The University of Sydney

The Complexity of Timetable Construction Problems

Technical Report Number 495

February 1995

Tim B Cooper and Jeffrey H Kingston

ISBN 0 86758 958 2

Basser Department of Computer Science
University of Sydney NSW 2006

The Complexity of Timetable Construction Problems

Tim B. Cooper and Jeffrey H. Kingston

Basser Department of Computer Science
The University of Sydney 2006
Australia

3 February, 1995

Abstract

This paper shows that timetable construction is NP-complete in a number of quite different ways that arise in practice, and discusses the prospects of overcoming these problems. A formal specification of the problem based on TTL, a timetable specification language, is given.

1. Introduction

The timetable construction problem is to assign times, teachers, students and rooms to a collection of meetings so that none of the participants has to attend two meetings simultaneously. This basic requirement is often augmented with others concerning limits on the workload of teachers, constraints on the way a meeting's times are spread through the week, and so on.

Many different techniques have been applied to the timetable construction problem. Previous work by the authors and others has attempted to break it into subproblems, in the hope that each will be efficiently solvable. In such work a detailed understanding of the inherent complexity of timetable construction is needed; a broad statement that the overall problem is NP-complete gives no guidance on the prospects of solving any particular subproblem. This paper fills in some of these details by identifying five independent NP-complete subproblems, and discussing the prospects of solving each in practice.

2. Specification of the timetable construction problem

In this section we present a specification of the timetable construction problem based on a *timetable specification language* called TTL. This language is formal yet flexible enough to specify instances encountered in practice. An earlier version of TTL appeared in [1].

A TTL instance consists of a *time group*, some *resource groups*, and some *meetings*. A formal grammar appears in Figure 1. Here is a typical time group:

instance → *timegroup* { *resourcegroup* } { *meeting* }
timegroup → **timegroup** *timegroupname* **is**
{ *timename*; }
end *timegroupname*;
resourcegroup → **group** *resourcegroupname* **is**
[**subgroups** *subgrouplist*;]
{ *resourcename* [**in** *subgrouplist*] ; }
end *resourcegroupname*;
subgrouplist → *resourcegroupname* { , *resourcegroupname* }
meeting → **meeting** *meetingname* **is**
{ *timeselect* | *resourceselect* }
end *meetingname*;
timeselect → *number timegroupname* [: *timeconditionlist*] ;
→ *timename*;
resourceselect → *number resourcegroupname*;
→ *resourcename*;
number → *integer* | **all**

Figure 1. Grammar of the TTL language. { ... } means zero or more of, [...] means optional. Time conditions and some unimportant extensions required in practice have been omitted.

timegroup *Times* **is**

*Mon*1; *Mon*2; *Mon*3; *Mon*4; *Mon*5; *Mon*6; *Mon*7; *Mon*8;
*Tue*1; *Tue*2; *Tue*3; *Tue*4; *Tue*5; *Tue*6; *Tue*7; *Tue*8;
*Wed*1; *Wed*2; *Wed*3; *Wed*4; *Wed*5; *Wed*6; *Wed*7; *Wed*8;
*Thu*1; *Thu*2; *Thu*3; *Thu*4; *Thu*5; *Thu*6; *Thu*7; *Thu*8;
*Fri*1; *Fri*2; *Fri*3; *Fri*4; *Fri*5; *Fri*6; *Fri*7; *Fri*8;

conditions (*omitted*)

end *Times*;

It lists the names of the times available for meetings, followed by *conditions* (see below). Here is a typical resource group:

group *Teachers* **is**

subgroups *English*, *Science*, *Computing*;

Smith **in** *English*, *Computing*;
Jones **in** *Science*, *Computing*;
Robinson **in** *English*;

end *Teachers*;

This group contains *resources* (*Smith*, *Jones*, and *Robinson*) which are available to attend meetings, and *subgroups* which are subsets of the set of resources defining functions that the resources are qualified to perform: teach English, etc. A resource may be in any number of subgroups. Typical instances have *Teachers*, *Rooms*, and *Students* resource groups.

