Techniques for Producing School Timetables on a Computer and their Application to other Scheduling Problems

By J. S. Appleby, D. V. Blake and E. A. Newman

This paper describes a method by which a timetable for a medium-sized school can be produced on a general-purpose digital computer in about 1½ hours. The techniques used and time taken by human beings and the computer program are compared. The use of this work in other scheduling problems is briefly described.

1. Introduction

This paper describes an attempt to produce school timetables on a general-purpose digital computer. In its simplest form the problem consists of arranging the periods for a number of classes, and their corresponding masters, over the week in such a way that no master teaches two classes at once. Although we have studied the methods by which timetables are produced by hand, we have not tried to mechanize the human process completely. The human being has the facility of looking at a problem as a whole, but not of keeping in mind all the details. The computer is rather the opposite: it remembers many more details, but it is difficult to program a computer to take a wider view of the problem. In the program described here, we have tried to do this by using rules which condense the detail and bring out the important features of the whole pattern.

When producing a school timetable by hand a schoolmaster usually knows "from experience" which parts of the table are likely to be most difficult and he does these parts first. In this respect we have followed the schoolmaster, but we have made the computer work out which parts are likely to be difficult. When the schoolmaster gets into difficulty he looks at the timetable as a whole and attempts to get out of the trouble by interchanging pairs of entries, often leading to a long series of interchanges. This technique is difficult to follow in a computer, and probably not desirable.

It is probably not possible to justify the production of school timetables on a computer purely on economic grounds. A master may take 100 hours or more to produce a timetable for a medium-sized school by hand. By computer he would take several hours preparing and punching the data and parameters, and about two hours of computer time. There is also considerable effort and cost involved in producing a program (or set of programs) which will cater for every school. We consider that the more important aspect of this work is in finding out how to use digital computers to do this type of logical work, and its value for other problems, which is described very briefly later in the paper. Originally we should have liked to study factory scheduling problems because of their more immediate importance. It turned out to be almost impossible for us to get the necessary information. We should also have needed the active co-operation of a factory to try out any results. The school timetable problem was chosen for detailed study because we were able to obtain all the necessary information (thanks to the efforts of some schoolmasters), there was no timing problem involved, and we were able to do comparatively abstract experiments on the computer.

2. Possible Methods of Solution

There are a number of possible methods of approach.

2.1 Produce a trial solution (probably by arranging the entries in a random way) and examine it to see which rules are broken; then interchange pairs of entries to try to decrease the number of rules broken. This can be done either at random or according to various criteria. The size of the problem makes this approach very difficult—for each of 35 periods we have, say, 70 binary variables (masters + classes occupied or not occupied) to satisfy only the simplest rule. Another difficulty is that the process may get into a loop, continually repeating a long series of interchanges, and this is difficult to detect in a large problem.

2.2 Try all the possible combinations in turn and test to see if the rules are obeyed. In a typical school with 30 classes and 35 periods per week, the number of combinations is very large—of the order of 10^{500}. Our experience suggests that the number of solutions is very small compared with this. It is possible to devise short cuts to ignore whole groups of combinations which do not obey the rules, but it is difficult to see how this method can ever be made practicable on a computer. (To be economic, the computer should find a solution in less than 3 hours, say 10^4 sec., and even with 10^4 tests/sec. this only allows us 10^8 tests.)

2.3 Starting with a blank timetable, entries are made at random. If the entry breaks a rule it is rejected and another entry made at random. If it is found to be impossible to make an entry, return will have to be made to an earlier point or to the beginning. The number of random trials required to find a solution is the same as the number of tests required in the method of 2.2.

2.4 A heuristic approach starting with a blank timetable in which entries are made into the table according to various criteria, the criteria themselves changing with the situation. Each entry now takes longer than it would if done at random, depending on the complexity of the

Table 1

Masters 1, 2, 3, 4, teaching 3 classes for 4 periods

Period

Class	1	2	3	4	Masters
1					1, 2, 2, 4
2					1, 2, 3, 4
3					1, 1, 2, 3

(*a*)

Period

Class	1	2	3	4
1	2	2	1	4
2		1	2	
3				

(*b*)

Period

Class	1	2	3	4
1	2	2	1	4
2			2	3
3				

(*c*)

criteria and the amount of work involved. However, a small increase in the work per entry gives an enormous decrease in the number of trials required to produce a solution. The optimum program will introduce just enough heuristics to give a high probability of producing a solution on the first trial.

