An integer linear programming model of a school timetabling problem

By N. L. Lawrie*

Earlier papers have defined the input requirements of a program for timetabling in terms of a list of lists of items, each item being a teacher, a class or set, a classroom or a piece of equipment. This paper describes an approach based on larger items of departments, group of pupils (generally year groups), and layouts. The problem is given an integer linear programming formulation, and computational methods used in obtaining solutions are discussed.

(Received March 1969)

There have been a number of approaches made in the last few years to the problem of constructing timetables for schools in Britain. Barraclough's work (Barraclough, 1965) has been developed and extended by Cox (1969), and, at the recent IFIP Conference in August 1968, Johnston and Wolfenden described a new approach. The Local Government Operational Research Unit (1967) has made a survey of programs and methods and has carried out field trials using a recently developed procedure due to Clementson (1968).

There has been general agreement among the authors mentioned above about the computer input requirements of a program for timetabling, and these requirements are well illustrated by Johnston and Wolfenden (1968). In essence, they form a list in which each entry specifies a list of items which must be available simultaneously for a specified number of periods in the week. An item may be a teacher, a class or set, classroom or piece of equipment, and a simply entry in the list of requirements would specify a teacher, a group of pupils, a room, and a number of periods.

The author (Lawrie, 1968) has described an approach to school timetabling which is based not on units of the teacher, the class or set, and the *event*, defined by Lions (1967) as '*either* a teacher and a class meeting for one period, *or* a teacher having a spare period *or* a class having a spare period', but on larger units of *departments, groups* of pupils (generally year groups), and *layouts.*

The aim of this paper is to describe the mathematical formulation given to the timetabling problem as it is posed here, and the computational methods used in obtaining solutions. No justification of the expression of the requirements in the form of layouts is given. This has been attempted in the paper by the author already referred to. In the next section, however, layouts are briefly described and illustrated, and some of the advantages and disadvantages of using them are mentioned.

Data requirements

Layouts

The data relating to curriculum of pupils and their organisation into classes and sets is expressed in a number of *layouts*, generally between 4 and 6 in total. The idea of a layout and its use are discussed by Lewis (1961), and the layouts developed for use with this method of timetabling are a development of those in Lewis's book. In essence, the layout is a statement of the curriculum and its organisation for a group of pupils, generally a year group, e.g. all the pupils in the first or other year of the school. The example of **Table 1** is a layout for the 3rd year of a comprehensive school with 1060 pupils. There are 260 pupils in the 3rd year.

Column A of the layout specifies a requirement for 5 classes or sets in English, 2 in Latin and 4 in Commerce for 6 periods in the week; column B specifies a requirement for 4 classes or sets in English, 1 in Homecraft, 5 in Technical subjects, 1 in Dress and Design, and 1 in 'X' which stands for a number of miscellaneous, minority time, subjects; the entries in the remaining columns may be interpreted similarly. On its own, the layout does not make clear which pupils will go to which classes or sets at any time, although this information will be available to the timetabler who has drawn up the layout, and may be set out beneath the layout, as illustrated in Table 1. It does not specify which teachers of English, Latin and Commerce will be required in column A or which rooms, although this data could be added beneath some or all of the columns and would not affect the formulation of the problem or require any alteration in principle to the computational method.

The advantages to the headmaster of expressing his requirements in the form of layouts have been discussed fully by the author (Lawrie, 1968). Two points will be made here.

1. A layout simplifies the planning of curriculum and organisation for a group of pupils by providing a compact notation which emphasises the staff requirements by department of the plan and omits explicit reference to classes or to how pupils are regrouped into sets from period to period. The notation makes it relatively straightforward to treat a year group as a unit for purposes of curriculum planning even in large schools, and should therefore help to avoid the segregation of pupils into separate streams which occurs in some schools simply because of the difficulty of planning for a large year group as a whole when the unit in which planning is done is the class.
2. A layout enables a certain amount of detail to be suppressed in planning. What is suppressed may vary from layout to layout. In Table 1, classes or sets in a notional department 'X' appear in several

**Department of Operational Research, University of Strathclyde*

columns, and the subjects which are grouped together and labelled 'X' in this instance are the non-examinable subjects occupying minority time:

Art (or Art Appreciation)
Music Appreciation or Class Singing
Physical Education
Religious Education.

It may be thought a disadvantage that the timetabler has to produce a layout for the year group as a whole since this transfers to him the burden (which apparently might be carried by the computer) of determining not merely curriculum and setting but also the overlap of sets (or classes) in different subjects over the week. If a year group is divided into 'streams' or 'sides' which do not mix with one another for much of the week, the layout *is* an additional burden and its specification fixes the simultaneous occurrence of unrelated classes which could well be left undetermined and at the disposal of a computer program. On the other hand, if the year group is not divided into 'streams' or 'sides' but is, like the year group in Table 1, set over different sub groups of pupils in different columns of the layout, the layout becomes necessary in order to specify the educational policy of the school.

