A Case Study for Timetabling in a Dutch Secondary School

Peter de Haan[1], Ronald Landman[2], Gerhard Post[1,3], and Henri Ruizenaar[3,4]

[1] ORTEC, Groningenweg 6k, 2803 PV Gouda, The Netherlands
[2] KLM Royal Dutch Airlines, Amsterdamseweg 55,
1182 GP Amstelveen, The Netherlands
[3] Department of Applied Mathematics, University Twente,
PO Box 217, 7500 AE Enschede, The Netherlands
[4] Stedelijk Lyceum, Locatie Kottenpark, Lyceumlaan 30,
7522 GK Enschede, The Netherlands

Abstract. This paper describes a case study for constructing the yearly schedule of a secondary school in the Netherlands. This construction is divided in three steps. In the first step we create cluster schemes containing the optional subjects. A cluster scheme consists of cluster lines, and a cluster line contains classes which will be taught simultaneously. Part of the problem is that the students are not yet assigned to the classes. Once the cluster schemes are fixed, it remains to schedule the lessons to time slots and rooms. We first schedule the lessons to day-parts, and once this is completed we schedule the lessons to time slots within the day-parts. Thanks to consistency checks in the day-part phase, going from day-parts to time slots is possible. Finally, in the third step, we improve the previously found schedule by a tabu search using ejection chains. Compared to hand-made schedules, the results are very promising.

1 Introduction

In the past 25 years a lot of research has been done on automated High School Timetabling. This research can be divided in two groups:

1. Theoretical oriented research and surveys, see for example, in chronological order [5,7,9,18,21,23,24]. These papers either define some concepts and/or methods, but do not describe real-life implementations.
2. Research based on several cases (usually high schools from the region). These papers (hopefully) define the problem they study, and explain that their methods perform quite well on the real-life cases considered. Examples of these papers are found below.

What is apparent from the studies in the second class, is that the problems differ widely among the countries. Of course, there are certain aspects that they all have in common. This could be named the basic high school timetabling feasibility problem replacing 'lesson' by 'event', this is the basic timetabling problem):

E.K. Burke and H. Rudová (Eds.): PATAT 2006, LNCS 3867, pp. 267–279, 2007.

Given a set of lessons with needed resources, and time slots,

Assign resources and a time slot to each lesson, such that the resources are not over-used.

We assume the situation that we need to construct a schedule for a week, which is repeated for a certain period like a year or semester. A lesson has usually the following resources with restrictions:

1. *Class*: the (virtual) group; can be used once per time slot. There are two principally different situations:
   - The classes are mutually disjoint: every student is in exactly one class. This is called the 'Class–Teacher model'.
   - The classes are not disjoint: the class depends on the subject (students have optional subjects). This occurs for example in Germany [4,11,25], the Netherlands [12,26], and New Zealand [27].

   In the first case, some intermediate cases can exist, where several classes are combined and reshuffled based on level (easy math vs. difficult math), or religion, or sex (physical education lessons).
2. *Subject*: the subject of the lesson; the (subject, class) combination can be used once per day.
3. *Students*: the students that constitute the class of the lesson; a student can be used once per time slot. In most countries the students are preassigned to the class. However in case of optional subjects, these students might have to be divided over different classes (the Netherlands: [12,20,26]).
4. *Teacher(s)*: the teachers of the lesson; a teacher can be used once per time-slot, if the teacher is available at that time. Usually a lesson has just one teacher, which in most countries seems to be preassigned, while in some it has to be assigned, for example in Australia [1], Greece [6], and the UK [28].
5. *Room(s)*: the rooms needed for the lesson; a room can be used once per time slot. Usually the lesson needs only one room, and this room has to be assigned; some papers mention that a class has its own room (Greece [22], Italy [19]). (Another possibility is that only room types – like music-room, gymnasium, etc. – are assigned.)

From this we see that already the basic feasibility problem has several variants: assign students or not, assign teachers or not, assign rooms or not.

