A University faculty timetable

By Mary Almond*

This paper describes a modification of the author's simple algorithm for producing a University faculty timetable.

(Received September 1968)

The timetabling problems at a large University may be classified into various categories. Very often the students select their courses from several different departments for the first year or two so that the basic problem is to plan a co-ordinated timetable for the whole faculty allowing the maximum freedom of permutation of courses. This is a large task involving several hundred courses. However, the timetable will not change very much from year to year and is virtually independent of staff changes. Hence the availability of lecturers need not be considered directly. A faculty timetable is discussed in this paper and the algorithm given earlier (Almond, 1966) is made more general and storage requirements reduced.

The second task is for each department to superimpose lectures for older students on these commitments for earlier years. In this case the lecturer's time is significant as his days are becoming fuller and it is necessary to think of assignments in a three-dimensional array (Almond, 1966, Yule, 1968).

A third problem might be allocation of lecture rooms if this were treated independently.

Fourthly on registration day the computer could be used to check that each student selects compatible courses, to ensure that classes and laboratories do not get overfull and to print out an individual timetable for each student.

Finally an examination timetable could be produced using the individual student timetables to draw up a conflict matrix (Wood, 1968).

Production of a faculty timetable

Data

To produce this timetable it is necessary to know the number and size of lecture theatres, if these are likely to prove a restriction, and in addition certain information must be collected from each department involved. The procedure adopted is to ask departments to complete a form for each of their courses. The following details are required:

1. Title of course.
2. Number of students.
3. Whether or not the course can be run in sections, e.g. it is often very convenient if laboratory periods can be duplicated.
4. Number of single periods per week.
5. Number and duration of multiple periods.
6. Times of courses which must be fixed, perhaps to suit the need of another faculty.
7. Preferred times or days in other cases.
8. Other courses within the department which must not conflict with this course owing to the non-availability of students, lecturers or rooms.
9. Courses in other departments which must not conflict.
10. Additional courses in other departments which if possible should not conflict. These should be listed in order of decreasing importance.
11. Any other special requirements.

The information supplied in answer to the first eight questions can be immediately used as data for the program. Unfortunately the requests for non-conflicting courses are often completely unrealistic and must be vetted by someone who understands the overall need of the faculty. In order to check the requests a conflict matrix was drawn (i.e. a symmetric Boolean matrix of dimensions 'number of courses' × 'number of courses' showing which courses should not conflict). The conflict data could be stored in the computer as a matrix, but it is more economical to store for each course a list of all courses with which it must not conflict. Furthermore, if the data is presented in this way then the position in the list can indicate the priority of the non-conflict.

The main part of the data consisted of a table of the information relevant to each course printed in a readable manner. A typical section is shown in **Table 1.** The courses are listed in some convenient order and will appear in the same order in the results. Fixed courses can be anywhere in the list and are recognised by the fact that they have no lectures or laboratories to be allocated. This makes it easy to fix or unfix a course at any stage.

Fixed or preferred times are indicated by the initial letter of the day, with R for Thursday to avoid confusion with Tuesday, and the hours numbered 1 to 7 or 1 to 8, e.g. W3 means the third lecture hour (probably 11.30 a.m.) on a Wednesday. A zero in the time column means that no particular time is preferred. If necessary extra data can indicate times to be avoided.

In the no-conflict list courses are referred to by numbers and the termination of the list is given by a figure in brackets saying how many of these non-conflicts (numbered from the left-hand side) are essential. The rest are desirable but can be relaxed if necessary.

Algorithms

The timetable is a matrix of 'courses' × 'time' and the first task is to clear this area and enter the times of the fixed courses.

* *Department of Computer Science, The University, Manchester 13*

The algorithm then lists the remaining courses in some suitable order for allocation. To do this a weight factor is associated with each course and the courses are then sorted in order of decreasing weight factor. Possible items to include in the weight factor are the number of essential non-conflicting courses, the number of students and the number of lecture and laboratory hours. These items can be combined in various ways to produce suitable weight factors and the different orderings of the courses for allocation will result in alternative versions of the timetable.

The allocation procedure passes twice through the list of courses searching for suitable times first for the multiple period courses and secondly for the single period courses. Let us consider a course some way down in the list which requires say two single periods. To find a suitable time for the first period the algorithm takes the first preferred time and looks through the list of non-conflicting courses to check if any are already inserted at this time. If this is the case then the second preferred time is tried in the same way. Let us suppose that this is successful. Next a time must be found for the second period. The first preferred time will not be tried and the second will be found occupied so assuming there are no more preferred times, the algorithm passes on to test other times of the week in some specified order. If Monday lectures are not popular then days can be tried in the order Tuesday, Thursday, Wednesday, Monday, Friday perhaps, and if the 9.30 a.m. lectures are not popular then the second period can be tried first and so on. Obviously a three-hour laboratory period must not start at 4.0 p.m. or just before lunch, so only certain starting times are acceptable. Lists of possible starting times in order of preference are included in the data for single periods and multiple periods of various lengths. In this way these times can be varied to produce alternative timetables. The algorithm searches through the times in the given order until one is found satisfying all desired conditions such as room restrictions, no-conflict restrictions and restrictions to spread the course evenly over the week.

