Downloaded from orbit.dtu.dk on: Apr 19, 2024

DTU Library

Elective course student sectioning at Danish high schools

Kristiansen, Simon; Stidsen, Thomas Riis

Published in:
Annals of Operations Research

Link to article, DOI:
10.1007/s10479-014-1593-7

Publication date:
2016

Document Version
Peer reviewed version

Link back to DTU Orbit

Citation (APA):
Kristiansen, S., & Stidsen, T. R. (2016). Elective course student sectioning at Danish high schools. *Annals of Operations Research*, *239*(1), 99-117. https://doi.org/10.1007/s10479-014-1593-7

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

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

Elective Course Student Sectioning at Danish High Schools

Simon Kristiansen · Thomas R. Stidsen

Received: date / Accepted: date

Abstract The *Elective Course Student Sectioning* (ECSS) problem is an yearly recurrent planning problem at the Danish high schools. The problem is of assigning students to elective course classes given their requests such that as many requests are fulfilled and the violation of the soft constraints is minimized. This paper presents an adaptive large neighborhood search heuristic for the ESCC. The algorithm is applied to 80 real-life instances from Danish high schools and compared with solutions found by using the state-of-the-art MIP solver Gurobi. The algorithm has been implemented in the commercial product Lectio, and is thereby available for approximately 200 high schools in Denmark.

Keywords: Education Timetabling; High School Timetabling; Student Sectioning; Elective Course Planning; Adaptive Large Neighborhood Search; Integer Programming

1 Introduction

The purpose of this article is to describe the *Elective Course Student Sectioning* (ECSS) problem and the solution methods used to solve it. ECSS serves as a pre-processing planning problem for the actual high school timetabling in Denmark and is an important yearly recurrent planning problem. The problem is concerned with assigning students to elective course classes given their requests, subject to various soft and hard constraints.

This paper is written in collaboration with the Danish software company MaCom A/S, whose main product is the cloud-based high school administration system Lectio. Lectio is used by the majority of all the Danish high schools. Due to the diversity of high schools, it is very important that the problem is formulated such that it covers all the high schools and such that the solution approach performs well for problems of different sizes and format.

In Section 2 the ECSS is described in detail with a literature review given in Section 3. In Section 4 the problem is formulated as an Mixed Integer Programming problem and the complexity of the problem is proven. The solution approach for the problem is explained in Section 5. Parameter tuning and the performance results are presented in Section 6 and 7, respectively. Finally Section 8 rounds up and concludes.

2 Problem description

It takes three years to complete a high school education in Denmark and every year the high schools need to create a timetable for the following year, hence they also need to solve the ECSS. Each year the students request some elective courses which they want to attend along with their mandatory courses. The problem is then to assign the students to elective course classes given

S. Kristiansen (E-mail: sikr@dtu.dk) · T.R. Stidsen (thst@dtu.dk)
Section of Operations Research, Department of Management Engineering, Technical University of Denmark
Building 426, Produktionstorvet, DK-2800 Kgs. Lyngby, Denmark

their requests and assign these classes to some time slots. The main goal of the ECSS is to fulfil as many of the students' elective course requests as possible while minimizing the number of classes created. The problem is of both educational and economical nature. First of all, the students are planning their high school education such that they have the necessary merits for applying for a university education afterwards. If a student is not granted his requests he might miss a merit to get admission to his desired education. Secondly if a student is not granted his request, it might entail that the student changes high school or drops out, and this impose some economic issues for the high school. The Danish high schools are self-governing and get a fee from the state based on the number of students which finish an education at the high school, i.e. a significant part of the high schools income is based on the students. It should be noted that the students in Denmark can freely choose between high schools. Another aspect of the economic issues is the creation of classes. For each created class there is a cost of approximately €27.000 p.a., i.e. it is not enough to grant all the requests it is also very important for the high schools to keep the number of created classes at a minimum.

A typically weekly schedule for a Danish high school consists of five days, each day divided into a given number of time slots where the teaching is performed. Each time slot then consists of a number of lectures and a student can of course only attend one course in each time slot. There exist two types of time slots in the Danish high schools. Time slots for the mandatory courses and time slots for the elective courses. This is due to that the mandatory courses are often taught in the common classes. I.e. a student has almost all his mandatory courses with the same fellow students. The elective courses are however mixed between these common classes, and hence it is more beneficial for the administration to have the elective course classes in some separate time slots. Figure 1 is an example of a typically weekly schedule containing five different time slots for elective courses.

	Monday	Tuesday	Wednesday	Thursday	Friday
8:15 9:45	Time1			Time3	
10:00 11:30					
	Lunch break				
12:00 13:30					
13:45 15:15		Time2		Time4	Time5

Fig. 1: *An example of a weekly schedule with four time slots each day. Five time slots (gray colored) are reserved for the elective courses whereas the mandatory courses are placed in remaining time slots (white colored)*

The time slots for elective courses are often chosen to be placed in the beginning or in the end of a day. This is to minimize the possibility of creating idle slots for students when creating the entire timetable. Due to the dividing of the time slots into elective and mandatory, the elective course planning can be seen as an independent part of the weekly schedule and therefore the mandatory courses and their respective time slots are neglected in the remainder of this article. The problem of assigning the mandatory courses is known as the classical High School Timetabling. (See e.g. Sørensen et al. [2012] and Post et al. [2012])

Ideally all the students should be assigned their first priority elective course requests. If a student is not granted his first priority requests, it would be preferable if his second priority request is granted. However it should be a noted that the outcome of the ECSS is an algorithm used in decision support software. If a student is not granted a first priority request, the high school administration wants to have a dialog with the given student before assigning him to his second priority. Hence we do only consider the first priority requests of the students.

3 Related work

As mentioned the ECSS serves as a pre-processing planning problem for the High School Timetabling (HSTT). The last couple of years more research has been done within HSTT and the *third International Timetabling Competition* (ITC2011) treats the HSTT (see e.g. Post et al. [2012] and Sørensen et al. [2012]). The ECSS is however not a very well researched area. Compared to the problem of ITC2011, the ECSS is a matter of which students should be in which elective course class, whereas the high school timetabling is concerned with the construction of a timetable according to some predefined course classes. Many articles have been made within education timetabling and within this research area ECSS is most related to *Student Sectioning* (Carter and Laporte [1998], Schaerf [1999], Burke and Petrovic [2002], Pillay [2010]).

The literature on Student Sectioning is however mostly concerned the universities. E.g. Erben and Keppler [1996], Rudova and Murray [2003], Müller and Murray [2010]. In Müller and Murray [2010] University Course Timetabling and Student Sectioning are combined and is solved using Iterative Forward Search algorithm. In de Haan et al. [2007] *optional subject* for the students is used when constructing the high school timetabling in the Netherlands, and *cluster schemes* are created to maintain the students' optional courses. The optional subjects are similar to the elective courses of this paper. At the high schools in Denmark it is not an option to have the planning of elective classes incorporated in the general timetabling. The high school, wants to separate planning tools, they use them as decision support and make changes during each part of the process.

