THE CONSTRUCTION OF ACADEMIC TIMETABLES

BY COMPUTER

by

JOHN NIGEL GAUNT BRITTAN

Thesis submitted for the degree of Doctor of

Philosophy in the University of London

London University Institute of Computer Science

April 1973

ABSTRACT

This thesis describes research into the construction of academic teaching timetables with the aid of a computer.

After an outline of the process of producing the timetable and a brief review of existing work in the field of timetabling by computer, an idealised problem used in the more theoretical work and known as the "timetable problem" is contrasted with the real timetabling situation in schools. It is concluded that the timetable problem is not a close enough approximation to the real problem in British secondary schools for work based on it to be capable of yielding results that are practically useful. An heuristic approach based on as full a description of the real problem as possible has therefore been adopted.

Possible methods of data storage are examined and the choice of an original method for use in practical development work justified. The various algorithmic processes involved in timetable construction are discussed from a theoretical point of view. The remainder of the thesis deals with the computer timetabling system which has been developed and the investigations carried out and results achieved with this system.

To be successful the solution of a timetabling problem almost invariably requires some departure from the original problem specification. An important part of this research has been the development of a fully interactive version of the timetable construction program. In this system, user and computer cooperate in the solution process, with the user able to modify the problem when necessary in the light of his

own knowledge and experience and of the detailed information on the current partial timetable presented by the computer.

It is concluded that computer timetabling is now technically feasible at an economic cost. Man-computer interaction in this field offers new possibilities which should be further investigated.

The critical survey of storage structures and algorithms, the basic concepts underlying the timetabling system produced, particularly the interactive features, are all believed to be original and to make a significant contribution to knowledge in this field.

Dependence on the work of others, wherever appropriate, has been explicitly stated in the text.

ACKNOWLEDGEMENTS

The work reported in this thesis was carried out by the author as a part-time student at the University of London Institute of Computer Science, while employed on the academic staff of the Royal Military College of Science.

My sincere thanks are due to my supervisor, Professor Keith Wolfenden for his encouragement, friendship and inspiration throughout the project: and also to:

Professor R.A. Buckingham, the Director of the Institute for accepting me as a part-time Research student;

Dr. F.J.M. Farley, Dean of the Royal Military College of Science, for encouraging me to start on this line of research and for his continued support and encouragement;

Mr. P.R. Farley, lately of the Royal Military College of Science, for the development of routines for visual display units;

Squadron Leader W.A. McIlroy of Research Branch, R.A.F. Training Command, for cooperation on trials and experiments using the computer timetabling system;

Commander David Brittan, my brother, for help with translation;

Professor K.L. Hunt for making it possible for me to undertake this project;

The following head teachers for allowing me to use data from their schools in this work:

Wing Commander I.J.S. Corderey, R.A.F., Officer Commanding No. 9 School of Technical Training, R.A.F. Newton;

Mr. J. Harrison, The American School in London;

Mr. S. Hughes, Burnage High School;

Mr. K.D. Smith, Aylesbury Grammar School;

Miss A.J. Towns, Faringdon County Grammar School.

The many people not mentioned above who have helped me in various ways.

Finally I would like to thank My wife Rosemary for her patient and unfailing support during this project.

CONTENTS

		Page
GLOSSARY OF TERMS		10
CHAPTER 1 -	INTRODUCTION	15
1.1	The timetabling problem in British Secondary Schools	15
1.2	Related problems	17
1.3	The preparation of the curriculum	20
1.4	The timetable construction phase	21
1.5	The role of the computer	25
1.6	Problems in using a computer	25
1.7	Survey of existing work	27
CHAPTER 2 -	THE "TIMETABLE PROBLEM" AND THE TIMETABLING PROBLEM	36
2.1	The timetable problem	36
2.2	A possible extension	38
2.3	The timetabling problem	39
2.4	The two problems compared	42
CHAPTER 3 -	STORAGE STRUCTURES	45
3.1	Data storage and computer timetabling	45
3.2	Essential features	45
3.3	Desirable features	46
3.4	Possible storage structures	48
3.5	Schools used as examples	51
3.6	Quantitative comparisons	55
3.7	Discussion	56
CHAPTER 4 -	ALGORITHMS I - EXISTENCE AND EVENT SELECTION	60
4.1	Processes	60
4.2	Existence	62
4.3	Event selection	65