After the groups come the meetings, which are collections of *slots* which are to be assigned elements of the various groups, subject to certain restrictions. For example, here is a typical meeting expressing five Science classes which meet simultaneously for six times per week:

meeting 10-*Science* **is**
*Year*10;
5 *Science*;
5 *ScienceLab*;
6 *Times*: 1 *Double*, 5 *Days*, 5 *Nice*;
end 10-*Science*;

There is one slot which must contain the *Year*10 resource from the *Students* group; five slots which must contain resources from the *Science* subgroup; five resources from the *ScienceLab* subgroup of the *Rooms* group, and six times from the *Times* group, which must satisfy three conditions: there must be at least one double time (i.e. two adjacent times), the times must be spread over at least five days, and all but one of them must be nice (i.e. not last on any day). These conditions are defined in the time group, but lack of space prevents us from explaining them in detail here. Formally, the meaning is that the eleven selected resources will all be occupied together for the six times; in fact, it is clear that the Year 10 students will be split into five groups.

Although most meetings are similar to 10-*Science*, there are some exceptions, for example Faculty meetings:

meeting *EnglishFaculty* **is**
all *English*;
1 *Times*;
end *EnglishFaculty*;

and meetings which ensure that each teacher teaches at most 30 out of the 40 possible times each week (say), and has at least one free time each day:

meeting *SmithFree* **is**
Smith;
10 *Times*: 5 *Days*;
end *SmithFree*;

Real instances may have two hundred or more meetings altogether.

This completes the presentation of the TTL language. It has been used successfully by the authors, with some unimportant extensions, to specify high school instances [1]. It is easy to formulate extensions for specifying the multiple sections needed in university timetabling, and to express preferences in selections. More difficult would be relations between meetings (must be in same building, or not on same day, etc.) as are needed for example with multiple campuses.

We will denote by TIMETABLE CONSTRUCTION (TTC) the decision problem of

determining whether an assignment of times and resources to all the slots exists which satisfies the various conditions and is such that no resource is assigned to two meetings which share a time. This is polynomial-time equivalent to the problem of actually producing such an assignment: given TTC, one can construct a solution if it exists by taking the first slot and trying each possible assignment to it until TTC indicates that a solution exists, then repeating on the other slots in turn.

3. NP-completeness results

In determining the complexity of the TTC problem defined in Section 2, formally it is sufficient to prove TTC $\in$ **NP** (obvious) and to demonstrate one transformation from any known NP-complete problem to TTC. But we wish to show that TTC is NP-complete in several independent ways, all of which arise in practice. For this it is necessary to find transformations which construct TTC instances that resemble special cases of the TTC problem that arise in real instances.

The well-known formulation of the timetable construction problem given by Gotlieb [5] assumes that each meeting contains exactly one nominated student group, one nominated teacher, and any number of times which may be freely chosen. Csima [2] showed that Gotlieb's problem is in **P** if the teachers are initially available at all times, and Even, Itai, and Shamir [3] showed that it is NP-complete if each teacher may be assumed initially unavailable at an arbitrary subset of the times.

More relevant in practice was the work of Karp [6] showing that graph colouring is NP-complete. At that time the connection between graph colouring and timetable construction (revisited below) was already well known [7, 8].

3.1. Intractability owing to student choice

We begin with a well-known result relating timetable construction to graph colouring. It shows NP-completeness when each student is granted a free choice from a wide range of subjects, as is characteristic of university timetable construction.

Theorem 1. GRAPH K-COLOURABILITY $\propto$ TTC.

Proof. Recall that the NP-complete GRAPH K-COLOURABILITY problem asks whether it is possible to assign a colour to each vertex of a graph G in such a way that no two adjacent vertices have the same colour; at most $K \leq |V|$ distinct colours are allowed. Let $G = (V, E)$ with $V = \{v_1, \ldots, v_n\}$ and $E = \{e_1, \ldots, e_m\}$. Construct the TTC instance

timegroup T **is**
  $t_1; \ldots; t_K;$
end T;

group R **is**
  $r_1; \ldots; r_m;$
end R;

meeting M_1 **is**
  1 T;
  $c_{1,1}$; …; c_{1,k_1};
end M_1;

...

meeting M_n **is**
  1 T;
  $c_{n,1}$; …; c_{n,k_n};
end M_n;

where the resources $c_{i,1}$, …, c_{i,k_i} selected by meeting M_i are exactly those resources r_j such that e_j is adjacent to v_i in G.

