The preparation of examination time-tables using a small-store computer

By A. J. Cole*

A method of preparing time-tables subject to certain imposed conditions is described. The method, which is founded on a sorting technique, has produced minimum or near minimum solutions in practical applications.

1. Introduction

In this paper we describe a method of preparing examination time-tables using a small-store computer without the addition of magnetic tape or some other backing store. Since most universities and technical colleges either have or have the use of such a machine the following method should be of some general interest. It has the merit that additional "local" conditions can be easily written in, and by the addition of such extra conditions it can be adapted for the preparation of weekly lecture and practical-class time-tables. It should be noted here that it is possible that, because of the essential limit on the number of periods available in this case, there may be no solution to the problem.

Several solutions or partial solutions of the general time-table problem have been described in other papers. For example, Appleby, Blake and Newman (1961) and Csima and Gotlieb (1963) considered the preparation of school time-tables, and Sherman (1963) the scheduling of university classes. In the school problem, in addition to the limit on the number of periods available in one week, it is also necessary that each class shall be occupied all of the time. This condition is not required in the corresponding university or enlightened technical college problem. Further, the allocation of lecturers to classes does not apply to the examination problem, and is not usually of any great difficulty in the weekly lecture case. Sherman describes an integer linear-programming solution to a simplification of the typical problem. In an unspecified medium-size computer he can handle up to fifty subjects, and comments "Even with the aid of the most powerful computers in existence it is presently hopeless to think in terms of optimizing the schedules of a large school." The emphasis in the present paper is on producing a reasonable solution which can handle as many subjects as possible without taking too much computer time.

The program written at Leicester University for an Elliott 803 computer with a minimum 4096-word store can handle up to 340 subjects with a maximum of fifteen papers per subject. The local conditions imposed are described below, but subject to these, the solutions obtained have either been the best possible or something very near to it, in the sense that the final time-table produced takes up a minimum or near minimum number of periods. In addition, the time-tables produced from actual data do not usually have the largest number of papers per period allocated to the first period, but build up to a maximum and then decrease again.

2. Statement of the problem

When the time-tables have been traditionally prepared by hand it is difficult to formulate explicitly some of the conditions which have been imposed, sometimes subconsciously, to account for the needs and whims of certain examiners. The following programmed conditions have been found to be sufficient to reproduce most if not all of these possibilities.

(i) The papers corresponding to a certain subject either *must* or *must not* or *need not* occur in consecutive periods.

(ii) No paper of subject A must precede any paper of subject B.

(iii) The papers of subject C must all appear in odd (morning) periods unless they are also required to be consecutive, in which case the first paper must be in an odd period.

(iv) The total number of students sitting papers in any one period is either bounded above or is restricted by the accommodation available and the condition that all students sitting the same paper must be in the same room.

The problem of adequate accommodation did not apply in Leicester University so the program did not include this condition. It has been indicated below how these conditions can be included.

Condition (ii) is of particular use when, for example, a particular general-degree subject has several papers, the first of which is also to be taken by other students taking it as a subsidiary. In this case the paper causes fewer clashes with other subjects if it is treated as two different subjects A and B, both of which are to be taken by general-degree students, and a condition of type (ii) imposed. In cases in which practical examinations have to be split into two groups preferably in the morning and afternoon of the same day, this can be arranged by making two papers appear in consecutive periods subject to condition (iii).

* *Computing Laboratory, Department of Mathematics, The University, Leicester.*

The problem then is to prepare a time-table such that no student is required to sit more than one paper per period subject to any or all of the conditions (i) to (iv), and such that the total number of periods taken is a minimum or near minimum.

3. Input data

Four input data tapes are required. The first records the subjects to be taken by each student. A typical data line would be

$$m_1\ m_2\ \ldots\ m_p\ *$$

where each m_j represents a subject number and * terminates the subjects for a particular student. If student numbers are to be taken into consideration this input line can be modified to

$$m_1\ m_2\ \ldots\ m_p\ /\ k\ *$$

where k is the number of students taking this combination of subjects. The subjects have all been numbered, but if it is preferred to give them their actual titles, then they can be translated into numbers by an auxiliary translation program. In the rest of this paper it will be assumed that all subjects are numbered. Since this data tape is the longest and most complicated it is advisable to incorporate correction symbols in the input routine.

The second data tape consists of rows of three integers u, v, w, where u is a subject number, v is the number of papers in that subject and w is 2, 1 or 0, according as the papers are to be in non-consecutive, consecutive or arbitrary periods. The third tape consists of pairs of integers A, B, where subject A must not precede subject B, and the fourth consists of a list of subjects which must appear in odd periods. Each of these tapes must be terminated by some agreed symbol.

Table 1 shows the four tapes prepared for a time-table for the general degree. Tapes 1 and 2 appear only in part.

