Department of Applied Mathematics
Faculty of EEMCS

University of Twente
The Netherlands

P.O. Box 217
7500 AE Enschede
The Netherlands
Phone: +31-53-4893400
Fax: +31-53-4893114
Email: memo@math.utwente.nl
www.math.utwente.nl/publications

Memorandum No. 1707

Clusterschemes in Dutch secondary schools

G.F. Post and H.W.A. Ruizenaar

January, 2004

ISSN 0169-2690

Clusterschemes in Dutch secondary schools

Gerhard Post and Henri Ruizenaar*

Department of Mathematical Sciences, University Twente,
P.O. Box 217, 7500 AE Enschede, The Netherlands

Abstract. The first step in constructing timetables in secondary schools in Netherlands consists of constructing the clusterschemes for the higher classes. A clusterscheme contains clusterlines with optional subjects that will be taught in parallel; the problem is to divide these optional subjects in clusterlines, such that the number of hours needed is as low as possible. We describe an efficient branch-and-bound method for this problem. Moreover we describe a fast heuristic to assign students to subjectgroups. Some test results are presented.

Keywords: Timetabling, High school, Scheduling.

AMS Subject classification: 90B35.

1 Introduction

The construction of (year) timetables in secondary schools is a well-known problem. The precise form of this problem varies from country to country. A specific point in Dutch secondary schools is that the students are divided up in divisions, and have optional subjects in the higher classes. These optional subjects form the principal bottleneck in constructing timetables, for at least two reasons:

- When considering a possible arrangement, a single student can make the timetable impossible.
- Without special methods, like the one of clusterschemes we describe below, the evaluation of a timetable is very time-consuming as we have to consider individual students all the time.

The solution which is usually followed in the Netherlands is to make 'clusterschemes' for the higher classes. This means that we combine several groups in a (cluster)line, and we schedule all the groups in the line at the same time-slots[1]. The arrangements of groups in a line should be such that all students can attend their optional subjects. Once a clusterscheme is constructed, we only need to place this line in the timetable, instead of all different groups and students. A clusterscheme is good when it occupies as few as possible time-slots. A bad clusterscheme will give rise to a timetable with many free periods, which we try to avoid.

* This research was supported by NWO grant 636.000.000.02N18.

[1] It can happen that not all subjects have the same amount of lessons. However we need to reserve as many time-slots as lessons for the subject with the most lessons.

1.1 Literature

A nice survey on timetabling called 'A survey of timetabling' by A. Schaerf [4] appeared in 1995. Here one can find references to papers on timetabling using direct methods, network methods, tabu search, simulated annealing, genetic algorithms, ... Since then many more papers appeared, usually with the same techniques. Papers, not directly for the Dutch situation, but applicable nevertheless are [1] and [7]. They describe, comparable with the situation in the Netherlands, secondary schools where (a part of) the subjects is student-depending.

On the specific Dutch situation, there are not so many scientific papers. Most recently there are studies by R. Willemen en H. ten Eikelder, which consider several complexity problems, and also propose a new method for the Dutch secondary schools [6]. In particular the cluster schemes described above are not used. In 1999 appeared a master's thesis on the clustering problem by B. van Kesteren [3]. He proposes a branch-and-bound algorithm, with randomization in case the process seems to get stuck. Apart form these papers, and some other Master's studies, there exist some older publications, [2, 5]. In [5], which appeared just after the introduction of optional subjects in the Netherlands, Simons proposes a computer program to improve a cluster scheme, using some local search procedure. In [2], the author describes experiments with heuristics to construct timetables, based on some heuristic strategies. It is assumed that the cluster schemes and assignments are already settled.

2 Problem description

2.1 Background

Let us describe the Dutch secondary school system in some more detail. After primary school students 12 years old are coming into the *brugklas* (the first class) of the 3 divisions *VMBO*, *HAVO*, and *VWO* or a combination of those three. After this first year students continue classes in several directions on their way to the final exams of the divisions VMBO (4 years), HAVO (5 years) or VWO (6 years). These 3 divisions prepare the student for *MBO* (middle professional education), *HBO* (higher professional education) and university, respectively. The *upper classes* of secondary schools are the subdivisions 3 and 4 VMBO, 4 and 5 HAVO, and 4, 5 and 6 VWO. All the other classes are called the lower classes. In the upper classes of secondary schools students can follow, depending on their interest and ability, one of 4 *profiles*. In these upper classes the students choose, in addition to the compulsory subjects, a number of optional subjects.