Suppose a K-colouring $f : V \to \{1, \ldots, K\}$ exists for G. Assign t_k where $k = f(v_i)$ to M_i for all i. The condition $f(v_i) \neq f(v_j)$ whenever $\{v_i, v_j\} \in E$ guarantees that meetings which share any resource receive different times, so the TTC instance is solved. Conversely, a successful time assignment defines a successful graph colouring. □

Taking each r_i to represent one student, this transformation shows that assigning times to university classes such that all students can attend their choices is NP-complete even when each meeting occupies only one time, each student chooses just two meetings, and teacher and room constraints are ignored. It also demonstrates that university examination timetable construction is NP-complete.

In universities, this problem is avoided by publishing the timetable in advance and requiring students to choose only combinations of subjects permitted by the timetable. Large classes are divided into *sections* (alternative offerings of the same subject) which run at different times. Choosing appropriate sections for just one student after times are fix edis NP-complete (Section 3.3), but sections provide sufficient freedom in practice to make solutions fairly easy to find by searching.

In high schools known to the author, student choice is limited by deciding in advance that certain groups of meetings will occur simultaneously, and inviting students to choose one meeting from each group. The decision as to which meetings to group in this way is often influenced by a preliminary survey of student preferences, which of course makes it into an NP-complete graph colouring problem too.

3.2. Intractability owing to varying meeting size

Meetings occupy more than one time each. A typical pattern in high schools might be six times for English and Mathematics, fiv efor Science, three for Sport, and so on. When meetings of such varying sizes are assigned to teachers, it can be difficult to assign exactly the 30 (say) times that comprise each teacher's workload. Overloading is forbidden by industrial agreement, and underloading one teacher implies overloading another. This leads to NP-completeness even disregarding restrictions imposed by teachers' qualifications and the need to avoid clashes:

Theorem 2. BIN PACKING (with unary encoding) $\propto$ TTC.

Proof. Recall that the NP-complete BIN PACKING problem asks whether a set of items U =

$\{u_1, \ldots, u_n\}$, each with a positive integer size $s(u_i)$, can be packed into B bins each of capacity C in such a way that no bin is overfull. We assume that these numbers are encoded in unary rather than binary; since BIN PACKING is NP-complete in the strong sense [4], this version is NP-complete. We transform to the TTC instance

timegroup T **is**
  $t_1; \ldots; t_C;$
end T;

group R **is**
  $r_1; \ldots; r_B;$
end R;

meeting M_1 **is**
  $s(u_1)$ T;
  1 R;
end M_1;

...

meeting M_n **is**
  $s(u_n)$ T;
  1 R;
end M_n;

Given the initial unary encoding, this transformation clearly has polynomial complexity.

Suppose that the BIN PACKING instance has solution $f : U \to \{1, \ldots, B\}$. Assign r_k where $k = f(u_i)$ to meeting M_i for all i; then, for each r_j in R, the total time requirements of all meetings containing resource r_j will be at most C, and we may assign any disjoint sets of times to these meetings. Conversely, from any solution to the TTC instance we may deduce a bin packing by assigning $f(u_i) = k$ where M_i contains r_k. □

In high schools known to the author, some meetings in the junior years are split into two in the following way in order to create small fragments to fill the bins:

meeting M_i^1 **is**
  $(s(u_i) - k)$ T;
  1 R;
end M_i^1;

meeting M_i^2 **is**
  k T;
  1 R;
end M_i^2;

for some k, allowing two teachers to share the meeting. This is called a *split assignment*, and it is the major form of compromise permitted in high school timetable construction. Universities are not subject to this problem, because face-to-face workloads are lighter and more flexible.

3.3. Intractability owing to time-incoherence

The bin packing NP-completeness just explained would vanish if all meetings were of equal size, they were aligned in time, and each teacher's workload were a multiple of the meeting size. Meeting sizes and workloads are not under the control of timetable construction programs, but the alignment of meetings in time is. It was called *time-coherence*, and shown to be a powerful heuristic in practice, in [1].

There are several ways to define time-coherence formally. One simple way is to define the time-incoherence $i(M)$ of a set of meetings M to be the number of pairs of meetings from M that share at least one time. We can then define the decision problem TTC-TC to be TTC augmented with the requirement that $i(M)$ not exceed a given bound K. Unfortunately, enforcing time-coherence is NP-complete:

Theorem 3. BIN PACKING (with unary encoding) $\propto$ TTC-TC.

Proof. We remind the reader that the purpose of this theorem is not just to prove the result (we have already done so in Theorem 2) but to construct a TTC instance which establishes an independent source of NP-completeness.

