Course Opening, Assignment and Timetabling with Student Preferences

Sacha Varone and David Schindl
Economie d'Entreprise, Geneva School of Business Administration, Route de Drize 7, 1227, Carouge, Switzerland

Keywords: Optimization, Scheduling, Application.

Abstract: We consider the following problem of course scheduling and assignment of students. Students express their preferences for each course from several sets of proposed courses and each student has to take a certain number of courses from each set. A minimum number of students is required to open a course and a maximum number of students is specified for each course. The courses have to be scheduled on a limited number of periods so that simultaneous courses have no students in common. This problem can be seen as a generalization of the Student Project Allocation problem. It consists in determining which courses to open, specifying the schedule for these opened courses, and assigning students to them, so that their preferences are maximized. Our model is an Integer Programming problem, which we solve with a common available solver using an iterative process.

1 INTRODUCTION

We present in this paper a real case from the Geneva School of Business Administration (HEG), which was modelled as an integer programming problem and solved with available solvers using exact methods. The problem is the following: according to the Bologna Declaration on the European space for higher education , students may choose some of their courses from a list of courses. The timetabling problem begins therefore with the following question: which courses to open? The decision is based mainly on the number of students and on their preferences for each course.

In the literature, the most similar problem is the Student-Project Allocation problem (SPA), in which a list of projects is available for students who express preferences over the projects. Methods for solving the SPA problem are, among others, building an integer program and solving it with a solver (Anwar and Bahaj, 2003), defining a linear program for some particular cases (Saber and Ghosh, 2001), or developing specific heuristics like a genetic algorithm (Harper et al., 2005). Manlove and O'Malley (Manlove and O'Malley, 2008) show the complexity and give an approximation algorithm for a generalized SPA problem in which both the students and the lecturers have preferences over the projects. The main difference between the SPA problem and our case study is that in the SPA problem there is no restriction on the minimum number of students assigned to a project. On the contrary, our case requires a minimum number of students to be assigned to a course in order to open it. In other words, the SPA problem has a lower bound of $L = 0$ and may have an upper bound of $U > 0$ on the number of students, for each project assignment. Our case study has a lower bound of $L \geq 0$ and an upper bound of $U \geq L$. In that sense, the SPA problem may be seen as a particular case of the problem we present in this paper. More generally, our problem belongs to the so called timetabling problems, which is an active research area (e.g., (Rudová et al., 2011)).

2 PROBLEM DESCRIPTION

During the last year of bachelor education at the Geneva School of Business Administration, students have to take, apart from the required courses, one major course and one or two minors courses, selected from a list of courses. A major course is a full-day course for the whole academic year, whereas a minor course is a four-hour evening course for one semester. Since the number of minor courses exceeds the number of available evenings, some of them have to be scheduled at the same evening.

Selecting a major course and some minor courses can also be done through a predefined portfolio of courses. It consists of one major course and two minor courses, one in the fall semester and one in the spring semester. Students may or may not choose a portfolio. If they do, then only one portfolio can be chosen from a set of portfolios. Those having chosen a portfolio are assigned to the courses contained in the portfolio, with a priority over other students' course assignment. This priority has been stated by

Varone S. and Schindl D..
Course Opening, Assignment and Timetabling with Student Preferences.
DOI: 10.5220/0004189901530158
In *Proceedings of the 2nd International Conference on Operations Research and Enterprise Systems* (ICORES-2013), pages 153-158
ISBN: 978-989-8565-40-2

the management team of the school.

The capacity of each course, as well as the capacity of each portfolio, is limited, and therefore some students will not be assigned to their preferred course. There is also a minimum number of students required to open a course or a portfolio.

The Geneva School of Business Administration has two types of students: full-time students and part-time students. The former have to be assigned to one major course, two minor courses during the fall semester and one minor course during the spring semester. The latter, referring to students working part-time and studying part-time, have to be assigned to one major course, one minor course during the Fall semester and one minor course during the Spring semester.