		Page
4.4	Event selection criteria used in computer timetabling	67
4.5	Current and inherent difficulty	68
4.6	Other event selection criteria	72
4.7	Frequency of selection	72
CHAPTER 5 -	ALGORITHMS II - TIME AND RESOURCE SELECTION	74
5.1	Interaction between time selection and resource selection	74
5.2	Steps in time selection	75
5.3	The elimination of unacceptable periods	77
5.4	Existence testing in relation to other steps	78
5.5	Interference	82
5.6	Timetable quality	88
5.7	Resource selection	90
5.8	Learning techniques	94
CHAPTER 6 -	ALGORITHMS III - CONFLICT RESOLUTION AND USER GUIDANCE	97
6.1	Detection of conflicts	97
6.2	Resolution of conflicts	97
6.3	Retreat and displacement compared	107
6.4	Modification of the problem	114
6.5	Man-computer interaction	116
CHAPTER 7 -	COMPUTER TIMETABLING SYSTEMS	123
7.1	Early work	123
7.2	SCOCOST - general philosophy	125
7.3	Data base	129
7.4	Input program	129
7.5	Main construction program	136
7.6	Room allocation program	137
7.7	Output program	143

		Page
CHAPTER 8 - TRIALS AND RESULTS		146
8.1	RAF Newton	146
8.2	Data sets, difficulty factors and system performance	150
8.3	Insertion rate, time and tdepth	154
8.4	Quality of timetables	157
8.5	Comparison of manual and computer timetables	162
CHAPTER 9 - INVESTIGATIONS INTO ALGORITHMS		165
9.1	Existence	165
9.2	Event selection	171
9.3	Time selection	177
9.4	Discussion	180
CHAPTER 10 - TIMETABLE CONSTRUCTION BY MAN-COMPUTER TEAM		181
10.1	Setting up facilities	181
10.2	The commentary	182
10.3	The command language INTERACT	187
10.4	Additional facilities	191
10.5	Closing down facilities	201
10.6	Experience in man-computer timetable construction	203
10.7	Retreat and displacement	206
10.8	Discussion	206
CHAPTER 11 - CONCLUSIONS		209
11.1	The Timetable problem	209
11.2	Overall practicability	209
11.3	Storage Structure	210
11.4	Algorithms	210
11.5	The user interface	212

Page

11.6 Future development 213

APPENDIX 1 - DATA STORAGE CALCULATIONS 214

APPENDIX 2 - DATA VARIABLES USED IN SCOCOST 218

APPENDIX 3 - SCOCOST DATA ARRAYS 219

APPENDIX 4 - EXAMPLES OF INPUTS TO AND OUTPUTS FROM SCOCOST 223

REFERENCES 237

GLOSSARY OF TERMS

Certain terms are used in this thesis in a specialized sense and it is important that the meanings of these terms are appreciated from the start. They are divided into four groups: time elements; resource elements; elements involving combinations of time and resources; and miscellaneous terms.

TIME ELEMENTS

1) Period

The period is the smallest indivisible time element in the timetable and is used as the basic unit of time. In a British secondary school a period is typically 40 minutes, although all periods in one timetable need not be of equal duration.

2) Week, day

The week is the duration of the timetable, typically 35 to 40 periods. Normally this will occupy a real week, but this is not necessarily the case. A week may be divided into a number of days each comprising 7 or 8 periods in a typical school.

RESOURCE ELEMENTS

3) Class, teacher, room

A class is any group of one or more pupils which can be treated as an indivisible unit for timetabling purposes. The terms teacher and room are used in their normal sense, except when qualified by the term multiple (see 8).

4) Resource

A general term covering classes, teachers and

rooms.

ELEMENTS INVOLVING COMBINATIONS OF TIME AND RESOURCES

5) Lesson, subject

A lesson involves a teacher (or a team of teachers) teaching one or more classes in one room for one or more consecutive periods. The term subject is used to indicate a prescribed area of study.

Example

It has been decided to timetable simultaneously a set of three optional subjects, Biology, Latin and Art, available to the Vth form of a school. The Vth form is divided into two classes Va and Vb. Two double period lessons and three single period lessons are required during the week for each subject. Miss Smith takes Biology in the Biology laboratory, Mr. Jones takes Latin in any non specialist classroom and Mrs. Wilks takes Art in the Art room.

6) Event e(i, z), duration, association

An event e(i, z) is a lesson, or group of lessons occurring simultaneously, belonging to the ith requirement (see 7) for a duration of z consecutive periods. In the above example the simultaneous occurrence of a double period lesson of Biology, Latin and Art constitutes a double period event. An event will normally be regarded simply as the time coincidence of a specified set of resources, without regard to breakdown (if any) into lessons. Thus any event in the example is associated with the following resources: Miss Smith, Mr. Jones, Mrs. Wilks, Class Va, Class Vb, Biology laboratory, Art room and one room from the group of non specialist classrooms.

7) Requirement ρ_i, value m(i)

A requirement ρ_i is a specification of the number and duration of a set of events e(i, z), all of which are associated with the same resources. Thus a requirement is associated with the same set of resources as each of the events which comprise it.

The value m(i) of a requirement ρ_i is an ordered n-tuple showing the number of events of duration 1,2,3,...,n which have to be scheduled during the week for that requirement. The dimension of the n-tuple is the maximum duration of any event in the entire problem. If this is 5 periods, say, and ρ_i comprises 3 single period events and 2 double period events, as above, then m(i) =<3,2,0,0,0>.