As far as the objective function is concerned, the variants are even more diverse. Here we mention two cases, which sometimes appear as hard constraints:

1. Compact schedules for classes, which means schedules for classes without idle times. Here an idle time (for a class or teacher) is defined as a free time slot between the first and last lesson of the day. In several cases this is automatic, as a class has as many lessons as timeslots available, as in Brazil [16,17], Italy [8,19], Spain [3], Switzerland [10], and the UK [28]. In cases with optional subjects it is usual impossible to have compact schedules. These schedules, on the student level now, are not often considered for quality.

2. Compact schedules for teachers. This is mentioned in most of the papers above. Sometimes, as with classes, this is almost automatic (this assumes that all, or at least almost all, teachers work full-time).

Considering all cases above, it is difficult to judge which problems are harder than others. Probably most cases have aspects that are difficult to handle; either feasibility is hard to attain, or it can be hard to obtain high quality schedules. Similarly it is difficult to judge the capabilities of different methods; how well would a method do on cases outside the studied ones? The lack of exchangeable benchmarks is obvious here.

As for the Netherlands, we think that the high school timetabling problem is quite difficult; we will try to explain that in the next section. Since De Gans [12] in 1981 no study seems to be published internationally on real-life data from the Netherlands. (Willemen [26] focuses on complexity issues.) Clearly a lot has changed since 1981, and it is worth studying the situation as it stands now. The process is still very dynamic, see Section 6.

2 Problem Description

The impetus for this research was a request in 2003 from the Kottenpark (a location of 'Het Stedelijk Lyceum' in Enschede) to assist with creating the year schedule. In the Kottenpark the timetable is still mainly made by hand, and checked by computer. The reason for not using the commercial engine is mainly quality: the engine is not able to generate any complete solution, and the part that is generated is of poor quality.

In 2004, the Kottenpark had around 1000 students, 36 school classes, 71 teachers, and 40 rooms. There were 1049 lessons to be scheduled. As such it is a school of average size in the Netherlands. There are 38 time slots available for lessons. The occupation of time slots by the students ranges from 76% (29 lessons per week) up to 92% (35 lessons per week). In timetabling this school the following difficulties are encountered.

- In the upper years, up to two-thirds of the lessons of the students are in optional subjects. To handle these subjects, a cluster scheme (see below) is constructed, which requires a certain number of time slots (the length of the cluster scheme). It is sometimes difficult to reduce this length to get an a priori schedulable situation.
- Around 75% of the teachers work part-time at the Kottenpark. Consequently (by collective labor agreement) they are entitled to have 1, 2 or 3 days without lessons; usually teachers have preferences for these days.
- Teachers have up to 26 lessons, most of them less. Hence avoiding idle time is not automatic. The hand-made schedule contains 128 idle times for teachers, an average of 1.9 per teacher.
- The lower years (without optional subjects) should have no idle times.
- The two gymnasiums are used for 100% of the time, often as a block (two consecutive time slots), and always two classes (of the same age) combined.

All these circumstances are quite common in the Netherlands. It seems that scheduling the students, and the high amount of part-time teachers are quite exceptional compared to other countries. Our approach consists of three phases, devised to handle these problems:

1. *Construct cluster schemes*: the students are assigned to classes for the optional subjects, and the classes to cluster lines.
2. *Create a feasible schedule*: assign all lessons to time slots, such that there are no clashes.
3. *Improve the schedule*: improve the feasible schedule.

In the next sections, we will describe the three phases of our algorithm in more detail.

3 Constructing the Cluster Schemes

3.1 Motivation

As one can easily imagine, scheduling the optional subjects constitutes a major bottleneck in constructing (good) timetables. To get this process under control, *cluster schemes* are constructed first. In the Netherlands it seems that all schools first construct cluster schemes. This is a way to avoid using individual students, essential when the schedule is constructed by hand. The system with optional subjects was introduced in 1968 (Mammoetwet), while the revision of 1995 ('Second Phase') made things even more complicated. The report of Simons [20] concerns the construction of cluster schemes, while De Gans [12] assumes that the cluster schemes are already created.