As previously described, the NP-complete BIN PACKING problem asks whether a set of items $U = \{u_1, \ldots, u_n\}$, each with a positive integer size $s(u_i)$, can be packed into B bins each of capacity C so that no bin is overfull. Transform to a TTC-TC instance whose groups are

timegroup T **is**
  $t_1; \ldots; t_{BC}$;
end T;

group R **is**
  r;
end R;

and whose meetings are $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_B$ where the X_i are

meeting X_i **is**
  r;
  $s(u_i)$ T;
end X_i;

and the Y_j are

meeting Y_j **is**
  $t_{(j-1)C+1}; \ldots; t_{jC}$;
end Y_j;

and the bound on $i(M)$ is $K = n$.

Suppose that the BIN PACKING instance has solution $f : U \to \{1, \ldots, B\}$. For each meeting X_i, choose any $s(u_i)$ times (not already chosen) from the set $S_k = \{t_{(k-1)C+1}, \ldots, t_{kC}\}$ where $k = f(u_i)$. This is possible because f guarantees that at most C times will be chosen from S_k. All requirements are satisfied, and each X_i overlaps with exactly one Y_j, so $i(M) = n$.

Conversely, suppose that the TTC-TC instance has a solution with $i(M) \leq n$. We must have $i(M) \geq n$ since each X_i must overlap at least one Y_j, so $i(M) = n$ and each X_i overlaps exactly one Y_j. Setting $f(u_i) = j$ then defines a bin packing for the u_i. □

Any reasonable definition of time-coherence would permit the same transformation. In practice then, when meeting sizes vary we cannot expect to maintain time-coherence.

This inevitable loss of time-coherence causes severe problems in practice. To illustrate this, we present a tranformation which shows that, in the absence of time-coherence, the problem of assigning meetings to just one teacher is NP-complete:

Theorem 4. EXACT COVER BY 3-SETS $\propto$ TTC.

Proof. Recall that the NP-complete EXACT COVER BY 3-SETS problem is as follows. We are given a set $X = \{x_1, \ldots, x_{3q}\}$, and a collection of 3-subsets of X called $C = \{C_1, \ldots, C_n\}$ with $n \geq q$. The problem asks for an exact cover of X, that is, a subcollection $C' \subseteq C$ such that every element of X occurs in exactly one element of C'.

Let $C_j = \{c_{j,1}, c_{j,2}, c_{j,3}\}$ for $1 \leq j \leq n$, where each $c_{j,k} = x_i$ for some i such that $1 \leq i \leq 3q$. We transform an instance of EXACT COVER BY 3-SETS to a TTC instance whose groups are

timegroup T **is**
  $x_1; \ldots; x_{3q};$
end T;

group R **is**
  $r_1; \ldots; r_{n-q}; z;$
end R;

and whose meetings are $A_1, \ldots, A_{n-q}$ and $B_1, \ldots, B_n$ where the A_i are

meeting A_i **is**
  $(3q - 3)$ T;
  r_i;
end A_i;

and the B_j are

meeting B_j **is**
  $c_{j,1}; c_{j,2}; c_{j,3};$
  1 R;
end B_j;

Now suppose X has exact cover C'. Assign z to each B_j such that $C_j \in C'$. This is feasible since no time x_i appears in two elements of C', and it takes care of q of the n meetings B_j. To each of the remaining $n - q$ meetings B_k assign one of the r_j. This leaves r_j free at all times except $c_{k,1}$, $c_{k,2}$, and $c_{k,3}$, so we may assign the $3q - 3$ times $T - \{c_{k,1}, c_{k,2}, c_{k,3}\}$ to A_j. The converse is similar: in any solution to the TTC instance, z must be assigned to exactly q of the B_j, and these define an exact cover for X. □

The constructed instance amounts to assigning meetings to a resource z (after their times have been fixed) so as to maximize the number of times that z is used. The importance of being able to do this was discussed in relation to bin packing (Section 3.2), but now we find that time-incoherence makes the problem NP-complete even when bin packing problems are absent.

3.4. Intractability owing to conditions on times

As explained in Section 2, the choice of a meeting's times is often constrained by requirements for double times, an even spread of times through the week, and so on. The language for specifying these time conditions in TTL is sufficiently general that it permits a transformation from the archetypal NP-complete problem, SATISIFIABILITY, containing exactly one meeting with no resources but with a complex condition on the choice of times. However, this problem is not encountered in practice: in real instances the difficulty arises from the need to satisfy the simpler time conditions of several meetings simultaneously. The following transformation establishes this NP-completeness:

Theorem 5. EXACT COVER BY 3-SETS $\propto$ TTC.

