Penalising Patterns in Timetables: Novel Integer Programming Formulations

Edmund K. Burke[1], Jakub Mareček[12†], Andrew J. Parkes[1], and Hana Rudová[2]

[1] Automated Scheduling, Optimisation and Planning Group
The University of Nottingham School of Computer Science and IT
Jubilee Campus in Wollaton Road, Nottingham NG8 1BB, UK

[2] Masaryk University Faculty of Informatics
Botanická 68a, Brno 602 00, The Czech Republic

1 Introduction

Many complex timetabling problems, such as employee rostering [1] and university course timetabling [2, 3], have an underpinning bounded graph colouring component, a pattern penalisation component and a number of side constraints. The bounded graph colouring component corresponds to hard constraints such as "each student attends all events of courses of his choice", "no student can be in two rooms at the same time", or "there can be at most a given number of events taking place during each period". Despite the intractability and hardness of approximation of bounded graph colouring, it is often easy to generate feasible solutions for instances with hundreds of events and hundreds of distinct enrolments. See [4] for more details. However, real-world timetabling systems [5] have to cope with much more challenging requirements, such as "students should not have gaps in their individual daily timetables", which often make the problem over-constrained. The key to tackling this challenge is a suitable formulation of "soft" constraints, which count and minimise penalties incurred by matches of various patterns. Several integer programming formulations of such pattern penalising constraints are presented and discussed in this paper.

[†] Corresponding author: Jakub Mareček, e-mail: jakub@marecek.cz.

2 Udine Course Timetabling

Throughout the paper, the Udine Course Timetabling Problem is used as an illustrative example of timetabling with soft constraints. The problem has been formulated by Schaerf and Di Gaspero [6, 7] at the University of Udine. Its input can be outlined as follows:

- C, T, R, D, P are sets representing courses, teachers, rooms, days, and periods, respectively
- U is a set representing distinct enrolments in courses ("curricula"), with Inc being the mapping from curricula to non-disjunctive subsets of C
- F is a set of pairs $\langle c, p \rangle \in C \times P$, representing deprecated periods p of courses c
- HasEC maps courses to numbers of weekly unit-length events
- HasStud maps courses to numbers of enrolled students
- HasMinD maps courses to recommended minimum numbers of days of instruction per week
- Teaches maps teachers to disjunctive sets of elements of C
- HasCap maps rooms to their capacity
- HasP maps days to tuples of corresponding periods in ascending order.

Given this input, the task is to assign events to rooms and periods so that:

- for each course, HasEC[c] events are timetabled
- no two events take place in the same room at the same period
- no two events of one course are timetabled at the same period
- events of no two courses in a single curriculum are taught at the same time
- events of no two courses taught by a single teacher are timetabled at the same period
- for all $\langle c, p \rangle \in F$, events of course c are not taught at period p
- the number of students left without a seat, summed across all events, is minimised with weight 1
- the number of events timetabled for a curriculum outside of a single consecutive block of two or more events per day, summed across all curricula, is minimised with weight 2
- the number of missing days of instruction, summed across all courses, is minimised with weight 5.

At the heart of most timetabling and rostering problems, including the Udine problem, there are multiple simultaneous `all_different` constraints. The problem of finding a bi-partite matching satisfying multiple simultaneous `all_different` constraints is closely related to the Graph Colouring Problem [4]. In integer programming, most researchers [8, for example] study a natural assignment-type formulation, although some focus also on a binary encoded formulation [9], a scheduling formulation [10], and four other distinct formulations. A brief survey can be seen in [11]. For timetabling applications, a clique-based formulation has recently been proposed [11]: assuming there are courses with multiple events per week, it is possible to use array T of binary decision variables indexed with periods, rooms and courses. Notice the difference between courses and events. $T[p, r, c]$ being set to one indicates course c is being taught in room r at period p. The corresponding hard constraints are:

$$\forall c \in C \sum_{p \in P} \sum_{r \in R} T[p, r, c] = \text{HasEC}[c] \tag{1}$$

$$\forall p \in P, \forall r \in R \sum_{c \in C} T[p, r, c] \leq 1 \tag{2}$$

$$\forall p \in P, \forall c \in C \sum_{r \in R} T[p, r, c] \leq 1 \tag{3}$$

$$\forall p \in P, \forall t \in T \sum_{r \in R} \sum_{c \in \text{Teaches}[t]} T[p, r, c] \leq 1 \tag{4}$$

$$\forall p \in P, \forall u \in U \sum_{r \in R} \sum_{c \in \text{Inc}[u]} T[p, r, c] \leq 1 \tag{5}$$

$$\forall \langle c, p \rangle \in F \sum_{r \in R} T[p, r, c] = 0 \tag{6}$$

Soft constraints in timetabling problems vary widely from institution to institution [12], but most notably penalise patterns in timetables [13]. Their integer programming formulations, although often crucial for the performance of the model, are still largely unexplored. Although instances of up to two hundred events with dozens of distinct enrolments are now being solved to optimum almost routinely [14], larger instances are still approached only via heuristics.