Table 2 illustrates five layouts which describe the curricula and organisation of a six-year comprehensive school, labelled school B in Table 5. Notice that plans for years 5 and 6 are run together in one layout and that the layout of Table 1 is identical to the 3rd year layout in Table 2. Much of the detail in years 1 and 2 has been suppressed since most teaching in these years is to mixed ability classes and could have been specified in the layout in a wide variety of ways. Rather than do this it was felt better to defer decisions about the overlapping of such classes in the timetable to a later stage.

What have been specified in the 1st and 2nd year layouts are:

1. the way in which practical classes are organised and, in particular, how two of the periods of Technical/Homecraft are used by two language pupils for Latin;

Table 1

Layout for a 3rd year of 260 pupils

Column:	A	B	C	D	E	F	G	H	I	J
Periods:	*6*	*6*	*4*	*2*	*7*	*3*	*4*	*4*	*2*	*2*
Sets:	5E	4E	4F	6F	6M	4M	3Hi	3Ch	2Bi	2Bi
	2Lt	1H	1H	5X	4X	6X	2G	3Ph	2Ph	2Ph
	4C	5T	5T				2EO	2C	2C	2C
		1DD	2X				4GSc	1A	1A	1A
		1X	1C					1Mu	1Mu	1Mu
								1Ge	1Ge	1Ge
								1Sp	1Sp	1Sp
								3MS	3X	1H
								1Hi		4T
										2X

Examples of pupil curricula

	A	B	C	D	E	F	G	H	I	J
Non academic:										
Building	E	T	T	X	M	X	GSc	MS	X	T
Commerce	C	E	C	F	X	M	GSc	MS	X	X
Academic:										
Sc/Ge/Sp	Lt	E	F	F	M/X	X/M	Hi/G		Sc/Ge/Sp	
Commerce	C	E	F	F	M/X	X/M	Hi/G	—	Commerce	—
Home/Tech.	E	H/T	H/T	F	M/X	X/M	Hi/G	Ph/Ch	X	H/T

Abbreviations

E:	English	M:	Mathematics	Bi:	Biology
Lt:	Latin	Hi:	History	A:	Art
C:	Commerce	G:	Geography	Mu:	Music
H:	Homecraft	EO:	Economic Organisation	Ge:	German
T:	Technical	GSc:	General Science	Sp:	Spanish
DD:	Dress and Design	Ch:	Chemistry	MS:	Modern Studies
F:	French	Ph:	Physics		

X: Miscellaneous ('minority time') subjects consisting of Art and Music Appreciation, Physical and Religious Education.

2. the way in which sets in English and Mathematics are organised. There are enough staff in the school to make possible the simplest arrangement—the simultaneous occurrence of all sets in English or Mathematics in each year.

What goes unspecified is the organisation of the rest of the curriculum which consists of French, History, Geography, Latin (only two of five Latin periods is specified in either layout), as well as the subjects classed as 'X'. This larger group of subjects is classed as 'Y' in the layouts.

Table 3 shows the data of Table 2 prepared for computer input. Each column of each layout gives rise to a line in Table 3 (in effect Johnston and Wolfenden's

Table 2

Layouts for school B (a 6-year school of 1,060 pupils)

1st year layout (150 pupils)

Column:	A	B	C	D	E
Periods:	*6*	*6*	*6*	*2*	*20*
Sets:	5E	5M	4Sc	1H	5Y
			2H	1T	
			2T	4Sc	
				2Lt	

2nd year layout (180 pupils)

Column:	A	B	C	D	E	F
Periods:	*6*	*6*	*4*	*2*	*2*	*20*
Sets:	6E	8M	6Sc	3T	2T	5Y
			2H	3H	1H	
			2T	3Sc	3Sc	
					2Lt	

3rd year layout (260 pupils)

Column:	A	B	C	D	E	F	G	H	I	J
Periods:	*6*	*6*	*4*	*2*	*7*	*3*	*4*	*4*	*2*	*2*
Sets:	5E	4E	4F	6F	6M	4M	3Hi	3Ch	2Bi	2Bi
	2Lt	1H	1H	5X	4X	6X	2G	3Ph	2Ph	2Ph
	4C	5T	5T				2EO	2C	2C	2C
		1DD	2X				4GSc	1A	1A	1A
		1X	1C					1Mu	1Mu	1Mu
								1Ge	1Ge	1Ge
								1Sp	1Sp	1Sp
								3MS	3X	1H
								1Hi		4T
										2X

4th year layout (260 pupils)

Column:	A	B	C	D	E	F	G	H
Periods:	*6*	*6*	*6*	*7*	*3*	*4*	*4*	*4*
Sets:	4E	5E	6F	7M	2M	4Hi	4Ph	6Ch
	3Lt	1H	1H	2X	7X	4G	2Bi	1X
	3C	2T	2T			1X	2Ge	2Ge
		3X	2X				1Gk	1Gk
							1Mu	1Mu
							1A	1A
							3C	3C

5th and 6th year layout (210 pupils)

Column:	A	B	C	D	E	F	G	H
Periods:	*6*	*4*	*7*	*4*	*4*	*6*	*6*	*3*
Sets:	7E	2Hi	7M	4Ph	4Ch	2Lt	6F	7X
		2G	1X	2Bi	2Ge	3T	3T	
		1MS		2Ge	2A	2H	2H	
		3X		2A	2Mu	1C	1X	
				2Mu	1C	3X		
				1C	1X			
				1X				

requirement list) which specifies a number of periods and the staff required by department for these periods. At the foot of Table 3, the number of teaching staff in each department is given.

