Adaptive Large Neighborhood Search for Student Sectioning at Danish high schools

Simon Kristiansen · Thomas R. Stidsen

Keywords Student Sectioning · Elective Course Planning · Adaptive Large Neighborhood Search · Educational Timetabling · High School Planning · Metaheuristics

Mathematics Subject Classification (2000) 90-08 · 90B35 · 90C10 · 90C59

1 Introduction

Student Sectioning is one of the less studied subjects within Educational Timetabling (Pillay (2010)). Student Sectioning is normally used at universities (e.g. Müller and Murray (2010) and Cheng et al (2003)), whereas this paper is concerned high schools in Denmark. I.e. a more generalized model is needed since this problem should be applicable for approximately 200 different high schools. The concerned problem is also known as Elective Course Planning Problem (ECP), described in Kristiansen et al (2011b). However in this paper we will also try to pack the students more convenient in the classes. I.e. minimize the number of common classes in each class and to have a more even distribution between classes of same course. The ECP is a preceding planning problem of the actual High School Timetabling in Denmark. The Internation Timetabling Competition 2011 was devoted to High School Timetabling (see e.g. Post et al (2012) and Sørensen et al (2012)).

The students request some elective courses, and the problem is then to assign course classes to time slots and then assign students to the classes given their requests. We want to maximize the number of fulfilled requests and to minimize the number of classes. The problem is to both please the students and to insure good economic. It cost approximately € 27,000 p.a. to create a class, and as the high schools are self-governing, ECP is a crucial problem. If the requests are not granted, the students might change school and as the high school are payed upon the number of students, they will lose revenue.

Fig. 1: An example of a weekly schedule with four modules each day and a total of five time slots for elective courses. The white time slots are used for mandatory courses.
As mentioned it is also desired to have an even distribution on the number of students in classes of same course. E.g. 40 students requests the same given course, and with 28 as upper bound on the class size, we aim at having a distribution of 20-20.

2 Integer programming model

The ECPP is formulated as an IP model. A high school has a set of students \( s \in S \), a set of offered courses \( e \in E \), a set of classes \( c \in C \) and a set of time slots \( b \in B \). Each students is assigned to a common class, \( q \in Q \), this is denoted by \( I_{q,s} \in \{0, 1\} \). Each course belongs to one of the course subjects given by the set \( f \in F \). The maximum number of classes of each subject in a time slot is given by \( M_f \in \mathbb{R}^+ \). \( K_{c,f} \in \{0, 1\} \) denotes whether course \( c \) is teaching subject \( f \) or not. For each course class there exist an upper bound on the class sizes, \( U_c \in \mathbb{R}^+ \). \( A_{c,s} \in \mathbb{R}^+ \) indicates whether student \( s \) is locked to course class \( c \). I.e. the student must be assigned to the given course class. \( R_{c,s} \in \{0, 1\} \) indicates whether student \( s \) has requested course \( c \) or not. \( D_{c,e} \in \{0, 1\} \) denotes whether class \( c \) is teaching course \( e \), or not. The total maximum number of classes which can be created is given by \( P \in \mathbb{R}^+ \). The parameters \( J_{c,c'} \in \{0, 1\} \) and \( H_{c,c'} \in \{0, 1\} \) indicates whether two classes cannot be placed in the same time slot or should be placed in the same time slot, respectively. The decision whether student \( s \) is assigned to class \( c \) in block \( b \) is defined by \( x_{c,b,s} \in \{0, 1\} \), while the decision whether course class \( c \) is assigned to time slot \( b \) is given by \( y_{c,b} \in \{0, 1\} \). The binary variable \( z_{c,q} \in \{0, 1\} \) takes value 1 if common class \( q \) is present in class \( c \). The variables \( w_{c,c'}^+ \in \mathbb{N}_0 \) and \( w_{c,c'}^- \in \mathbb{N}_0 \) counts the difference of the number of students between classes of same course. The objectives are weighted in respect to each other, given by \( \alpha_{c,s}, \beta_c, \gamma \) and \( \delta \).