Out of the three soft constraints in the Udine Course Timetabling problem, the minimisation of the number of students left without a seat can be formulated using a single term in the objective function:

$$\sum_{r \in R} \sum_{p \in P} \sum_{\substack{c \in C \\ \text{HasStud}[c] > \\ \text{HasCap}[r]}} T[p, r, c] \left(\text{HasStud}[c] - \text{HasCap}[r] \right) .$$

The second soft constraint, minimising the number of missing days of instruction summed across all courses, can be formulated using two auxiliary arrays of decision variables. The first binary array, CTT, is indexed with courses and days. $\mathrm{CTT}[c, d]$ being set to one indicates there are some events of course c held on day d. The other array of integers, Miss, is indexed with courses. The value of $\mathrm{Miss}[c]$ is bounded below by zero and above by the number of days in a week and represents the number of days course c is short of its recommended days of instruction. This enables addition of the following constraints:

$$\forall_{\substack{c\in C\\ d\in D\\ p\in \mathrm{HasP}[d]}} \sum_{r\in R} T[p, r, c] \leq \mathrm{CTT}[c, d] \tag{7}$$

$$\forall_{\substack{c\in C\\ d\in D}} \sum_{r\in R} \sum_{p\in \mathrm{HasP}[d]} T[p, r, c] \geq \mathrm{CTT}[c, d] \tag{8}$$

$$\forall_{c\in C} \sum_{d\in D} \mathrm{CTT}[c, d] \geq \mathrm{HasMinD}[c] - \mathrm{Miss}[c] \tag{9}$$

The term $5 \sum_{c\in C} \mathrm{Miss}[c]$ can then be added to the objective function.

3 Pattern Penalisation by Feature

The natural formulation of penalisation of patterns of classes and free periods in daily timetables for curricula goes "feature by feature", where feature is described relatively to a particular position in a daily timetable, for instance as "a class in period one with period two free". This formulation then uses an auxiliary binary array M, indexed with curricula, days and features, where $M[u, d, f]$ being set to one indicates feature f is present in the timetable for curriculum u and day d. The number of features to check, $|\mathrm{Check}|$, obviously depends on the number of periods per day. In the case of four periods per day, we have the following constraints:

$$\forall_{u\in U, d\in D, \forall \langle p_1, p_2, p_3, p_4\rangle \in \mathrm{HasP}[d]}$$
$$\sum_{c\in \mathrm{Inc}[u]} \sum_{r\in R} (T[p_1, r, c] - T[p_2, r, c]) \leq M[u, d, 1] \tag{10}$$

$$\forall_{u\in U, d\in D, \forall \langle p_1, p_2, p_3, p_4\rangle \in \mathrm{HasP}[d]}$$
$$\sum_{c\in \mathrm{Inc}[u]} \sum_{r\in R} (T[p_4, r, c] - T[p_3, r, c]) \leq M[u, d, 2] \tag{11}$$

$$\forall u{\in}U, d{\in}D, \forall\langle p_1,p_2,p_3,p_4\rangle{\in}\mathrm{HasP}[d]$$
$$\sum_{c\in\mathrm{Inc}[u]}\sum_{r\in R}(T[p_2,r,c]-T[p_1,r,c]-T[p_3,r,c]) \leq M[u,d,3] \quad (12)$$

$$\forall u{\in}U, d{\in}D, \forall\langle p_1,p_2,p_3,p_4\rangle{\in}\mathrm{HasP}[d]$$
$$\sum_{c\in\mathrm{Inc}[u]}\sum_{r\in R}(T[p_3,r,c]-T[p_2,r,c]-T[p_4,r,c]) \leq M[u,d,4]\ . \quad (13)$$

The third term in the objective function is then

$$2\sum_{u\in U}\sum_{d\in D}\sum_{s\in\mathrm{Check}} M[u,d,s]\ . \quad (14)$$

This formulation, referred to as T (for "traditional") below, leaves plenty of room for improvement in terms of performance.

