An algorithm for constructing University timetables

By Mary Almond*

This paper gives details of a simple heuristic approach to the University timetable problem. The method is used to construct a timetable for one department and an integrated timetable for all departments in a Science Faculty.

Scheduling lectures is tedious and frustrating work, and the problem of applying computers to this task is currently being investigated in many countries. Several of the published reports discuss theoretical solutions only (Gotlieb, 1963; Csima and Gotlieb, 1963; Sherman 1963). Other authors have achieved some practical success in constructing school or University class timetables (Appleby, Blake and Newman, 1961) in preparing examination tables (Broder, 1964; Cole, 1964) and in assigning students to sections of a class according to a previously prepared timetable (Bossert and Harmon, 1963).

This paper describes algorithms for a heuristic approach starting with a blank timetable and making class-lecturer assignments so as to satisfy complex conditions. Two problems have been considered:

(*a*) a timetable for one department in which courses for each class are fixed; and
(*b*) a timetable for a whole faculty in which courses offered by different departments may be combined in various ways to suit individual students.

The algorithms have been written in ALGOL 60 and used on the University of London Atlas computer. The resulting timetables will be used in the Mathematics Department and the Science Faculty at Queen Mary College, University of London.

PROBLEM (a): Timetable for one department

Statement of the Problem

Given a set of lecturers, a set of classes and a Class Requirements matrix with integer elements representing the number of hours lecturer (*l*) is to meet class (*c*) during each week, the problem is to allocate times (*t*) to these hours satisfying certain given conditions. Hence there are 3 variables, *l*, *c*, *t* and it is convenient to think of assignments being made in a 3-dimensional Boolean array as shown in **Fig. 1.** The algorithm uses the three 2-dimensional arrays which form the three rectangular faces of this brick. Face 1 is the Class Requirements matrix. Face 2 is of dimensions (lecturer) × (time) and is called the Lecturer Availability matrix. It is a Boolean matrix whose coefficients are false for hours when the lecturer is free and true for hours when he is unavailable. The Timetable, face 3, is an integer matrix of size (class) × (time) whose coefficient in row *c* and column *t* will be the name of the lecturer *l* meeting class *c* at time *t*.

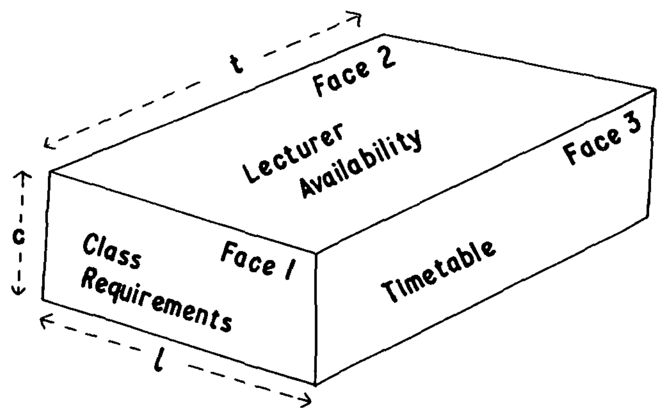

Fig. 1

Input

Initially the matrices in faces 1 and 2 contain input data. The Initial Class Requirements matrix gives the total number of hours each lecturer is to meet each class. The Initial Lecturer Availability matrix has entries true when a member of staff is lecturing to another department or has a free day. These two matrices are duplicated in the Current Requirements matrix and the Current Lecturer Availability matrix which can be repeatedly updated as the timetable is constructed.

Output

The matrix in face 3 is at first a null matrix, and finally it contains the completed timetable. It is printed in its existing form using a write-text procedure for lecturers' names.

Algorithm

The method for solution is to consider the entries in the Requirements matrix one-by-one and allocate to each a suitable lecture hour. When an allocation has been made at time *t* for class *i* and lecturer *l*, then one is subtracted from the integer in row *c* and column *l* of the Current Requirements matrix, the Current Lecturer Availability matrix is marked true at row *l* and column *t* and the value of *l* is inserted at a point on the *c*th row and *t*th column of the Timetable matrix. This process is illustrated in **Fig. 2.** In general the solution will not be unique, and different versions of the timetable may be obtained by scanning the Requirements matrix in different directions. If any allocation fails certain con-

* *Mathematics Department, Queen Mary College, Mile End Road, London, E.*1.