Proof. Once again we remind the reader that the purpose is to construct independent NP-complete instances of TTC, not merely to prove the result (which has been done before in Theorem 4).

As previously described, the NP-complete EXACT COVER BY 3-SETS problem is as follows. We are given a set $X = \{x_1, \ldots, x_{3q}\}$, and a collection of 3-subsets of X called $C = \{C_1, \ldots, C_n\}$ with $n \geq q$. The problem asks for an exact cover of X, that is, a subcollection $C' \subseteq C$ such that every element of X occurs in exactly one element of C'.

Let each $C_j = \{c_{j,1}, c_{j,2}, c_{j,3}\}$ where each $c_{j,k} = x_i$ for some i. We transform an instance of EXACT COVER BY 3-SETS to the TTC instance whose groups are

timegroup T **is**
  $x_1; \ldots; x_{3q}$;
end T;

group R **is**
  r;
end R;

and whose meetings are $M_1, \ldots, M_q$ where each M_j is

meeting M_j **is**
  3 T;
  r;
end M_j;

In addition, we impose on each M_j the time condition that the three times chosen must be $\{c_{k,1}, c_{k,2}, c_{k,3}\}$ for some k such that $1 \leq k \leq n$.

First suppose that the initial instance of EXACT COVER BY 3-SETS has a solution

$C' = \{C_1', \ldots, C_q'\}$. C' must have exactly q elements. For all j, assign the times of C_j' to meeting M_j. The collection of all these sets of times is pairwise disjoint, as required by the presence of r in each meeting, and each meeting's times satisfy the time condition.

Conversely, any solution to the TTC instance defines a collection of disjoint sets of times, each of which satisfies the time condition, and from this we obtain a solution to the EXACT COVER BY 3-SETS instance. □

The TTL instance constructed here has small meetings, all with the same time condition, which is a simple list of alternative time patterns as often occurs, for example, in university timetabling. This is good evidence of intractability in practice.

Nevertheless there are special cases which can be solved efficiently . If the C_j are pairwise disjoint the problem is obviously trivial. More generally, EXACT COVER BY 3-SETS is solvable in polynomial time if each x_i appears in at most two of the C_j [4].

We can identify a second easy special case based on the concept of a *simple time selection,* which we define as a time selection with a time condition requiring only that the times be chosen from a given arbitrary subset of the set of all times. The problem of assigning times to any number of time-disjoint meetings, each containing any number of simple time selections, can be solved by bipartite matching in a graph whose edges connect nodes representing time slots to nodes representing times.

Based on these two special cases and the observation that heuristic methods usually succeed on this problem, it seems likely that a restricted version exists which encompasses most of the cases encountered in practice, and which is solvable in polynomial time for sets of time-disjoint meetings. Heuristics are certainly adequate if occasional violations of the conditions are acceptable.

Incidentally, we can reinterpret the assignment of times $\{c_{k,1}, c_{k,2}, c_{k,3}\}$ to meeting M_j as the assignment of student r to section k of meeting M_j. This shows that the assignment of sections of university courses to even a single student (as discussed in Section 3.1) is NP-complete.

3.5. Intractability of assigning two forms simultaneously

One useful line of attack is to discover large subproblems that can be solved efficiently . One such is the matching subproblem introduced by de Werra [9] and generalized to 'meta-matching' by Cooper and Kingston [1], which assigns times to all the meetings of one *form* (all meetings having a nominated student group in common) simultaneously, in such a way that the demand for the various types of teachers and rooms does not exceed their supply at any time.

The question naturally arises as to whether it is possible to assign suitable times to two forms simultaneously in polynomial time. In the following proof of NP-completeness, MX_1, …, MX_q stand for the meetings assigned previously, and MY_1, …, MY_q and MZ_1, …, MZ_q for the meetings of the two forms to which we wish to assign times.

Theorem 6. THREE DIMENSIONAL MATCHING ∝ TTC.