8) Resource groups, multiple resources

Sometimes an event may involve a number of resources which are not specified uniquely but must be selected from a group of resources which are equivalent for this purpose. Such a group of resources is defined as a resource group and is usually designated by the type of resource involved, for example room group. In the example after 5) above, the Latin lessons take place in one room selected from a room group, which in this case includes all the non-specialist classrooms.

A multiple resource, always designated by type, is a set of unique resources (see 9) of that type associated with an event. Thus Miss Smith, Mr. Jones and Mrs. Wilks constitute a multiple teacher.

9) Unique resources, selectable resources

A unique resource is a resource uniquely associated with a requirement in its specification; a selectable

resource is one which has to be selected from a resource group. In the example Mrs. Smith, Mr. Jones, Mrs. Wilks, Class Va, Class Vb, Biology laboratory and Art room are unique resources while the room in which the Latin lesson is due to take place is a selectable resource.

10) Event period, teacher period

The event period and the teacher period are units of activity in a timetabling problem. An event period is the occurrence of one event for one period, while a teacher period denotes one teacher teaching one or more classes for one period. In the example, each of the double period events involves two event periods and six teacher periods.

11) Clashing requirements, tight sets, nearly tight sets, open sets

Two requirements ρ_i and ρ_j are said to clash if an event of ρ_i cannot take place at the same time as an event of ρ_j because the two requirements are associated with one or more common unique resources. A tight set of requirements is a set of requirements in which each member of the set clashes with every other member. The requirements uniquely associated with any one resource, for example, form a tight set. A nearly tight set is a set of requirements in which every member clashes with every other member or every other member but one. An open set is any set of requirements which is neither tight nor nearly tight.

MISCELLANEOUS TERMS

12) Available, suitable

When used to qualify a resource and a period, the

term available means that the resource is free to be committed to an event at that period. The term is also used to qualify a period or set of periods and an event or requirement. A set of periods {p, p+1,...,p+z-1} is said to be available for an event e(i, z) or a requirement ρ_i if all the unique resources, and enough selectable resources belonging to appropriate groups associated with this event, or this requirement, are available at each of these periods. The same set of periods is said to be suitable for an event e(i, z) if the insertion of e(i, z) at these periods would be acceptable from all points of view other than resource availability.

13) Conflict, forcing situation

A conflict is said to exist whenever, at any stage of the construction of a timetable, the partially constructed timetable is incompatible with a satisfactory final solution. A forcing situation is said to exist at any stage of timetable construction if at that stage it can be shown that all but one of a number of possible alternative steps will give rise to a conflict.

14) Normal distribution (This term and its definition are due to Johnston [22]*).

Events of a requirement are said to occur in a timetable with a normal distribution if on any day d there are h events of this requirement ($h \geq 0$) and on any other day there are k events ($k \geq 0$) and $|h - k| \leq 1$, for all days d in the week.

---

* Numbers given in square brackets refer throughout to items in the list of references on pages 237-240.

CHAPTER 1 - INTRODUCTION

1.1 The timetabling problem in British secondary schools

This thesis describes research into the construction of academic teaching timetables with the aid of a computer. It is primarily concerned with the timetabling problem in British secondary schools, which in outline is as follows. Given a curriculum specifying the courses to be run at the various levels in the school, the allocation of pupils and teachers to courses, a defined time structure with the teaching week typically subdivided into 35 to 40 periods of about 40 minutes each, and information about the material resources (rooms and equipment) available, to implement the curriculum by producing a timetable showing when and where every lesson is to take place. A universal constraint is the need to avoid timetabling a person to be in two different places simultaneously or two groups of people to be using the same indivisible facility at the same time. In addition, most schools have an agreed pattern of operation covering such matters as avoiding a high concentration of any subject in any one part of the week, and the need for double and triple period lessons for certain subjects.

The timetabling problem has been successfully handled in the past by head teachers or senior members of staff using fairly rudimentary aids, confined in the main to pencil, paper and rubber.

Recent trends towards comprehensive reorganisation of the British secondary school system have led to

larger schools and the desire for ever greater flexibility in schools' curricula so that the education provided for each pupil matches his aptitude and ability as far as possible. These developments have increased the timetabling problem considerably, and indeed the difficulty of timetabling in such situations is an important factor in limiting curriculum development. In his recent book on school organisation, Davies [9] writes

" ..... given a research project of a reasonable order, rules could be generalized by which to devise economical means of computerising school timetables from models produced by headmasters. Until this has been achieved, ..., the comprehensive school is hardly likely to become the subject of widespread and bold experiments .....".