If no suitable time can be found the desirable but not essential no-conflict restrictions can be relaxed one-by-one beginning with the least important. After a restriction is relaxed then the times of the week are again tried in order, in the hope that a suitable one will be found. Essential no-conflict restrictions are not relaxed but an omission message is printed and the program passes on to the next period. A note is kept of the number of omissions and neglected no-conflict restrictions for each course and these numbers can be introduced into the weight factor for a second attempt. The courses are relisted in some new order giving priority to those which proved difficult and a second timetable is produced. If it is assumed that the optimum timetable is one giving students the best possible selection of combinations of

Table 1. A section of input data

A	B	C	D	E	F	G	H	I
76	SC11	100	2	0	0	T1 R1	5, 6, 7, 8, 9, 16, 18, 19, 20, 21, 23, 54, 62, 63, 86	(14)
77	CA10	20	2	0	0	0	36, 37, 39, 40, 41, 42, 43, 44, 54, 55, 56, 57, 58, 64, 84, 85, 88, 89, 90, 91	(20)
78	PA10	10	2	0	0	M2 F2	82, 83, 84, 85, 87, 90, 91	(7)
79	HS70	6	1	0	0	T1	30, 31, 32, 33, 34, 35, 45, 46, 80, 86, 99, 100	(10)
80	HS71	6	1	0	0	R1	30, 31, 32, 33, 34, 35, 45, 79, 86, 99, 100	(10)
81	CE40	30	2	0	0	0	37, 44, 82, 84, 85	(5)
82	CE41	30	1	0	0	0	37, 44, 78, 81, 84, 85	(6)
83	GY10	22	2	0	0	0	78, 88, 89, 92	(4)
84	EY70	12	0	0		T1 R1		
85	EY81	12	0	0		M2		
86	FR60	66	0	0		M6 F5 T1 T5 T6 T7		
87	GL10	9	0	0		T1 T5 T6 T7 R3 F5 F6 F7		
88	GK51	15	2	0	0	M2 F2	83, 84, 85, 89, 90, 91, 10, 11, 13, 14, 41, 77	(8)
89	GK52	15	2	0	0	M5 W2	83, 84, 85, 89, 90, 91, 10, 11, 13, 14, 41, 77	(8)
90	AY10	30	2	0	0	T1 R1	42, 77, 78, 88, 89, 91, 92	(7)
91	AY11	30	2	0	0	M1 F1	42, 77, 78, 88, 89, 90, 92	(7)
92	LN50	22	4	0	0	M1 M5 W1 R5	10, 11, 13, 14, 16, 40, 42, 83, 84, 85, 88, 89, 90, 91	(14)
93	MP01	100	0	0		W1 W2 F1		
94	MU50	30	0	1	2	F1	44, 45, 46, 86, 95, 96	(6)
95	MU51	30	0	1	2	M2	44, 45, 46, 86, 94, 96	(6)

A. Course number.
B. Course title.
C. Number of students.
D. Number of single periods.
E. Number of multiple periods.
F. Length of multiple periods.
G. Times.
H. Courses which must not conflict.
I. Number of essential non-conflicts.

courses then this technique will improve the solution.

The actual best solution is rather subjective and not easy to define for the computer hence in practice several alternative versions of the timetable can be produced and the final choice left to the human beings involved.

Results

A section of the printed timetable is shown in **Table 2.** This layout seems convenient when a very large number of courses is involved.

In addition to the timetable it seems useful to print out for each course a complete list of all courses which do not conflict. These lists will include as many as possible of the no-conflict courses and several other besides which may or may not provide suitable combinations for the student.

This algorithm is being tried in Arts or Science Faculties at three different Universities. The timetable produced for Liverpool Arts Faculty involved over a hundred courses given to first-year students. The storage required for the data arrays was about 7000 words (i.e. 70 × number of courses). The program was written in Atlas Autocode and the execution time on Atlas was about 24 seconds or 0·24 seconds per course. For a science faculty having more laboratory classes the time was about 0·3 seconds per course.

Acknowledgements

The author wishes to acknowledge the assistance of many people in various registrar's departments especially F. T. Mattison of the Liverpool University Arts Faculty.

Table 2. A section of the results

	MONDAY							TUESDAY							WEDNESDAY							THURSDAY							FRIDAY						
	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7
SC11		*																				*													
CA10													*																	*					
PA10																*														*					
HS70		*																																	
HS71																						*													
CE40									*																					*					
CE41																							*												
GY10		*																					*												
EL70								*														*													
EL81		*																																	
FR60						*		*				*	*	*																				*	
GL10								*				*	*	*										*										*	*	*
GK51																	*																		*	
GK52					*																			*											
AY10								*														*													
AY11	*																												*						
LN50						*									*											*					*				
MP01															*	*													*						
MU50																													*	*					
MU51									*	*																									

References

ALMOND, M. (1966). An Algorithm for Constructing University Timetables, *The Computer Journal*, Vol. 8, p. 331.
WOOD, D. C. (1968). A system for computing University Examination Timetables, *The Computer Journal*, Vol. 11, p. 41.
YULE, M. P. (1968). Extensions to the Heuristic Algorithm for University Timetables, *The Computer Journal*, Vol. 10, p. 360.