As described above the problem we address is making clusterschemes for subdivisions in upper classes. For this we assume that the following facts are known:

- the subjects to choose in each subdivision and their number of weekly lessons;
- the optional subjects of all students;
- the number of groups of each subject.

It is possible that students from the same subdivision and the same profile have a different number of optional subjects. In addition to the optional subjects chosen by a student he attends the compulsory subjects in the subdivision. In constructing the clusterscheme we will discard them.

Depending on the number of students attending a subject, the subject can consist of one or more groups. The maximum number of students attending a group is set to approximately 32 by the school management. If the number of students is larger than 32, the group will be divided into 2 or more groups. If the number of students is too small, the management can decide to remove the group, or to merge this group with a small group of the same subject in another subdivision. Merging has no influence on making clusterschemes, but has strong influence on making time-tables in the upper classes. (The merged clusterlines are to be scheduled at the same time-slots). For making timetables for subdivisions are usually 40 periods (or time-slots) available: 5 days of 8 lessons each. Due to organization arrangements (meetings) it is not possible to use all of those 40 periods, so in practice timetables must consist of less than 40 periods. A part of these periods are spent on the compulsory subjects.

As one can easily imagine, scheduling the optional subjects form a major bottleneck in constructing (good) timetables. To get this process under control, a process called *clustering* is performed first. This means that we try to find a so-called *clusterscheme* for every subdivision. The clusterscheme consists of *clusterlines*. Each clusterline contains groups of optional subjects that will be scheduled at the same periods. The arrangement of groups and the assignment of a student to a group must be such that each student can attend the optional subjects he has chosen, i.e. for each student it is possible to make an assignment to groups of the optional subjects, such that these groups belong to different clusterlines.

Every clusterline takes a number of positions that equals the maximum of the number of lessons for subjects in that clusterline. This we call the *clusterline length*. The *clusterscheme length* is the sum of the lengths of the contained clusterlines.

The main goal is to minimize the length of all clusterschemes, because in that case the number of free periods (no contact lessons) for students as well as teachers will be minimal. Moreover it gives more freedom in timetabling the subdivision. Because of the restriction of the number of periods in a timetable in some subdivions it is critical to find this minimal length. It can happen that the school management decides to add extra groups to the subdivision, to decrease its clusterscheme length. This, however, we will not consider.

The secondary goal is to balance the groups of the same subject. If, for instance, there are three groups for the subject mathematics, and 79 students, then the best balance is that the groups contain 26, 26 and 27 students. Forbidden is the combination 23, 23 and 33, as this violates the maximum size (32) of a group, while the combination 32, 32, 15 is not desirable.

2.2 Sizes of instances

In the Netherlands a secondary school usually contains between 800 and 1500 students. These students are divided in divisions, and divisions are divided into 15 subdivisions, making that a subdivision in the upper classes contains up to (approximately) 100 students. Depending on the subdivision, the student can choose between 3 and 9 subjects out of 4 to 21 subjects.

3 Modeling and observations

3.1 The mathematical program

To make our problem absolutely clear, we formulate it as a mathematical program for one subdivision. The index j will be used for the subjects, i for the clusterlines, and k for the students of this subdivision. We introduce the decision variables x_{ij} and y_{ijk} by

$$x_{ij} = \begin{cases} 1 & \text{if a group of subject } j \text{ is in clusterline } i \\ 0 & \text{otherwise} \end{cases}$$

and

$$y_{ijk} = \begin{cases} 1 & \text{if student } k \text{ attends subject } j \text{ is in clusterline } i \\ 0 & \text{otherwise} \end{cases}$$

An instance provides values to the following variables:

$$\begin{aligned} L_j &= \text{the number of lessons of subject } j \\ G_j &= \text{the number of groups of subject } j \\ S_{jk} &= \begin{cases} 1 \text{ if student } k \text{ chose subject } j \\ 0 \text{ otherwise} \end{cases} \\ A_j &= \text{the average number of students in a group of subject } j \\ C &= \text{the maximum length of the clusterscheme} \end{aligned}$$