We have to construct a cluster scheme for years with optional subjects. Each cluster scheme consists of *cluster lines*. Each cluster line contains a number of classes with different optional subjects, that will be scheduled at the same time slots. The arrangement of classes in the cluster lines, and the assignment of students to classes of their optional subjects must be such that each student can attend the optional subjects he has chosen. So for each student it should be possible to make an assignment to classes of his optional subjects, such that these classes are in different cluster lines. Unfortunately, not all optional subjects have the same number of lessons, neither have all students the same number of optional subjects. Homogeneous tiling structures, as in [14], seem difficult to attain. (The high percentage of part-timers also breaks homogeneity.)

The classes in a cluster line have a number of lessons. The maximal number of lessons in a cluster line is called the *cluster line length*. The *cluster scheme length* is the sum of the lengths of the cluster lines it contains; it represents the number of time slots we have to reserve for the cluster scheme.

The main goal is to minimize the cluster scheme lengths. (Commercial software prescribes the maximal number of cluster lines, which is less informative.) For this there are the following reasons.

- If the cluster scheme length is too high, it might be that there are not enough time slots left for the obligatory subjects. In some years this causes a problem. It can even happen that the school management decides to add an extra class to decrease the cluster scheme length.
- A lower cluster scheme length increases the freedom in scheduling the remaining obligatory subjects.
- A lower cluster scheme length decreases the number of potential idle times. Here it is important to remember that not all students might be present in a cluster line. For the time slots of this cluster line these students are free. If these students have a lesson earlier and later on the day, it is an idle time.

The secondary goal in a cluster scheme is to balance the classes of a subject. If, for instance, there are three classes for the subject mathematics, and 79 students, then the best balance is that the classes contain 26, 26 and 27 students. The combination 23, 23 and 33 is forbidden, as this violates the maximum size (32 in our case) of a class, while the combination 32, 32, 15 is not desirable. Balancing is done for educational reasons, and for fairness towards the teachers.

3.2 Branch and Bound

The sizes of problem instances (up to 20 or 25 classes) make it worthwhile to attempt careful enumeration. If we have 20 classes and 8 cluster lines, then a priori there are 8^{20} possibilities for classes in lines, which is clearly out of range. However there are several ways to reduce the number of possibilities considerably. In the enumeration there are two possible approaches.

1. Decide per step in which classes a student takes his optional subjects. In this case we have to put the classes in lines, such that classes in a line have no students in common. This can be viewed as a graph coloring problem (classes are vertices, with edges between non-disjoint classes, and cluster lines are colors).
2. Decide per step in which lines the classes are put. In each step we have to solve a matching problem for each student, namely to decide in which lines a student takes his optional subjects.

Both approaches will try to extend partial assignments, and step to the next possibility in case the configuration gives no solution. We chose the second approach for two reasons:

- It seems easier to solve at each step a matching problem than a graph coloring one.
- It seems easier to estimate our objectives, the cluster scheme length and the balance of classes.

Hence our approach takes the following steps. We store, for instance, the best 100 solutions:

1. Place one student in a group for his k subjects, and place these groups in the first k cluster lines. These groups are fixed, and will never be moved.

2. Take the first not yet placed group G.
3. For cluster line 1 to the last cluster line, place G, and estimate the cluster scheme length. If too high skip the case, and move G to the next cluster line.
4. Assign all students to the placed groups (increment for G). If not all students can be assigned, stop the case and move G to the next line.
5. If there are non-placed groups left, return to 2.
6. Balance the groups, and calculate the objectives, the cluster scheme length and the penalty for not-balanced groups. We compare solutions lexicographically, first by cluster scheme length, and then by balance. If necessary store the solution. Continue with placing G in the next cluster line.

We give an overview of the methods we applied to speed up the search process. For more details we refer to the technical report [15].

3.3 Using Statistics

Calculating the matchings at each step is very time consuming. A first simplification is to group students that have the same optional subjects. Apart from this, we accumulate some statistics at the start. We will call an optional subject with k classes a 'k-grouper'.

- For any two 1-groupers, check if they have a student in common. If this is the case, the corresponding classes 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 classes have to occupy at least 3 lines.

To use these statistics in an efficient way, we decided to order the classes according to the number of classes of the subject; the 1-groupers first, then the 2-groupers, then the 3-groupers, and so on. Note that placing the 1-groupers is a graph coloring problem: the classes are the vertices of the graph, while two vertices are connected if the corresponding classes have a student in common (this is part of the statistics above).