The construction of school timetables appears to be a natural problem for a computer, and computer assisted timetabling has attracted attention from the earliest days of automatic digital computing. In a British context serious published work dates from the 1961 paper of Appleby, Blake and Newman [2]. Work on computer timetabling for schools has continued in many parts of the world, the nature and difficulty of the problem varying from country to country with local conditions. In the United States, where the problem is quite different from that in Great Britain, a good deal of success has been achieved and computers are widely used. Until very recently, little real success had

been achieved in applying computers to British schools and indeed no British school is yet using computer produced timetables on a regular basis. The Institute of Computer Science of the University of London has been interested in this field for the past 7 years and the present work follows naturally from the lines of research previously adopted at the Institute.

1.2 Related problems

All problems involving the scheduling of meetings of any kind have a general similarity with the timetabling problem in British secondary schools. To set this work in context the more obviously related problems are outlined below.

1.2.1 American High Schools

A large proportion of American high schools run a pupil based system in which every pupil has a very wide choice of optional subjects open to him. Also, unlike the great majority of his British counterparts, the American high school pupil may have as much as 30% private study or "unstructured time".

The starting point for timetabling, or scheduling as it is known in the U.S.A., is a list of courses to be offered. A number of sections is allocated to each course, depending on the number of pupils expected to opt for it, and teachers are then allocated to each section. A master timetable or schedule is now created which takes explicit account of teachers and rooms only. If this is done by hand the scheduler will know from his experience many of those combinations of sections which

should not be scheduled at the same time. The final step is to schedule each pupil by presenting his subject choices to the master schedule and attempting to allocate him to a section of each of his chosen courses. This procedure normally allows some modification of the master schedule in order that as many pupils as possible may be satisfied, particularly in respect of important subjects. Those who cannot be allocated to any section of one or more of their chosen courses are asked to change their choice.

The scheduling of the individual pupils, known as sectioning, is a most laborious task when done by hand and is a very suitable task for a computer. A number of successful computer scheduling and sectioning systems are running in the United States.

1.2.2 European schools

Most European schools have a broadly similar structure to British secondary schools but with rather less individual specialization and consequently fewer complications with regard to choice of subjects by students. One computer timetabling system is already in operation in Norway [30], and has produced timetables for a number of schools in Scandinavia.

1.2.3 Universities and Polytechnics

Universities traditionally present a less intrinsically difficult timetabling problem than schools for two main reasons. Firstly the loading on resources in terms of timetabled periods per week is usually much lower, and secondly most students pursue specialized courses with teaching mainly in only one department.

Timetables can therefore usually be compiled on a departmental basis. In recent years, however, the growth of degree courses cutting across traditional departmental boundaries has tended to complicate this picture somewhat. Computer programs capable of generating workable University timetables have been produced [20, 36] but these are not widely used.

1.2.4 Technical colleges

Technical colleges often present a more complex timetabling problem than Universities. The loadings on students and teachers are higher and an additional factor is the presence of part-time and day release courses which impose very severe time constraints on the time-tabling problem. It is believed that no British technical college uses a computer to produce its timetable.

1.2.5 Short course establishments

As a general rule in the institutions described above the timetable repeats itself regularly and usually on a weekly cycle. This is because they are concerned with what may be called long courses lasting a term, a year or more. There are, of course, many exceptions to this rule. Extra-mural departments of Universities, for example, run a number of concentrated courses on particular topics for durations as short as three days. The main timetable of the institution, however, is not normally affected to any marked degree by such activity within one department. There are many places, however, where such courses make up a large proportion of the instruction, and in some cases all of it, for example:

training establishments in the services, industrial training centres, business schools, and computer training schools.

Short courses are structured very differently from long courses. Usually successive weeks of the same course have quite different timetables and such courses make very precise demands on facilities and resources. The order in which formal lessons, practical work, discussion periods, private study and so on occur is normally critical. Short course establishments usually have a combination of different courses running simultaneously and the establishment itself, and more particularly departments and sets of resources within the establishment, experience peaks and troughs of activity which can be quite marked. As far as is known no short course establishment in Great Britain uses a computer for timetabling.

1.3 The preparation of the curriculum

There are two main stages in the production of a British secondary school timetable. The first is the preparation of the curriculum, which defines the problem for the second, the timetable construction phase.

A number of fundamental decisions governing the framework within which the timetable must operate have to be made by the head of the school before he can start to prepare the curriculum, even though difficulties met in this and subsequent phases may force changes in these decisions. The number of periods in the teaching week is one such decision.

The curriculum defines the courses to be made available to the pupils in the school. Its preparation, thoroughly discussed by Davies [9], is a most important phase and is a valid subject for research in its own right. However, it is only relevant to the current work to the extent that it plays a vital part in defining the problem which the timetable construction phase has to solve.

The VIth form pupils are normally offered a wide choice of subjects to be taken either at advanced level or for general study. With the tendency towards larger schools with larger VIth forms, this element of choice presents increased timetabling problems and at present many schools arbitrarily restrict the allowable combinations of courses in order to make timetable construction by manual means a manageable operation.