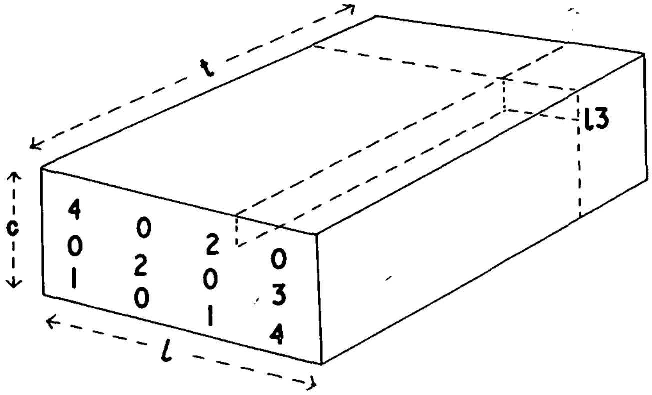

Fig. 2

ditions are changed, the Timetable is wiped clean and the whole process is restarted from the initial conditions. The ALGOL version of the program is given in **Table 1.**

Three predeclared procedures *allocate one lecture*, *alter conditions* and *copy initial matrices* are used.

(*i*) The *allocate* procedure (see **Fig. 3**), searches for a suitable lecture time avoiding the hours already filled and satisfying any desired conditions. For example the results illustrated in **Tables 2** and **3** meet the following conditions.

(*a*) Several lecturers take classes at fixed times in other departments or faculties.
(*b*) All members of staff have one free day each week.
(*c*) Senior members of staff do not like 9.30 a.m. lectures.
(*d*) Lecturers may ask for extra free hours which will be allowed if possible.
(*e*) A lecturer should not meet the same undergraduate class twice in any half day, but he might meet a class in both the morning and afternoon of one day.
(*f*) Lectures to postgraduate classes last for two consecutive hours and should begin at 10.30 a.m. or 2.30 p.m.
(*g*) Undergraduates have no lectures on Wednesday afternoons.
(*h*) If possible, no-one should be asked to lecture for three consecutive hours.
(*i*) All lectures should be given in the morning in preference to the afternoon.
(*j*) The classes are split into two or three groups for exercises.

To meet conditions (*a*), (*b*), (*c*), (*d*) appropriate entries true must be made in the Initial Lecturer Availability matrix. The allocation will then avoid these hours. Conditions (*e*), (*f*), (*g*), (*h*) are satisfied by a series of tests in the allocation procedure. The hours of the week are numbered in such a way that the mornings are always filled first, i.e., numbers 1 to 5 for the first hours of the mornings Monday to Friday, numbers 6 to 10 for the second hours of the day, and so on. When an

Table 1

```
timetable: for c := 1 step 1 until total classes do
               for n := 1 step 1 until number of hours do
                  begin allocate one lecture;
                     if fail then
                        begin alter conditions;
                              copy initial matrices;
                              goto timetable
                     end
                  end;
```

exercise class is being allocated it may use the same hour as a previous exercise class provided that the lecturers involved are available.

(ii) The *alter conditions* procedure which is called in when an allocation fails carries out a series of manoeuvres in an attempt to find a solution. First the classes are reordered so that the one proving difficult will be inserted in a blank timetable. If this is unsuccessful the lecturer whose hour cannot be allocated is given a different free day. Finally if all free days prove impossible the lecturer will have his extra free hours removed. or he may have to give three consecutive lectures. If this fails the procedure prints a postmortem and brings the program to a halt.

(iii) After an alteration in conditions the Initial Requirements and Initial Lecturer Availability matrices are recopied into the Current Requirements and Current Lecturer Availability matrices, and the timetable matrix is made null. The *allocation* procedure is then re-entered.

Result

Timetables produced by this program are shown in Tables 2 and 3. The program was contained in 32 512-word blocks on the Atlas computer. Compilation took approximately 9 seconds and execution for these examples took about $7\frac{1}{2}$ seconds.

Execution time will vary considerably with the difficulty of finding a solution. When the *alter conditions* procedure was not needed times of 5 to 8 seconds were taken, and each additional attempt took approximately 2 extra seconds. A few seconds would be saved if the Timetable were printed in terms of lecturer's numbers rather than names.