The ECSS for the Danish high schools was first described in Kristiansen et al. [2011]. The article gives a good overview of the problem, and Dantzig-Wolfe decomposition and explicit constraint branching is used for solving the problem. The approach however was only created to clarify the performance of an earlier solution approach used in the software Lectio, and was hence never released and the model misses some of the restriction which is incorporated in this article. It did however prove that the previous solution method for the ECSS was inefficient and that it lacked some restrictions including the fairness distribution which is described in the following section.

4 Integer programming model

In the following the ECSS is formulated as a MIP model which aims at maximizing the number of granted elective course requests while minimizing the violation of soft constraints and respecting the hard constraints. For the ECSS the high schools have a set of students $\mathcal{S}$, a set of offered courses $\mathcal{E}$, a set of classes $\mathcal{C}$ and a set of time slots $\mathcal{T}$. The parameter $D_{c,e} \in \{0,1\}$ denotes whether course class c is teaching course e and parameter $R_{e,s} \in \{0,1\}$ indicates whether student s has requested course e or not. The decision whether student s is assigned course class c in time slot t is defined by the binary variable $x_{s,c,t} \in \{0,1\}$, whereas the binary variable $y_{c,t} \in \{0,1\}$ takes value 1 if course class c is assigned time slot t, zero otherwise. In the following the MIP formulation is divided into small sections.

4.1 Availabilities

It is not allowed to assign a student to a class of a course he has not requested, and it is obviously not possible to assign a student to more than one course class in each time slot. Neither is it possible to assign a course class to more than one time slot. The following constraints make sure that these restrictions are maintained.

$$\sum_{c,t} D_{c,e} \cdot x_{s,c,t} \leq R_{e,s} \qquad \forall\, e, s \tag{1}$$

$$\sum_{c} x_{s,c,t} \leq 1 \qquad \forall\, t, s \tag{2}$$

$$\sum_{t} y_{c,t} \leq 1 \qquad \forall\, c \tag{3}$$

As the elective courses can have duration of more than one year some of the elective courses may be a continuation from the previous school year. For these course classes the students are locked. I.e. the students which were assigned the course class the previous year, must be assigned the class this year also. Let $A_{c,s} \in \{0,1\}$ take value 1 if student s is locked to course class c, zero otherwise. The following constraints are then imposed.

$$x_{s,c,t} \leq y_{c,t} \qquad \forall\, c,t,s, A_{c,s} = 0 \tag{4}$$

$$x_{s,c,t} = y_{c,t} \qquad \forall\, c,t,s, A_{c,s} = 1 \tag{5}$$

The basic objectives for the ECSS are to maximize the number of granted request and to minimize the number of created elective course classes.

$$\sum_{c,t,s} \alpha_{c,s} \cdot x_{s,c,t} - \sum_{c,t} \beta_c \cdot y_{c,t} \tag{6}$$

4.2 Resource limitations

There exists some resource limitation when solving the ECSS. Firstly, it is determined by the Danish educational legislation, that the class size in high schools may not exceed 28 students. This is to make sure that the students have the best possibly environment. However some course classes might have an even more restricted upper limit. For classes where the students are locked, the upper class size is equal to the number of locked students, such that no new students can be assigned to the given class. Let the parameter $U_c \in \mathbb{N}$ denote the upper class size for course class c.

$$\sum_{s} x_{s,c,t} \leq U_c \qquad \forall\, c,t \tag{7}$$

Furthermore, as the price for creating an elective course class is approximately €27.000 p.a., a high school often has an upper limit on how many classes they can afford to create each year. Let $P \in \mathbb{N}$ be the maximum number of classes which can be created in total.

$$\sum_{c,t} y_{c,t} \leq P \tag{8}$$

The limitation of classes which can be created of a given course is given by the set of classes $\mathcal{C}$ and the parameter $D_{c,e}$.

There also exists some resource limitation on the number of course classes with the same subject which can be taught in the same time slot. Let $\mathcal{F}$ be the set of course subjects of a high school. Each course is teaching a course subjects, given by $K_{c,f} \in \{0,1\}$. Due to the limited resources on e.g. rooms and equipment, it is often not possible to assign all class of a course to the same given time slot. E.g. if a high schools only have x physic class rooms, it is not possible to assign more than x physic classes to a given time slot. Let $B_{f,t} \in \mathbb{N}$ be the maximum number course classes of subject f which can be taught in time slot t. This imposes the following constraints

$$\sum_{c} K_{c,f} \cdot y_{c,t} \leq B_{f,t} \qquad \forall\, f,t, B_{f,t} > 0 \tag{9}$$

4.3 Class positions

Some course classes cannot be assigned to the same time slot. E.g. two courses with the same preassigned teacher are not allowed to share the same time slot. Let the parameter $J_{c,c'} \in \{0,1\}$ take value 1 if the course classes c and c' cannot be in the same time slot, zero otherwise. The constraints are given by

$$y_{c,t} + y_{c',t} \leq 1 \qquad \forall\, c,c',t, J_{c,c'} = 1 \tag{10}$$

On the contrary some courses are forced to be placed in the same time slot. Let $L_{e,e'} \in \{0,1\}$ denote whether classes of course e should be assigned the same time as classes of course e'. As not

all course classes are being assigned to a time slot, a new variable is introduced such that only the assigned classes are considered for this constraint. Let the binary variable $h_{e,t} \in \{0,1\}$, take value 1 if course e is placed in time slot t. We then get the following constraints.

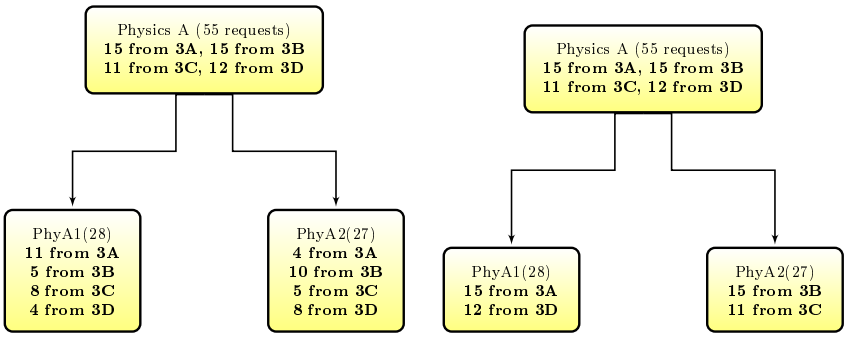
$$y_{c,t} \leq D_{c,e} \cdot h_{e,t} \qquad \forall\, c, e, t \tag{11}$$

$$h_{e,t} = h_{e,t'} \qquad \forall\, e, e', t, L_{e,e'} = 1 \tag{12}$$

$$\sum_{e} h_{e,t} \leq 1 \qquad \forall, e, e', t, L_{e,e'} = 1 \tag{13}$$

4.4 Common classes

When a student is enrolled at a high school he is assigned to a common class. Many of the mandatory courses in the Danish high schools are taught in classes exactly equal to a common class. It has the advantage that the students in a common class are quite familiar with each other, and it makes it easier to collaborate between mandatory classes of different subject, as the students attending the two classes are the same. Hence it would also be beneficial to have as few common classes representing in each elective course class. As the elective courses can be selected by all the students it is most unlikely that only students from one common class have requested a given elective course. Figure 2 gives an example of the handling of common classes.