Once the courses and groups of pupils taking each course have been established, the assignment of teachers and specialist rooms to individual courses takes place. A basic rule of curriculum preparation is that no teacher, room or pupil can be occupied for more periods than there are in the teaching week. This is a necessary, but not a sufficient condition for the existence of a timetable for any curriculum.

1.4 The timetable construction phase

A completed timetable must specify when and where every lesson is to take place. Normally the "where" element is established either before or after the "when" element. The allocation of specialist rooms to lessons

is part of the curriculum preparation stage, while non-specialist rooms are usually allocated to lessons once the time for each lesson has been determined.

For a small school with few optional subjects and where a well established pattern has developed timetable construction is not too difficult a problem, although it can easily take an experienced member of staff two or three weeks. In a large school or one where the head wishes to introduce extensive changes to the curriculum it is an exceedingly laborious task which can easily take two or three months' effort by a senior teacher, often the deputy head.

The manual method described by Lewis [26] is to keep track of all resources continuously on large sheets of paper or cardboard so that at any time it can be seen how any class, teacher or room is occupied at any period in the week. The first step is to mark part-time staff as not available for appropriate periods. Then various "preassigned" events are inserted, for example, those based on radio or television programmes which must occur at specific times. The timetabling of the bulk of the events which are not so heavily time constrained can now take place.

The complex interactions between the various elements in the timetabling problem makes it necessary for most people to work one step at a time and since it is difficult to foresee the effect of any step on future possibilities, this limitation makes the selection of each step most important. The essential questions are:

1) Which event is to be inserted next?

2) At which period or set of periods is it to be inserted?

It is worth noting that if a "correct" answer could always be found to question 2) then question 1) would be unimportant since insertions would always be made in a way compatible with a satisfactory final timetable. However, no method of insuring this has yet been found and the order in which events are selected for insertion remains most important. Fortunately, the person constructing the timetable has his own experience and that of others to guide him in this selection. The general rule is "difficult before easy" and Lewis amplifies this by suggesting categories of events likely to prove especially difficult, for example, VIth form events, those events belonging to setted subjects and those events which are associated with heavily used specialist rooms. General rules of this kind, however, are not always applicable in particular circumstances.

Once an event has been selected for insertion a period or set of periods must be chosen for it. It may be that the state of the partial timetable is such that no choice is possible; where this is not the case a number of criteria govern the selection. These criteria fall into three main groups as follows:

1) Obligatory or nearly obligatory rules based on educational considerations, such as the need to spread lessons of important subjects evenly over the week or to avoid interrupting

multiple period events by a long break such as lunch.

2) Desirable educational features: for example, many schools try to avoid having Mathematics lessons in the last period of the afternoon.

3) Constraints based on personal preferences: for example, married women teachers often prefer to be free for the first and last periods of the day if possible.

In general groups 1), 2) and 3) are in descending order of importance.

Frequently it is found that an event cannot be inserted because there are no suitable and available periods for it (glossary 12). In this case there are three options open: attempting to remove the difficulty by amending the partial timetable to permit insertion of the event; modifying the problem in some way perhaps by relaxing one of the original constraints; adopting some combination of the two previous courses of action.

Once the time for every event has been decided, the timetable is completed by allocating appropriate non-specialist rooms to lessons.

Inevitably, there is interaction between the curriculum preparation and timetable construction phases. Minor changes to the specification of the timetabling problem during the construction phase are almost inevitable. Difficulties during timetable construction may also cause major alterations to the curriculum, particularly if the desired curriculum is substantially different from that in operation the

previous year.

1.5 The role of the computer

Timetable construction is, in the main, a mechanistic process which involves keeping check of the many different elements of the problem. This type of process is, in principle, well suited to a computer. If computers can produce satisfactory timetables at an economic cost then much time and effort of senior members of staff could be saved by using computers in the construction process.

Apart from this direct benefit in the timetable construction phase itself, a computer timetabling system could help at the curriculum preparation stage by enabling the detailed implications of different approaches to the curriculum to be examined. It may also be possible to place fewer restrictions on pupils' choice of subjects. At the level of the educational administrator such a system might enable the timetabling implications of various possible developments in schools to be examined quickly and the cost of these developments in terms of resources determined more accurately than at present.

1.6 Problems in using a computer

The main problem to be overcome before any of the possible benefits mentioned in 1.5 can be obtained is to develop a computer program capable of constructing timetables which a school could even consider using in practice. The difficulty in producing such a program is due partly to the inherent difficulty of the school

timetabling problem itself, which involves a highly complex pattern of interactions between requirements (glossary 7) and resources, with most resources occupied for every period or nearly every period of the week. Then there is the difficulty of describing the problem to a computer program, particularly those aspects which are difficult to define precisely. Furthermore certain features of school life such as the existence of specially awkward classes of pupils, and the need to avoid too concentrated exposure of one teacher to such classes may be difficult even to admit explicitly!