PROBLEM (b): Timetable for the Faculty

Statement of the problem

Given a set of lecturers, a set of courses on individual topics and a Course Requirements matrix with integer elements representing the number of hours lecturer (l) meets course (c) during each week, the problem is to allocate times (t) to these hours so that a student may take as many suitable combinations of courses as possible.

As before the assignments are made in a 3-dimensional array, see **Fig. 4.** Side c is now of length corresponding to the total number of courses. Again the algorithm

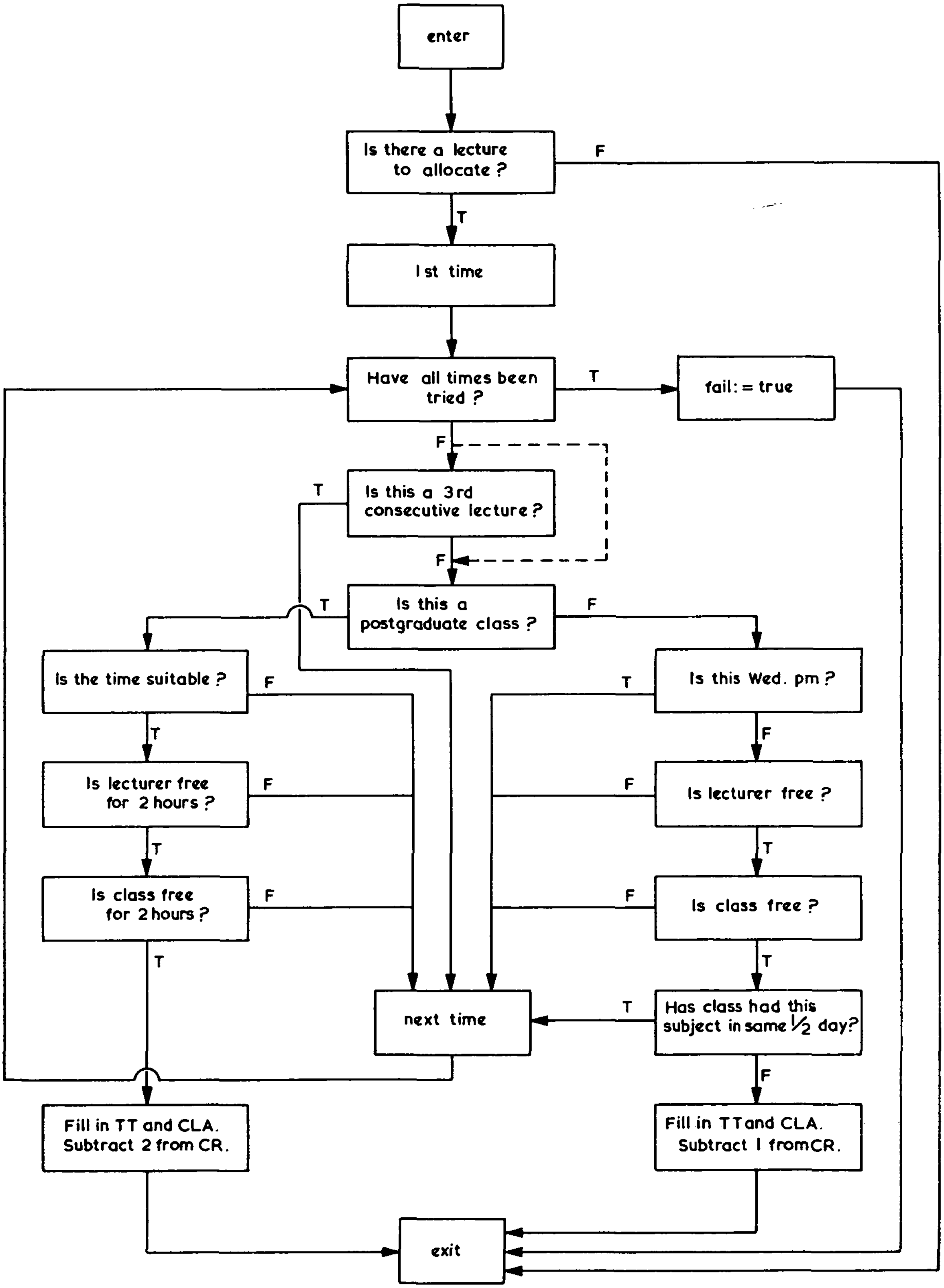
enter
Is there a lecture to allocate ?
F
T
1st time
Have all times been tried ?
T
fail: = true
F
Is this a 3rd consecutive lecture ?
T
F
Is this a postgraduate class ?
T
F
Is the time suitable ?
F
T
Is lecturer free for 2 hours ?
F
T
Is class free for 2 hours ?
F
T
Is this Wed. pm ?
T
F
Is lecturer free ?
F
T
Is class free ?
F
T
next time
Has class had this subject in same ½ day?
T
F
Fill in TT and CLA. Subtract 2 from CR.
Fill in TT and CLA. Subtract 1 from CR.
exit