(a) Two classes with four common classes represented in each class. All possible common classes are represented in both class

(b) Two classes with only two common classes represented in each. Neither of the common classes are represented in more than one class

Fig. 2: *Two examples of representing common classes in classes of same course. 55 students from four different common classes have requested* Physics A *as elective course. With an upper class size of 28, two classes are needed to fulfill all requests. Figure 2(a) is the worst possible scenario where all possible common classes is represented in each class, whereas 2(b) is the best solution where only two common classes is represented in each class. I.e. no common class is represented more than once.*

We want to minimize the number of common classes represented in each created elective course class. Let $\mathcal{Q}$ be the set of common classes for a high school, and let $I_{s,q} \in \{0,1\}$ denotes whether student s is part of common class q or not. The decision variable $z_{c,q} \in \{0,1\}$ indicates whether common class q is represented in course class c, or not. There is no need to minimize the number of common classes represented in classes where the students are locked, as these cannot be improved. Let parameter $A_c \in \{0,1\}$ denotes whether course class c is locked or not. This gives the following constraints.

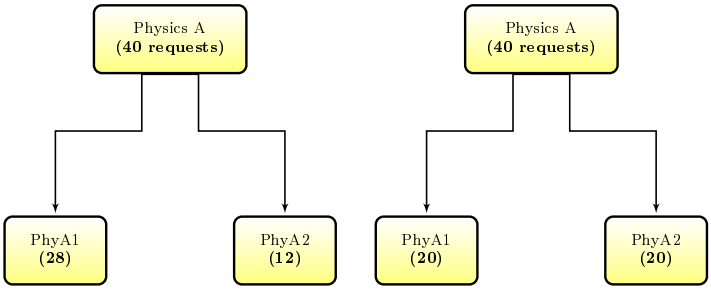
$$\sum_{t} I_{s,q} \cdot x_{s,c,t} \leq z_{c,q} \qquad \forall\, c, q, s, A_c = 0 \tag{14}$$

with a contribution to the objective given by

$$-\gamma \cdot \sum_{c,q} z_{c,q} \tag{15}$$

4.5 Even distribution

When having two classes of same course, it is then highly appreciated to have approximately the same amount of students attending both classes. This is due to fairness of both the students and the teachers. See Figure 3 for an illustration of two different distributions of students in two classes of same course.



(a) The worst distribution (b) The best distribution

Fig. 3: *Two examples of distribution of students in two classes of same course. 40 students have requested* Physics A *as elective course and there is an upper class size of 28. I.e. two classes are needed to fulfill all requests. Figure 3(a) is the worst possible scenario on distribution the students into the classes, whereas 3(b) is the best*

We want to minimize the difference between the numbers of students in classes of same course. Only unlocked assigned classes are considered. Let variable $w_{c,c'} \in [0,1]$ be the ratio of the difference between c and c', and let $g_{c,c'} \in [0,1]$ be a slack variable helping with determination of the difference ratio. Let $\rho(i)$ denote the ordinal number of i. The following constraints are imposed

$$\sum_{t,s} x_{s,c,t} - \sum_{t,s} x_{s,c',t} \leq g_{c,c'} \cdot U_c \; \forall\, c, c', D_{c,e} = D_{c',e} = 1, \; A_c = A_{c'} = 0, \rho(c) < \rho(c') \tag{16}$$

$$\sum_{t,s} x_{s,c',t} - \sum_{t,s} x_{s,c,t} \leq g_{c,c'} \cdot U_c \; \forall\, c, c', D_{c,e} = D_{c',e} = 1, \; A_c = A_{c'} = 0, \rho(c) < \rho(c') \tag{17}$$

$$\sum_{t} (y_{c,t} + y_{c',t}) - 2 + g_{c,c'} \leq w_{c,c'} \quad \forall\, c, c', D_{c,e} = D_{c',e} = 1, \; A_c = A_{c'} = 0, \rho(c) < \rho(c') \tag{18}$$

$$\tag{19}$$

The contribution to the objective is then given by

$$-\delta \cdot \sum_{c,c'} U_c \cdot w_{c,c'} \tag{20}$$

An entire model of the ECSS is given in (21)

4.6 IP model for ECSS

$$\max \sum_{c,t,s} \alpha_{c,s} \cdot x_{s,c,t} - \sum_{c,t} \beta_c \cdot y_{c,t} - \gamma \cdot \sum_{c,q} z_{c,q} - \delta \cdot \sum_{c,c'} U_c \cdot w_{c,c'} \tag{21}$$

$$
\begin{aligned}
\text{s.t.} \quad & \sum_{c} x_{s,c,t} && \leq 1 && \forall\, t, s \\
& \sum_{c,t} D_{c,e} \cdot x_{s,c,t} && \leq R_{e,s} && \forall\, e, s \\
& \sum_{t} y_{c,t} && \leq 1 && \forall\, c \\
& \sum_{s} x_{s,c,t} && \leq U_c && \forall\, c, t \\
& \sum_{c,t} y_{c,t} && \leq P && \\
& x_{s,c,t} && \leq y_{c,t} && \forall\, c, t, s, A_{c,s} = 0 \\
& x_{s,c,t} && = y_{c,t} && \forall\, c, t, s, A_{c,s} = 1 \\
& \sum_{c} K_{c,f} \cdot y_{c,t} && \leq B_{f,t} && \forall\, f, t, B_{f,t} > 0 \\
& y_{c,t} + y_{c',t} && \leq 1 && \forall\, c, c', t, J_{c,c'} = 1 \\
& y_{c,t} && \leq D_{c,e} \cdot h_{e,t} && \forall\, c, e, t \\
& h_{e,t} && = h_{e,t'} && \forall\, e, e', t, L_{e,e'} = 1 \\
& \sum_{e} h_{e,t} && \leq 1 && \forall, e, e', t, L_{e,e'} = 1 \\
& \sum_{t} I_{s,q} \cdot x_{s,c,t} && \leq z_{c,q} && \forall\, c, q, s, A_c = 0 \\
& \sum_{t,s} x_{s,c,t} - \sum_{t,s} x_{s,c',t} && \leq g_{c,c'} \cdot U_c && \forall\, c, c', D_{c,e} = D_{c',e} = 1, \\
& && && A_c = A_{c'} = 0, \rho(c) < \rho(c') \\
& \sum_{t,s} x_{s,c',t} - \sum_{t,s} x_{s,c,t} && \leq g_{c,c'} \cdot U_c && \forall\, c, c', D_{c,e} = D_{c',e} = 1, \\
& && && A_c = A_{c'} = 0, \rho(c) < \rho(c') \\
& \sum_{t} (y_{c,t} + y_{c',t}) - 2 + g_{c,c'} && \leq w_{c,c'} && \forall\, c, c', D_{c,e} = D_{c',e} = 1, \\
& && && A_c = A_{c'} = 0, \rho(c) < \rho(c') \\
& x_{s,c,t} \in \{0,1\} && && \\
& y_{c,t} \in \{0,1\} && && \\
& z_{c,q} \in \{0,1\} && && \\
& w_{c,c'} \in [0,1] && && \\
& g_{c,c'} \in [0,1] && && \\
& h_{e,t} \in \{0,1\} && &&
\end{aligned}
$$