Course Selection. Prior to 2010, students expressed their preferences by filling out a paper questionnaire, in which they specified their preferences for three major courses, and three minor courses each semester. Since the data treatment and the optimization were done manually, it was tedious and time consuming to do it each year for about 150 students. Such problems often occur in universities and their solution involves building an automated process, as it was done in (Hinkin and Thompson, 2002) with their SchedulExpert software, or using goal programming to build a course schedule with preferences (Badri et al., 1998; Schniederjans and Kim, 1987). It was decided that each student will express his own preferences for each available course, major and minor, for each semester, using a web form.

Students have to rank all the major courses by preference. They also rank all the minor courses in each semester. A strict ordering is therefore expressed for each type of course (major and minor in the fall and minor in the spring). Then, they may indicate if they are willing to take one of the predefined portfolios.

Data Process. To avoid the previously used paper form, we designed a web form linked to a database to collect the data. The web form uses an internal authentication via the Lightweight Directory Access Protocol (LDAP[1]). Once the data has been collected, a model is created using a mathematical modeling language, for example AMPL. The model is then sent with the data and the commands to a solver, for example Gurobi, which returns the solution files. The ultimate goal is to produce a worksheet so that the project leader may easily use the result.

[1]"an Internet protocol for accessing distributed directory services"

3 MODEL FORMULATION

We built an integer programming model, with binary variables in order to define the assignments of students to portfolios (s) and to courses (x). Two types of binary variables refer to the decision of whether or not to open a course (z_j) and a portfolio (z_p). The binary variables (y) define the assignments of courses to periods.
First, we introduce the relevant notation.

J	Set of courses
I	Set of students
H	Types of courses: major or minor
K	Set of periods
P	Set of portfolios
$T = (t_{ip})$	Selection of portfolio
$Q = (q_{pj})$	Definition of portfolios
$A = (a_{ij})$	Matrix of preferences

More precisely,

$$t_{ip} = \begin{cases} 1 & \text{if portfolio } p \text{ is selected by student } i \\ 0 & \text{otherwise} \end{cases}$$

$$q_{pj} = \begin{cases} 1 & \text{if course } j \in \text{portfolio } p \\ 0 & \text{otherwise} \end{cases}$$

Student i ranks a_{ij} the course j, starting with 1 as the preferred course.

Moreover, in order to balance the weight between a major course and a minor course, we introduce a cost matrix $C = (c_{ij})$ such that

$$c_{ij} = \begin{cases} 2a_{ij} & \text{if course } j \text{ is a major course} \\ a_{ij} & \text{otherwise} \end{cases}$$

This means that the utility of assigning a major course with rank 1 and two minor courses with rank 2 is the same as the utility of assigning a major course with rank 2 and two minor courses with rank 1.

The binary variables are of three types: those related to assignment, those related to scheduling, and those related to opening.

$$s_{ip} = \begin{cases} 1 & \text{if student } i \text{ is assigned to portfolio } p \\ 0 & \text{otherwise} \end{cases}$$

$$x_{ij} = \begin{cases} 1 & \text{if student } i \text{ is assigned to course } j \\ 0 & \text{otherwise} \end{cases}$$

$$y_{jk} = \begin{cases} 1 & \text{if course } j \text{ is scheduled in period } k \\ 0 & \text{otherwise} \end{cases}$$

$$z_j = \begin{cases} 1 & \text{if course } j \text{ is opened} \\ 0 & \text{otherwise} \end{cases}$$

$$z_p = \begin{cases} 1 & \text{if portfolio } p \text{ is opened} \\ 0 & \text{otherwise} \end{cases}$$

3.1 Portfolio Problem

The opening of a portfolio only depends on the number of students assigned to it. If this number is smaller than the minimum number needed to viably sustain the portfolio, then the portfolio is not opened. The minimum number of students is given by the academic department, which is a common practice (see (Saber and Ghosh, 2001) for example).