\[
\begin{align*}
\text{max} & \quad \sum_{c,b,s} \alpha_{c,s} \cdot x_{c,b,s} - \sum_{c,b} \beta_c \cdot y_{c,b} - \gamma \cdot \sum_{c,q} z_{c,q} - \delta \cdot \sum_{c,c'} (w_{c,c'}^+ + w_{c,c'}^-) / 2 \\
\text{s.t.} & \quad \sum_{c} x_{c,b,s} \leq 1 \quad \forall b, s \\
& \quad \sum_{b} y_{c,b} \leq 1 \quad \forall c \\
& \quad \sum_{c,b} x_{c,b,s} \cdot D_{c,e} \leq R_{e,s} \quad \forall e, s \\
& \quad \sum_{s} x_{c,b,s} \leq U_c \quad \forall c, b \\
& \quad \sum_{b} I_{q,s} \cdot x_{c,b,s} \leq z_{c,q} \quad \forall c, q, s \sum_{s} A_{c,s} = 0 \\
& \quad \sum_{c,s} x_{c,b,s} - \sum_{c,s} x_{c',b,s} = w_{c,c'}^+ - w_{c,c'}^- \quad \forall c, c', b, s, D_{c,e} = D_{c',e} = 1, \sum_{s} A_{c,s} = \sum_{s} A_{c',s} = 0 \\
& \quad x_{c,b,s} \leq y_{c,b} \quad \forall c, b, s, A_{c,s} = 0 \\
& \quad x_{c,b,s} = y_{c,b} \quad \forall c, b, s, A_{c,s} = 1 \\
& \quad \sum_{c,b} y_{c,b} \leq P \\
& \quad y_{c,b} + y_{c',b} \leq 1 \quad \forall c, c', b, s, J_{c,c'} = 1 \\
& \quad y_{c,b} = y_{c',b} \quad \forall c, c', b, s, H_{c,c'} = 1 \\
& \quad \sum_{c} K_{c,f} \cdot y_{c,b} \leq M_f \quad \forall b, f, M_f > 0 \\
& \quad x_{c,b,s} \in \{0, 1\} \\
& \quad y_{c,b} \in \{0, 1\} \\
& \quad z_{c,q} \in \{0, 1\} \\
& \quad w_{c,c'}^+ \in \mathbb{N}_0 \\
& \quad w_{c,c'}^- \in \mathbb{N}_0
\end{align*}
\]
of students in a course class. Constraints (6) are counting the number of common classes used while constraints (7) are used for equal distribution of the students in classes of same course. Constraints (8) is the connection between the two variables $x_{c,b,s}$ and $y_{c,b}$ and make sure that a student cannot be assigned a class which is not yet assigned to a time slot. Constraints (6) and (7) are soft constraints. Constraints (9) shall ensure that if students are locked to a course class and the class is assigned a timeslot, the students should be assigned to the class. Constraints (10) ensures that the total number of created classes does not exceed maximum. Constraints (11) ensure that classes which cannot be placed in same time slot are not done so and constraints (12) ensure that classes which should be placed in same time slot are satisfied. Finally, constraints (13) make sure that the resource limit on subjects $f$ are respected.

3 Solution method

It has been chosen to attempt Adaptive Large Neighborhood Search (ALNS) to establish solutions to the ECPP. ALNS was first applied and is still mainly used on Vehicle Routing Problems (Azi et al. (2010); Laporte et al. (2010); Salazar-Aguilar et al. (2011); Ribeiro and Laporte (2012)). It has however also been applied on a few other problems such as Lot-sizing (Muller and Spoorendonk (2011)) and Scheduling problems (Muller (2010); Kristiansen et al. (2011a); Sørensen and Stidsen (2012)). Pisinger and Ropke (2010) is recommended for additional reading on ALNS. The pseudo code is given in Algorithm 1.

\begin{algorithm}
\begin{algorithmic}[1]
\STATE \textbf{Input:} a feasible solution $x_{q,b,s}, q \in \mathbb{N}$
\STATE $x_{best} = x, \pi = (1,...,1)$
\REPEAT\STATE $x' = x$
\STATE select destroy and repair methods $d \in \Omega^-$ and $r \in \Omega^+$ using $\pi$
\STATE remove $q$ requests from $x'$ using $d$
\STATE reinsert removed requests into $x'$ using $r$
\IF { $c(x') > c(x_{best})$ }  
\STATE $x_{best} = x'$
\ENDIF
\IF { accept($x'$, $x$) } 
\STATE $x = x'$
\ENDIF
\STATE update $\pi$
\UNTIL { stop-criterion met }
\RETURN $x_{best}$
\end{algorithmic}
\end{algorithm}

The neighborhoods are implicitly defined by several destroy and repair methods. In each iteration, a destroy and a repair method is chosen upon some performance indicators which is updated after each iteration.

For this implementation of ALNS two different types of moves are used: Assign/unassign a class with students to/from a time slot and assign/unassign a student to/from an assigned class. Based on these moves a total of 8 different destroy methods and 4 repair methods are implemented. The destroy methods are simple random removal heuristics and Shaw heuristics (Shaw (1997)), where less or more related classes are removed from the solution. The repair methods are basic greedy algorithms and regret heuristics (Potvin and Rousseau (1993)), which aims at inserting the request which we will regret most if not inserted immediately.
4 Results

The algorithm is implemented in Lectio, and is hence available for use for approximately 200 different high schools in Denmark. This gives the possibilities for a huge amount of data for further testing and research. Table 1 shows the size and the computational results from 7 different datasets.

<table>
<thead>
<tr>
<th></th>
<th>No. of student requests</th>
<th>No. of courses</th>
<th>No. of blocks</th>
<th>No. of assigned classes</th>
<th>Assigned requests</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vejen</td>
<td>382</td>
<td>586</td>
<td>29</td>
<td>3</td>
<td>36</td>
</tr>
<tr>
<td>Silkeborg</td>
<td>927</td>
<td>1789</td>
<td>65</td>
<td>5</td>
<td>77</td>
</tr>
<tr>
<td>Falkoner</td>
<td>421</td>
<td>1080</td>
<td>49</td>
<td>4</td>
<td>66</td>
</tr>
<tr>
<td>Vordingborg</td>
<td>415</td>
<td>1462</td>
<td>61</td>
<td>5</td>
<td>68</td>
</tr>
<tr>
<td>Alsion</td>
<td>385</td>
<td>650</td>
<td>31</td>
<td>5</td>
<td>31</td>
</tr>
<tr>
<td>Holtebro</td>
<td>345</td>
<td>567</td>
<td>18</td>
<td>5</td>
<td>29</td>
</tr>
<tr>
<td>Frederikssund</td>
<td>170</td>
<td>273</td>
<td>18</td>
<td>4</td>
<td>18</td>
</tr>
</tbody>
</table>

Table 1: Results for a given set of real-life problems at Danish high schools.

References


1 Lectio is developed by MaCom A/S and is a cloud-based high school administration, which handles all sorts of administrative tasks for the high schools, including a GUI and a heuristic-based solver for the ECPP.