4.7 Complexity

In the following the complexity of ECSS is proven to be $\mathcal{NP}$-hard. This is done by showing that a well-known $\mathcal{NP}$-hard problem, the *Max K-Cut* (Karp [1972]), is polynomial reducible to ECSS ([Wolsey, 1998, chap. 6]). The Max K-Cut problem can be formulated as follows:

Given a multigraph $G = (V, E)$, we wish to divide the vertices of the graph into K disjoint cuts, such that the total number of edges going from one cut to another is maximized. Let $v_1 \dots v_{|K|}$ be the vertices, and let $k_1 \dots k_{|K|}$ denote the cuts. The binary variable $x_{k,n}$ takes the value 1 if vertex v is in cut k and zero otherwise. The Max K-cut is then given by

$$\max \sum_{(v,v') \in E} \sum_{k,k':k'>k} x_{k,v} \cdot x_{k',v'} \tag{22}$$

$$\text{s.t.} \quad \sum_{k} x_{k,v} = 1 \qquad \forall\, v \tag{23}$$

For the ECSS we make the assumption that not more than U_c students have requested a given course. I.e. only one class is created for each course e. For each vertex $v \in V$ in graph G, create a course e. If vertex $v = e$ and vertex $v' = e'$ is adjacent, we create a student s with a request for both courses of v and v', i.e. $R_{s,e} = R_{s,e'} = 1$. We want to find an optimal solution with K time slot. Each cut represents a time slot.

To answer whether G is solvable using K cuts, solve the ECSS instance and check whether all meeting requests are assigned a course using K time slots, i.e. $\sum_{s,c,t} x_{s,c,t} = |\mathcal{R}|$. I.e. a Max K-cut can be found by solving the corresponding ECSS for K time slots as given by reduction, ECSS is therefore $\mathcal{NP}$-hard.

5 Solution methods

It has been chosen to use *Adaptive Large Neighborhood Search* (ALNS) for the ECSS. ALNS is a hyperheuristic first described by Pisinger and Ropke [2005] and is an extension of Large Neighborhood Search (LNS) [Shaw, 1998]. Where most neighborhood search algorithms explicitly define the neighborhood, LNS defines the neighborhood implicitly by a removal and an insertion method. ALNS consists of several removal and insertion heuristics. The ALNS framework has the advantage of having many different neighborhoods, such that the algorithm can explore a large part of the solution space. A pseudo code for the ALNS algorithm as it is presented in Pisinger and Ropke [2010] is shown in Algorithm 1.

Algorithm 1: Adaptive Large Neighborhood Search

Input: a feasible solution $x_{s,c,t}$
1 solution $x^{best} = x$; $\pi = (1, \ldots, 1)$
2 **repeat**
3 select a removal $d \in \Omega^-$ and an insertion heuristic $r \in \Omega^+$ using π
4 $x' = r(d(x))$
5 **if** $c(x') > c(x^{best})$ **then**
6 $x^{best} = x'$
7 **if** *accept*(x', x) **then**
8 $x = x'$
9 update π
10 **until** *stop-criterion met*
11 **return** x^{best}

The set of removal and insertion heuristics are denoted Ω^- and Ω^+, respectively. The variable $\pi \in \mathbb{R}$ which stores the weight of each removal and insertion heuristic is introduces in line 1. Line 5 checks whether the new solution is better than the best known solution. $c(x)$ denotes the objective value of solution x.

In line 7 the solution is evaluated using an accept function. Finally, in line 9 the weights are adjusted based on the performance of each removal and insertion heuristic.

ALNS has been used with success on various problems, especially in *Vehicle Routing Problems*; see e.g. Ropke and Pisinger [2006], Laporte et al. [2010], Azi et al. [2010], Ribeiro and Laporte [2012], Lei et al. [2011]. Other problems such as *Lot-Sizing* (Muller and Spoorendonk [2010]), *Resource Constraint Project Scheduling* (Muller [2009]), *Home Health Care Problem* (Steeg and Schröder [2008]) and *Consultation Scheduling Problem* (Kristiansen et al. [2013]) have also applied ALNS with success.

5.1 Algorithm setup

In the following we describe the main elements of the ALNS used for the ECSS. The user-controlled parameters for the ALNS are tuned in Section 6.

Adaptive search strategy: The choice of the removal and insertion heuristic is governed by a scoring scheme where each heuristic is assigned a weight which is updated due to its past behavior. The search is divided into segments of N_{it} consecutive iterations. Let π_h^i be the measure of the performance of heuristic h in segment i. In the first segment all heuristics are assigned the same weight, $\pi_h^{i_0} = 1$. The probability of choosing heuristic h in segment i is given by $\frac{\pi_{h'}^i}{\sum_h \pi_h^i}$. After N_{it}

iterations the weights are adjusted according to the score obtained during the segment

$$\pi_h^{i+1} = \eta \cdot \frac{\bar{\pi}_h^i}{a_h^i} + (1-\eta)\pi_h^i \tag{24}$$

where $\eta \in [0,1]$ is the *reaction factor* and $\bar{\pi}_h^i$ is the number of times heuristic h has been used in segment i. $\bar{\pi}_h^i$ is the observed weight of the heuristic h in segment i and this is updated in each iteration the heuristic is used. Let x be the current solution and x' be the new found solution. The following scaling parameter for updating π_h^i is introduced

$$\bar{\pi}_h^i = \bar{\pi}_h^i + 5^{\min(\sigma \cdot \text{gap},1)} \tag{25}$$

where gap $= \frac{c(x')-c(x)}{c(x)}$ and $\sigma \in \mathbb{R}^+$ is a parameter which needs tuning.

Acceptance criteria: The accept criteria used in this paper is borrowed from Ropke and Pisinger [2006] and is bases on *Simulated Annealing* (SA). A solution, x is always accepted if $c(x) > c(x^{best})$. However if $c(x) \leq c(x^{best})$, x is accepted with the probability

$$\exp\left(-\frac{c(x^{best}) - c(x)}{T}\right) \tag{26}$$

where the *temperature* T is updated by $T = d_{\text{SA}} \cdot T$. Having $d_{\text{SA}} \in]0,1[$ as the *cooling rate*. The initial temperature is selected using a *temperature control* parameter, $w_{\text{SA}} \in]0,1[$, such that the solution is accepted with probability of 0.5 if the solution is w_{SA} percent worse than the initial solution x_0. Using an acceptance strategy where worse solutions can be accepted with a small probability makes it easier to expand and change neighborhoods.

Stopping criteria: The selection of removal and insertion heuristics are repeated until one of the following stopping criteria is met: (1) the running time exceed the maximum running time of 60 seconds; or (2) the number of iterations without any improvements in the objectives reaches 1,000.

5.2 Removal and insertion heuristics

In this section the removal and the insertion heuristics used for the ECSS are described. Let $m \in \mathbb{N}$ be the number of classes which should be removed from a solution x and let $\bar{\mathcal{C}} \subseteq \mathcal{C}$ be the set of unassigned classes.