Fig. 3

Table 2

Timetable for the Mathematics Department, Version 1

CLASS	HI	HII	HIII	PIII	PG
Monday					
9.30	Thomas	Carter	King	Shaw	—
10.30	Fisher	Exercises	Hughes	King	Peters
11.30	Shaw	Thomas	Pratt	—	Peters
12.30	—	—	Rose	—	—
2.30	—	—	Pratt	—	Gray
3.30	—	—	—	—	Gray
Tuesday					
9.30	Lewis	White	Shaw	—	—
10.30	Fisher	Rogers	White	Exercises	—
11.30	Andrew	Green	Fuller	King	—
12.30	—	—	Reader	—	—
2.30	—	Green	King	—	—
3.30	—	—	—	—	—
Wednesday					
9.30	Thomas	Exercises	Rogers	—	—
10.30	Andrew	Carter	Reader	—	Fisher
11.30	—	Thomas	Green	—	Fisher
12.30	—	—	—	—	—
2.30	—	—	—	—	—
3.30	—	—	—	—	—
Thursday					
9.30	Shaw	Rowland	Rose	—	—
10.30	Lewis	Andrew	Hughes	—	Grant
11.30	Thomas	White	Shaw	—	Grant
12.30	—	—	Fuller	—	—
2.30	—	—	Green	—	—
3.30	—	—	—	—	—
Friday					
9.30	Shaw	Rowland	King	—	—
10.30	Andrew	Rogers	White	Shaw	Coles
11.30	—	Andrew	Rogers	—	Coles
12.30	—	—	Rose	—	—
2.30	—	—	White	—	—
3.30	—	—	—	—	—

uses the three rectangular faces of this brick for the Course Requirements, Lecturer Availability and Timetable matrices. In addition, a 2-dimensional Boolean array known as the Conflicts matrix, lists groups of courses whose lecture times must not conflict. For example, in row 1 chemistry, physics, mechanics and their associated laboratory classes could all be given the value true.

Input

Input data must include information for the Initial Course Requirements and the Initial Lecturer Availability matrices. Again lecturers will have free days and may be occupied with lectures in other faculties. As before these matrices are copied into Current versions which can be updated during the allocation.

Groups of courses which should be available for a student are put into rows of the Conflicts matrix. These groups may be in two categories, essential and desirable.

Output

In keeping with the usual Faculty convention the timetable is printed as a matrix of dimensions (day of week) × (time of day) whose coefficients are lists of the lectures taking place at that hour. To produce this timetable, face 3 of the brick in Fig. 4 is stored as a

Table 3

Timetable for the Mathematics Department, Version 2

(Version 2 uses the same data as Version 1, but the Requirements matrix is scanned in the opposite direction)

CLASS	HI	HII	HIII	PIII	PG
Monday					
9.30	Lewis	Rogers	Reader	Shaw	—
10.30	Shaw	Thomas	Rose	King	Gray
11.30	Thomas	Carter	Rogers	—	Gray
12.30	—	—	Pratt	—	—
2.30	—	Carter	King	—	Fisher
3.30	—	—	Pratt	—	Fisher
Tuesday					
9.30	Andrew	Green	Fuller	Exercises	—
10.30	Fisher	Exercises	Shaw	King	—
11.30	Lewis	White	King	—	—
12.30	—	Thomas	White	—	—
2.30	—	—	Hughes	—	—
3.30	—	—	—	—	—
Wednesday					
9.30	Andrew	Exercises	Green	—	—
10.30	Thomas	Green	White	—	Coles
11.30	Fisher	Rogers	Hughes	—	Coles
12.30	—	—	—	—	—
2.30	—	—	—	—	—
3.30	—	—	—	—	—
Thursday					
9.30	Shaw	Andrew	Rose	—	—
10.30	Andrew	Rowland	Green	—	Grant
11.30	Thomas	White	Fuller	—	Grant
12.30	—	—	White	—	—
2.30	—	—	—	—	—
3.30	—	—	—	—	—
Friday					
9.30	Shaw	Andrew	Reader	—	—
10.30	—	Rowland	Rogers	Shaw	Peters
11.30	—	—	Shaw	—	Peters
12.30	—	—	Rose	—	—
2.30	—	—	King	—	—
3.30	—	—	—	—	—