The mathematical problem we try to solve is then

$$\text{Minimize} \left\{ \alpha \sum_i C_i + \beta \sum_{i,j} x_{ij} (\sum_k y_{ijk} - A_j)^2 \right\}$$

such that

$$C_i = \max_j (L_i x_{ij}) \qquad (\forall i) \qquad \textit{(definition of clusterline length } C_i\textit{)} \qquad (1)$$

$$\sum_i C_i \leq C \qquad \textit{(length of clusterscheme is at most } C\textit{)} \qquad (2)$$

$$\sum_i x_{ij} = G_j \qquad (\forall j) \qquad \textit{(all groups of } j \textit{ must be put in a line)} \qquad (3)$$

$$\sum_i y_{ijk} = S_{jk} \qquad (\forall j,k) \qquad \textit{(k must attend his optional subjects)} \tag{4}$$

$$\sum_j y_{ijk} \leq 1 \qquad (\forall i,k) \qquad \textit{(k can attend at most 1 subject in a line)} \tag{5}$$

$$y_{ijk} \leq x_{ij} \qquad (\forall i,j,k) \qquad \textit{(subject must be in a line if it is chosen)} \tag{6}$$

Apart from the non-linear term in the objective function this mathematical program is (easily converted to) an integer linear program. The constants α and β are chosen, such that the first term $\alpha \sum_i C_i$ in the objective function dominates the second term, i.e. $\alpha >> \beta > 0$. We omitted one constraint, namely the maximum group size. We consider this as a soft constraint, which is handled (more or less) by the term $\sum_{i,j} x_{ij}(\sum_k y_{ijk} - A_j)^2$ in the objective function.

3.2 The number of clusterlines

For j and k the ranges of the sums are clear, being all subjects, and all students, respectively. For i this is not so clear; the number of clusterlines, let us call it I, is not known in advance. However we can always assume that $I = \sum_j G_j$, as in this case we have for each group a clusterline available. Of course, this will not be a good clusterscheme, as $\sum_i C_i = \sum_j G_j L_j$; probably the constraint (2) is not satisfied. In case of a feasible solution, we can make some estimates. The number of non-empty clusterlines is denoted by I^*.

- Let N_k be the number of subjects chosen by student k. Then the number of non-empty clusterlines is at least N_k.

- Suppose ℓ is the minimum number of lessons for all subjects, i.e.

  $$\ell = \min_j L_j,$$

  and student k has chosen N_k subjects, that totally take $\mathcal{L}_k$ lessons. Then

  $$I^* \leq N_k + (C - \mathcal{L}_k)/\ell \tag{7}$$

  Proof. Suppose the clusterlines 1 to N_k are attended by student k. Then we have

  $$\sum_1^{N_k} C_i \geq \mathcal{L}_k,$$

  and hence, by using constraint (2),

  $$C \geq \sum_i C_i \geq \mathcal{L}_k + (I^* - N_k)\ell$$

  and the claim easily follows.

It is superfluous to have empty clusterlines. In our computations we therefore will always assume that the number of clusterlines is

$$I = \min_k (N_k + (C - \mathcal{L}_k)/\ell). \tag{8}$$

In case that $I > N_k$ for a certain k we have infeasibility, and can stop the computation.

3.3 The strategy

Looking at our mathematical program, we note that for fixed x_{ij}, so a fixed clusterscheme, we can check whether the clusterscheme length is C or less, hence whether constraint (2) is satisfied. If not we can stop, if yes, we consider the remaining problem of constraints (4) and (5). For each k we have a matching problem in a bipartite graph. On the left side of the bipartite graph we have the subjects of student k (those subjects j for which $S_{jk} = 1$), and on the right side the clusterlines. We connect subject j and clusterline i in case $x_{ij} = 1$. The constraints (4) and (5) tell that we need a matching of cardinality N_k.

Instead of fixing all subjects, we can also consider fixing the first j^* subjects. This way we get a partial clusterscheme: a scheme in which only the groups of the subjects 1 to j^* are put into the clusterscheme. In this partial clusterscheme we must be able to assign the subjects 1 to j^* to all students. Otherwise we can stop. In more detail, we order the subjects, and consider one by one the groups of this subject in a clusterline. As soon as all groups of a subject are placed, we try to assign the students with this optional subject to a group. Thus we obtain the following basic branch-and-bound method for our program.

```
00  program
01    active group := first group;
02    while first group is not tried in all lines do
03      find next possible line for the active group;
04      if such line does not exist then
05        active group := previous group
06      else
07        if group is last group of subject then
08          assign all students to a group of this subject;
09          if all assignments are possible then
10            if active group = last group then
11              consider the solution
12            else active group := next group
13            end if
14          else << do nothing >>
15          end if of line 09
16        else active group := next group
17        end if of line 07
18      end if of line 04
19    end while
20  end program.
```

This enumeration is the basis of our method. However efficient branching and efficient bounding have to be added to make it work for real life instances. In the next two sections we describe the ways we try to improve the performance of the algorithm.