5.2.1 Random removal heuristic

The simple removal heuristic removes m elective course classes with students from the solution. The classes are selected at random. This heuristic tends not to give better solutions, but it helps diversify the search. Furthermore a random removal which only removes classes which aren't locked. I.e. it does not remove classes which are a continuation from previous years.

5.2.2 Shaw removal heuristic

The general idea of Shaw removal heuristic is to remove parts of the solution which are somewhat related, as it is expected that they then are reasonably easy to reshuffle, and then creating a new, perhaps better solution (Shaw [1997],Ropke and Pisinger [2006]). Let the *relatedness measure* between meeting i and j be defined by $M(i;j) \in [0;1]$, where a high level corresponds to much relatedness between i and j. Algorithm 2 present a pseudo code for the Shaw Removal heuristic for the ECSS.

Algorithm 2: Shaw removal

Input: A feasible solution $x_{s,c,t}$, $m \in \mathbb{N}$, $p_{\text{shaw}} \in \mathbb{R}^+$
class: c = a randomly selected class with students from $x_{s,c,t}$
set of classes : $D = \{c\}$
while $|D| < m$ **do**
  $\hat{c}$ = randomly selected class from D
  L = all classes from $x_{s,c,t}$ not in D, sorted by similarity to $\hat{c}$
  choose a random number $b^{p_{\text{shaw}}} \in [0, 1[$
  l = element number $b^{p_{\text{shaw}}} \cdot |L|$
  $D = D \cup L[l]$
remove the classes with students in D from $x_{s,c,t}$

The Shaw removal heuristic of this paper is based on the how many students which have requested both course e of c and e' of c'. Let $\mathcal{S}_e$ indicates the set of students which has requested course e. The relatedness measure is then the percentage of students which has requested both courses of the two classes.

$$M(c, c') = \frac{|\mathcal{S}_e \cap \mathcal{S}_{e'}|}{\min(|\mathcal{S}_e|, |\mathcal{S}_{e'}|)} \qquad \text{where } D_{c,e} = D_{c',e'} = 1 \tag{27}$$

A high value of M means that the courses are much related.

Two Shaw heuristics for the ECSS is implemented. One sorted with increasing similarity, (i.e. removing those most related), and one with decreasing similarity (i.e. of those related, remove those less related).

5.2.3 Basic greedy insertion heuristic

A basic greedy algorithm is implemented as one of the insertion heuristic for the ALNS. It simple assigns a class with students to a time slot in order of contribution to the objective. The process is repeated until no more classes with an improvement of the solution can be assigned.

The initial solution for the ALNS is constructed by means of a Basic Greedy Algorithm.

5.2.4 Regret-k insertion heuristic

The regret heuristic is a greedy algorithm with a look-ahead function incorporated. I.e. it tries to improve the myopic behavior of the greedy heuristic. As the name indicates, the heuristic aims at inserting the course class which will be regretted most if not inserted at the given iteration. For each of the unassigned course classes $\bar{c} \in \bar{\mathcal{C}}$, the regret-2 heuristic calculates a regret value equal to the difference in profit between two solutions in which $\bar{c}$ is assigned to its best time slot and its second best time slot. The unassigned class with the highest regret value, is the class which will be regretted most if not inserted in its best time slot, hence this is inserted. Let $o_{\bar{c}}^k$ denote the regret value by inserting class c into the kth best position. The regret value of $\bar{c}$, $r_{\bar{c}}$, is given by

$$r_{\bar{c}} = \sum_{h_2}^{k} (o_{\bar{c}}^1 - o_{\bar{c}}^k) \tag{28}$$

In each iteration the heuristic chooses to insert class $\bar{c}$ according to $\max_{\bar{c} \in \bar{\mathcal{C}}}\{r_{\bar{c}}\}$ For the ECSS, it has been chosen to use at Regret-2, -3 and -4 as insertion methods.

5.2.5 Coupled removal and insertion heuristic

In each iteration the ALNS heuristic chooses a removal and an insertion heuristic based on how well the pair has been performing previously. However some of the removal and the insertion heuristics might not be a good matching due to the structure of the given heuristics. By pairing some of the heuristic, we simple declare which given insertion heuristic a removal heuristic should be paired with. For the ECSS we have created these coupled pairs, all only concerning students. I.e. only

removing and inserting students, not classes. The other previous mentioned heuristics is concerning assigning/unassigning classes with students.

Three coupled pairs have been created. Let $\hat{E}$ be the set of courses where more than one class are created for the given course.

- Remove all students from classes in $\hat{E}$ and insert them using a basic greedy algorithm
- Remove all students from classes in $\hat{E}$ and insert them greedily based on common classes.
- Remove all students from classes in $\hat{E}$ located in the same given time slot $\hat{t}$ and insert them greedily.

5.3 Using exact methods within ALNS

The performance of the ALNS algorithm used of ECSS is evaluated by comparing it with solution found using a state-of-the-art MIP solver (see Section 7). It can also be an advantage to include some exact methods in the ALNS algorithm using a MIP solver. Heuristics which embedded exact solution methods are known as matheuristics.

In this paper we will try to embed exact solution methods within the ALNS by introducing some exact repair methods. We have only focused on the students as we did for the coupled constraints. I.e. the method removes all assigned students or all students assigned to classes in $\hat{E}$. All the classes are fixed. If some students are not removed, they are locked to the respective classes as well. Then by using a MIP solver we try to optimize the problem.

The performance of the ALNS with the exact method is evaluated in Section 7.3.

6 Parameter tuning

For tuning the free parameters of the heuristic, the *F-Race* algorithm has been chosen [Birattari, 2005]. A race algorithm sequentially process data instances using a set of all possible parameter configurations. After each iteration, the parameter configurations which are proven to be statistically inferior are eliminated from the set. In F-Race the *Friedman Two-way Analysis of Variance by Ranks* is used for determining whether any of the parameter configurations are statistically inferior. F-Race has previously been successfully used for parameter tuning for meta-heuristics. (see. e.g. Chiarandini et al. [2006], Pellegrini and Birattari [2007], Kristiansen et al. [2013]). The drawbacks of F-Race are that all possible parameter configurations are considered. I.e. if many parameters with a wide range of values exist, the F-Race becomes inefficient and impractical. In Balaprakash et al. [2007] *Iterative F-Race* (I/F-Race) is introduced. I/F-Race uses a probabilistic model on the set of parameter configurations, such that only a subset of the parameter configurations is generated in each iteration.

In this paper a *manually* I/F-Race is used for tuning. I.e. after each iteration the new configurations are manually created based on the results from the previous iterations.

Table 1 lists the best found parameter configurations. Data instances from 50 different Danish high schools are used. For the SA based acceptance criteria two parameters are tuned. The temperature control, w_{SA}, and the decay parameter, d_{SA}. N_{it} defines the number of iterations between reset. For the ALNS scoring scheme, the tuned parameters are the reaction factor η and the scale parameter σ. ξ_{start} and ξ_{end} are the destroy percentage in the beginning and the end of the running time, respectively. Lastly p_{shaw} is the random indicator in the Shaw removal heuristic.