3. Existence of Solutions

One difficulty of this class of problem is to know whether a solution exists, because it is obviously not worth trying to find a solution if none exists. At first sight it appears to be desirable—almost necessary—to find a criterion which shows that solutions exist for a given set of data. Further consideration shows that it is not necessarily very helpful to be able to prove that solutions exist unless the method of proof leads to the solution, and this is unlikely. On the other hand, for a given set of initial data or for data at some intermediate stage, it is helpful to prove that no solution exists.

This can be illustrated by an almost trivial timetable example. Suppose that one is producing a timetable from the data shown in Table 1(*a*), where 4 masters are teaching 3 classes for 4 periods (the masters having some free periods), and suppose that one reaches either of the partial timetables shown in Table 1(*b*) or 1(*c*). Then it is clearly impossible to complete the table; we have devised rules which show this, and these partial timetables can be rejected immediately.

The best type of program is one which has a high probability of reaching a solution if one exists, with rapid indication of failure, preferably giving the cause of failure. If the program then fails to give a solution for a given set of data, one should modify the data to remove the cause of failure. It would be of little help to know that in some of these cases solutions existed, if they could not be found reasonably quickly and economically.

4. Statement of the Problem

For any given school we can assume that some of the data are fixed:

number of days per cycle, a cycle usually being a week and the timetable being repeated every week unchanged;
number of periods per day;
number and distribution of classes. For most schools there will be a number of parallel classes for each age-group.

The headmaster, in consultation with his staff, produces a schedule showing:

subjects for each class;
number of periods for each subject;
allocation of teachers for each subject and class.

The schedule will be in the form:

Class 1 to have 4 periods of Science with master 7;
6 periods of Mathematics with master 10;
4 periods of English with master 4;
2 periods of History with master 4.

For our present work we have assumed that this allocation of teachers is fixed. In some schools the data are more flexible. One is given a list of the subjects which each master can teach and the range of age groups for which he is suited. It is then left to the person producing the timetable to work out the detailed allocation of masters to classes to suit the timetable requirements. Also, when producing a timetable by hand, it is quite common to make minor changes to the master/class allocation if difficulties arise. We do not allow any changes, and thus make the computer's task much harder.

There are a number of other points which must be considered before timetabling can start. The list is not exhaustive and may vary from school to school.

4.1 Some subjects require two or more consecutive periods. These must not straddle a break or lunch period, and this limits the number of possible periods in which these multiples can start.

4.2 The children in any age-group are usually divided into a number of parallel streams or classes according to their overall ability. They may also be divided into "sets" according to their ability in particular subjects. Thus a child in Class 1 of an age-group may be in Set 1 for French and Set 4 for Mathematics. Thus an age-group comprising 3 classes may be split into 4 sets for Mathematics, and each set will contain children from all three classes. This means that all children in the age-group must take Mathematics at the same time, and in

general when setting of a subject occurs, all classes in the age-group must take the subject at the same time.

4.3 Where there are a number of periods per week in the same subject they should be spread over the week, e.g. on a 5-day week, 5 periods should preferably be spread one per day and in no circumstances should there be more than two periods in one day.

4.4 For some subjects there may be a limited number of special rooms, laboratories, etc.

4.5 There may be special requirements for subjects like games and P.T. which must not occur on the same day or immediately after lunch.

5. Program 1

The first program was quite simple in conception, and although it proved inadequate the later programs are based on it. The idea was to start with a blank timetable and to fill in entries one at a time, each entry consisting of one or more classes for one or two periods (i.e. one line of input data). The line which is to be entered next is chosen according to some criterion of how difficult it is to fit, the most difficult being taken first (Fig. 1). Having chosen a line, there will usually be several possible periods into which to put it, and it is put into that period which causes least interference with the remaining lines. The process proceeds either until all the entries have been made, or until it can be shown that it is impossible to do so. At first there will appear to be a lot of freedom in choosing the period into which to put the selected line, but we feel that it is essential to choose the period very carefully; a few badly placed entries at the start can cause havoc later on.

Input data

It is convenient to arrange the data as shown in Table 2, each line referring to a particular master/class combination.

Comparing this with the data provided by the headmaster, it will be noted that there is no reference to the subject: Class 1, which has 4 periods of English with master 4 and 2 periods of History with master 4, is entered as 6 periods with master 4. The timetable is then made up without reference to the subject, and it is a trivial exercise to separate the subjects afterwards. Line number 15 shows the effect of setting—described in Section 4.2—all three classes and their corresponding masters must be entered as a single entry.