If the number of students who wish to take a portfolio exceeds its capacity, then some of them can not be assigned to it. The students selected to attend the portfolio are those with the highest sum of preferences for the courses in the portfolio. This has been defined during the ranking procedure, in which all courses have to be ranked by the students. In other words, thanks to the complete ordering defined for each set of courses (majors and minors during the fall semester and minors during the spring semester), it is possible to give a score to the portfolio choice of each student. This selection process is expressed by the objective function (1). In case of a similar portfolio choice, with equal score for several students, and a number of students exceeding the upper limit, the selection may be done at random or by the solver itself.

$$\max \sum_{p\in P}\sum_{i\in I}\Big(|J| - \sum_{p'\in P} t_{ip'} \sum_{j\in J} q_{p'j}a_{ij}\Big)\, s_{ip} \tag{1}$$

$$s_{ip} \leq t_{ip} \quad \forall\, i\in I, p\in P \tag{2}$$

$$\sum_{i\in I} s_{ip} \geq 15 z_p \quad \forall\, p\in P \tag{3}$$

$$\sum_{i\in I} s_{ip} \leq 30 z_p \quad \forall\, p\in P \tag{4}$$

The objective function (1) indicates that the preferences of the students have to be maximized. Constraint (2) ensures that portfolio p is assigned only to students who have selected it. The minimum number of students necessary to open a portfolio p is expressed by constraint (3), whereas the capacity of portfolio p is defined by constraint (4). The values of these parameters in our school are 15, respectively 30.

3.2 Course Problem

The objective is to maximize the preferences of the students, or equivalently, to minimize the dissatisfaction of the students (5).

Each student should be assigned to the right number of courses of each type. This number of courses is defined by a function $n(i,h)$, where i represents student $i \in I$ and $h \in H$ represents the type of course. The values of the function n, considered as a parameter, are necessary as some students (full-time or part-time) may already have successfully passed a major or a minor course. The number of courses to be taken by each student is expressed by equation (6). The bounds on the capacity of each course also have to be respected (7)(8). Each course may be opened and in that case a single period has to be assigned to it (9).

Since several minor courses may be taken by the same student, a non-simultaneous course constraint has to be valid (10). To understand this constraint, suppose that a student i is assigned to courses j and j'. This would mean that $x_{ij} = x_{ij'} = 1$. The simultaneity of courses j and j' would be expressed by assigning the period k to both j and j', that is to say $y_{jk} = y_{j'k} = 1$. Therefore, in such a case, the left-hand side of inequality (10) would be equal to 4, which would violate the constraint. Suppose now that either student i is not assigned to j or j', or courses j and j' are not simultaneously given. Then the maximum value of the left-hand side of inequality (10) would be 3, which satisfies the constraint.

Finally, the students assigned to a portfolio have to take the courses in it (11). This last constraint (11) is linear if the values of z_p and s_{ip} are fixed. Notice that the portfolio capacity should not be set to a value larger than the courses' capacities, otherwise the student to course assignment may produce an infeasible solution. This will indeed be the case since, as explained in section 4, the portfolio problem is solved before the course problem.

The model is presented below:

$$\min \sum_{j\in J}\sum_{i\in I} c_{ij}x_{ij} \tag{5}$$

$$\sum_{j\in h} x_{ij} = n(i,h) \quad \forall i\in I, h\in H \tag{6}$$

$$\sum_{i\in I} x_{ij} \leq 30 z_j \quad \forall j\in J \tag{7}$$

$$\sum_{i\in I} x_{ij} \geq 15 z_j \quad \forall j\in J \tag{8}$$

$$\sum_{k\in K} y_{jk} = z_j \quad \forall j\in J \tag{9}$$

$$x_{ij} + x_{ij'} + y_{jk} + y_{j'k} \leq 3 \quad \forall i\in I, k\in K,\ j, j'\in J \tag{10}$$

$$z_p t_{ip} q_{pj} s_{ip} \leq x_{ij} \quad \forall i\in I \tag{11}$$

4 SOLUTION METHODOLOGY

We first solve the portfolio allocation problem, as it has priority over the course allocation. Once a solution is found, constraint (11) ensures that students enrolled in a portfolio are assigned to the corresponding major and minor courses. Then we solve the remaining assignment and scheduling problem, which consists in optimizing the students assignment to courses along with the courses' schedule.