Notice that there are apparent discrepancies between columns X and Y of Table 3 and the sets in 'X' and 'Y' specified in the layouts. For example, column I of the 3rd year layout in Table 2 specifies 3 classes or sets in 'X' and yet line 20 (the corresponding line) of Table 3 specifies 5 teachers required in department X. The explanation is that teachers available to teach the notional subject 'X' come from the departments of Music, Art, Physical Education and Religious Education, and that where some of the subjects taught by these departments occur explicitly in the layout, as they do in columns H, I and J of the 3rd year layout in Table 2, the number of staff required to teach these subjects is deducted from the number available to teach in department 'X' (or, equivalently, added to the number of sets in 'X' already specified). The effect of the single classes in each of Music and Art in the columns referred to is included in the appropriate lines of Table 3 (lines 19, 20 and 21) by adding 2 to the staff requirement in department 'X' for each line. Apparent discrepancies in column Y have the same explanation.

Notice too that the layouts in Tables 1 and 2 are expressed in terms of the *subjects taught*, whereas Table 3 is expressed in terms of *departmental requirements*. The link between these two is given in the following paragraphs.

Table 3
Staff requirements for each line of each layout

LINE NO.	PERIODS	E	M	S	T	H	C	L	F	Z	U	A	X	Y
1	6	5	–	–	–	–	–	–	–	–	–	–	–	–
2	6	–	5	–	–	–	–	–	–	–	–	–	–	–
3	6	–	–	4	2	2	–	–	–	–	–	–	–	–
4	2	–	–	4	1	1	–	2	–	–	–	–	–	2
5	20	–	–	–	–	–	–	–	–	–	–	–	–	5
6	6	6	–	–	–	–	–	–	–	–	–	–	–	–
7	6	–	8	–	–	–	–	–	–	–	–	–	–	–
8	4	–	–	6	2	2	–	–	–	–	–	–	–	–
9	2	–	–	3	3	3	–	–	–	–	–	–	–	–
10	2	–	–	3	2	1	–	2	–	–	–	–	–	2
11	20	–	–	–	–	–	–	–	–	–	–	–	–	5
12	6	5	–	–	–	–	4	2	–	–	–	–	–	2
13	6	4	–	–	5	2	–	–	–	–	–	–	1	1
14	4	–	–	–	5	1	1	–	4	–	–	–	2	6
15	2	–	–	–	–	–	–	–	6	–	–	–	5	11
16	7	–	6	–	–	–	–	–	–	–	–	–	4	4
17	3	–	4	–	–	–	–	–	–	–	–	–	6	6
18	4	–	–	4	–	–	2	–	–	5	–	–	–	5
19	4	–	–	6	–	–	2	–	2	4	1	1	2	8
20	2	–	–	4	–	–	2	–	2	–	1	1	5	7
21	2	–	–	4	4	1	2	–	2	–	1	1	4	6
22	6	4	–	–	–	–	3	3	–	–	–	–	–	3
23	6	5	–	–	2	1	–	–	–	–	–	–	3	3
24	6	–	–	–	2	1	–	–	6	–	–	–	2	8
25	7	–	7	–	–	–	–	–	–	–	–	–	2	2
26	3	–	2	–	–	–	–	–	–	–	–	–	7	7
27	4	–	–	–	–	–	–	–	–	8	–	–	1	9
28	4	–	–	6	–	–	3	1	2	–	1	1	2	5
29	4	–	–	6	–	–	3	1	2	–	1	1	3	6
30	6	7	–	–	–	–	–	–	–	–	–	–	–	–
31	4	–	–	–	–	–	–	–	–	5	–	–	3	8
32	7	–	7	–	–	–	–	–	–	–	–	–	1	1
33	4	–	–	6	–	–	1	–	2	–	2	2	5	7
34	4	–	–	4	–	–	1	–	2	–	2	2	5	7
35	6	–	–	–	3	2	1	2	–	–	–	–	3	5
36	6	–	–	–	3	2	–	–	6	–	–	–	1	7
37	3	–	–	–	–	–	–	–	–	–	–	–	7	7
Staff Numbers (by Department)		9	9	9	6	4	4	3	9	8	3	4	12	32

Other data requirements

The data still requiring to be specified after layouts have been determined consists of:

1. figures of staffing by department;
2. subjects taught by actual and notional departments;
3. agreed codes to be used in presenting input and in printing output.

In school B, staffing and subjects taught by departments were as follows:

DEPARTMENT	NUMBER OF STAFF	SUBJECTS TAUGHT
English	9	English
Mathematics	9	Mathematics
Science	9	General Science, Physics, Chemistry, Biology
Technical	6	Technical
Homecraft	4	Homecraft, Dress and Design
Commerce	4	Commerce, Economic Organisation
Classics	3	Latin, Greek, Y
Modern Languages	9	French, German, Spanish, Y
Modern Studies	8	Modern Studies, History, Geography, Y
Music	3	Music, X, Y
Art	4	Art, X, Y
Physical Education	4	PE, X, Y
Religious Education	1	RE, X, Y

Notice that the number of teachers available to teach the notional subject 'X' is 12, and the number available to teach the notional subject 'Y' is 32.