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

3.4 Symmetry

All lines are equivalent. Hence in case of eight lines, we gain a factor 8! by removing equivalent solutions. We remove symmetric solutions by:

- Fixing one (difficult) student to classes, and fixing these classes in different lines.
- Only place a class in a cluster line, when all previous lines are non-empty.

– For two classes of the same subject we assume that the line number of the first class is lower than the line number of the second class. This holds if these classes that are not fixed by the difficult student (as described above).

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

$$\frac{\text{line 1: } S3\ S1}{\text{line 2: } S2\ S3} \quad \text{and} \quad \frac{\text{line 1: } S3\ S2}{\text{line 2: } S1\ S3}$$

If the fixed class 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.

3.5 Bounding

The next part we have to take care of is bounding. In our problem we have two things to bound on: first the length of the cluster scheme, and second the balancing of the classes.

The length of the cluster scheme can be estimated by the classes that were placed in lines. In the case that not all classes are placed yet, we have a lower bound for the cluster scheme length, which can be used for bounding: as soon as the partial cluster scheme has a length exceeding the best obtained cluster scheme, we prune the search tree.

At the moment we start to place the classes of a new subject j, we place it in line i, and calculate (by matching) the maximum number of students M_{ij} that can be assigned. In the partial cluster scheme this is an exact calculation; this number however, can decrease when new subjects are placed. As soon as all classes of a subject are placed in cluster lines, we can estimate how far the classes necessarily will deviate from the average size; again we prune if the deviation is higher than the best found.

3.6 Balancing Heuristic

Once a *complete* cluster scheme has been found, and all classes have been put in cluster lines, the assignments of students to classes have to be reconsidered. The assignment were made by the Hungarian method, and hence by first fit; no attention was paid to the number of students in the classes. In particular we prefer that classes of the same subject contain approximately the same number of students. Our heuristic for balancing is a greedy algorithm; here M_{ij} is as above, and A_j denotes the average size for classes of subject j.

1. Find the line i and subject j where $M_{ij} - A_j$ is negative and minimal. We assign as many as possible students to this class, and discard the combination (i, j) in the sequel. We continue until all (i, j) with $M_{ij} < A_j$ are treated.

2. If for remaining combinations (i, j) we have that $M_{ij} - A_j$ is non-negative, we turn our attention to classes which are still below average, and proceed in the same way, with M_{ij} replaced by the number of currently assigned students.

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

3.7 Pruning Based on Computation Time

The program we developed contains an option to prune parts of the search tree, based on the time spent in the subtree: one can prescribe that for search depth d only s seconds are allowed. Here d is usually between 1 and 4, while s is taken as a few seconds. In this way we can do a quick scan of the search space within one or two minutes. Especially for the harder cases, it turns out that solutions are found much quicker this way.

3.8 Results

The methods above have been used at the Kottenpark during the last three years. When running the program an upper bound for the cluster scheme length must be given. For most years good solutions are found within a few seconds, if at least one feasible solution exists. If no solutions exist, the search can take several minutes or hours, as the complete search tree has to be checked.

4 Creating a Feasible Schedule

4.1 Motivation

Schools consider the creation of the cluster schemes as a preliminary phase; usually a different application is provided for it, without interaction with the timetabling itself. Once the cluster schemes are found, the next phase starts to assign the lessons to time slots. Instead of assigning the lessons directly to time slots, we will first assign them to day-parts. For this the days at Kottenpark are divided in two day-parts: mornings of five time slots, and afternoons with three time slots, except for the Thursday afternoon, which consists of one time slot. There are several reasons to do this intermediate step.

- Several constraints are on daily, or on day-part level. These are the constraints that lessons of a class must be on different days, teachers are not available on certain days or day-parts, and the number of teaching days for teachers should be limited for part-time teachers. Such constraints can be handled very well in this phase.
- It is unclear to which time slot we should assign a certain lesson, if we do not know about all lessons to be assigned. Hence we will do a lot of useless reshuffling.

– Assigning lessons to 40 time slots in a week is much harder than assigning lessons to the 5 or 3 time slots in a day-part for 10 times.