Proof: Recall that in the NP-complete THREE DIMENSIONAL MATCHING problem we are given three sets X, Y, and Z, each containing q elements, and a set $M \subseteq X \times Y \times Z$. The problem is to determine whether M contains a matching, that is, a subset $M' \subseteq M$ such that $|M'| = q$ and every element of X, Y, and Z occurs exactly once in M'. We transform this to a TTC instance

whose groups are

timegroup T **is**
  $t_1; \ldots; t_q;$
end T;

group R **is**
  $r_X; r_Y; r_Z;$
end R;

and whose meetings are $MX_1, \ldots, MX_q$, $MY_1, \ldots, MY_q$, and $MZ_1, \ldots, MZ_q$. These meetings all have the same form, typified by

meeting MX_i **is**
  1 T;
  r_X;
end MX_i;

where the MX_i select r_X, the MY_i select r_Y, and the MZ_i select r_Z.

But now, for each triple $m_j = (x_a, y_b, z_c)$ in $\overline{M}$, the complement of M in $X \times Y \times Z$, we create two new resources α_j and β_j and a resource subgroup R_j whose members are α_j and β_j, and we add the selection 1 R_j to MX_a, MY_b, and MZ_c. This completes the transformation.

Suppose first that M contains a matching M'. For each triple $m_k' = (x_a, y_b, z_c)$ in M', where $1 \le k \le q$, assign time t_k to MX_a, MY_b, and MZ_c. Since M' contains each x_i exactly once, each MX_i is assigned exactly one time, and these times are distinct, as required by the presence of r_X in each one. Similar remarks apply to the MY_i and the MZ_i.

It remains to check that all the 1 R_j selections are satisfied. Since R_j has two elements, the only possible violation would be if all three meetings MX_a, MY_b, and MZ_c scheduled for time t_k contained 1 R_j for some particular j. But by construction this would imply $(x_a, y_b, z_c) \in \overline{M}$, contradicting $(x_a, y_b, z_c) \in M'$.

Conversely, if the TTC instance has a solution, the presence of r_X ensures that the MX_i are assigned different times, and similarly for the MY_i and the MZ_i. It follows that the solution can be expressed as a set of q triples (MX_a, MY_b, MZ_c) of meetings that occur simultaneously. By replacing each meeting by the corresponding element of X, Y, or Z, we arrive at a matching $S \subseteq X \times Y \times Z$. Since MX_a, MY_b, and MZ_c occur simultaneously, they cannot all contain the selection 1 R_j for any particular j, so by construction the corresponding (x_a, y_b, z_c) cannot be an element of $\overline{M}$. Hence $S \subseteq M$. □

The complexity of the set $\overline{M}$ is easily achievable in real instances, owing to ‘elective’ meetings which select a number of teachers and rooms of arbitrary types. This would seem to rule out all hope of assigning two forms simultaneously.

4. Conclusion

This paper has demonstrated that the timetable construction problem is NP-complete in five quite independent ways. This explains why timetable construction is so difficult.

The instances constructed in our transformations are such as actually occur in practice. This is important, because it ensures that the intractability is real, not merely an artifact of the method of specification. Where known we have indicated special cases and compromises which may be used to work around the problems.

Against these negative results we can set the limited size of timetable construction instances. High schools with more than 100 teachers are rare; a week of more than 40 times is also rare. University problems are larger but seem to be easier. As ingenuity and computing power increase, timetable construction will become feasible in practice.

References

1. Tim B. Cooper and Jeffrey H. Kingston. The solution of real instances of the timetabling problem. *The Computer Journal* **36**, 645–653 (1993).

2. J. Csima. *Investigations on a Time-Table Problem*. Ph.D. thesis, School of Graduate Studies, University of Toronto, 1965.

3. S. Even, A. Itai, and A. Shamir. On the complexity of timetable and multicommodity flow problems. *SIAM Journal on Computing* **5**, 691–703 (1976).

4. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, 1979.

5. C. C. Gotlieb. The construction of class-teacher timetables. In *Proc. IFIP Congress*, pages 73–77, 1962.

6. R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher (eds.), *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.

7. G. Schmidt and T. Strohlein. Timetable construction—an annotated bibliography. *The Computer Journal* **23**, 307–316 (1980).

8. D. J. A. Welsh and M. B. Powell. An upper bound for the chromatic number of a graph and its application to timetabling problems. *The Computer Journal* **10**, 85–86 (1967).

9. D. de Werra. Construction of school timetables by flow methods. *INFOR – Canadian Journal of Operations Research and Information Processing* **9**, 12–22 (1971).