The following single letter codes are used to label departments in Table 3 and in **Table 4** which is an 'outline timetable' (defined in the next section) for school B, based on the layouts of Table 2.

E: English	H: Homecraft	Z: Modern Studies
M: Mathematics	C: Commerce	U: Music
S: Science	L: Classics	A: Art
T: Technical	F: Modern Languages	X: department 'X'
		Y: department 'Y'.

Mathematical formulation

Once layouts have been obtained, the problem of constructing the timetable is dealt with in two stages.

Table 4

An outline timetable for the requirements of Table 3

PERIODS	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Year 1	5Y	5M	5M	5Y	5Y	4S 2T 2H	4S 2T 2H	5Y	5M	5Y	5Y	5Y	5Y	5M	5Y	5Y	5Y	5E	5Y	5Y
Year 2	3S 2T 1H 2L	5Y	5Y	8M	8M	5Y	8M	6S 2T 2H	3S 3T 3H	5Y	5Y	5Y	5Y	6E	6E	5Y	5Y	5Y	6S 2T 2H	6S 2T 2H
Year 3	4S 2C 5Z	4S 2C 5Z	4S 2C 5Z	6S 2C 2F 4Z 1U 1A	6S 2C 2F 4Z 1U 1A	4S 2C 2F 1U 1A 3X	4S 2C 5Z	5E 4C 2L	6S 2C 2F 4Z 1U 1A	6M 4X	6M 4X	6M 4X	6M 4X	5T 1H 1C 4F 2X	6M 4X	4S 4T 1H 2C 2F 1U 1A 2X	4S 4T 1H 2C 2F 1U 1A 2X	4E 5T 2H 1X	6M 4X	6M 4X
Year 4	7M 2X	5E 2T 1H 3X	5E 2T 1H 3X	2T 1H 6F 2X	2T 1H 6F 2X	5E 2T 1H 3X	2T 1H 6F 2X	2M 7X	5E 2T 1H 3X	6S 3C 1L 2F 1U 1A	6S 3C 1L 2F 1U 1A	6S 3C 1L 2F 1U 1A	6S 3C 1L 2F 1U 1A	8Z 1X	8Z 1X	2T 1H 6F 2X	2T 1H 6F 2X	6S 3C 1L 2F 1U 1A 1X	4E 3C 3L	4E 3C 3L
Years 5 and 6	3T 2H 6F 1X	4S 1C 2F 2U 2A 1X	4S 1C 2F 2U 2A 1X	3T 2H 1C 2L 3X	3T 2H 1C 2L 3X	7M 1X	7E	7M 1X	7X	7E	7E	7E	7E	6S 1C 2F 2U 2A 1X	3T 2H 1C 2L 3X	7M 1X	7M 1X	7M 1X	3T 2H 6F 1X	3T 2H 6F 1X

1. The overlapping of layouts with one another in such a way that staffing restrictions are not violated during any period of the week. This results in what is called an 'outline timetable' in what follows and is illustrated for school B in Table 4.
2. The permutation of the outline timetable to meet requirements on distribution of classes and sets in various subjects over the week.

So far, only the first of these stages has been programmed, and the mathematical formulation given below is for this stage only.

We define an *arrangement* as a set of columns, one from each layout, which does not violate any staffing restriction. For the five layouts of Table 2, for example, column B of layout 1, column F of layout 2, column G of layout 3, column B of layout 4 and column E of layout 5 can be verified as forming an arrangement since the total staff by department to teach the classes or sets specified in these columns is, department by department, less than the teaching staff available. The problem can now be seen as one of finding as many arrangements, not necessarily all different, as there are periods in the week. It has a Linear Programming formulation analogous to the trim problems discussed, for example, by Gilmore and Gomory (1961). The constraints which must be satisfied by any solution are that each column of each layout must occur the number of times specified for it in its layout.

Formally, we suppose that all possible arrangements are serially numbered, and define for each j an integer valued, nonnegative variable, x_j, as the number of occurrences in the outline timetable of the jth arrangement. There are thus as many variables as arrangements, N say, and as many constraints, m say, as the total number of columns in the layouts of the problem (37 in the case of the problem of Table 2).

We suppose the columns of the layouts numbered consecutively from 1 (column A of layout 1) upwards to m as in Table 3, and define

b_i = the number of occurrences of column i specified in its layout.

$i = 1, 2, \ldots, m.$

a_{ij} = 1 if column i appears in arrangement j
= 0 otherwise.