Boolean matrix of size (course) × (time). The output procedure must scan the columns of this matrix to form the lists of lectures for each hour.

Algorithm

The basic algorithm is unchanged. The previous ALGOL program is repeated with courses replacing classes in the outermost cycle.

The *allocation* procedure must satisfy the same conditions as before. In addition some of the courses now include laboratory classes which require from 2 to 5 consecutive hours. These are allocated first by putting them at the head of the list of courses. The number of lectures allocated at each hour can be limited by the number of lecture theatres available.

The procedure also scans the Conflicts matrix for any groups of courses containing the course which is being allocated, and ensures that its lecture hour will not coincide with any of the other courses in any of these groups.

The block diagram for procedure *allocate* is shown in **Fig. 5** and the ALGOL version is given in **Table 4.**

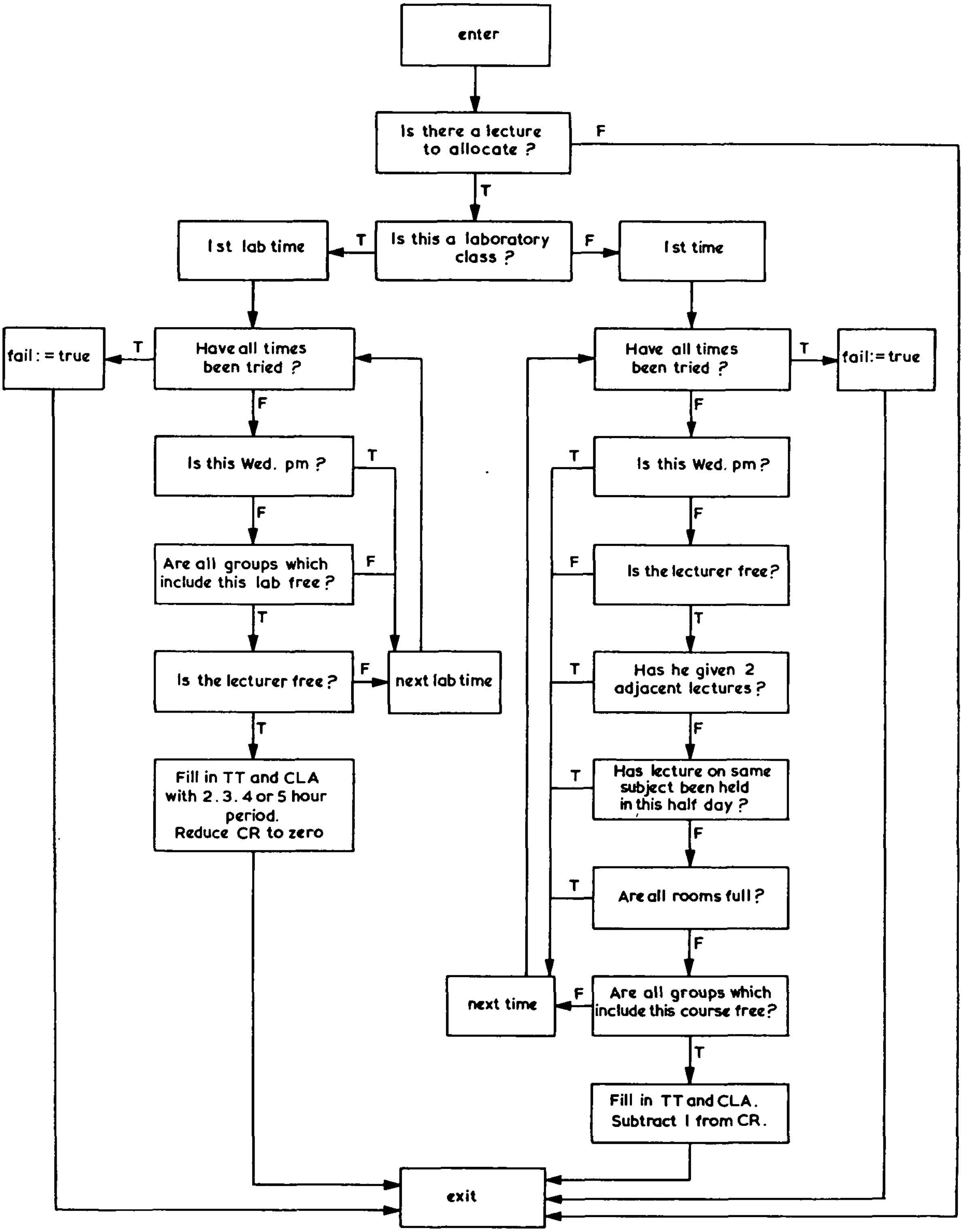
enter
Is there a lecture to allocate ?
F
T
Ist lab time
T
Is this a laboratory class ?
F
Ist time
fail : = true
T
Have all times been tried ?
F
Is this Wed. pm ?
T
F
Are all groups which include this lab free ?
F
T
Is the lecturer free ?
F
next lab time
T
Fill in TT and CLA with 2.3.4 or 5 hour period. Reduce CR to zero
Have all times been tried ?
T
fail:= true
F
T
Is this Wed. pm ?
F
F
Is the lecturer free ?
T
T
Has he given 2 adjacent lectures ?
F
T
Has lecture on same subject been held in this half day ?
F
T
Are all rooms full ?
F
next time
F
Are all groups which include this course free ?
T
Fill in TT and CLA. Subtract I from CR.
exit