4 Pattern Penalisation by Enumeration

Considerable improvement in the performance of pattern penalisation can be gained by introducing the concept of the enumeration of patterns. It is obviously possible to pre-compute a set B of $n+2$ tuples $w, x_1, \ldots, x_n, m$, where n is the number of periods per day, x_i is one if there is instruction in period i of the daily pattern and minus one otherwise, w is the penalty attached to the pattern, and m is the sum of positive values x_i in the patterns decremented by one. There are several applications of this concept.

In a purely enumerative Formulation E, the array M is replaced with an array W indexed with curricula and days, and constraints (10)–(13) are (in the case of four periods per day) replaced with:

$$\forall\langle w,x_1,x_2,x_3,x_4,m\rangle{\in}B\ \forall u{\in}U\ \forall d{\in}D\ \forall\langle p_1,p_2,p_3,p_4\rangle{\in}\mathrm{HasP}[d]$$
$$\begin{aligned} w\ (x_1 \sum_{c\in\mathrm{Inc}[u]}\sum_{r\in R} T[p_1,r,c] + x_2 \sum_{c\in\mathrm{Inc}[u]}\sum_{r\in R} T[p_2,r,c] \\ + x_3 \sum_{c\in\mathrm{Inc}[u]}\sum_{r\in R} T[p_3,r,c] + x_4 \sum_{c\in\mathrm{Inc}[u]}\sum_{r\in R} T[p_4,r,c] - m) \\ \leq W[u,d]\ . \quad (15)\end{aligned}$$

The corresponding Term 14 in the objective function is then replaced with $2\sum_{u\in U}\sum_{d\in D} W[u,d]$.

In an alternative Formulation ET, both arrays M and W are used, together with constraints (10)-(15). Term 14 in the objective function can then be replaced with:

$$\sum_{u \in U} \sum_{d \in D} \sum_{s \in \text{Check}} M[u,d,s] + \sum_{u \in U} \sum_{d \in D} W[u,d] \ . \tag{16}$$

This Term 16 could perhaps better guide the search than either of the terms involving only M or only W.

In yet another Formulation ETP, Formulation ET is strengthened using constraints:

$$\sum_{u \in U} \sum_{d \in D} W[u,d] - \sum_{u \in U} \sum_{d \in D} \sum_{s \in \text{Check}} M[u,d,s] = 0 \tag{17}$$

$$\forall u \in U \sum_{d \in D} W[u,d] - \sum_{d \in D} \sum_{s \in \text{Check}} M[u,d,s] = 0 \tag{18}$$

$$\substack{\forall u \in U \\ \forall d \in D} W[u,d] - \sum_{s \in \text{Check}} M[u,d,s] = 0 \ . \tag{19}$$

Finally in Formulation TP, the original Formulation T could be strengthened using constraints resembling (15), whose right-hand side is replaced with $\sum_{s \in \text{Check}} M[u,d,s]$.

5 Empirical Results

The five formulations, together with Formulation C of the decision version of graph colouring, have been encoded in Zimpl [15] and tested on four real-life instances from the University of Udine School of Engineering [6] and 18 semi-randomly generated instances, available from the authors' website[3]. The results in Table 1 have been obtained with SCIP 0.82 using SoPlex, the present-best freely-available integer programming solver from Berlin [16], running on Linux-based Sun V20z with dual Opteron 248 and 2 GB of memory. Notice that all constraints were given explicitly in these experiments, although extended formulations seem to promise considerably better performance when the additional constraints are added only dynamically in a branch-and-cut algorithm. Judging from less systematic experiments using larger instances and ILOG CPLEX version 10.01, the performance gains are the more pronounced, the larger and the denser the instances are.

[3] `http://cs.nott.ac.uk/~jxm/timetabling`

Table 1. Performance of five formulations of pattern penalisation: instance name, number of events, occupancy in percent, and either the run time needed to reach the optimality, or the gap remaining after two hours of solving. A blank indicates no feasible solution has been found