$i = 1, 2, \ldots, m.$
$j = 1, 2, \ldots, N.$

Table 4 continued

PERIODS	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
Year 1	5E	5Y	5Y	5Y	5E	5E	4S 2T 2H	4S 2T 2H	4S 2T 2H	4S 1T 1H 2L	5E	5Y	5Y	5Y	5M	5E	5M	4S 2T 2H	4S 1T 1H 2L	5Y
Year 2	5Y	6E	6E	6E	8M	6S 2T 2H	5Y	5Y	5Y	5Y	5Y	5Y	5Y	8M	3S 2T 1H 2L	8M	6E	5Y	5Y	3S 3T 3H
Year 3	5T 1H 1C 4F 2X	4M 6X	4M 6X	4M 6X	4E 5T 2H 1X	6F 5X	5E 4C 2L	5E 4C 2L	5E 4C 2L	4E 5T 2H 1X	5T 1H 1C 4F 2X	4E 5T 2H 1X	4E 5T 2H 1X	5E 4C 2L	4S 2C 2F 1U 1A 3X	4E 5T 2H 1X	5T 1H 1C 4F 2X	5E 4C 2L	6F 5X	6S 2C 2F 4Z 1U 1A
Year 4	4E 3C 3L	6S 3C 1L 2F 1U 1A 1X	6S 3C 1L 2F 1U 1A 1X	6S 3C 1L 2F 1U 1A 1X	8Z 1X	4E 3C 3L	7M 2X	7M 2X	7M 2X	7M 2X	7M 2X	4E 3C 3L	4E 3C 3L	2T 1H 6F 2X	2M 7X	8Z 1X	2M 7X	7M 2X	5E 2T 1H 3X	5E 2T 1H 3X
Years 5 and 6	5Z 3X	3T 2H 1C 2L 3X	3T 2H 1C 2L 3X	3T 2H 1C 2L 3X	4S 1C 2F 2U 2A 1X	7M 1X	3T 2H 6F 1X	3T 2H 6F 1X	3T 2H 6F 1X	5Z 3X	5Z 3X	6S 1C 2F 2U 2A 1X	6S 1C 2F 2U 2A 1X	7X	7E	6S 1C 2F 2U 2A 1X	5Z 3X	7X	4S 1C 2F 2U 2A 1X	7M 1X

The constraints of the problem may now be written in the form:

$$\sum_{j=1}^{N} a_{ij}x_j = b_i \qquad i = 1, 2, \ldots, m. \tag{1}$$

The matrix of coefficients (a_{ij}) is by its definition a zero-one matrix and has in general between 85% and 90% zeros. It may be noted that

$$\sum_{i \varepsilon\, layout\, k} a_{ij} = 1$$

since each arrangement contains one and only one column from any layout, and that

$$\sum_{i \varepsilon\, layout\, k} b_i = N_p,$$

the number of periods in the week, since the layout specifies an activity (column) for each period of the week. Hence

$$\sum_{i \varepsilon\, layout\, k} \sum_{j=1}^{N} a_{ij}x_j = \sum_{i \varepsilon\, layout\, k} b_i$$

reduces to

$$\sum_{j=1}^{N} x_j = N_p.$$

Thus the m constraint equations are not independent. If there are K layouts, $K - 1$ of the equations, one from each of any $K - 1$ of the layouts, may be omitted.

Although a set of $K - 1$ equations is omitted in the Linear Programming runs, the explanation of details of the computational method assumes that all m equations are present in order to simplify the exposition.

In typical problems the number of variables is large and greatly in excess of the number of equations. Details of 4 recent problems taken from actual schools are given in Table 5.

An objective function

Most frequently, headmasters specify what they regard as a good timetable in terms of the distributions over the week of classes and sets in the various subjects, sometimes also in terms of the distribution of free periods for staff or for senior pupils, sometimes in terms of the mix of periods each day—not too many or too few periods daily of 'academic' subjects. These imprecisely defined criteria can be seen to be relevant to the second stage of calculation mentioned above, but not to the first stage. At that stage we are interested only in feasible (integer) solutions and hence most recent work has omitted an objective function. It is clear, however, that some outline timetables might lead to better permuted timetables at the second stage than others, and the aim of the numerical procedures described in the next section has been to produce a number of alternative solutions at stage one.

In earlier work the objective function

$$\textit{maximise} \sum_{j=1}^{N} c_j x_j$$

was used where c_j was defined as the minimum of the b_i values of the columns in arrangement j and hence is an upper bound for x_j. This was an attempt to give preference in an outline timetable to arrangements which could occur to a higher multiplicity, and hence to reduce the number of different arrangements occurring at non zero level in a solution. Such an objective function, however, has little relation to the criteria normally quoted by headmasters and has been dropped in favour of producing a number of alternative (integer) solutions.

The computational method

In the first version of the method, all variables were generated prior to the Linear Programming run which used a code, written in ALGOL for the ICL 1905, incorporating Gomory's Method of Integer Forms (Gomory, 1958) and making use of the objective function mentioned above.

In the current version of the method, variables are generated only as required, no objective function is used, and Gomory's MIF, while still incorporated, is supplemented by an *ad hoc* procedure, described below, which first rounds a rational solution to integer values and then completes the partial solution thus obtained using an enumerative procedure. The Method of Integer Forms has been retained along with the *ad hoc* procedure although no longer strictly necessary. It serves, however, as a means of moving from one feasible solution to an essentially different solution and hence produces different starting points for the *ad hoc* procedure. The program of this method is also in ALGOL except for a few procedures, one for double length integer arithmetic, written in PLAN. It has solved successfully a number of problems of which those listed in **Table 5** are typical.

