in T \quad (1b)$$

where X_t is the set of all the sections in timeslot t .

3.3 Hard constraints

3.3.1 One instructor per section

Each section needs only one instructor.

$$\sum_i x_{ic} \leq 1 \quad \forall c \in C. \quad (2)$$

3.3.2 Course leaders

Some multi-section courses are handled by adjunct (which we do not schedule) with the exception that one faculty member must be leader and teach one of the sections. This is handled by extracting subsets of the sections C_j for $j \in I^L$, the index set of the partitions requiring leaders. The model constraint is then

$$\sum_{c \in C_j} z_c \geq 1 \quad \forall j \in I^L. \quad (3)$$

3.3.3 Load constraint

Each instructor has to teach a certain total number of credits indicated by the lower bound L_i and the upper bound U_i . Each course has a certain number of credits indicated by M_c . The model constraints to ensure that the exact number of credit is met for each instructor is

$$L_i \leq \sum_c M_c x_{ic} \leq U_i \quad \forall i \in I \quad (4)$$

3.3.4 No cloning constraint

Each instructor can lecture only one section during a given timeslot.

$$y_{it} \leq 1 \quad \forall i \in I, \forall t \in T. \quad (5)$$

But unfortunately, our schedule includes some overlapping timeslots: a 1440h-1547h as well as a 1530h-1717h. To take this into consideration, we construct a set of pairs of overlapping timeslots O and model the constraint as

$$x_{ic_1} + x_{ic_2} \leq 1 \quad \forall i \in I, \forall (c_1, c_2) \in O. \quad (6)$$

3.3.5 Incompetence constraint

Preventing certain instructors from being assigned certain courses is of course, trivial. Assuming instructor i is deemed incompetent to teach section c , we generate the constraint

$$x_{ic} = 0. \quad (7)$$

3.4 Soft Constraints

Since the major goal of the model is to maximize instructor satisfaction, understood as giving them, as far as possible, both the courses they want and the times they want, the following constraints are part of the objective function.

3.4.1 Preference set

First, we introduce a non-negative variable r_i^j defined by

$$r_i^j = \sum_{s \in S_j} y_{is} \quad \forall i \in I, j \in \{0, \dots, N_{PS}\}. \quad (8)$$

Note that r_i^j counts the number of instances where instructor i is assigned a section in preference set S_j . Assuming that the instructor allocates a weight ws_i^j to a preference set S_j , the following term $ws_i^j \times r_i^j$ in a maximizing objective function will try to increase r_i^j if the weight is positive and reduce it to 0 if the weight is negative.

3.4.2 Preference pair

The preference pairs are handled similarly. First, we introduce non-negative variables rr_{kl}^j and rrr_i^j . The former, as seen in (9a)-(9c), indicates whether sections k and l of preference pair P_j are assigned to instructor i .

$$y_{ik} + y_{il} - rr_{ikl}^j \leq 1 \quad \forall i \in I, \forall (k, l) \in P_j, \forall j \in \{0, \dots, N_{PP}\}, \quad (9a)$$

$$rr_{ikl}^j \leq y_{ik} \quad \forall i \in I, \forall (k, l) \in P_j, \forall j \in \{0, \dots, N_{PP}\}, \quad (9b)$$

$$rr_{ikl}^j \leq y_{il} \quad \forall i \in I, \forall (k, l) \in P_j, \forall j \in \{0, \dots, N_{PP}\}, \quad (9c)$$

$$rrr_i^j = \sum_{(k,l) \in P_j} rr_{ikl}^j \quad \forall i \in I, \forall j \in \{0, \dots, N_{PP}\}. \quad (9d)$$

The latter, as seen in (9d), counts the number of such assignments correspond to preference pair P_j . If the instructor allocates a weight wp_i^j to a set of preference pairs P_j , we then add the following term $wp_i^j \times rrr_i^j$ into the objective function

3.5 Objective function

Since covering our upper-level courses is a department priority, we have a weight wf_c of non-negative value for each course — with service courses having value 0. This allows some flexibility in the assignation of courses, independently of the preferences of the faculty members. This also potentially leaves lower-level courses open for to adjunct which we schedule later, when we know exactly which sections need to be covered.

The goal is to maximize satisfaction of faculty preferences and department preferences, modeled as

$$\max \sum_{i \in I} \sum_{c \in C} x_{ic} w_c^c + \sum_{i \in I} \sum_{j=0}^{N_{PS}} w s_i^j r_i^j + \sum_{i \in I} \sum_{j=0}^{N_{PP}} w p_i^j r r r_i^j - \sum_{c \in C} w f_c (1 - z_c) \quad (10)$$

4 Experimental results

The model was written in GNU Mathprog² and has been used on a real schedule for the first time during the Fall semester 2005 and every semester since then. The complete model, as well as some typical datasets are available at <http://homepage.oakland.edu/~kruk/research>. In a typical semester, the number of instructors is almost 30, the number of sections is a little below 100. Recall that we do not allocate an instructor to all sections as the left-over positions will be manned by part-timers and graduate students. The resulting number of variables in a typical instance is around 5000 with about half being integer and the number of constraints hovers just under 10000. Even on an old Pentium computer, most instances are solved in seconds or minutes with all the solvers tried (CPLEX³, Cbc⁴ and glpk⁵).