Table 1

Input data tapes

TAPE 1	TAPE 2	TAPE 3	TAPE 4
2 3 6 10 5 11 *	1 1 0	10 6	11
2 3 5 11 13 *	2 1 0	11 5	21
14 15 16 17 *	3 1 0	16 15	
2 3 6 10 1 18 *	4 1 0	18 1	
2 3 19 1 18 *	5 1 0	21 20	
23 20 21 24 *		27 26	
14 23 24 *	11 2 1	28 8	
23 24 19 *	13 2 0	30 29	
9 8 28 29 30 *	14 3 0	34 33	
9 1 18 29 30 *		12 4	
9 22 7 *	24 3 2		

4. The incompatibility table

Two subjects are said to be incompatible if any student takes them both: from the student/subject list a complete table of incompatibilities of subject can be drawn up. It is sometimes convenient to be able artificially to make two subjects mutually incompatible. This can be done by introducing a spurious student taking these two subjects. If student numbers are being counted then k defined above must be set to zero. In the Leicester program a maximum of 340 subjects can be handled. Since each word of the 803 computer contains 39 bits, eight full words and 28 bits of a ninth are needed to record the incompatibilities for each subject. The spare 11 bits of the ninth word are used for another purpose. As each student's list of subjects is read in, a record of incompatibilities in the form of bit markers is made in the appropriate positions of the nine words allocated to each subject. Care must be taken here to see that a particular incompatibility has not already been recorded. When the incompatibility table has been formed it is useful to offer as an optional output an incompatibility list by subjects. Such a list is illustrated in Table 2.

At this stage Tape 2 can be read in and its data recorded in the spare 11 bits mentioned above. Four bits are used to record the number v of papers, and four are reserved to count the papers as they are allocated to periods. Two bits are used to record the "consecutive" condition digits. The spare bit is used for another purpose described later.

5. The grand sort

For the rest of this paper we can forget that the incompatibilities for each subject are stored in nine machine words, and regard them simply as an $N \times N$ array of bits, where N is the highest subject number used. From a number of trials it has been established that a very satisfactory ordering of subjects for subsequent allocation in the time-table is first by the total number p of incompatibilities per subject, followed by sub-ordering first by "consecutive" bits q, then by number of papers r, and finally by the original ordering s of the subjects. Corresponding to each subject a composite word is built up, and the N such words are sorted in

$+p$	q	r	$N - s$

order of magnitude. The least-significant part $N - s$ of each composite word is then amended to become s again, and the sequence $s_1, s_2, \ldots, s_N$ of these numbers determines the new ordering of the incompatibility array by subjects. That is, the old row s_j has to become the new row j and at the same time the old column s_j has to become the new column j. This interchange can be done using only one extra location, by at the jth stage picking out s_j, and if $s_j > j$ carrying out an interchange

Table 2

Incompatibility list by subjects

SUBJECT	INCOMPATIBILITIES																			
1	2	3	4	6	8	9	10	13	15	16	18	19	23	25	28	29	30			
2	1	3	5	6	7	10	11	13	15	16	17	18	19	23	26	27				
3	1	2	5	6	7	10	11	13	15	16	17	18	19	23	26	27				
4	1	9	12	15	16	18	25	29	30											
5	2	3	6	10	11	13	15	16	17	23										
6	1	2	3	5	7	10	11	13	18											
7	2	3	6	8	9	10	19	22	26	27	28	29	30							
8	1	7	9	14	18	24	26	27	28	29	30	31	32	33	34					
9	1	4	7	8	12	15	16	18	19	22	24	26	27	28	29	30	31	32	33	34
10	1	2	3	5	6	7	11	13	18											
11	2	3	5	6	10	13	15	16	17	23										
12	4	9	29	30																
13	1	2	3	5	6	10	11	18	19											
14	8	15	16	17	19	20	21	23	24	25	28									
15	1	2	3	4	5	9	11	14	16	17	18	25	26	27	29	30	31			
16	1	2	3	4	5	9	11	14	15	17	18	25	26	27	29	30	31			
17	2	3	5	11	14	15	16													
18	1	2	3	4	6	8	9	10	13	15	16	19	23	25	28	29	30			
19	1	2	3	7	9	13	14	18	23	24	26	27	29	30						
20	14	21	23	24																
21	14	20	23	24																
22	7	9	29	30																
23	1	2	3	5	11	14	18	19	20	21	24									
24	8	9	14	19	20	21	23	25	28	29	30									
25	1	4	14	15	16	18	24	26	27											
26	2	3	7	8	9	15	16	19	25	27	28	29	30							
27	2	3	7	8	9	15	16	19	25	26	28	29	30							
28	1	7	8	9	14	18	24	26	27	29	30	31	32	33	34					
29	1	4	7	8	9	12	15	16	18	19	22	24	26	27	28	30	31	32		
30	1	4	7	8	9	12	15	16	18	19	22	24	26	27	28	29	31	32		
31	8	9	15	16	28	29	30	32												
32	8	9	28	29	30	31	33	34												
33	8	9	28	32	34															
34	8	9	28	32	33															

in an obvious manner. At this stage the new s_jth row is probably not in its final form. If $s_j = j$ nothing need be done, and if $s_j < j$ then pick out $s_j' = s_j$. If s_j' is still less than j then repeat the process and after at most $j - 1$ steps a value greater than or equal to j will have been obtained. This value indicates the row and column to be interchanged with the jth row and column.