Table 5

Data relating to schools studied

Schools	B	D	G	W
Number of pupils	1,060	700	980	490
Number of staff	74	45	62	31
Pupil : Teacher ratio	14·3	15·6	15·8	15·8
Number of layouts	5	5	3	4
Number of lines	37	45	28	48
Total number of arrangements	1,454	1,459	210	2,363
Teaching periods available	2,960	1,890	2,480	1,395
Teaching periods specified in layouts	1,789	1,282	1,208	905
Percentage of available teacher periods specified	60·4	67·8	48·7	64·9
Computer running times (minutes)	2·7	11·9	1·3	7·0

The inverse matrix of the basis is held in explicit form as a matrix of integers with their common denominator, D, held separately. Variables generated for entry into the basis are numbered in order of their entry and have their coefficients in the constraints of the problem stored in compact form with zeros omitted.

The program consists of a Phase I procedure at the end of which cuts are made, one at a time, unless the solution found at the end of Phase I is in integers. After each cut there is an immediate return to Phase I in order to restore primal feasibility. Cuts are made until either the program terminates with an integer solution as a result of a cut, or sufficient integer solutions have been obtained by use of the *ad hoc* procedure which is called prior to each cut.

Generation of variables

In Phase I the simplex multipliers, $\pi_1, \pi_2, \ldots, \pi_m$ are available at each iteration and the problem of finding the best variable to enter the basis at the next iteration—taken to be the variable pricing out most negative—is a zero-one Integer Linear Programming problem. We define

$$y_i = 1 \text{ if the arrangement selected involves the } i\text{th column}$$
$$= 0 \text{ otherwise.} \qquad i = 1, 2, \ldots, m.$$

The problem then is to

$$\textit{minimise} \sum_{i=1}^{m} \pi_i y_i \tag{2}$$

subject to the constraints

$$\left.\begin{array}{ll} \sum_{i \varepsilon \, layout \, k} y_i = 1 & k = 1, 2, \ldots, K \\ \sum_{i=1}^{m} d_{ir} y_i \leqslant D_r & r = 1, 2, \ldots, S \end{array}\right\} \tag{3}$$

where D_r is the number of staff in the rth department, d_{ir} is the number of staff in the rth department used in column i, and S is the number of departments in the school. D_r and d_{ir} are tabulated for school B in Table 3 ($S = 13$).

The method of solution employed for this sub-problem is one of partial enumeration. The constraints

$$\sum_{i \varepsilon \, layout \, k} y_i = 1 \qquad k = 1, 2, \ldots, K$$

express the condition that only one line from each layout may be selected. Lines in each layout are examined in increasing order of their π_i values and an enumerative scheme set up capable of examining all possible combinations of lines, one from each layout. In the case of the problem of Table 3 where there are 5 lines in the 1st layout, 6 in the 2nd, and 10, 8, 8 lines respectively in the 3rd, 4th and 5th layouts, complete enumeration would involve the examination of $5 \times 6 \times 10 \times 8 \times 8 = 19{,}200$ combinations of y_i values. However, only combinations —arrangements—which satisfy constraints (3) are enumerated. In addition, the value of

$$\sum \pi_i y_i = \pi, \text{ say}$$

is calculated for the first arrangement found, and the constraint

$$\sum \pi_i y_i < \pi$$

is then added to the constraints (3) so that branches with a lower bound $\geqslant \pi$ need not be examined. This constraint is successively sharpened if further arrangements are found with smaller values of π, until the arrangement minimising (2) has been found.

After Phase I has been completed and the first cut has been made, the problem of generating variables to enter the basis is more complex. It has the same form as during Phase I (with additional, nonlinear terms in the objective function) and may be solved in the same way. It has not been programmed and is not discussed further here. In the present version of the method only those variables already generated by the end of Phase I are used during the derivation of cuts and the re-entries to Phase I required.

The *ad hoc* procedure

This procedure may be used following a rounding procedure to obtain one or more solutions in integers from any non negative, non integer basic solution. It has been developed because the use of the Method of Integer Forms on its own did not lead readily to an integer solution. Runs made on the larger problems (e.g. schools B, D and W) were allowed to continue until several cuts had been made and until a considerable time had elapsed without reaching an integer solution. The same experience was repeated when one of these problems was run using two commercially available ILP codes. For these runs all the variables of the problem were generated prior to using the codes.

The usefulness of the procedure depends on the fact that the coefficients of the constraint equations are all non negative and that, if values of certain variables are fixed in such a way that one of the equations is satisfied, the value of all remaining variables in that equation must be zero.

It has one parameter, q, in the control of the user. q is the number of arrangements to be found by the *ad hoc* procedure. Alternatively, since the sum of the basic variables is N_p, the number of periods in the week, the rounding procedure must round basic variables to integer values until the sum of the rounded variables equals $N_p - q$. Typically, q has been set to 4, 5 or 6.