4.1 Portfolio Allocation

Since portfolio allocation is given priority over course allocation, this sub-problem can be solved first. Its goal is to define which portfolio to open and which not, and which students to assign to each opened portfolio. In our case, even though we have a binary program, the size of the instance is small so that usage of common solvers allows to easily find an optimal solution within a few seconds.

4.2 Course Allocation

Once the portfolio sub-problem is solved, it is necessary to assign students enrolled in a portfolio to the corresponding courses. This will fix some of the variables and hence reduce the size of the instance.

A major difficulty in solving "difficult" binary programs, in the sense of NP-hard problem, is that the time to solve them optimally increases exponentially with the size of the instance. One approach is to build a specific heuristic which solves the problem but often without a guarantee of optimality, or in some cases with a bound on the gap between the solution and the optimal solution. This approach is described in section 4.3. Another approach, described by Algorithm 2, is to arbitrarily fix some variables to values they are likely to take at optimality, so as to reduce the size of the problem in the hope that it will become small enough to be tractable by available solvers.

Instead of considering all available courses, a subset of them may be selected with the aim of solving the subproblem optimally (Algorithm 2). The selection of the subset of courses is based on the expressed preferences of the students: a maximum rank is defined as an upper bound on the possible assignment of a student to a course. Then, for each course, if the number of students with a preference lower than or equal to this maximum rank is lower than the required lower bound, then the course is not opened (Algorithm 1). Since there is a limited capacity for each course, it may imply that the problem is infeasible. In that case the maximum rank is increased by one unit and this process is run iteratively.

4.3 Further Ideas

A direct attempt to solve the course problem presented in Section 3.2 may not result in a solution, due to the intrinsic complexity of this binary problem. Then an heuristic procedure would be necessary.

Algorithm 1: Course elimination.

```
Require: MaxRank
1: for all $j \in J$ do
2:     if $\sum_i \delta(a_{ij} \leq \text{MaxRank}) < 15$ then
3:         $J = J \backslash \{j\}$
4:     end if
5: end for
```

Algorithm 2: Direct method : IterateElimination procedure.

```
1: Presolve {Course assignment with respect to portfolio assignment}
2: MaxOrder = 2 {Initialization of maximal preference}
3: repeat
4:     MaxOrder = MaxOrder + 1
5:     Call algorithm 1(MaxOrder)
6:     Solve the courses alloc. problem (5)-(11)
7: until Problem solved or MaxOrder= $\min\{|\{j : j \in h\}| : h \in H\}$
```

The idea expressed by Algorithm 3 is the following: a first difficulty in solving the courses allocation problem (5)-(11) is the assignment of courses to periods, so that constraint (10) is satisfied. In our application, the set of periods associated to each type of courses (major, minor in spring, minor in autumn) are disjoint: i.e., the set of available periods for a major course is different from the one for a minor course, and the set of available periods for a minor course in spring is different from the one for a minor course in autumn. This assumption is made as a requirement for Algorithm 3. The first step is therefore to choose the assignment of periods to courses (line 1). To proceed, one should avoid two courses with a lot of high preferences to be scheduled at the same period.

Algorithm 3: Heuristic for the courses allocation problem.

```
Require: Non overlapping periods between types of courses
1: Assign periods to courses
2: Build the associated flow network
3: Solve the associated min cost max flow problem
4: while feasible solution is not found do
5:     Eliminate 1 course with insufficient number of students
6:     Solve the associated min cost max flow problem
7: end while
```

A second difficulty lies in the minimal number of students necessary to open a course. This constraint (8) is temporarily relaxed. The second step (line 2) consists of building a transshipment network $G = (V, A, c, w)$ with capacities c and weights w on the arcs. The set of nodes V contains a source and a sink, a node (i,h) for each student with each type of course, a node (i,k) for each student with each period, and the courses with their already associated periods (j,k). The set of arcs is (source, (i,h)), $((i,h),(i,k))$, $((i,k),(j,k))$, ((j,k), sink). The capacity c and weight w functions are defined as follows:

Arcs A	Capacity c	Weight w
(source,(i,h))	$n(i,h)$	0
$((i,h),(i,k))$	1	0
$((i,k),(j,k))$	1	a_{ij}
((j,k), sink)	30	0

Since the capacities and the weights are all integers, this transshipment problem can be solved optimally using continuous variables, as stated by the well known integrality theorem; this problem is known under the name of min cost max flow problem and is polynomially solvable: *For every network flow problem with integer data, every basic feasible solution and, in particular, every basic optimal solution assigns integer flow to every arc.*

Once the problem solved, if some courses do not contain enough students, one of these is eliminated from the set of courses. The choice of the course to be eliminated may depend on the preferences expressed by the students, as stated by Algorithm 1. Technically, the elimination of a course may be done by either removing the course from the network model, or by setting the capacity of the course to 0. And this process iterates until all courses have the minimal number of students (while loop) required to be opened.

Of course, in order to get a feasible solution, prior to running Algorithm 3, one has to check that the number of students in each type of course (major, minor in spring, minor in autumn) is less than or equal to the sum of the capacities.

One could also extend the model by changing some unused arc costs or capacities. For instance, one could take into account students preferences over days with costs on the arcs $((i,h),(j,k))$, or even by putting a zero capacity if a student is not available a given day. Moreover, if a course has a cost which is linear with the number of students (for instance if some material has to be bought for each student), one could put a cost, say b on each arc ((j,k), sink), expressing that each student following the course costs an additional amount of b.

5 APPLICATION

The student assignment problem to major and minor courses was successfully applied at the Geneva School of Business Administration. In 2011, it involved 146 students, of which 98 full-time students and 48 part-time students, 14 major courses, 10 minor courses during the fall semester and 9 minor courses during the spring semester. The minimum number of students necessary to open a portfolio or a course is 15, and the capacity of each course is 30.

The generated model contains 5904 variables and 66541 constraints. It is solved optimally within a few seconds on a Intel Core 2 Quad PC with 4 Gb memory, using the AMPL modeling language and the Gurobi 4.5 solver. It was therefore, with our specific data set, not necessary to apply the heuristic described by Algorithm 3.

6 CONCLUSIONS

In this article, we described a successful application of operations research tools. We created a convenient way of data collection, modelled an assignment problem as a binary problem and finally solved it using available solvers.

Solving an NP-hard problem, or at least one viewed as ”difficult” from a practical point of view, requires several steps: data collection, modeling, solving and presenting the results. It is often agreed that the development of a specific heuristic is required even for middle-size “difficult” problems. We learned from our experience in the field of practical optimization of “difficult” problems that before developing home-made heuristics, it makes sense to try solving the problem with available commercial or open-source solvers.

REFERENCES

Anwar, A. A. and Bahaj, A. S. (2003). Student project allocation using integer programming. *IEEE Transactions on Education*, 46:359–367.

Badri, M. A., Davis, D. L., Davis, D. F., and Hollingsworth, J. (1998). A multi-objective course scheduling model: Combining faculty preferences for courses and times. *Computers & OR*, 25(4):303–316.

Harper, P. R., de Senna, V., Vieira, I. T., and Shahani, A. K. (2005). A genetic algorithm for the project assignment problem. *Computers & Operations Research*, 32(5):1255 – 1265.

Hinkin, T. R. and Thompson, G. M. (2002). Schedulexpert: Scheduling courses in the cornell university school of hotel administration. *Interfaces*, 32:45–57.

Manlove, D. F. and O'Malley, G. (2008). Student-project allocation with preferences over projects. *Journal of Discrete Algorithms*, 6(4):553 – 560. Selected papers from the 1st Algorithms and Complexity in Durham Workshop (ACiD 2005), 1st Algorithms and Complexity in Durham Workshop (ACiD 2005).

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

Saber, H. M. and Ghosh, J. B. (2001). Assigning students to academic majors. *Omega*, 29(6):513–523.

Schniederjans, M. J. and Kim, G. C. (1987). A goal programming model to optimize departmental preference in course assignments. *Computers & OR*, 14(2):87–96.