Each line specifies a number of requirements or constraints—classes, masters, special rooms, etc.—and two lines clash, i.e. cannot occur in the same period, if they have any constraint in common. For this purpose all types of constraint have exactly the same significance and in practice we allocate one marker digit to each constraint and test for a clash between two lines by a simple "AND" operation. It is convenient and economical for us to work out a complete set of tables showing clashes between any pair of lines and store them, rather than work them out each time they are required.

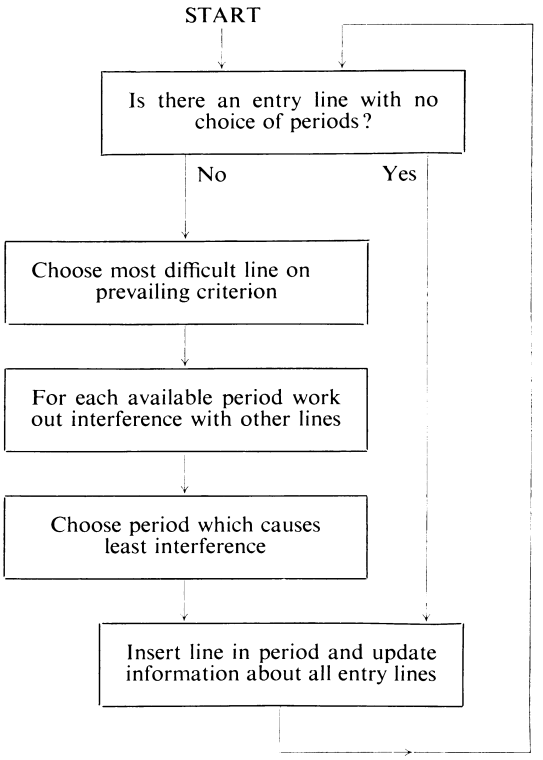

Fig. 1.—Basic flow chart of Program 1

Table 2

Input Data

LINE NO.	CLASS	MASTER	OTHER CONSTRAINTS, IF ANY	NO. OF PERIODS SINGLE	NO. OF PERIODS DOUBLE
0	1	7	—	4	—
1	1	10	—	6	—
2	1	4	—	6	—
9	2	7	—	1	1
15	1, 2, 3	8, 9, 14	—	4	1

Other information about each line is also stored and updated after each entry has been made:

a set of marker digits, one for each period still available;
the number of periods available to single entries;
the number of periods in which doubles can start;
indicators showing the spread over the week.

Criteria derived from data

When an entry is made from a line, the appropriate period digits are cancelled in all the other lines with which it clashes. Thus, at any stage, the period-available digits show the periods into which a single can be entered. The number of these digits, P_S, and the number of single entries remaining, N_S, are kept up to date.

The corresponding P_D for doubles is not so easy to update, as one has to take into account the position of the breaks. The program has been written in a general form with 8 periods/day, and a single code word is all that is necessary to define the position of the breaks.

Some simple function of P and N, such as $P - N$ or P/N, gives a useful working number to determine the amount of freedom in entering a line. It can easily be tested as follows:

$P - N$	P/N	
negative	<1	insufficient periods remaining to make the outstanding entries
0	$=1$	exactly sufficient periods, i.e. no choice
positive	>1	measure of the choice still available.

Besides trying to keep $(P - N)$'s as large as possible, the program attempts to satisfy secondary criteria such as good distribution over the week (see Section 4.3) as shown in the following Sections.

Choosing the next line to enter

1. If there is any line with no choice, then this is entered in the prescribed periods. It may happen that this causes the $P - N$ of another line to go negative, and this is one of the indications of failure of the program.

2. If no lines are forced, then we choose the next line on the basis of some criterion of how difficult it is to enter—some function of N, P or the other properties of the lines. So far we have mostly used an empirical criterion, though there is some theoretical basis for using $P - N$ when this is small and P/N when there are no small $(P - N)$'s. It is also quite useful to use a secondary criterion to choose between several lines whose primary criteria are equal.

Where to put the selected line

When a line has been selected we examine the marker digits showing which periods are available. Because of the requirement of Section 4.3 to spread the entries from any line over the week, we first select the digits from those days which have no entry from this line, and try to fit the entry into one of these periods. If this is found to be impossible we try the digits from the days which have one entry.

We choose the best period by assessing the interference which the selected line causes to all the other lines with which it clashes, e.g. if the selected line clashes with line x, then for every period into which the selected line can be put we work out

$$A = \Sigma \frac{R_{Sx}N_{Sx}}{(P - N)_{Sx}} + \Sigma \frac{2R_{Dx}N_{Dx}}{(P - N)_{Dx}} \qquad (1)$$

for all x, where

R_{Sx} = reduction in the number of single periods available to line x caused by putting the selected line in that period,

N_{Sx}, $(P - N)_{Sx}$ are the values N_S and $(P - N)_S$ for line x,

R_{Dx}, N_{Dx}, $(P - N)_{Dx}$ are the corresponding terms for interference with doubles.