Of course there are certain drawbacks. The most important one is that a feasible day-part schedule does not imply a feasible time slot schedule. This problem we address in Section 4.3. Moreover we restrict the search space; while mentioned above as an advantage, it could prevent us from improving certain aspects in the solution.

4.2 Direct Heuristic

We proceed to schedule the lessons to day-parts. To do this, the lessons are grouped by the class, or for the optional subjects, by the cluster line. For uniformity we create an artificial cluster line (with one class) for the compulsory subjects. Hence in a stage of the direct heuristic we consider a cluster line, and try to assign the lessons of the cluster lines to day-parts.

The method we use is a dynamic priority rule. At each stage we estimate the difficulty of the cluster lines to be scheduled. The difficulty is based on the weight of the cluster line (originally all weights are 0), and the availabilities of the resources of the cluster line. We will schedule the most difficult cluster line first. If this scheduling process breaks down, because a particular cluster line cannot be assigned any more, we raise the weight of this cluster line, and restart.

4.3 Compatibility Checking

If we assign cluster lines to day-parts, some conditions have to be checked. The obvious necessary conditions are that a resource is scheduled for at most the number of time slots that it is available. This, however, is not enough to guarantee schedulability on time slot level. We can take certain measures which in practice are sufficient. These measures consist of creating time slot schedules for day-parts and resources that get tight. More specifically, if for a resource the slack in time slots is 1 or 0, we decide to do this check. In that case, all lessons of this resource are taken, as well as the neighbors, and the neighbors' neighbor (the compatibility graph). Here a neighbor is defined to be a lesson with a common teacher, class, or student. We try to color this graph where the colors are the available time slots. If we do not succeed within a certain time limit, we assume that no coloring exists, and reject the day-part for this lesson.

Here some special attention has to be taken towards the resource 'room type'. (We do not really schedule the rooms, but only make sure that we have enough rooms of the required room type available.) The lessons with the required room types are not necessarily neighbors in the compatibility graph. Hence, when constructing the subgraph, we take all lessons with the required room type, and add all neighbors and neighbors' neighbors.

4.4 Assigning the Time Slots

Once all lessons have been assigned to day-parts, we try to assign them to time slots. For this we use a graph coloring heuristic, which colors the nodes one by one (first fit). To sort the nodes, we use the weight of the nodes, where the weight is originally the degree of the node. Each time a node cannot be colored any more, we increase the weight of this node, and backtrack.

4.5 Results

Thanks to the checks described in Section 4.3 all lessons are scheduled after a few restarts (see Section 4.2), which takes a few minutes. The quite extensive compatibility tests turn out to be beneficial on the running time. We tried to influence the coloring with regard to idle times for teachers. For this we also started off with random orderings of the nodes (see Section 4.4), instead of ordering by degree. Allowing 30 seconds per day-part for this random search reduced the total number of idle times for teachers from 142 to 114. Repeating the random searches 20 times, the total number of idle times dropped to 95. Hence on this aspect we beat the hand-made schedule. Unfortunately there are still idle times for the classes of the lower years, which we did not take into account.

5 Improving the Schedule

5.1 Motivation

The previous phase aimed at assigning all lessons to time slots. Not much attention was paid to quality yet; the emphasis was on finding a feasible schedule. In the current phase, we try to improve the feasible schedule we found. Ejection chains [2] combined with tabu search [13] seem to be very appropriate for improving schedules. The quality of a schedule is determined per resource by the idle times, and the division of lessons over the days for the resource. Hence we can find a resource, with a low quality schedule, improve this schedule by shifting some lessons, and prevent shifting back by placing this shift on a tabu list. An improvement for one resource (teacher/class) automatically implies that for other resources (class/teacher) some repairs have to be done; the shifted lessons can have clashes, due to other resources. Such lessons, 'conflict-lessons', again have their own conflicts, etc. Hence we run quite naturally into an ejection chain of improvements.

5.2 Selecting the Shifts

We perform a tabu search, in which each step by itself is a chain of shifts. Here a shift means that a lesson is moved from one time slot to another. We explain how we find the shifts that we execute.