Suppose that the variables of some non integer basic solution are $x_1, x_2, \ldots, x_m$. The rounding procedure rounds down these values to the nearest integer, $int[x_i]$, $i = 1, 2, \ldots, m$, and terminates if $\sum_{j=1}^{N} int\,[x_i] > N_p - q$. Otherwise fractional parts are examined in decreasing order of magnitude and the corresponding variable values increased by 1 until the sum of these rounded values equals $N_p - q$.

Denote by b^* the vector of right-hand sides which is obtained when the rounded variable values are substituted in the basic equations (1) of the problem. b^* is compared with b, the vector of true right-hand sides. If $b_i^* > b_i$ for any i, the partial solution obtained by rounding cannot be completed by adding non negative variables and the *ad hoc* procedure is not entered. If however $b_i^* \leqslant b_i$ for all i, the partial solution may be completable by adding non negative integer variables and the *ad hoc* procedure is entered. Constraints for which $b_i^* = b_i$ may be dropped from consideration as may all variables which have non zero coefficients in such constraints.

The problem remaining is to find integer solutions to the equations

$$\sum_{j=1}^{N} a_{ij}x_j = b_i - b_i^*.$$

The number of constraints with non zero right-hand sides is generally about half the number in the original problem (for $q = 4, 5, 6$); the number of variables which can occur at non zero level is greatly reduced; and instead of requiring to find a solution involving N_p arrangements (or variables) only q arrangements have to be found.

This problem can be tackled readily using an enumerative or branching procedure similar in its logic to the procedure for generating arrangements. This is the *ad hoc* procedure which forms part of the present program. It is worth noting that in generating arrangements the variables branched on are the y_i. In the *ad hoc* procedure the variables branched on are the x_j, i.e. arrangements themselves. It should be noted too that there are no bounds to calculate in the *ad hoc* procedure since there is no objective function associated with the constraints. A particular branch may only be ignored if it can be shown there are no feasible solutions along it. As in the main program x_j variables are generated as required.

Results

During the summer of 1968, timetabling problems were examined in detail in four six year, comprehensive schools listed in Table 5. Layouts were prepared in conjunction with the headmasters of the schools (one set of layouts, for school B, is shown in Table 2), and one or more outline timetables was constructed for each school using the program already described.

In two cases, the layouts prepared in the first instance gave rise to the result 'no feasible solutions' at the end of Phase I of the program, and this led in one case (school D) to alterations and corrections to the layouts, and in the other (school W), where an attempt was being made to devise a curriculum and form of organisation for the school using substantially fewer teachers than at present, to small increases in the number of staff specified as being available. Layouts were prepared as shown below.

YEAR \ SCHOOL	B	D	G	W
1	L1	L1	omitted	L1
2	L2	L2	L1	L2
3	L3	L3	L2	L3
4	L4	L4	L3	L4
5	L5	L5	L3	omitted
6	L5	omitted	L3	omitted

The reasons for omitting certain years from the study were partly educational and partly to simplify the subsequent computing problem. For example, in school D, only 40 pupils were involved in year 6 and it was felt that their requirements could best be met after an outline timetable had been formed. In school G, the first year is taught largely in mixed ability groups which it was felt could be fitted in in a variety of ways, against a specified outline timetable. The same reason (teaching in mixed ability groups) led to the simplification of layouts for years 1 and 2 in school B.

It is worth noting that the size of computing problem which arises in finding an outline timetable for a school depends more on the complexity of curriculum and organisation in the school than on its size. This shows clearly in Table 5 where the smallest school, W, requires more lines to express its requirements in years 1–4 than does school B for its entire 6 years. Running times in Table 5 exclude times spent in the *ad hoc* procedure.

All the schools studied had a relatively low pupil teacher ratio but were not simpler to deal with for this reason, since schools with a good staffing position tend to commit their staff to as much teaching (to smaller classes) as schools which are less well staffed.

The *ad hoc* procedure has been used extensively on a limited number of problems and the results of certain tests are tabulated in **Table 6**. Machine running times are very variable, but half of the 42 completed runs took 10 seconds or less. Runs were terminated after 10 solutions had been found, after all solutions had been found if the number of solutions was less than 10, or after it had been shown that no solutions existed.

Some running times were long, but, since most entries to the procedure produced several solutions quickly, there seems a case for setting an upper limit, say 20 seconds, to the running time of the procedure and returning to the main program after that time if no solutions have been obtained. Running times decrease as q decreases but the number of solutions decreases also. A good value for q would seem to be 3 or 4 if the rounding procedure were modified (as it could be readily) to ensure that it invariably produced a partial solution for which $b^* \leqslant b$.

Table 6

Results of using the *ad hoc* procedure

RUNNING TIME (t)	$t \leqslant 10$ SECS	$10 < t \leqslant 20$ SECS	$t > 20$ SECS	UNKNOWN
NO OF CASES	22	9	7	4
$q = 3$	3*	1	–	–
4	4	5*	–	1
5	6	–	3	–
6	5	1	2	2
7	4	2	1	1
8	–	–	1	–

* One run in each of these categories gave rise to no solutions.