This is also largely empirical, but it seems logical that it should be of the form $RN \,.\, f(P - N)$ or $RN \,.\, f(P)$, and the change in A must obviously be large for small values of P or $P - N$.

We now choose the period for which A is minimum. If putting the selected line into a period reduces the periods available to a line whose $P - N$ is zero, then, with our formula, A is infinite for that period and we must avoid that period.

Updating the entry lines

Having selected a line and decided where to put it, the last stage in the process is to update all the information in the lines with which the selected line clashes. During this process we test all $(P - N)$'s which change: if any goes negative the program has failed; if a $P - N$ goes to zero the associated line has no freedom and we note its reference number to be entered next.

Results

Program 1 was run on DEUCE and can solve some pilot problems of the size shown by the example of Table 3. This has 6 classes and 7 masters with 5 periods/day and 3 days/week. The day was divided into 2, 2, 1 periods so that doubles could only start on periods 1 or 3 on each day. The problem is reasonably realistic, and its size was chosen so that it was possible to include setting, double periods, and distribution over the week without being so large that it took a long time to run on the computer. However, there are some problems of this size which Program 1 cannot solve, and we know how to devise such problems.

Shortcomings of Program 1

The most obvious shortcoming is that lines which clash with one another, e.g. a number of lines which involve the same class or the same master, compete for the same periods. This is illustrated by the entry lines of Table 4. We have shown period-available digits for a single day of 7 periods, commas denoting the mid-morning break and lunch. There is no difficulty here about making any 4 out of these 5 entries, but it is clearly impossible to make all 5 entries because there are only 4 periods available. This leads logically to the next stage of considering not only individual lines, but also groups of lines.

6. Program 2

The logic for dealing with groups of lines causes a very considerable increase in program size and the work which the computer has to do; this is dealt with in Program 2.

Table 3

Example

CLASSES

		1	2	3	4	5	6
DAY 1	1	1, 2, 3			7	4	6
	2						5
	3	6	5	2	4	7	1
	4		2	5	3, 4		
	5	3	2, 7		4, 5, 6		
DAY 2	6	3	2, 7		4, 5, 6		
	7	1	7	3			
	8	1, 2, 3			7	4	6
	9				6	7	5
	10	6	5	2	3, 4		1
DAY 3	11	1, 2, 3			6	4	5
	12	3	1	7	5		6
	13	6	5	2	7	3	1
	14	7			3, 4		
	15	1	2, 7		4, 5, 6		

Table 4

C	M	N_S	PERIODS AVAILABLE	$P - N$
1	7	2	100, 11, 01	2
1	10	1	100, 10, 01	2
1	4	2	100, 10, 01	1

Table 5

LINE NO.	N_S	N_D	PERIODS AVAILABLE	
1	1	–	111, 11, 10	
2	1	–	111, 11, 00	
3	–	1	111, 10, 00	
4	1	–	011, 10, 00	
5	1	–	010, 10, 00	
	$N = 6$		111, 11, 10	$P - N = 0$

Groups of lines

If we take a group consisting of two or more entry lines such that each line clashes with every other line, i.e. no two members of the group can appear in the same period, then we can formulate some useful properties of the group. Considering the lines of Table 4, which all involve Class 1, we could write

	N	Periods available	$P - N$
Class 1	5	100, 11, 01	−1

N is the sum of the number of entries in the individual lines, doubles counting as two. The periods available have a period digit whenever one occurs in any member of the group (the logical "OR" of the individual lines). We can now count the period-available digits and thus get $P - N$ for the group. Because of the way in which we have defined a group, N and $P - N$ have the same significance for a group as they have for a single line. The fact that $P - N$ is negative in this example shows that there are not enough periods available to make all five entries in the group.

Group information can be used in several ways:

1. To give an indication of failure. This does not sound very useful, but indication of failure may be given long before it is shown by individual lines.
2. To try to avoid reducing the group $P - N$, and in particular to try to avoid it going negative. Most classes, usually all in the lower school, are occupied for every period of the week, and the group of these classes will have $P - N = 0$. Masters will have several free periods and the group $P - N$ will start with a value equal to the number of free periods.

Group information is used as an additional term in equation (1), i.e. for every period into which the selected line can be put, we work out the interference caused to all the groups.

$$A_G = W \cdot \Sigma \frac{R \cdot N}{P - N}$$

where R = reduction in the periods available to the group caused by inserting the selected line into that period,

and W is a weighting factor so that we can alter the relative effect of interference caused to groups compared with that to lines.