Instance	Ev.	Occ.	C	T	TP	E	ET	ETP
udine1	207	86	(2 s)	(1417 s)	(1231 s)	500.00 %	(655 s)	(916 s)
udine2	223	93	(10 s)	42/0	49/0		44/0	48/0
udine3	252	97	(25 s)	638.21 %	470.54 %			755.46 %
udine4	250	100	(28 s)	4800.00 %	4500.00 %			
rand1	100	70	(3 s)	(345 s)	(434 s)	37/0	(1134 s)	(567 s)
rand2	100	70	(2 s)	(888 s)	(610 s)	0.28 %	(615 s)	0.02 %
rand3	100	70	(1 s)	(561 s)	(440 s)	0.52 %	(751 s)	(1228 s)
rand4	200	70	(21 s)	1.62 %	2.25 %		2.25 %	0.50 %
rand5	200	70	(30 s)	2.89 %	0.11 %		0.03 %	0.31 %
rand6	200	70	(25 s)	0.57 %	0.82 %			0.79 %
rand7	300	70	(213 s)	0.63 %				
rand8	300	70	(132 s)	8.76 %				
rand9	300	70	(113 s)	1.22 %				
rand10	100	80	(0 s)	(363 s)	(2417 s)	28.24 %	(337 s)	(1154 s)
rand11	100	80	(2 s)	(242 s)	(428 s)	15.88 %	(572 s)	(679 s)
rand12	100	80	(0 s)	(403 s)	(511 s)	58.59 %	(635 s)	(413 s)
rand13	200	80	(31 s)	4.88 %	4.88 %		40.24 %	19.51 %
rand14	200	80	(34 s)	0.25 %	0.11 %		0.52 %	
rand15	200	80	(38 s)	0.24 %	0.43 %			0.73 %
rand16	300	80	(136 s)	42.46 %				
rand17	300	80	(143 s)					
rand18	300	80	(165 s)	1.27 %				

6 Conclusions

The formulation of soft constraints penalising patterns in timetables of individual students or groups of students is crucial for the performance of integer programming formulations of timetabling with soft constraints. The presented formulations penalise patterns not only by feature, but also by enumeration of patterns over daily timetables. They might prove to be a good starting point for further research into branch-and-cut algorithms for timetabling with soft constraints.

Acknowledgements. The authors are grateful to Andrea Schaerf and Luca Di Gaspero, who kindly maintain the Udine CTT Problem. Hana Rudová has been, in part, supported by Project MSM0021622419 of MŠMT.

References

1. Ernst, A.T., Jiang, H., Krishnamoorthy, M., Owens, B., Stiehr, G.: An annotated bibliography of personnel scheduling and rostering. Ann. Oper. Res. **127** (2004) 21–144
2. Burke, E.K., Petrovic, S.: Recent research directions in automated timetabling. European J. Oper. Res. **140**(2) (2002) 266–280
3. Petrovic, S., Burke, E.K.: University timetabling. In: Handbook of Scheduling: Algorithms, Models, and Performance Analysis. CRC Press, Boca Raton, FL (2004) 1001–1023
4. Burke, E.K., de Werra, D., Kingston, J.H.: Applications to timetabling. In: Handbook of Graph Theory. CRC, London, UK (2004) 445–474
5. McCollum, B.: University timetabling: Bridging the gap between research and practice. In: Practice and Theory of Automated Timetabling, PATAT 2006, Berlin, Springer (2007)
6. Gaspero, L.D., Schaerf, A.: Multi neighborhood local search with application to the course timetabling problem. In: Practice and Theory of Automated Timetabling, PATAT 2002, Berlin, Springer (2003) 262–275
7. Gaspero, L.D., Schaerf, A.: Neighborhood portfolio approach for local search applied to timetabling problems. J. Math. Model. Algorithms **5**(1) (2006) 65–89
8. Méndez-Díaz, I., Zabala, P.: A cutting plane algorithm for graph coloring. Discrete App. Math. (2008) In press.
9. Lee, J.: All-different polytopes. J. Comb. Optim. **6**(3) (2002) 335–352
10. Williams, H.P., Yan, H.: Representations of the all_different predicate of constraint satisfaction in integer programming. INFORMS J. Comput. **13**(2) (2001) 96–103
11. Burke, E.K., Mareček, J., Parkes, A.J., Rudová, H.: On a clique-based integer programming formulation of vertex colouring with applications in course timetabling. Technical report (2007) at http://arxiv.org/abs/0710.3603.
12. Burke, E.K., Elliman, D., Ford, P.H., Weare, R.F.: Examination timetabling in british universities: A survey. In: Practice and Theory of Automated Timetabling, PATAT 1995, Berlin, Springer (1996) 76–90
13. Rudová, H., Murray, K.: University course timetabling with soft constraints. In: Practice and Theory of Automated Timetabling, PATAT 2002, Berlin, Springer (2003) 310–328
14. Avella, P., Vasil'ev, I.: A computational study of a cutting plane algorithm for university course timetabling. J. Scheduling **8**(6) (2005) 497–514
15. Koch, T.: Rapid Mathematical Programming. PhD thesis, Berlin (2004)
16. Achterberg, T.: Constraint Integer Programming. PhD thesis, Berlin (2007)

Index