Data tapes 3 and 4 can now be read in and stored in consecutive locations. The pairs of numbers on tape 3 should be stored in single locations in reverse order.

6. The allocation of subjects to periods

We will suppose here that the first $t - 1$ periods have been completed and that we are about to start on the tth. We will also suppose that an N-bit word W has been recorded to indicate which subjects occurred in the last period, and also that the number v of papers per subject has been reduced by 1 each time that subject has appeared in the first $t - 1$ periods, and that a second number v' equal to the number of papers allocated so far has been recorded for each subject. Further, we will suppose that any subject occurring in period $t - 1$ and which was required *not* to appear in consecutive periods will have been recorded with a marker digit in the spare bit mentioned at the end of Section 4. When $t = 1$, all these marker bits will be zero, and also $v' = 0$, v has its original value and $W \equiv 0$.

The first step is to search through the bits of W. For any non-zero bit w of W the "consecutive period" bit of the corresponding subject is examined. If it is 0 then w is removed from W. Otherwise the paper count v of w is examined. If it is zero then w is removed from W. This leaves in W all subjects appearing in period $t - 1$ which must appear in consecutive periods and which have not exhausted their number of papers. They must also be mutually compatible since they appeared together in period $t - 1$. Their incompatibilities are then recorded in a second word W' and the paper counts v and v' of these subjects are reduced and increased by 1, respectively.

The list of subjects is now searched to find the first whose paper count v has not been reduced to zero. If this subject has not already been recorded in either W or W' the various lists of conditions are searched to see if it is permissible to include it in W. (It is at this stage that a condition on the number of students sitting in one period can be applied.) As one of these conditions it must be remembered to examine the "non-consecutive period" bit. If this is 1, then the non-consecutive marker bit must be examined. If this is 1, then it must be set back to 0, and this subject is rejected for the tth period. If it is 0 and the other conditions are also satisfied then it is set to 1. A subject which satisfies all the conditions is now included in W, its incompatibilities are recorded in W', and its paper counts are amended. A search for the next permissible subject is then made, and the process continued until subject N has been examined. The set of permissible subjects in W is now put out, each together with the current value of the paper count v'. W' is now cleared and the papers to be allocated in period $t + 1$ are determined by repeating the operations described in the last two paragraphs.

The process is terminated when all the paper counts v have been reduced to zero.

7. Conclusion

Table 3 shows the First Year General Degree time-table produced from the actual data for June 1963 at Leicester University. In all there were 34 very incompatible subjects with a total of 57 papers. The notation $a - b$ means subject a, paper b. By an inspection of the subject incompatibility list of Table 2 it can be seen that at least fourteen periods are required to accommodate

Table 3.

A complete time-table

Period 1	9 — 1	2 — 1	14 — 1			
Period 2	9 — 2	3 — 1	14 — 2			
Period 3	29 — 1	14 — 3	5 — 1	33 — 1		
Period 4	30 — 1	23 — 1	25 — 1	13 — 1	17 — 1	34 — 1
Period 5	30 — 2	23 — 2	25 — 2	13 — 2	17 — 2	
Period 6	1 — 1	7 — 1	24 — 1	31 — 1		
Period 7	15 — 1	8 — 1	19 — 1	6 — 1	20 — 1	22 — 1
Period 8	16 — 1	28 — 1	19 — 2	10 — 1	20 — 2	22 — 2
Period 9	18 — 1	7 — 2	24 — 2	11 — 1	31 — 2	
Period 10	19 — 3	11 — 2	25 — 3	32 — 1		
Period 11	26 — 1	23 — 3	4 — 1	32 — 2		
Period 12	27 — 1	24 — 3	12 — 1			
Period 13	27 — 2	21 — 1				
Period 14	27 — 3	21 — 2				

the mutually incompatible subjects. Since the solution obtained takes exactly fourteen periods it is the best possible in this case.

If a subject whose papers must not be in consecutive periods appears at the end of the table, this may lead to an excessive number of periods used. As suggested above this can be rectified by introducing some spurious subject numbers and making the offending subject incompatible with them. By this means it will be given a higher priority in the "grand sort" but will not affect the other conditions adversely.