Fig. 5

Table 4

ALGOL version of procedure *allocate*

```
procedure allocate (c, l); integer c, l;
begin integer t, j, g, p;
  if CR[c, l] ≠ 0 then
    begin if c< laboratory then goto lab; t := 0;
    next time: t := t + 1; if t ⩽ 40 then
      begin if CLA[l, t] then goto next time;
        for j := 23,28,33,38 do if t = j then goto next time;
        for j := 1,2,3,4,5,21,22,24,25 do if t = j and ((CLA[l, t + 5] and CLA[l, t + 10]) or
          (TT[c, t + 5] or TT[c, t + 10] or TT[c, t + 15])) then goto next time;
        for j := 6,7,8,9,10,26,27,29,30 do if t = j and ((CLA[l, t + 5] and
          (CLA[l, t − 5] or CLA[l, t + 10])) or (TT[c, t − 5] or TT[c, t + 5] or TT[c, t + 10]))
          then goto next time;
        for j := 11,12,13,14,15,31,32,34,35 do if t = j and ((CLA[l,t − 5] and
          (CLA[l, t − 10] or CLA[l, t + 5])) or (TT[c, t − 10] or TT[c, t − 5] or TT[c, t + 5]))
          then goto next time;
        for j := 16,17,18,19,20,36,37,39,40 do if t = j and ((CLA[l, t − 10] and CLA[l, t − 5])
          or (TT[c, t − 15] or TT[c, t − 10] or TT[c, t − 5])) then goto next time;


      if rms[t] > rooms then goto next time;
    for g := 1 step 1 until course do
      if C[g, c] then
        for p := 1 step 1 until lecture do
          if C[g, p] then begin if TT[p, t] then goto next time end;
      TT[c, t] := CLA[l, t] := true; CR[c, l] := CR[c, l] − 1; rms[t] := rms[t] + 1
    end
    else fail := true; goto exit;
    lab: t := 10;
      next lab time: t := t + 1; if t ⩽ 15 then
        begin if t = 13 and CR[c, l]>2 then goto next lab time;
        for g := 1 step 1 until course do
        if C[g, c] then
          for p := 1 step 1 until lecture do
          if C[g, p] then begin if TT[p, t] or TT[p, t + 10] and CR[c, l]>2
                      then goto next lab time end;


    if (CR[c, l] = 2 or CR[c, l] = 5) and not CLA[l, t] and not CLA[l, t + 5] then
      begin TT[c, t] := TT[c, t+5] := CLA[l, t] := CLA[l, t+5] := true;
          CR[c, l] := CR[c, l]−2
      end;
    if (CR[c, l]=3 or CR[c, l]=4) and not CLA[l, t + 10]
      and not CLA[l, t + 15] and not CLA[l, t+20] then
      begin    TT[c, t+10] := TT[c, t+15] := TT[c, t+20] := true;
               CLA[l, t+10] := CLA [l, t+15] := CLA[l, t+20] := true;
               CR[c, l] := CR[c, l]−3
      end;
      if CR[c, l]=1 and not  CLA[l, t+25] then
        begin TT[c, t+25] := CLA[l, t+25] := true; CR[c, l] := 0 end
      end
      else fail := true
    end;
  exit: end allocate;
```