Table 1: *Final values of tuned parameters, found by the F-Race algorithm with confidence level* $\alpha = 0.05$.

Parameter	w_{SA}	d_{SA}	N_{it}	η	σ	ξ_{start}	ξ_{end}	p_{shaw}
Value	0.01	0.99	50	0.30	5000	0.30	0.0033	20

7 Performance

The purpose of this section is to evaluate the performance of the ALNS algorithm by comparing it with an upper bound found solving the IP model in the state-of-art MIP solver Gurobi 5.01. Both the ALNS and the Gurobi implementation was coded in C# 4.0 and all tests are performed using nUnit 2.6 on a machine with an Intel i7-930@2.8GHz CPU and 12GB of RAM under the Windows operating system. No parallelization has been implemented for improving the performance.

The ALNS algorithm for the ECSS presented in this article was launched for use in Lectio in mid-January 2012 and is as mentioned available for approximately 200 different high schools in Denmark. Up to this date over 500 data sets shared among the high schools, are available in the Lectio database.

7.1 Defining weights

The objectives of the problem are weighted in respect to each other, and the selection of the weights in the implementation in Lectio and for this article has been greatly assisted by MaCom A/S.

The profit of assigning a student to a course is depending on the educational level of the course. Let $\alpha_{c,s} \in \mathbb{N}$ denote the profit of assigning student s to class c. Then $\alpha_{c,s}$ is given by

$$\alpha_{c,s} = \begin{cases} 95 & \text{If the course level of } c \text{ is at a } \textit{basic} \text{ level} \\ 100 & \text{If the course level of } c \text{ is at a } \textit{intermediate} \text{ level} \\ 105 & \text{If the course level of } c \text{ is at a } \textit{advanced} \text{ level} \\ 150 & \text{If } A_{c,s} = 1 \end{cases} \tag{29}$$

Notice that if the student is locked to the class the profit is quite high. This is to give preferential treatments to the classes which are a continuation from the previous year.

The cost of creating an elective course class is depending on the minimum number of classes which is necessary to fulfill all the request for a given course. Let $MIN_e \in \mathbb{Z}^+$ be the minimum theoretical number of classes needed to fulfill all requests for course e. The cost of creating elective course classes is then given by the following

$$\beta_c = \begin{cases} 150 & \rho(c) > Min_e \text{ where } D_{c,e} = 1 \\ 80 & otherwise \end{cases} \tag{30}$$

The cost of each represented common classes in an elective course class is given by

$$\gamma = 10 \tag{31}$$

For each student which the two classes differ from each other is penalized by

$$\delta = 1 \tag{32}$$

7.2 Performance of ALNS compared with Gurobi

As some of the data sets in the database might be duplicates, it has been chosen to evaluate the ALNS algorithm on 80 unique data sets. The data sets are selected randomly, and should cover all possible kinds of setups for the ECSS. The runtime is set to 60 seconds, which is the running time selected upon conversations between MaCom A/S and the users of Lectio. In order to reduce the eventual influence of stochastic behavior, 10 runs on each instance are performed. The Gurobi solver is run for 1 hour, as we want to have good upper bounds.

The percentage gap between the solution found using ALNS and the upper bound is calculated by $\frac{UB - \bar{x}_{ALNS}}{UB}$.

In Table 2 it is seen that the ALNS in average finds solutions less than 1% from optimum and some of the big instances the ALNS outperforms the solutions found using Gurobi. Moreover some of the instances are solved to optimality using ALNS. This is satisfying results.

Table 2: *ALNS for the ECSS on 80 datasets compared with an upper bound using Gurobi 5.0.1. For each dataset is listed the number of students "$|\mathcal{S}|$", number of requests "$|\mathcal{R}|$" and number of courses "$|\mathcal{E}|$", which indicates the size of the given instance. For Gurobi is listed the final objective value, "x", the best bound "UB" and the reported gap between these. For the ALNS, the mean performance of the algorithm over 10 runs, "$\bar{x}$" and column "σ" is the standard deviation. Finally column "Gap(%)" is the percentage difference between ALNS and Gurobi.*