1. First we select the resource A (teacher or class) with the worst schedule, which is not tabu.
2. For resource A we consider all lessons and all free time slots. For each combination (lesson, free time slot) we calculate the cost change for resource A for moving a lesson to a free time slot. With respect to other resources of the lesson, we only make sure that there is not more than one conflict lesson.
3. We select the $C = 20$ best candidates. The selected shifts we call the *first shift candidates*.
4. Executing the first shift candidates, two things can happen:
   - The other resources have no clashes; in this case the chain of shifts is ended.
   - There is one conflict-lesson (we did not allow more than one!), due to resource B. We shift the conflict-lesson to a free time slot of resource B, and calculate what is the best for resource B. Again we only consider time slots with at most one conflict-lesson; time slots without conflict-lesson are preferred.

   If in the second case no new conflict arises, then the chain of shifts is ended. Otherwise we proceed until a maximum of $D = 10$ shifts.
5. The chain with the highest cost reduction is executed; the reverse move of first shift is made tabu for $L = 10$ moves. If no chain is found at all, resource A itself is made tabu also for L moves.

5.3 Results

We let the algorithm run for 2500 iterations. Usually the best schedule is found within 1500 iterations. With the parameters as above the algorithm runs for less than one minute. Experiments were executed with different sets of constraints. In case of the default Kottenpark set, the cost is reduced by more than 70%. The total number of idle times of the teachers reduced to 48, while the idle times of the lower years disappeared. Moreover the spreading of lessons for the teachers was improved considerably.

6 Conclusion

The presented study is performed with data from a specific Dutch school, but we believe that these data are representative for many schools in the Netherlands. Unfortunately not all constraints are incorporated yet, which makes comparison to the real timetable not completely fair. Comparing what *is* included we see a huge improvement in quality; for instance the number of free periods for teachers drops from 128 (hand-made) to 48, maintaining compact schedules for the lower years.

After constructing the cluster schemes, we used a two-phase approach to obtain feasible schedules; the first phase (Section 4) was designed to handle several constraints related to part-time teachers. Viewing the result in the second phase

(Section 5), one can wonder at the effectiveness of this approach; many lessons were moved from one day to another, improving the spreading of lessons for part-time teachers. Nevertheless the two-phase method is quite effective in obtaining a feasible schedule.

In 2005, the Kottenpark introduced a new educational system in the two lower years. In this system classes of 60 students are constructed. Most subjects are taught with two teachers: the first teacher belongs to the subject (preassigned as before), while the second teacher is one of the two preassigned to the class. The 24 lessons of one class have to be divided between these two teachers, where there is some preference for subjects, but split assignments are allowed. Because of this, the program as described here is used operationally only for constructing the cluster schemes.

Acknowledgements. This research has been supported by the Netherlands Organization for Scientific Research, grant 636.000.000.02N18 (Leraar in Onderzoek), and by BSIK grant 03018 (BRICKS: Basic Research in Informatics for Creating the Knowledge Society).

References

1. Abramson, D.: Constructing school timetables using simulated annealing: sequential and parallel algorithms. Management Science 37, 98–113 (1991)
2. Ahuja, R.K., Ergu, O., Orlin, J.B., Punnen, A.P.: A survey of very large-scale neighborhood search techniques. Discrete and Applied Mathematics 123, 75–102 (2002)
3. Alvarez-Valdes, R., Martin, G., Tamarit, J.M.: Constructing good solutions for the Spanish school timetabling problem. Journal of Operational Research Society 47, 1203–1215 (1996)
4. Bufé, M., Fischer, T., Gubbels, H., Häcker, C., Hasprich, O., Scheibel, C., Karsten Weicker, K., Weicker, N., Wenig, M., Wolfangel, C.: Automated solution of a highly constrained school timetabling problem – preliminary results. In: Boers, E.J.W., Gottlieb, J., Lanzi, P.L., Smith, R.E., Cagnoni, S., Hart, E., Raidl, G.R., Tijink, H. (eds.) EvoIASP 2001, EvoWorkshops 2001, EvoFlight 2001, EvoSTIM 2001, EvoCOP 2001, and EvoLearn 2001. LNCS, vol. 2037, pp. 431–440. Springer, Heidelberg (2001)
5. Burke, E.K., Petrovic, S.: Recent research directions in automated timetabling. European Journal of Operational Research 140, 266–280 (2002)
6. Birbis, T., Daskalali, S., Housos, E.: Timetabling for Greek high schools. Journal of the Operational Research Society 48, 1191–1200 (1997)
7. Carter, M.W., Laporte, G.: Recent developments in practical course timetabling. In: Burke, E.K., Carter, M. (eds.) PATAT 1997. LNCS, vol. 1408, pp. 3–19. Springer, Heidelberg (1998)
8. Colorni, A., Dorigo, M., Maniezzo, V.: Metaheuristics for high school timetabling. Computational Optimization and Applications 9, 275–298 (1998)
9. Cooper, T.B., Kingston, J.: The solution of real instances of the timetabling problem. The Computer Journal 36, 645–653 (1993)
10. Costa, D.: A tabu search algorithm for computing an operational timetable. European Journal of Operational Research 76, 98–110 (1994)