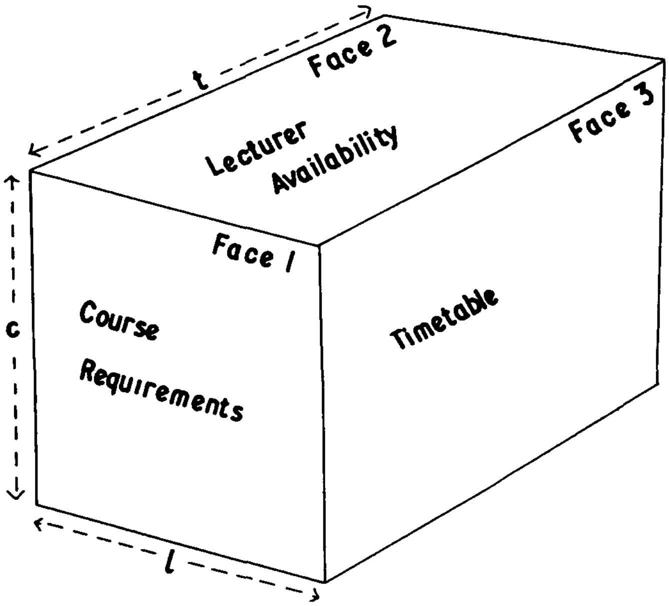

Fig. 4

The *alter conditions* procedure again tries first a reordering of the courses. If this fails then the lecturers' free days can be adjusted, one or two extra lecture theatres may be used, or finally those groups of courses which are desirable rather than essential can be neglected.

Results

Typical results are illustrated in **Tables 5** and **6**. The program was contained in 40 storage blocks of Atlas, compilation took about 10sec, execution for **Table 5** 12·5 sec and for **Table 6** 40 sec.

Possible groups of courses

A student is expected to take three courses at once, hence an extra procedure was added which would scan the final Boolean timetable and list all possible combinations of three courses. For example, there are over 100 possible groups of three courses for the timetable in **Table 5**.

Timetables and lists of possible courses were produced for each of a student's first four semesters.

An extra program then takes as input data these four lists of three courses and also any prerequisite courses for courses in semesters 2 to 4, and produces a list of

Table 5

Science Faculty Timetable for Semester 1

Courses are represented by the department initial (B Botany, Z Zoology, C Chemistry, P Physics, M Mathematics, G Geology, g Geography), followed by a reference number within the department.

	9.30	10.30	11.30	12.30	2.0	3.0	4.0	5.0
Monday	B1 Z1 Z2	C2 M2	B1lab Z1lab Z2lab M3 M4	-------- -------- -------- M5 M6	-------- -------- --------	-------- -------- --------	---- ---- ----	
Tuesday	B1 Z1 Z2	P1 M1 G1	Z3lab P2 M2 g1	-------- C3 M6 g2	-------- C1lab g3lab	-------- -------- --------	---- -------- ----	 ----
Wednesday	Z3 C1 P3	P1 M1 G1	P2 M3 M5	C3 M7 g1				
Thursday	Z3 C1 P3	P1 M1 G1	M3 M4	M7 g1	C2lab G1lab	-------- --------	-------- ----	----
Friday	C2 g3	P2 M2 M5	M4 M6	M7 g4	P1lab P3lab	-------- --------	---- ----	