4 Branching

In the program above, a complete enumeration is described. For instances from real life this program will not stop in our life-time. Let us for example consider an instance with 17 subjects, of which 10 subject have only one group, 6 subjects have 2 groups and one subject has 3 groups. There are 100 students that take 6 optional subjects. The average group size will be 24 in this case. Suppose we restrict our search to clusterschemes with at most 8 lines. (It is a priori uncertain if such a solution exists or not). The number of possible group arrangements with 8 lines will be

$$8^{10} \cdot \binom{8}{2}^6 \cdot \binom{8}{3} \approx 2.9 \cdot 10^{19}.$$

It is clear that we have to avoid considering all these possibilities. In particular constructing all 100 matchings should be avoided. For this we apply two methods. The first method is removing symmetry, the second method is preprocessing the students.

4.1 Symmetry

A permutation of the lines gives a new solution (usually) which is totally equivalent with the original one. Hence if we remove the symmetry in a scheme with 8 lines, the reduction is by a factor 8!. This is clearly worthwhile. The way we do it is by choosing the 'best' student k (to be explained in subsection 4.3), and place groups of the subjects of this student in the first N_k lines. These groups we will fix throughout. This reduces the symmetry already greatly, but does not remove it completely. To reduce the symmetry even more we apply the following methods:

- *Compactness.* If we place a group in line i, then all lines $1, \ldots, i-1$ are non-empty.
- *Interchange of groups.* For two groups of the same subject we assume that the linenumber of the first group is lower than the line number of the second group. This holds if these groups are not fixed by the best student (as described above).

Even this does not remove all symmetry. The situation that can occur is that a subject $S3$ with (say) 2 groups is fixed in line 1. The subjects $S1$ and $S2$ are placed before the non-fixed group of $S3$ is. Then the following can happen

$$\begin{array}{l}\text{line 1: } S3\ S1\\ \hline \text{line 2: } S2\ S3\end{array} \quad \text{and} \quad \begin{array}{l}\text{line 1: } S3\ S2\\ \hline \text{line 2: } S1\ S3\end{array}$$

If the fixed group of $S3$ is not there, the second solution is forbidden; line 1 would still be empty at the moment we start to place subject $S1$. It seems hard to avoid this kind of symmetry.

4.2 Ordering of subjects

Calculating the matchings at each step is very time consuming. For this reason we accumulate some statistics at the start. Let us call an optional subject with k groups a 'k-grouper'.

- For any two 1-groupers, check if they have a student in common. If this is the case, the corresponding groups have to be placed in different lines.
- For any two 1-groupers, and a 2-grouper, check if there is a student with this combination of subjects. If this is the case, the corresponding 4 groups have to occupy at least 3 lines.

To use these statistics in the most efficient way, we decided to order the groups according to the number of groups; the 1-groupers first, then the 2-groupers, then the 3-groupers, and so on. Note that placing the 1-groupers is now a graph coloring problem: the groups are the vertices of the graph, while two vertices are connected if the corresponding groups have a student in common (this is part of the statistics above). We have to construct all coloring with a certain maximum number of colors. Since a color corresponds to a line in the clusterscheme, the maximum number of colors is restricted by (8), the maximum number of (non-empty) lines we need to consider. Hence for the 1-groupers, we can specify line 03 in the program above: the next possible line corresponds to the next possible color, where the previous subjects are already colored.

When placing a 2-grouper, we similarly use the statistics. At the moment we try to place the second group of this subject, we collect all 1-groupers in the two corresponding lines. If the two 1-groupers with the 2-grouper is chosen by a student, the combination is forbidden, and need not be considered.