11. Drexl, A., Salewski, F.: Distribution requirements and compactness constraints in school timetabling. European Journal of Operational Research 102, 193–214 (1997)
12. de Gans, O.B.: A computer timetabling system for secondary schools in the Netherlands. European Journal of Operational Research 7, 175–182 (1981)
13. Glover, F.W., Laguna, M.: Tabu Search. Kluwer, Norwell, MA (1997)
14. Kingston, J.H.: A tiling algorithm for high school timetabling. In: Burke, E.K., Trick, M.A. (eds.) PATAT 2004. LNCS, vol. 3616, pp. 208–225. Springer, Heidelberg (2005)
15. Post, G.F.H., Ruizenaar, W.A.: Clusterschemes in Dutch secondary schools. Memorandum 1707, University of Twente (2004), `http://www.math.utwente.nl/publications/2004/1707abs.html`
16. Ribeiro Filho, G., Lorena, L.A.N.: A constructive approach to school timetabling. In: Boers, E.J.W., Gottlieb, J., Lanzi, P.L., Smith, R.E., Cagnoni, S., Hart, E., Raidl, G.R., Tijink, H. (eds.) EvoIASP 2001, EvoWorkshops 2001, EvoFlight 2001, EvoSTIM 2001, EvoCOP 2001, and EvoLearn 2001. LNCS, vol. 2037, pp. 130–139. Springer, Heidelberg (2001)
17. Santos, H.G., Ochi, L.S., Souza, M.J.F.: An efficient tabu search heuristic for the school timetabling problem. In: Ribeiro, C.C., Martins, S.L. (eds.) WEA 2004. LNCS, vol. 3059, pp. 468–481. Springer, Heidelberg (2004)
18. Schaerf, A.: A survey of automated timetabling. CWI Report CS-R9567, CWI, The Netherlands (1995)
19. Schaerf, A.: Local search techniques for large high school timetabling problems. IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans 29, 368–377 (1999)
20. Simons, J.L.: 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)
21. Smith, K.A., Abramson, D., Duke, D.: Hopfield neural networks for timetabling: formulations, methods and comparative results. Computers and Industrial Engineering 44, 283–305 (2003)
22. Valouxis, C., Housos, E.: Constraint programming approach for school timetabling. Computers and Operations Research 30, 1555–1572 (2003)
23. de Werra, D.: An introduction to timetabling. European Journal of Operational Research 19, 151–162 (1985)
24. de Werra, D.: On a multiconstrained model for chromatic scheduling. Discrete Applied Mathematics 94, 171–180 (1999)
25. Wilke, P., Gröbner, M., Oster, N.: A hybrid algorithm for school timetabling. In: McKay, B., Slaney, J.K. (eds.) AI 2002: Advances in Artificial Intelligence. LNCS (LNAI), vol. 2557, pp. 455–464. Springer, Heidelberg (2002)
26. Willemen, R.J.: School timetable construction; algorithms and complexity, Ph.D. Thesis, Technical University Eindhoven, The Netherlands (2002)
27. Wood, J., Whitaker, D.: Student centred school timetabling. Journal of Operational Research Society 49, 1146–1152 (1998)
28. Wright, M.: School timetabling using heuristic search. Journal of Operational Research Society 47, 347–357 (1996)