The time taken for the above illustration on an 803 computer with an addition time of 576 μsec, was about $12\frac{1}{2}$ minutes. If all 340 subjects are used the program will take several hours, depending in a complex way upon the number of conditions imposed. If no conditions of types (ii) or (iii) of Section 2 are imposed the times are considerably reduced. For example, with 340 subjects with an average of $1\frac{1}{2}$ papers per subject, and with no imposed conditions of the type "paper *A* must not precede paper *B*," the time taken was about ninety minutes.

References

APPLEBY, J. S., BLAKE, D. V., and NEWMAN, E. A. (1961). "Techniques for producing School Timetables on a Computer and their Application to other Scheduling Problems," *The Computer Journal*, Vol. 3, p. 237.

CSIMA, J., and GOTLIEB, C. C. (1963). "A Computer Method for Constructing School Timetables." Presented at the Eighteenth Annual Conference of the Association for Computing Machinery, Denver, Colorado.

SHERMAN, G. R. (1963). "A Combinatorial Problem Arising from Scheduling of University Classes," *Journal of the Tennessee Academy of Science*, Vol. 38, No. 3, p. 115.

Note on a machine algorithm for conversion from reflected binary to natural binary

Frequently in special-purpose machines an input is provided in reflected binary (or Gray) code, but arithmetic in the processor is done in natural binary. Reflected binary arithmetic is too difficult to implement and cannot be done so economically at the present time. The problem is to convert from reflected to natural binary in the optimum fashion.

Call the reflected code register G and the natural binary register B, both of n bits. The bits of each are subscripted from least to greatest significance thus:

$$G = g_n g_{n-1} \ldots g_2 g_1 \tag{1}$$

$$B = b_n b_{n-1} \ldots b_2 b_1 \tag{2}$$

The conventional solution is to attach a logical converter between G and B. When actuated, this converter produces an output b_j representing the jth natural bit in terms of the jth, $(j+1)$th, ... nth reflected bit. This relation is succinctly expressed if we permit logic and arithmetic to be mixed,

$$b_j = \overline{\sum_{i=j+1}^{n} g_i \,(\mathrm{mod}\ 2)}\, g_j \vee \sum_{i=j+1}^{n} g_i \,(\mathrm{mod}\ 2) \bar{g}_j. \tag{3}$$

The summation is arithmetic and requires that all 1s between $j+1$ and n be added together. The expression "mod 2" requires that if this sum is odd, 1 is substituted for it; if even, 0 is substituted. The overbar means that 0 and 1 are interchanged beneath it.

Although the equation (3) has been simplified, the logic to do it has not. The logic to form a sum mod 2 is not complicated, but it is extensive, as those familiar with parity checkers will testify.

A simple relation exists between *this* natural bit, *this* reflected bit and the *previous* natural bit,

$$b_j = \overline{b_{j+1}} g_j \vee b_{j+1} \bar{g}_j. \tag{4}$$

Conversion is performed from left to right, more-significant bits being converted first. If the *last* bit g_{j+1} converted into a 0(b_{j+1} is 0), then *this* natural bit b_j is the same as *this* reflected bit g_j; if b_{j+1} was found to be 1, then g_j is reversed to get the proper value for b_j.

With (4), a simple command can be incorporated into the processor for the conversion. A counter designated i keeps track of the number of bits so far converted. This algorithm, particularly suitable to serial machines, is micro-programmed below:

(5) $\quad 0 \rightarrow B$

(6) $\quad 0 \rightarrow i$

(7) $\quad \bar{b}_1 g_n \vee b_1 \bar{g}_n \rightarrow b_n$

(8) $\quad B \overset{1}{\leftarrow} (B)$

(9) $\quad G \overset{1}{\leftarrow} (G)$

(10) $\quad i + 1 \rightarrow i$

(11) $\quad i : n < \Rightarrow (7); = \Rightarrow \mathrm{end}$

To begin, the natural binary register is cleared (5) and the counter set to 0 (6). The left-hand (most-significant) bit of G and B is processed. The natural bit, b_n, is set according to (7). A circular shift of one place for G and B is done (8), (9), which moves each bit one position left; the *most*-significant bit, b_n or g_n, is moved to the *least* significant position, b_1 or g_1. The counter is incremented (10) and tested (11) to see if all bits have been converted. If not, the next bit is processed by entering the microprogram at (7).

Since B is cleared in (5), b_1 is 0 the first time (7) is performed, so that b_n is set to g_n. Other times (7) is done, b_n is set to g_n if the last natural bit (b_1) was 0 and to $\bar{g}_n$ otherwise.

IVAN FLORES.

Computer Design Consultant,
Norwalk, Connecticut, U.S.A.