Table 6

Science Faculty Timetable for Semester 4

	9.30	10.30	11.30	12.30	2.0	3.0	4.0	5.0
Monday	B2 B7 Z4 Z10 C12 P11 M8 M34 M35 g15	B9 C4 C16 P9 M10 M28 M36 M37 G5 g5	B2lab B7lab Z4lab Z10lab C17 P12 M11 M14 M29 g15lab	M16 M17 M31	C12lab P11lab M33			
Tuesday	B2 B7 Z4 Z10 C12 P11 M8 M11 M34 M35	Z13 C5 P9 M10 M28 M38 G2 G6	B8lab Z5lab Z11lab P5lab P12 M13 M30 g8 g19	M17 g20	C16lab g17lab			
Wednesday	B8 Z5 Z11 Z12 C12 P5 M9 g7	Z13 C5 P10 M10 M28 G2 G5	B10 P4 P12 M15 M30 g20	M12 M16 M17 M32				
Thursday	B8 Z5 Z11 Z12 C16 P4 M9 M36 M37 g5	C17 P10 M8 M11 M29 M38 G2 g16	B9lab Z12lab C6 M14 M15 M30 G5lab	M12 M32	C4lab M33			
Friday	B9 C4 C16 P9 M9 M14 M34 M35 G5 g5	B10 C17 P4 P10 M13 M29 M36 M37 g6 g18	B10lab Z13lab M12 M15 M16 M31 M38 G6lab	C6 M32	C5lab C17lab G2lab			

Table 7

Some possible combinations of courses

SEMESTER 1			SEMESTER 2			SEMESTER 3			SEMESTER 4		
Z3	M4	M6	Z5	M11	M16	P7	P8	M26	P9	P10	P11
M1	M6	M7	M13	M15	M17	M23	M25	M27	M34	M36	M38
P3	M1	M3	P5	M8	M10	P6	M18	M19	M28	M29	M30
C1	C2	M4	C4	C5	M11	C7	C8	M2	C12	C16	C17
Z1	C1	C2	Z4	C4	C5	B4	B5	B6	B7	B8	B10
Z2	C1	G1	B2	C4	G2	C7	G3	G4	C12	G5	G6
Z3	P1	M5	B2	Z5	M12	Z6	Z7	Z8	Z10	Z11	Z13

all possible groups of twelve courses which a student could study during his first four semesters using the given timetables. A section of the results of this program is illustrated in **Table 7.**

The basic principles of these algorithms for producing timetables seem very simple and it is hoped that other people may be able to adapt them for their own purposes.

Acknowledgements

The author wishes to thank Dr. B. H. Chirgwin of Queen Mary College, who had previously produced the Mathematics Department timetables, for impetus in starting this work and for many helpful discussions; also Professor V. C. A. Ferraro, Head of the Mathematics Department, for his encouragement.

References

GOTLIEB, C. C. (1963). "The Construction of Class-Teacher Timetable," *Proc. IFIP Congress* 62, Munich, North Holland Pub. Co., Amsterdam.

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.

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.

BRODER, S. (1964). "Final Examination Scheduling," *Communications of the ACM*, Vol. 7, No. 8, p. 494.

COLE, A. J. (1964). "The preparation of examination timetables using a small store computer," *The Computer Journal*, Vol. 7, No. 2, p. 117.

BOSSERT, W. H., and HARMON, J. B. (1963). *Student sectioning on the IBM* 7090, IBM Corp., Cambridge, Mass.

Notice: Newsletter for numerical analysts

The ACM Special Interest Committee on Numerical Mathematics (SICNUM) will begin publication of a newsletter in order to provide numerical analysts with a fast means of communication below the journal level. The newsletter is free and will be sent upon request. (ACM membership is *not* required.) The newsletter will appear as frequently as there is sufficient material.

Material for the newsletter is solicited. The material will not be refereed. It should be submitted in duplicate on bond in a form ready for publication. Material appropriate for journal publication should *not* be sent to the newsletter. Requests to receive the newsletter and material for the newsletter should be sent to the chairman of SICNUM:

DR. J. F. TRAUB, Mathematics Research Center,
Bell Telephone Laboratories, Incorporated,
Murray Hill, New Jersey.

A partial list of the type of material which is appropriate for publication in the newsletter follows:

(*a*) Material relating to numerical algorithms.
(*b*) Announcement of available programs, especially packages; critical discussions of existing programs.
(*c*) Announcement of availability of technical reports; announcement of new books.
(*d*) Announcement of meetings of interest to numerical analysts.
(*e*) Letters to the editor—thinkpieces.
(*f*) Problems and their solutions.
(*g*) Discussions of curricula.
(*h*) Personal news.