In principal we could accumulate more statistics: for any combination of subjects we could check if a student has chosen this combination. We decided to stop with the statistics above for two reasons:

1. The amount of combinations is exponential in the number of subjects. Hence checking it all will take a lot of time.
2. Our instances contain usually many 1-groupers, not so many 2-groupers, and (maybe) some 3-groupers. Thus by the ordering of subjects chosen, and the statistics above we avoided already most of the matchings. Moreover the remaining cases are less restrictive, as the more groups for a subject, the more possibilities to avoid conflicts.

4.3 The best student

For an efficient implementation we ordered the subjects increasingly with the number of groups (see above). The next problem is to choose a student for whom we fix groups in clusterlines. We want to have as much information as possible from this student. If we take a student with only 1-groupers, we have no additional information: the information in the statistics about 1-groupers ensures that this student can be placed anyhow. A similar statement holds partly for the 2-groupers a student has chosen. Hence we come to the following definition of a best student.

> The best student is a student with the most 3-groupers; among these students the one with the highest number of lessons is better.

In some instances there are no 3-groupers. In this case we simply choose a student with the maximum number of lessons. This number of lessons is important for bounding, see subsection 5. Note that fixing a 3-grouper reduces the number of combinations for this subject from $\binom{I}{3}$ to $\binom{I-1}{2}$. For $I = 10$ the difference is a factor 5.

4.4 Ordering the students

Our computer program proceeds subject by subject. If all groups of a subject are placed in lines, we check for all students if assignment to a group is possible. Of course we stop as soon as one student can not be placed. In a certain configuration one can imagine that one student is more restrictive than another. It would be a pity if such a student A is at the end of the list of students we check. For this reason we decided to move up student A each time he is the first that can not be assigned.

In a real life instance millions of matchings are considered for approximately 100 students. By moving up restrictive students step by step, we only keep the history of the last tries we made. If in a neighborhood of the current partial clusterscheme a student is the most restrictive, he will be on top of the list. Here 'neighborhood' is defined by our enumeration: two partial clusterschemes are 'near' means that they are close in the enumeration.

5 Bounding

The next part we have to take care of is bounding. In our method we have two things to bound on: first the length of the scheme, and second, after assigning all students, the balancing of the groups.

5.1 The length of the clusterscheme

As we explained in subsection 3.1 our main object is to minimize the length of the scheme. To restrict the search space the user can choose the maximum

length, see equation (2). During the progress of the search, we keep track of all clusterline lengths. Note that the fixed groups already contribute to the length of the clusterscheme from the beginning. Moreover, based on equation (8), we have an upperbound for the number of lines. In this way efficient bounding is possible.

5.2 The size of groups

The number of students in a group of a subject should be close to the average number of students for this subject. We put quadratic costs on the deviation from the average, see subsection 3.1. The calculation of the deviation is not satisfactory: usually the average is fractional, and we can not avoid deviation from the average. For this reason we allow deviation 1; so instead of taking $(y_{ijk} - A_j)$, we take $|y_{ijk} - A_j| - 1$ in case $x_{ij} = 1$ and $|y_{ijk} - A_j| > 1$. Otherwise the deviation is not taken into account. Moreover, before taking the square, we apply the ceil function, to get an integer-valued cost function.

We have a hard constraint with respect to the maximum group-size: the number of students in a group should not exceed a given maximum. In our implementation we do not steer the matching in any way, for efficiency reasons. The consequence of this is, that the groups tend to be very unbalanced, thus exceeding the maximum value quite easily. This is because we match as many as possible students in the first group of the subject, then the second group, and so on. Consequently when we find a complete clusterscheme, we have to balance the groups. This heuristic we describe in subsection 5.3. Though we do not enforce that the maximum size is not passed, we nevertheless take a measure to avoid it in some cases. What we do is, that at the moment we start to place the groups of a new subject j, we place it in line i, and calculate the maximum number of students S_{ij} that can be assigned. In the partial clusterscheme this is an exact calculation. This number however, can decrease when new subjects are placed. We use the number S_{ij} in four ways.

1. Based on the maximum group-size, we can calculate the minimum group-size by the formula
$$m_j = G_j A_j - (G_j - 1)M_j,$$
where m_j is the minimum number and M_j is the (given) maximum number of students in a group of subject j.
Hence we need $S_{ij} \geq m_j$.
2. Suppose we place the groups of subject j in the lines $i_1, i_2, \ldots$. Then necessarily
$$\sum_k S_{i_k,j} \geq G_j A_j \qquad (G_j A_j \text{ is the total number of students in subject } j).$$
3. If $S_{ij} < A_j - 1$ we surely have costs for placing the group in this line. Thus we have a lower bound for the actual cost, which we use for bounding.
4. We use S_{ij} to balance the groups, see subsection 5.3.