				Gurobi 5.01			ALNS		
	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{R}\rvert$	$\lvert\mathcal{E}\rvert$	x	UB	Gap[%]	$\bar{x}$	σ	Gap[%]
Aabenraa	20	20	3	1630.0	1630.0	0.0	1630.0	0.0	0.0
Aalborg	212	539	16	79010.0	79010.0	0.0	79010.0	0.0	0.0
Aarhus1	341	471	34	67745.0	67745.0	0.0	67745.0	0.0	0.0
Aarhus2	338	481	28	59134.0	59451.0	0.5	59099.6	4.8	0.6
Aars1	219	365	23	39315.0	39315.0	0.0	39315.0	0.0	0.0
Aars2	220	585	29	65615.0	65615.0	0.0	65615.0	0.0	0.0
Alssund	183	338	17	32827.0	33003.0	0.5	32824.6	3.6	0.5
Bagsvaerd1	49	75	10	10350.0	10350.0	0.0	10350.0	0.0	0.0
Bagsvaerd2	110	152	21	20210.0	20210.0	0.0	20210.0	0.0	0.0
Broenderslev1	312	515	22	49458.0	49943.0	1.0	49292.6	158.6	1.3
Broenderslev2	312	514	22	49357.0	49835.0	1.0	49089.9	149.1	1.5
CPHWEST1	249	426	32	48405.0	48498.0	0.2	48405.0	0.0	0.2
CPHWEST2	251	480	33	56041.0	56057.0	0.0	56041.0	0.0	0.0
DetFrie1	49	49	3	7110.0	7110.0	0.0	7110.0	0.0	0.0
DetFrie2	112	112	3	10839.0	10839.0	0.0	10825.0	0.0	0.1
Dronninglund1	299	522	27	75495.0	75495.0	0.0	75495.0	0.0	0.0
Dronninglund3	297	519	25	75380.0	75380.0	0.0	75380.0	0.0	0.0
Esbjerg	595	789	34	90050.0	90713.0	0.7	90006.6	27.7	0.8
EUCNORD	335	735	45	95080.0	95089.0	0.0	94993.4	78.3	0.1
Falkoner1	421	1080	49	131264.0	131728.0	0.4	130877.7	66.9	0.7
Falkoner3	649	1666	85	241015.0	241015.0	0.0	240641.0	788.5	0.2
Falkoner4	537	1376	57	177440.0	177674.0	0.1	177368.2	25.3	0.2
Falkoner5	431	617	42	64789.0	65210.0	0.7	64763.2	5.3	0.7
Falkoner6	297	456	34	41395.0	41828.0	1.1	41373.4	4.2	1.1
Falkoner7	742	1656	76	215578.0	216076.0	0.2	215446.0	53.0	0.3
Falkoner8	446	1266	56	160169.0	160362.0	0.1	160083.9	63.3	0.2
Falkoner10	335	520	34	50782.0	51428.0	1.3	50756.8	13.3	1.3
Fjerritslev	456	822	71	113706.0	113716.0	0.0	113701.2	2.5	0.0
Frederikssund1	294	475	25	53300.0	53300.0	0.0	53300.0	0.0	0.0
Frederikssund2	193	351	18	51130.0	51130.0	0.0	51130.0	0.0	0.0
Greve1	306	922	31	102351.0	103436.0	1.1	102347.8	89.8	1.1
Greve2	306	892	31	99500.0	100689.0	1.2	98980.8	246.5	1.7
Gribskov1	394	648	33	71895.0	72000.0	0.2	71325.6	129.7	1.0
Gribskov3	220	474	24	46632.0	46747.0	0.3	45968.0	296.2	1.7
GUAasiaat	71	82	12	11820.0	11820.0	0.0	11820.0	0.0	0.0
Haderslev1	447	1034	37	112065.0	113494.0	1.3	110729.1	1160.8	2.5
Haderslev2	470	1063	64	150480.0	150490.0	0.0	150480.0	0.0	0.0
Hasseris1	400	508	21	54019.0	54074.0	0.1	53846.0	92.2	0.4
Hasseris2	400	508	21	50859.0	51035.0	0.4	50384.8	92.8	1.3
HoejeTaastrup1	241	416	19	41010.0	41014.0	0.0	40909.4	80.9	0.3
HoejeTaastrup3	233	380	17	55330.0	55330.0	0.0	55330.0	0.0	0.0
Holstebro1	93	202	9	29420.0	29420.0	0.0	29420.0	0.0	0.0
Holstebro2	626	912	35	111064.0	111577.0	0.5	111021.0	6.7	0.5
Holstebro3	93	202	9	29420.0	29420.0	0.0	29420.0	0.0	0.0
Horsens	380	662	33	96660.0	96660.0	0.0	96660.0	0.0	0.0
Koebenhavns	289	816	31	100920.0	100930.0	0.0	100842.2	39.9	0.1
KoebenhavnsTek	166	169	7	24790.0	24790.0	0.0	24790.0	0.0	0.0
Koege	369	546	31	79360.0	79360.0	0.0	79360.0	0.0	0.0
Kongsholm1	383	760	46	109570.0	109570.0	0.0	107661.0	1655.6	1.8
Kongsholm2	365	974	40	126975.0	128862.0	1.5	126381.3	717.6	2.0
Langkaer	503	795	51	113540.0	113540.0	0.0	113540.0	0.0	0.0
Mariagerfjord1	365	521	24	75670.0	75670.0	0.0	75670.0	0.0	0.0
Mariagerfjord2	382	611	29	88170.0	88170.0	0.0	88170.0	0.0	0.0
Middelfart	390	1332	61	170243.0	170558.0	0.2	169663.3	232.0	0.5
Munkensdam	482	6456	231	349400.0	349400.0	0.0	349400.0	0.0	0.0
Noerresundby	563	1456	55	180916.0	181614.0	0.4	180807.5	50.1	0.5
NZahles	189	271	20	31955.0	31958.0	0.0	31955.0	0.0	0.0
Oeregaard1	239	489	13	71510.0	71510.0	0.0	71510.0	0.0	0.0
Oeregaard2	547	826	27	96376.0	97212.0	0.9	96425.0	11.6	0.8
Risskov1	539	784	38	91856.0	92529.0	0.7	91941.6	2.8	0.6
Risskov2	258	480	20	48993.0	49733.0	1.5	49038.8	3.8	1.4
Roedovre	350	868	34	100907.0	101414.0	0.5	100871.6	13.4	0.5
RoskildeKatedral	383	1145	40	119565.0	128359.0	7.4	126507.0	185.8	1.5
RoskildeTek1	358	529	21	77670.0	77670.0	0.0	77670.0	0.0	0.0
RoskildeTek2	358	688	30	91518.0	91520.0	0.0	91500.0	0.0	0.0
Rybners1	238	313	15	32454.0	32457.0	0.0	32336.6	130.0	0.4
Rybners2	352	443	11	40462.0	41245.0	1.9	40423.2	5.1	2.0
RybnersGym	192	384	20	55600.0	55600.0	0.0	55600.0	0.0	0.0
Rysensteen	285	570	19	56198.0	56429.0	0.4	55803.4	141.5	1.1
Skanderborg	245	439	18	41782.0	42246.0	1.1	41787.6	10.3	1.1

Continued on next page

Table 2 – continued from previous page

				Gurobi 5.01			ALNS		
	$\|\mathcal{S}\|$	$\|\mathcal{R}\|$	$\|\mathcal{E}\|$	x	UB	Gap[%]	$\bar{x}$	σ	Gap[%]
Slagelse1	974	1660	54	158025.0	180308.0	14.1	177314.5	129.9	1.7
Slagelse2	751	1345	45	127620.0	150895.0	18.2	148045.5	277.4	1.9
Slagelse3	1272	2221	57	131200.0	234645.0	78.9	229250.5	308.0	2.4
Slagelse6	1261	2289	113	329730.0	329730.0	0.0	329730.0	0.0	0.0
Slagelse7	329	508	23	63830.0	63836.0	0.0	63824.8	6.4	0.0
Struer	534	805	42	103161.0	103170.0	0.0	103140.2	5.4	0.0
Taarnby	298	760	29	110490.0	110490.0	0.0	110490.0	0.0	0.0
Varde2	230	677	30	98680.0	98680.0	0.0	98680.0	0.0	0.0
Vejlefjord	100	226	35	29985.0	29985.0	0.0	29985.0	0.0	0.0
Viby	232	472	18	47438.0	47798.0	0.8	47451.0	1.7	0.7
Average	359.5	756.6	34.4	-	-	1.7	-	-	0.5
Max	1272.0	6456.0	231.0	-	-	79.1	-	-	2.3

7.3 Performance using ALNS with exact methods

The exact methods are implemented as a repair method and as a hill climber, and both are tested using a running time of 2 and 5 seconds. For the hill climber this means that the running time for the ALNS is shortened by the running time of the hill climber such that the total running time still is 60 seconds.

The performance of the ALNS with some exact solution methods embedded are shown in Table 3. For the exact methods Gurobi 5.0.1 is used.

Table 3: *Average performance using Gurobi and ALNS with different exact solution methods incorporated. The average is taken over 80 different dataset. The second column is the performance using Gurobi 5.01 and the third column is ALNS without any exact methods. Column 4 and 5 are average performance of the ALNS with a exact repair method. And Column 6 and 7 are the average performance using ALNS with a Hill Climber attached. Both tested with running time of 2 and 5 seconds*

	Gurobi 5.01	ALNS	w. exact rep. (2 sec)	w. exact rep (5 sec)	w. exact HC (2 sec)	w. exact HC (5 sec)	w. exact rep & HC (2 sec)
Average	1.77	0.52	0.61	0.66	0.52	0.50	0.64
Max	78.90	2.50	2.70	4.07	2.72	2.20	3.45

As it is seen all the different ALNS algorithms outperform Gurobi on the average performance. This is due to the bad performance of Gurobi on the large instances. Furthermore, it is seen that the pure performs better than many of the ALNS with exact methods. This is mainly due to the running time and the scoring scheme. The running time has an important influence on the solution results. As the running time for the algorithm is 60 seconds, the running time of the exact repair methods cannot be too long. If it's more than 2-5 seconds it makes a significant decrease in the number of iterations we are able to perform. Yet, when having a low running time for the exact repair method it may results in poor performance of large instances and the solutions might not fulfill the acceptance criteria.