There are notable variations in difficulty from one semester to the next. In CPLEX, the presolver will eliminate between 10% and 30% of the rows and columns. The branch and bound code will never solve at the root node but will visit from 10 to 1000 nodes with the gap being reduced from 5% down to zero; so that even the first integer solution is likely to be acceptable in practice if the solution time to prove optimality ever became a problem. Most of the effective cuts are generic Gomory fractional cuts, on the order of a few dozens, with some cover cuts in addition, for some instances.

5 Conclusion

By the subjective reaction of the chair of the department and our colleagues, the implementation of this system has been a success. This is in contrast to the usual litany of complaints following a hand-crafted schedule. This is not entirely surprising since preferences, even if they were known with certainty, were handled in a ad-hoc fashion, hence often forgotten. More to the point, with an interface allowing each instructor to put weights on each combination of courses and times, the true relative importance of each choice was clearly spelled out and taken into consideration. Some instructors in a certain semester want a specific course with no concern for times; during other semesters, the time preferences trump any choice of courses. The model allows complete flexibility in this manner. Even more telling is that, after the initial run, we were asked to add a few more preferences (related to two sections during three consecutive timeslots), which the model accommodates trivially, suggesting that it is fairly robust to change. The program is now adopted by the department as a permanent procedure.

That all solvers solve the problem very rapidly is somewhat surprising considering the number of integer variables and constraints. This is partly due to the advances in general-purpose solvers, undoubtedly, but not exclusively. All the solvers produce different solutions every semester. This multiplicity of optimal solutions plays to our advantage and indicates that a two-phase approach to the university timetabling problem, with the first phase solely concerned with students and

²<http://www.gnu.org/software/glpk/glpk.html>

³<http://www.ilog.com/products/cplex/>

⁴<http://www.coin-or.org/Cbc/index.html>

⁵<http://www.gnu.org/software/glpk/>

room requirements while the second caters to instructor's preference is a natural and efficient decomposition of the problem. This may be the more interesting result of the experiment since it suggests that adding a varied and flexible list of instructor preferences to a timetabling model will likely not unduly affect the solving time.

6 Acknowledgements

The authors would like to thank the anonymous referees for valuable comments and for pointing out some pertinent case studies.

References

- [1] S.M. Al-Yakoob and H.D. Sherali (2006) Mathematical programming models and algorithms for a class-faculty assignment problem. *European J. Oper. Res.* **2**(173), 488 – 507.
- [2] E. Burke and S. Petrovic (2002) Recent research directions in automated timetabling. *Eur. J. Oper. Res.* **140**(2), 266 – 280.
- [3] E.K. Burke and M.W. Carter, editors (1998) The Practice and Theory of Automated Timetabling II: Selected papers from the 2nd International conference, *Lecture Notes in Computer Science* **1408**, Springer.
- [4] E.K. Burke and P. De Causmaecker, editors (2003) The Practice and Theory of Automated Timetabling IV: Selected papers from the 4th International conference, *Lecture Notes in Computer Science* **2740**, Springer.
- [5] E.K. Burke and W. Erben, editors (2000) The Practice and Theory of Automated Timetabling III: Selected papers from the 3rd International conference, *Lecture Notes in Computer Science*, **2079**, Springer.
- [6] E.K. Burke and P. Ross, editors (1996) The Practice and Theory of Automated Timetabling: Selected papers from the 1st International conference, *Lecture Notes in Computer Science* **1153**, Springer.
- [7] E.K. Burke and H. Rudová, editors (2006) The Practice and Theory of Automated Timetabling VI: Selected papers from the 6th International conference, ISBN 80-210-3726-1, Brno, Czech Republic, Masaryk University.
- [8] E.K. Burke and M. Trick, editors (2005) The Practice and Theory of Automated Timetabling V: Selected papers from the 5th International conference, *Lecture Notes in Computer Science* **3616**, Springer.
- [9] S. Daskalaki, T. Birbas, and E. Housos (2004) An integer programming formulation for a case study in university timetabling, *European J. Oper. Res.* **153**(1), 117 – 135.
- [10] T. Hinkin and G. Thompson (2002) Schedulexpert: Scheduling courses in the cornell university school of hotel administration, *Interfaces*, **32**(6), 45 – 57.
- [11] Badri M. A. and Davis D. L., Davis D. F., and Hollingsworth J (1998) A multi-objective course scheduling model : Combining faculty preferences for courses and times, *Computers and operations research*, **25**(4), 303 – 316.

- [12] C. Martin (2004) Ohio university's college of business uses integer programming to schedule classes, *Interfaces*, **34**(6), 460 – 465.
- [13] Barry McCollum (2006), University timetabling: Bridging the gap between research and practice, *Burke and Rudová* [7], 15 – 35.
- [14] A. Schaefer and L. Di Gaspero (2006), Timetabling research: State-of-the-art and discussion, *Burke and Rudová* [7], 52 – 62.