In any given timetabling problem most of the rules and constraints can be violated under exceptional circumstances. When a human being constructs the timetable he can, when such circumstances arise, decide which of the possible departures from the original specification are acceptable, possibly by reference to his colleagues. The use of a computer for timetable construction, however well designed for the purpose, inevitably introduces a degree of inflexibility, owing to the limitations of current techniques of man-computer communication.

Cost is clearly an important consideration. In general, the use of a computer for timetabling will involve clearly identifiable additional expense. This is not easy to justify in view of the heavy pressure on educational budgets and the fact that the main direct benefit of using a computer is a saving of time of staff who would be employed anyway. Thus a quantifiable cost has to be set against a largely unquantifiable benefit.

Nevertheless, the author has been told that some local authorities would be prepared to spend up to £200 per school on a computer produced timetable.

1.7 Survey of existing work

In 1968 Folkers published a comprehensive survey of work on computer timetabling up to that time in a report entitled "Timetable Construction by Computer" [15]. His report forms the starting point for this brief survey.

Folkers, while acknowledging the difficulty of giving a clear cut sub-division, identifies three main categories of published work on computer timetabling.

1) Primarily heuristic methods.

2) Methods based on existence conditions.

3) Methods using mathematical models.

The word "heuristic" is widely used in the literature of computer timetabling, and before discussing the first category it is worth a short digression to examine the different ways in which the word has been used. As Norman [32] has pointed out, the word was originally used with the connotation of discovery. It was in this sense that Archimedes used it when he leapt out of his bath on discovering a way to determine whether or not the gold in Heiro's crown had been adulterated. A more recent use of the word is to describe a learning process, so that a child learns heuristically if he learns by experiment or by finding out for himself.

The most widespread use of "heuristic" has the connotation of intuition or reasonableness: it is in

this sense that it is used in the literature on timetabling to describe strategies and procedures for the solution of timetabling problems, the effectiveness of which cannot be proved analytically.

Folkers's first category embraces the predominantly heuristic approaches which all follow the fundamental principle of manual timetable construction. That is, assignments of time periods to lessons of the curriculum are carried out in a sequence which is determined by some heuristic criterion concerning the relative difficulty of such assignments. Several papers falling into this category are mentioned and the work of Berghuis et al [3] and Appleby et al [2] are discussed in more detail as being to some extent representative.

In the second category, based largely on a paper by Gotlieb [16] published in 1962, the emphasis is shifted towards the establishment of conditions for the existence of a solution to the problem as defined by a given set of input data using the techniques of combinatorial analysis. These conditions, derived from a theorem on set representatives by Hall [19], are necessary conditions only; no sufficiency criteria with respect to the existence of solutions to timetabling problems with arbitrary input data have yet been established. Work on the principles established by Gotlieb has been continued notably by Lions [27, 28], who has described the incorporation of a procedure based on these principles in a practical program for constructing timetables. This incorporates an heuristic algorithm

for distributing the curriculum requirements over the days of the week, with the Gotlieb procedure then applied to the construction of each day's timetable. The use of the heuristic algorithm for assigning lessons to days is made necessary by the very heavy demands on computer storage made by the basic procedure; this represents a distinct practical limitation.

In the third category, attempts are made to translate the timetabling problem into a model amenable to solution by existing mathematical techniques. The approaches that have been tried include the representation of the problem in terms of partitioning, notably by Sherman [33], graph colouring (Welsh [34]) and integer programming (Lawrie [23]). The work described in this category is largely theoretical, and up to the time that Folkers's report was published none of this work had reached the stage at which solutions to practical problems could be produced. A difficulty with these approaches is that the timetabling problem when formulated as a mathematical model is apt to be prohibitively large. For the linear programming formulation, for example, Lawrie [24] mentions the need for 10^6 variables to represent a realistic sized problem!

Since the publication of Folkers's report there have been important developments in the field.

1.7.1 Heuristic systems

Johnston's 1970 thesis [22] describes an approach to computer timetabling based on the successive timetabling of events which currently appear to be most

"difficult". If a conflict (glossary 13) is detected, that is, no time can be found for some event e(i,z), a retreat strategy is followed. Events are removed from the existing partial timetable, on a last-in first-out basis, until an event e(j,z') is encountered for which a choice of times for assignment exists. At this point e(j,z') is re-assigned to a time different from that chosen previously and the insertion algorithm proceeds in the hope that the previous conflict in relation to e(i,z) will not recur.

Johnston's main conclusion is that his method by itself is not effective. However, he does point out its potential if embedded in an interactive system of computer timetabling, and this interactive approach, based on a different detailed strategy, is extensively developed in the present work.