3. For the special case when the group $P - N$ is zero we examine the period-available digits to see if any is produced by only one entry line, as shown by period 6 in Table 5, which is produced by line 1. Because the group $P - N = 0$, a member of the group must be entered into each of the periods available: therefore, line 1 must be put into period 6. When this has been done it will be found that line 2 must be put into period 5, line 3 into periods 1 and 2, and thus line 4 into period 3 and line 5 into period 4. There is thus only one solution to this

example, and it is important to make these forced entries as soon as they become known. Otherwise there is a danger that in this case the program might take line 5, because it has only two periods available, and enter it in period 2, making a solution impossible.

Choice of groups

Ideally one should take all possible groups of lines, i.e. all possible pairs of lines with a constraint in common, all possible triples, etc. This would lead to a prohibitive number of groups as there is a great deal of computer work involved in working out and using group information.

We have compromised by using the following groups:

(1) those lines involving one class,
(2) those lines involving one master,
(3) those lines involving several classes.

The groups involving several classes require a little more explanation. Consider an age-group (classes 1, 2 and 3) in which setting occurs, and suppose that some entries have been made in this part of the timetable—Table 6(*a*)—and suppose that the remaining lines in the group of class 1 are shown in Table 6(*b*). It is clear from the partial timetable that only period 7 is available for lines involving both class 1 and class 2, and with lines *a* and *b* outstanding a solution is impossible. Had we had a group whose lines all involved both class 1 and class 2, this situation would probably not have arisen, and if it had it would have been detected immediately:

Group of	N	Periods available	$P - N$
classes 1 *and* 2	2	000,00,01	−1

This example also illustrates the danger of having too few groups—the whole group may be satisfactory, but a sub-group not. The case where a sub-group is found to be negative when the remainder of the group is entered can be catered for by an iterative process. If the current state of the working data is stored each time a given number of entries has been made, one can go back to the last point at which data was stored and add the sub-group to the list of groups. This new group is then checked and if its $P - N$ is zero or positive one can proceed from that point again. If not, one has to go farther back—if necessary to the beginning. The program will then try to stop the new group from going negative. If at any point a group or sub-group can be found whose $P - N$ is negative, then the timetable cannot be completed.

(In the example of Table 6 we could also say that line *c* caused the trouble because only line *c* contributes to periods 3 and 4 in the period-available digits. Therefore, when line *c* is entered P is reduced by 2 or 3 and $P - N$ by 1 or 2. The effective group $P - N$ is really −1 and some addition to the program would cater for this. The case where two or more lines give trouble of this kind could be covered in principle, but the extra program would be very large.)

Table 6(*a*)

		Periods						
		1	2	3	4	5	6	7
Class	1	×	×			×		
	2		×	×	×		×	
	3		×		×			

Table 6(*b*)

LINE NO.	C	M	N	PERIODS AVAILABLE	
a	1, 2	4, 5	1	000, 00, 01	
b	1, 2, 3	12, 20, 22	1	000, 00, 01	
c	1	8	1	001, 10, 01	
	Group of Class 1		3	001, 10, 01	$P - N = 0$

Results

Program 2 was run on ACE. For this type of problem ACE is probably at least five times as fast as DEUCE, and our ACE Program 2 is slightly faster overall than Program 1 on DEUCE.

We believe that Program 2 will solve all pilot problems of the size shown in Table 3, and we have produced a complete timetable for a school of 26 classes × 5 days/week × 7 periods/day with 35 masters. This involved a total of 26 × 35 = 910 squares in the timetable, but because of setting and double periods the actual number of entries is about 480 from 190 entry lines. Initially entries are made at the rate of 4–5 per minute, but the program gradually speeds up and the total computer time is about 1½ hours. This was completely realistic, involving all the restrictions met in the actual problem.

The complete timetable, which is shown in Table 7, has some shortcomings, nearly all in the form of poor distribution over the week, e.g. class 16 has master 8 for three periods over the week and two of these occur on one day. Each school will have its own idea about some features of the timetable. If two periods occur on the same day, some schools will prefer that they shall make a double period, while others will prefer them to be separate—this could be catered for by an extension to the program. Some schools consider that the masters' free periods should be spread over the week, some prefer them to be grouped, others consider this point of secondary importance. All features of this kind could be added to the program, but the first requirement is to get a program which can be relied on to ensure that all the primary rules are satisfied, and preferably gives some freedom of arrangement to cater for secondary rules.