5.3 Balancing the groups

Once a clusterscheme has been found the assignment of students has to be reconsidered. As explained above, usually the groups are very unbalanced. We face here a difficult problem, at least in mathematical sense: how to assign the students such that the groups are optimally balanced, or stated differently: how to minimize the cost? The problem seems to become even worse, when one realizes that there are usually many solution to the clusterscheme problem, if there is at least one solution. However this last fact is what saves us. We can not put too much time in balancing, but if we have a reasonable heuristic, probably one of the clusterschemes we found will be balanced optimally.

The heuristic for balancing we use is a simple greedy algorithm.

1. Find the line i and subject j where $S_{ij} - A_j$ is negative and minimal. We assign as many as possible students to this group, and discard the combination (i, j) in the sequel. We continue till all (i, j) with $S_{ij} < A_j$ are treated.
2. If for all non-fixed combinations (i, j) have that $S_{ij} - A_j$ is non-negative, we turn our attention to subjectgroups which are still below average, so where $\sum_k y_{ijk} < A_j$, and proceed in the same way, with S_{ij} replaced by $\sum_k y_{ijk}$.

We could continue with balancing of groups above average, but we do not do so. In practice there seems to be no need for it.

6 Results

We tested our program with the data from the location 'Kottenpark' of the 'Stedelijk Lyceum' in Enschede. Available data where the data of the years 2002 and 2003. We could compare these results to the result obtained by the commercial package that is in use at this school. It turned out that our program obtained in nearly half of the cases solutions with a lower clusterscheme length; the number of clusterlines was always the same. In half of the cases where there was a difference in length, this difference was 2 hours, in the other cases one hour.

For the year 2003 the results obtained by our program are used in the 'Kottenpark College' timetable of this year. Apart from lower lengths, also the balancing of the groups turned out to be better than the division of the students found by the commercial package. A summary of the results for the year 2003 is given in the table below.

	Number of students	Number of subjects	Length (our)	Length (commercial)
VMBO-T 3	72	3 or 4 out of 9	9	9
VMBO-T 4	62	4 or 5 out of 11	26	26
HAVO 4	90	5 to 8 out of 19	21	21
HAVO 5	56	3 to 7 out of 19	**21**	23
VWO 4	52	3 to 7 out of 17	21	21
VWO 5	72	4 to 9 out of 21	**23**	25
VWO 6	36	5 to 9 out of 18	**26**	27

7 Conclusion

Constructing the clusterscheme is the first step in constructing the complete timetable. As we explained, this step is very important, as it serves as a framework for the timetable. Several extensions should be considered in the future. First of all the partially declustering. By this we mean to split off a number of lessons of some optional subjects, and try to put them into one of the remaining clusterlines, with the aim of reducing the clusterscheme length even further. Another problem that should be taken into account is the availability of the teachers. It is no use to construct a clusterline with two (part-time) teachers, that are present on different days in the week. Taking into account this, means that we look ahead to the real timetable. Ideally these two parts interact with each other. How such aspects can be taken into account, is a field of future research.

References

1. A. Drexl and F. Salewski, 'Distribution requirements and compactness constraints in school timetabling', European Journal of Operational Research **102**, (1997), pp. 193-214.
2. O.B. de Gans, 'A computer timetabling system for secondary schools in the Netherlands', European Journal of Operational Research **7**, (1981), pp. 175-182.
3. B. van Kesteren, 'The clustering problem in Dutch high schools: changing metrics in search space', Master's thesis, Leiden University, The Netherlands (1999).
4. A. Schaerf, 'A survey of automated timetabling', CWI report CS-R9567, CWI, The Netherlands (1995).
5. J.L. Simons, 'ABC: Een programma dat automatisch blokken construeert bij de vakdifferentiatie binnen het algemeen voortgezet onderwijs', Technical report NLR TR 74107 U, NLR, The Netherlands (1974).
6. R.J. Willemen, 'School timetable construction; algorithms and complexity', PhD-thesis, Technical University Eindhoven, The Netherlands (2002).
7. J. Wood and D. Whitaker, 'Student centred school timetabling', Journal of Operational Research Society **49**, (1998), pp. 1146-1152.