A system which is in operation in New Zealand has recently (1972) undergone limited trials in Great Britain with some success, but no account of the principles on which it is based has been published apart from a short summary by Groom in [17].

1.7.2 The work of Michalsen

One of the most important works on computer timetabling yet produced was published in December 1970 by Michalsen [31], entitled 'A working strategy for general school scheduling', the English translation of which became available in July 1972. This gives a detailed description of the principles of the first commercially successful computer timetabling system in

Europe. The system is operated by Nordata Software Ltd., Trondheim and has produced over 100 timetables for schools and colleges in Norway, Sweden and Denmark every year since 1970.

Michalsen's approach is as follows. Suppose that s steps in the allocation process have been completed and the (s+1)th step is about to start. An outstanding requirement ρ_i is selected which has the minimum "degree of freedom", that is, the requirement ρ_i with the fewest available periods on any day on which an event e(i,1) must be assigned. (As far as the internal operation of the system is concerned, all events have a duration of <u>one period</u>. Double period events are represented by two single period events with an indication that they must occur consecutively on the same day. Triple period events are similarly represented.)

If there is more than one requirement with minimum degree of freedom then that which has the maximum number of clashing requirements (glossary 11) is selected for assignment. Other factors such as a user assigned allocation priority may also be used to weight the selection process in favour of certain requirements.

Available periods for an event e(i,1) of requirement ρ_i are ranked according to desirable criteria such as the need for a normal distribution (glossary 14). Each period p of rank 1 is then examined and a "clashing picture matrix" M generated for each, covering all periods q and outstanding requirements ρ_j where M[q,j] = 1 if an event e(j,1) could be allocated at period q now

but not after the assignment of e(i,1) to period p, and 0 otherwise.

This clashing picture matrix is used to update a "freedom picture matrix" FPM for each period p where $FPM_p[q,j] = 1$ if an event of ρ_j can be allocated to period q and 0 otherwise. Each FPM_p is now examined to establish whether or not certain necessary conditions for existence are satisfied. The most important of these is that for every tight set of requirements (glossary 11), the number of periods for which at least one requirement belonging to the set is available (glossary 12) is not less than the total number of outstanding event periods (glossary 10) for all the requirements in the set. These conditions are extensions of the necessary conditions established by Gotlieb [16] for the existence of a solution to a much simpler class of problem.

If no period p of rank 1 satisfies the necessary conditions, then the process is repeated for the periods in rank 2 and so on. If no period can be found in any of the ranks the event e(i,1) is removed from the problem and manual adjustment is necessary. If only one period is satisfactory then this is selected for the assignment of e(i,1). If more than one period p satisfies the necessary conditions, a measure of the degree to which the assignment of e(i,1) to p would cause loss of freedom for future allocations is computed for each such p and the period which minimizes this function is selected for e(i,1).

The cycle described is then repeated for the (s+2)th step unless a forcing situation (glossary 13) exists in which case the necessary forced assignments are carried out before the complete cycle is repeated.

The system is highly successful in its application to the timetabling of Norwegian schools. These schools do not appear to present such a difficult timetabling problem as British grammar and comprehensive schools, for two reasons. Firstly they do not have the same complicated pattern of optional subjects which make many British secondary schools so difficult from a timetabling point of view; and secondly they have a more flexible time structure with an optional period at the beginning and end of each day which may be used in the last resort when timetabling difficulties occur. Despite this there is every reason to believe that the system, or a modified version of it, would be successful in Great Britain and indeed one satisfactory trial has already taken place on a grammar-technical school in South Shields [13].

However Michalsen's method is based on a combination of exact and heuristic principles and is not completely general, particularly in regard to multiple period events, and manual adjustments to the timetable produced are often necessary. The computer program is large, some 15000-20000 FORTRAN statements, and the number of data variables for a school with 20 classes is between ½ and 1 million, although a high proportion of these are logical variables requiring

only single bit storage. Such a program requires a large computer system and approaches making less expensive demands for facilities must be worth investigating.

1.7.3 Mathematical models

In a doctoral thesis De Werra [11] examines a graph theoretic approach to timetabling and describes a program, EHOR, to construct timetables. This program works through the periods of the week in order, selecting the most favourable combination of events for assignment at each period. A network is constructed initially with a node for each class c_i and teacher t_j and an arc representing every meeting of (c_i, t_j) that must occur in the timetable. Every arc is allocated a cost, which is the sum of the "margins" of the individual resources (c_i and t_j) and the (c_i, t_j) requirement. De Werra defines the margin of a resource or a requirement as the difference between the number of periods available and the number still to be assigned to it. That combination of events is selected for each period which has the minimum total cost, by solving a least cost flow problem. As each event is assigned the corresponding cost is set to infinity so that it will not be reassigned at a later period.