7. Program 3

Program 2 also was found to have a serious shortcoming, though it does manage to produce a solution in some cases. We consider the number of periods available for a given class, i.e. we produce a table of

Table 7

	CLASSES																										
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	0
MONDAY																											
1	1	F	20	5, 15, 30		6, 12, 31, 32			8	4	16	7, 27, 33			11, 24, 25, 26			2, 22, 23			29	3	21	10, 14, 17			
2	F	F	1*	30	1*	9	4	8	31	12	20										28	35	21				
3	22	6, 31	25		11, 13	19	4	21	2, 14, 29			5, 17, 24, 32			9, 12, 16			35	7	28	18	34		20	36	10	
4	22		4	26, 32		14, 17, 25			5, 13, 16, 23			19	20	21	3, 15, 27, 33			6, 12, 29			34	9	28	18	7	10	
5	22	6	19			17, 25, 33						24, 29, 31			2	8	20	7, 9, 27, 30			3, 10, 15			4	35	28	
6	22, 25	16	20	4, 19		15, 27, 33			7, 11, 30			24, 29, 31			1, 2, 17, 23			8	35	14	3, 5, 32			18	28	6	
7			F	4, 19								5, 26, 32			2, 17, 23, 31			8	28	14	10, 21, 29			35	34		24
TUESDAY																											
8	17	16	19	5, 15, 33		11, 13, 22, 32			2, 10, 29			14, 23, 25			3, 18, 26			7, 9, 27, 30			12, 24, 31			4	28	35	
9	22	31	20	5, 15, 33		6, 13, 26, 32			7, 11, 30			19	25	4	9, 12, 16			8	34	28	10, 21, 29			18	24	3	
10	22	F	4	26, 32		15, 27, 33			2	8	20	14, 23, 25			9, 12, 16			3, 5, 11, 13						35	28	6	
11	22	6	F	4, 19		13, 17, 25, 33			8	12	16	24, 29, 31			2	23	20	7, 9, 27, 30			3, 10, 15			34*	35	34*	
12		16	25	27	13	9	21	8	31	4	19	5, 17, 24, 32			2	23	20	6, 12, 29						18	7	28	
13	19		F	15, 27		6, 12, 31, 32			5, 13, 16, 23			7, 30, 33			11, 24, 25, 26			8	20	14	29	9	21	4	18	10	
14	F	F	1	30	11, 13	19	34	21	5, 16, 23, 26			8	25	24	3, 15, 27, 33			6, 12, 29			35	9	28	10, 14, 17			
WEDNESDAY																											
15	22	16	F	7, 15, 30		6, 12, 31, 33			5, 11, 26			14, 23, 25			2	8	20	34		35	29	9	21	4	24	28	
16	19		4	30	11, 13	14, 17, 25, 32			5, 16, 23, 26			7, 27, 33			2	8	20	6, 12, 29			3, 10, 15			18	35	28	
17	22	6	F			15, 27, 33			31	8	19	5, 17, 24, 32			9, 12, 16			34	20	14	10, 21, 29			4	28	35	
18	F	F	20	26, 32		9	4	21	2, 14, 29			16, 23, 25			3, 15, 27, 33			8	28	34	12, 24, 31			35	7	10	
19	17	16				19	21	8				33	25	4	34*	23	34*	3, 5, 11, 13			12, 24, 31			36	18		
20	F	31	25	5, 15, 27		6, 13, 22, 26, 32			7, 11, 30			34	20	4	1, 2, 17, 23			3, 14, 16, 19			10, 21, 29			18	28	36	
21	1	6	19	5, 15, 27		34*	4	34*	8	12	16	7, 30, 33			11, 24, 25, 26			2, 22, 23			35	3	28	10, 14, 17			
THURSDAY																											
22	8, 12, 14					15, 27, 33			31	4	16	19	34	21	11, 24, 25, 26			2, 22, 23			18	9	35	20	7	3	
23	19		F	7, 15, 30		6, 11, 13, 22, 32			2	12	20	16, 23, 25			3, 18, 26			8	28	35	29	9	21	10, 14, 17			
24	25	6	F	5, 15, 27		13, 22, 26, 32			7, 11, 30			8	20	4	2, 9, 17, 23			3, 14, 16, 19			12, 24, 31			34		35	
25	17	F	20	34		6, 11, 13, 26			31	12	16	8	25	21	9, 15, 27, 33			2, 22, 23			3, 5, 32			4	35	28	
26	34			26, 32		19	4	21	2, 10, 29			14, 23, 25						3, 5, 11, 13			12, 24, 31			35	28	6	
27	22	16	1	5, 15, 33		14, 17, 25			11, 13, 26			24, 29, 31			2	23	20	7, 9, 27, 30			28	35	21	18	36	10	
28	22	16	19	33	11, 13	6, 12, 31			8	34		5, 17, 24, 32			2	23	20				3, 10, 15			18	28	36	
FRIDAY																											
29	25	F	20	30	11, 13	15, 27, 33			31	4	19	5, 17, 24, 32			3, 18, 26			6, 12, 29			34	9	35	36	7	28	16
30		6	F						34	12	19	5, 26, 32			2, 17, 23, 31			8	28	14	29	9	21	18		35	
31	22		F	7, 15, 30		6, 12, 31, 33			5, 13, 26			8	25	34	2, 9, 17, 23			3, 14, 16, 19			10, 21, 29				28	35	
32	8, 12, 14					19	21	17	13, 16, 23, 26			24, 29, 31			2	34		35	20	28	3, 5, 32			4	18	10	
33	1, 26*	16	F	1, 26*		14, 17, 25, 32			7, 11, 30						9, 15, 27, 33			2, 22, 23			28	34		4	35	3	
34	22	F	4	5, 15, 33		6, 12, 13, 31			2	8	16	19	20	21	11, 24, 25, 26			7, 9, 27, 30			18	3	28	10, 14, 17			
35	17	6	25	26, 32		19	4	21	2, 14, 29			7, 27, 33			9, 12, 16			8	35	34	3, 10, 15			20	24	28	