Acknowledgements

The author is grateful to the Scottish Council for Research in Education for a grant in support of this work, to the headmasters of the schools co-operating in the study, and to Miss S. J. Turner for assistance in the collection of data and in computer programming.

References

BARRACLOUGH, E. D. (1965). The application of a digital computer to the construction of timetables, *The Computer Journal*, Vol. 8, p. 136.

CLEMENTSON, A. (1968). Computer Timetabling Data Manual. Published by the Local Government Operational Research Unit.

COX, N. S. M. (1969). Final Report on the Research Project into the Construction of Timetables by Automatic Computer. To be published by the University of Newcastle upon Tyne Computing Laboratory, 1969.

GILMORE, P. C., and GOMORY, R. E. (1961). A Linear Programming Approach to the Cutting-Stock Problem, *Operations Research*, Vol. 9, No. 6, p. 849.

GOMORY, R. E. (1958). An Algorithm for Integer Solutions to Linear Programs, *Recent Advances in Mathematical Programming*, ed. R. L. Graves and P. Wolfe, McGraw-Hill, New York, 1963.

JOHNSTON, H. C., and WOLFENDEN, K. (1968). Computer Aided Construction of School Timetables, paper presented at IFIP Conference.

LAWRIE, N. L. (1968). School Timetabling by Computer, *Aspects of Educational Technology*, Vol. II, Methuen, London, 1968.

LEWIS, C. F. (1961). *The School Timetable*, Cambridge University Press.

LIONS, J. (1967). Construction of Timetables for Ontario Schools Using a Computer, O.E.C.D. paper, Paris.

Local Government Operational Research Unit (1967). The Use of Computers for School Timetabling. Report No. C19. Reading.

Book Review

The Art of Computer Programming, by Donald E. Knuth Volume 1—Fundamental algorithms. 1968. 634 pp. 182s.; Volume 2—Seminumerical algorithms. 1969. 624 pp. (*Addison-Wesley*, 173s.)

These two heavy volumes are the first to be published of a series of seven forming a work with the general title 'The Art of Computer Programming'. When completed, this work will be an immense achievement. It is impossible to do justice even to the first two volumes in a short review, and it would be presumptuous to attempt a full evaluation without giving them a long and detailed study. I can only attempt to define the scope of the work and its flavour, and to make one or two scattered comments.

The books are addressed to people who approach things from the mathematical point of view. Such people enjoy proofs and like to be set puzzles. They will find plenty here. Some 25% of Volume 1 is taken up either in setting problems or in indicating their solutions. The author has taken the trouble to classify the exercises according to their difficulty and degree of mathematical orientation.

There is no doubt that, as well as being a mathematician, Dr. Knuth is a practical man when it comes to dealing with a computer. He discusses programming down to the level of assembly language, and uses for this purpose an imaginary computer known as MIX, whose assembly code is fairly typical of real assembly codes. He does not attempt to cater for the complete beginner, but he does give a comprehensive treatment of the finer points. I was delighted, for example, to find the first comprehensible account that I have seen of co-routines (although we shall have to wait for Volume 4 for a treatment of recursive co-routines). Nevertheless, the fact remains that the overwhelming, almost overbearing, atmosphere of the book is one of mathematics. It would be unfortunate if some ordinary mortal, attracted by the title and charmed by the style, were, nevertheless, led to conclude that he needed a high standard of mathematical knowledge in order to understand programming.

The first 119 pages of Volume 1 are about general mathematics and could just as well have appeared in a book on, for example, quantum mechanics. After that, the author goes on to describe his imaginary computer MIX, and in terms of it to discuss fundamental programming techniques such as subroutines, interpretive routines, trace routines, etc. He then discusses information structures, including stacks, queues, linked lists, trees, and so forth. However, it is not long before he passes from programming to mathematical problems suggested by programming, such as the enumeration of trees.

Volume 2 deals with random numbers and with algorithms for performing arithmetic operations. It is stated in the preface to contain a noticeably higher percentage of mathematical material than other volumes in the series. Methods of generating a series of pseudo-random numbers in a computer are exhaustively discussed. It is pointed out that some methods are better than others, and statistical techniques for evaluating them are developed.

In the next chapter, the author goes systematically through floating-point arithmetic, multi-precision arithmetic, radix conversion, rational arithmetic, and polynomial arithmetic. The material partly concerns the designer of the algorithms that are wired into a computer, and partly the designer of subroutines or procedures that are incorporated within programming languages. These subjects are of basic importance, and much time and money has been wasted because people have not understood them properly.

Here and there in the volumes there are historical notes. These, like everything else, are flavoured with the author's personality; they are adequate for their purpose, but not necessarily definitive from an historian's point of view.

There can be few people in university computer departments who will not get some value and pleasure from browsing in these volumes. Those who are able to learn by studying details, and who like to follow up interesting side issues, will read them systematically. Others will perhaps recall the story of the man who apologized for writing a long book, saying that, unfortunately, he could not spare the time to write a short one. However, Dr. Knuth, generous with his time as with his scholarship, has foreseen this criticism and plans to publish later a shortened version of the series. I hope that in preparing this he will have the non-mathematician in mind.

M. V. WILKES (Cambridge)