However, we can conclude that by embed some exact methods we might be able to improve the ALNS algorithm. For the ECSS using an exact hill-climber of 5 seconds at the end, improves the average performance a little.

8 Final remarks and outlook

In this paper the Elective Course Student Sectioning has been described in details and formulated as an MIP model. ALNS has proven to be a successful method to establish solutions to the problem. The ALNS algorithm has been implemented in the Cloud-based software system Lectio and is hence available for more than 200 Danish high schools. For testing the performance of the ALNS algorithm, 80 real life instances from different high schools have been used. In average ALNS finds

solutions within 1% of the optimum and for large instances the algorithm outperforms Gurobi, which is very satisfying results.

It was shown that for some of the instances it could be an advantage to embed some exact methods in the ALNS. However more testing is needed with open source MIP solvers, as Gurobi is not a possibility as all the clients need a license.

Of future research within student sectioning at high schools it could be interesting to expand the model such that it contains the creation of the common classes.

Acknowledgments

The authors thank Michael Bigom Herold from MaCom A/S for kindly helping determining the problem and setting the weights for the problem, and MaCom A/S for providing all the data.

References

N. Azi, M. Gendreau, and Jean-Yves Potvin. *An Adaptive Large Neighborhood Search for a Vehicle Routing Problem with Multiple Trips.* CIRRELT, 2010.

P. Balaprakash, M. Birattari, and T. Stützle. Improvement strategies for the f-race algorithm: sampling design and iterative refinement. In *Proceedings of the 4th international conference on Hybrid metaheuristics*, HM'07, pages 108–122, Berlin, Heidelberg, 2007. Springer-Verlag.

M. Birattari. *The Problem of Tuning Metaheuristics as seen from a Machine Learning Perspective*, volume 292 Dissertations in Artificial Intelligence - Infix. Springer, 1 edition, 2005.

E.K. Burke and S. Petrovic. Recent research directions in automated timetabling. *European Journal of Operational Research*, 140(2):266 – 280, 2002. ISSN 0377-2217. doi: 10.1016/S0377-2217(02)00069-3.

M.W. Carter and G. Laporte. Recent developments in practical course timetabling. In Edmund Burke and Michael Carter, editors, *Practice and Theory of Automated Timetabling II*, volume 1408 of *Lecture Notes in Computer Science*, pages 3–19. Springer Berlin / Heidelberg, 1998.

M. Chiarandini, M. Birattari, K. Socha, and O. Rossi-Doria. An effective hybrid algorithm for university course timetabling. *Journal of Scheduling*, 9:403–432, 2006. ISSN 1094-6136.

P. de Haan, R. Landman, G. Post, and H. Ruizenaar. A case study for timetabling in a dutch secondary school. In Edmund Burke and Hana Rudova, editors, *Practice and Theory of Automated Timetabling VI*, volume 3867 of *Lecture Notes in Computer Science*, pages 267–279. Springer Berlin / Heidelberg, 2007.

W. Erben and J. Keppler. A genetic algorithm solving a weekly course-timetabling problem. In Edmund Burke and Peter Ross, editors, *Practice and Theory of Automated Timetabling*, volume 1153 of *Lecture Notes in Computer Science*, pages 198–211. Springer Berlin / Heidelberg, 1996.

R.M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations: Proc. of a Symp. on the Complexity of Computer Computations*, 19:85–103, 1972.

Simon Kristiansen, Matias Sørensen, and Thomas R. Stidsen. Elective course planning. *European Journal of Operational Research*, 215(3):713 – 720, 2011. ISSN 0377-2217. doi: 10.1016/j.ejor.2011.06.039.

Simon Kristiansen, Matias Sørensen, Michael B. Herold, and Thomas R. Stidsen. The consultation timetabling problem at danish high schools. *Journal of Heuristics*, 19(3):465–495, 2013.

G. Laporte, R. Musmanno, and F. Vocaturo. An adaptive large neighbourhood search heuristic for the capacitated arc-routing problem with stochastic demands. *Transportation Science*, 44 (1):125–135, 2010.

H. Lei, G. Laporte, and B. Guo. The capacitated vehicle routing problem with stochastic demands and time windows. *Computers & Operations Research*, 38(12):1775 – 1783, 2011. ISSN 0305-0548. doi: DOI: 10.1016/j.cor.2011.02.007.

T. Müller and K. Murray. Comprehensive approach to student sectioning. *Annals of Operations Research*, 181:249–269, 2010. ISSN 0254-5330.

Laurent Flindt Muller and Simon Spoorendonk. A hybrid adaptive large neighborhood search algorithm applied to a lot-sizing problem. Technical report, DTU Management Engineering, 2010.

L.F. Muller. An adaptive large neighborhood search algorithm for the resource-constrained project scheduling problem. In *MIC 2009: The VIII Metaheuristics International Conference*, 2009.

P. Pellegrini and M. Birattari. Implementation effort and performance. pages 31–45. 2007.

N. Pillay. An overview of school timetabling research. In *Proceedings of the International Conference on the Theory and Practice of Automated Timetabling*, pages 321–335, Belfast, United Kingdom, 2010.

D. Pisinger and S. Ropke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34:2403–2435, August 2005. ISSN 0305-0548.

D. Pisinger and S. Ropke. Large neighborhood search. In Michel Gendreau and Jean-Yves Potvin, editors, *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*, pages 399–419. Springer US, 2010. ISBN 978-1-4419-1665-5.

Gerhard Post, Luca Di Gaspero, Jeffrey H. Kingston, Barry McCollum, and Andrea Schaerf. The third international timetabling competition. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, Son, Norway, August 2012.

Glaydston Mattos Ribeiro and Gilbert Laporte. An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem. *Computers & Operations Research*, 39(3):728 – 735, 2012. ISSN 0305-0548.

S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40:455–472, November 2006. ISSN 1526-5447.

H. Rudova and K. Murray. University course timetabling with soft constraints. In *Practice And Theory of Automated Timetabling IV.*, pages 310–328, 2003.

A. Schaerf. A survey of automated timetabling. *Artificial Intelligence Review*, 13:87–127, 1999. ISSN 0269-2821.

P. Shaw. A new local search algorithm providing high quality solutions to vehicle routing problems, 1997.

P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In Michael Maher and Jean-Francois Puget, editors, *Principles and Practice of Constraint Programming — CP98*, volume 1520 of *Lecture Notes in Computer Science*, pages 417–431. Springer Berlin / Heidelberg, 1998.

Matias Sørensen, Simon Kristiansen, and Thomas R. Stidsen. International timetabling competition 2011: An adaptive large neighborhood search algorithm. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, pages 489–492. SINTEF, 2012.

Jörg Steeg and Michael Schröder. A hybrid approach to solve the periodic home health care problem. In Jörg Kalcsics and Stefan Nickel, editors, *Operations Research Proceedings 2007*, volume 2007 of *Operations Research Proceedings*, pages 297–302. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-77903-2.

L.A. Wolsey. *Integer programming.* Wiley-Interscience publication, 1998.