F = Free period. * Classes combine for appropriate periods. Column 0 shows masters who are not available for particular periods.

masters against periods for a given class, and we also produce a table of classes against periods for a given master. This ensures that there are sufficient periods available for the single-master and single-class groups. We are not, however, ensuring that, for a given period, sufficient masters are available to take the required number of classes. For example, considering only period 1 of Table 8(*a*), if classes 2, 3 and 4 are occupied for all periods ($P - N$'s are zero), then each must have one entry in period 1. This is clearly impossible because we have two masters and three classes, Table 8(*b*), or M(asters) − C(lasses) = −1, and we can treat $M - C$ for a period in the same way as $P - N$, *providing* $P - N$ for all classes concerned is zero.

If $M - C = 0$ and there is only one entry for a particular master, as shown in Table 9, then that master/class must be entered in that period. Any other entry for that class will make $M - C$ negative. This is exactly analogous to the case where a group $P - N = 0$ and only one entry occurs for a period.

A comparatively small extension to Program 2 will allow $M - C$ to be monitored for each period and make this factor influence the choice of period into which the selected line is put. The neglect of $M - C$ has been shown to be a major cause of failure in Program 2.

Results

Program 3 has not been run on a computer. A timetable for a second and more difficult school has been produced partly by ACE using Program 2, and then taken over just before $M - C$ went negative and completed by hand. The rules for the manual program were chosen so as to avoid a lot of arithmetic, and were more in line with our ideas for future machine programs described below.

8. Future Programs

The extension to Program 3 means that we are considering in turn three planes from a cubic array of classes, masters and periods. A digit C_a, M_b, P_c in the array means that line C_a M_b has period c available. At the start these planes are not equivalent because the masters have free periods and the periods have excess masters, but the classes have no free periods. The program should make entries in such a way that they have least effect on the whole array. The best approach for this seems to be to make entries from those rows or columns which have fewest digits—the number of digits being weighted by the value of $P - N$ or $M - C$. This type of program lends itself somewhat to manual operation and is liable to be inefficient on a computer.

It is difficult to deal with the array as a whole, and this is made much worse by the practical difficulties such as:

(*a*) setting—several masters, several classes;
(*b*) several masters taking one class;
(*c*) one master taking several classes;
(*d*) double periods;
(*e*) spread over the week.

Table 8(*a*)

C	*M*	PERIOD 1
2	3	1
2	7	1
3	3	1
3	7	1
4	3	1
4	7	1

Table 8(*b*)

		PERIOD 1
		Masters
		3 7
Classes	2	1 1
	3	1 1
	4	1 1

Table 9

		Masters		
		3	7	10
Classes	2	1	1	1
	3	1	1	
	4	1	1	

9. Computer Times

Program 2 makes about 5 entries/minute into the timetable, and Program 3 will probably drop to 3 or 4/minute. Both of these programs involve a large number of logical operations and a fair amount of arithmetic. A direct comparison between the machine and a human being is not possible at the moment, because the only program which has been done by hand involved looking for rows and columns in the array with the smallest number of digits. Two human operators working together on this program can do about 10 entries/hour, and we should expect the computer to do about 3 entries/minute. On Programs 2 or 3, the rate by hand would probably be 5 entries/hour with two operators. This means that the machine is only 50–100 times as fast as a human being, despite the fact that we are considering a job which has been hand-programmed. At first sight this seems to be an incredibly small factor: it is partly accounted for by the fact that ACE has no random-access store. With some difficulty we were able to put the main working data and working program into the mercury delay-line stores each holding 32 words. This means that random access to data takes either 16 or 32 word-times, and an immediate-access store of even 2,000 words would probably increase the overall program speed by a factor of 4. This would make the machine faster than a human being by a factor of 200–400.

It is interesting to note that the time taken by human beings to complete a timetable using the methods described in this paper is of the same order as that taken by a schoolmaster using normal methods.

10. Relationship with other Scheduling Problems

Factory scheduling

The type of scheduling most closely resembling the school timetable approach is that of detailed scheduling of factory work-load on to machines and men. As far as we know, this problem has not been tackled successfully.

All that has been done to date is to allocate groups of jobs to sections of men and machines, and leave it to the foreman to do the detailed job allocation. The computer is then working on average job times and changeover times between jobs. This is probably adequate in many machine shops where such things as the changeover time between jobs is fairly constant for any pair of jobs or is small compared with the job-time. The simplified scheme is probably also adequate where job-times are fairly short compared with shift-time, and where there is not much loss in starting and stopping. However, if these conditions do not apply, there may be considerable loss in splitting the whole job into separate sections without interaction. If the problem is too big to be planned by a small group of men who can work closely together, it must be broken down into more or less independent sub-problems each of which is planned by a small group. This causes loss of flexibility and efficiency—just as on a large timetable one would lose flexibility by not being able to see the table as a whole.

The school timetable problem is in some ways easier and in others much harder than factory scheduling. It is easier because all the periods are of the same length or simple multiples of this length. School timetables are, however, made vastly more difficult by the fact that most classes are occupied for every period of the week. Masters have some free periods and may have a utilization of 85%. In factory scheduling this would be equivalent to a machine loading of 85% and a man utilization of 100%—the difficulty rises swiftly as one approaches 100%.

Air traffic control

There are many similarities between the way in which we have chosen to tackle the school timetable problem and a class of problem which includes road, rail, and air traffic control. We could imagine the entry line to be analogous to an aircraft wanting to enter the control area. Various possible flight paths can be allocated to it for periods of time into the future, and these obey similar rules for clashing. Before it can be accepted it has to be allocated to a group which has some freedom (i.e. its $P - N$ or equivalent term must be positive non-zero). The combination of flight paths allocated represents the timetable. There are, of course, differences—we are dealing with a continuously varying timetable of the factory scheduling type rather than the cyclic one of the school timetable.

11. Acknowledgements

The authors would like to thank Mr. B. Cook of Edinburgh Academy, Mr. A. C. Crowther of Altrincham Grammar School, and Mr. J. E. Bullard of Christ's Hospital who have supplied them with details of their timetables. The work described has been carried out as part of the research programme of the National Physical Laboratory, and this paper is published by permission of the Director of the Laboratory.

Work of 'New Generation' to be shown at London Computer Exhibition and Symposium

Electronic Computer Exhibition: Olympia, London, 3–12 October 1961
Business Computer Symposium: 4, 5, and 6 October 1961.

"The Work of the New Generation of Computers" will be the theme of the Electronic Computer Exhibition to be held at Olympia, London, from 3 to 12 October 1961. Emphasis will be placed at least as strongly on new applications as on new equipment.

At the same time the Business Computer Symposium, to be held at Olympia on 4, 5, and 6 October, will bring forward users from some of the largest and smallest firms in Britain to speak of their experience with computers and to exchange information. Plans are being made to reach out especially to the businessman who is currently faced with the decision to instal electronic data processing equipment. The exhibition will aim to talk to him in management terms.

Display boards on the stands will feature statements from directors of user companies underlining the value of a computer to their business. In order to widen the range of experience placed at their disposal, those attending the exhibition will be able to arrange, on exhibitors' stands, visits to computer system installations of particular interest to them—to take place after the exhibition.

The exhibition and symposium will be organized jointly by the Electronic Engineering Association and by the Office Appliance and Business Equipment Trade Association which will organize the Business Efficiency Exhibition to run concurrently at Olympia. A list of some 40 exhibitors is divided almost equally between manufacturers of computers and firms concerned with ancillary equipment.

Further particulars about exhibits and papers will be published in *The Computer Bulletin*, Vol. 5, Nos. 1 and 2.