Unfortunately, while simple one class - one teacher events can be represented on a network in this way, more complex events cannot and so De Werra has to use a two stage algorithm in EHOR. For each period p, the first stage assigns as many "complex" events as possible and the second selects the optimum combination

of simple events as above. De Werra has been forced into the same strategy as Lions [27, 28]. A theoretically elegant and powerful algorithm has been developed but it can only deal with events capable of being described in terms of this algorithm. Other events must be catered for by some other heuristic process.

Dempster [10] has published a paper on a graph colouring approach to timetabling following Welsh [34] and further papers have been promised.

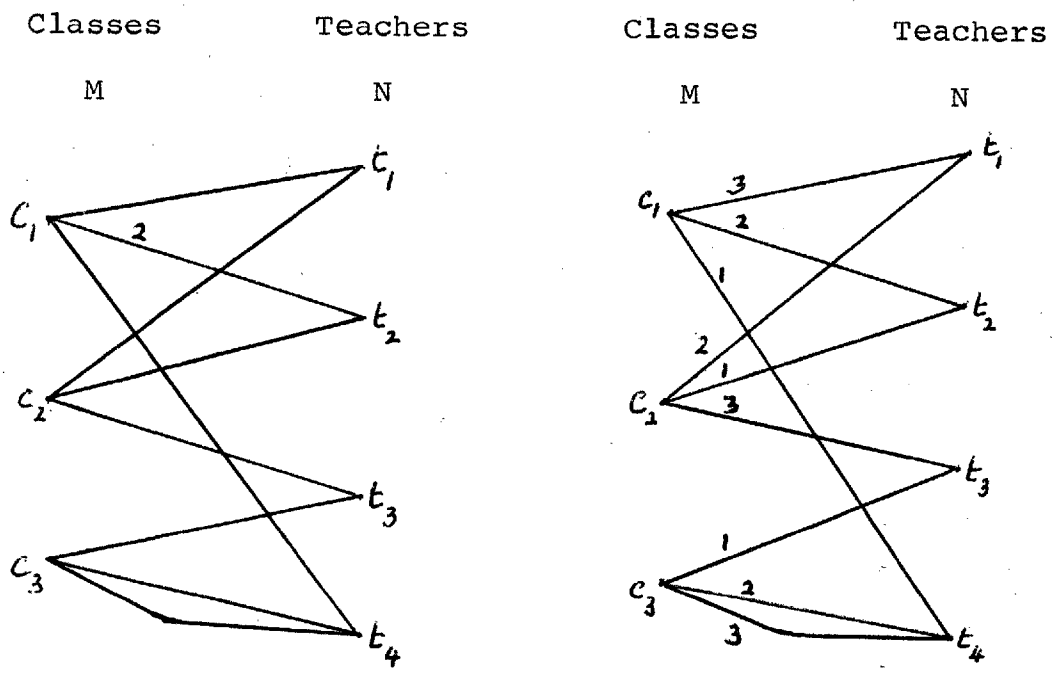
Apart from the question of the size of the problem, already mentioned, a serious drawback of all the more theoretical work in the second and third categories is that the problem has to be idealized in order to make it amenable to solution by the appropriate technique. In his paper Dempster defines such an idealized problem and calls it the "timetable problem". All the more theoretical work in this field is based on the timetable problem or limited extensions of it. It is therefore appropriate to examine the extent to which it models the real timetabling problem in British secondary schools in order to assess whether an approach based on the timetable problem is likely to yield useful results in such schools. A comparison between the timetable problem and the timetabling problem is given in Chapter 2.

CHAPTER 2 - THE "TIMETABLE PROBLEM" AND THE TIMETABLING PROBLEM

This chapter discusses the relationship between the mathematical problem known as the timetable problem [10] and the problem of constructing a timetable in a real school, to be called the timetabling problem. The timetable problem is defined, compared with the timetabling problem and its practical value assessed.

2.1 The timetable problem

The timetable problem may be represented as a bipartite multigraph, see figure 2.1, with m classes c_i ($i \in M=\{1,2,\ldots,m\}$) and n teachers t_j ($j \in N=\{1,2,\ldots,n\}$), forming two disjoint sets of nodes, which are connected by edges wherever the corresponding class and teacher must meet. Multiple meetings give rise to multiple edges. If there are p periods in the timetable the problem consists of finding a label $k, (k=1,2,\ldots,p)$, for every edge subject to the restriction that no two edges incident to a given node have the same label, that is one teacher can meet at most one class at any one period and conversely one class can be taught by at most one teacher at any one period. The basic problem may be modified so that in the initial state certain edges are already labelled, corresponding to preassigned meetings.



a) The problem in its initial state with c_1 and t_2 preassigned to meet at period 2

b) A possible solution to the problem

Figure 2.1

A three period, three class, four teacher timetable problem represented as a bipartite multigraph.

Following Dempster [10] an equivalent definition involving matrices is as follows. The basic timetable problem T(R) concerning m classes c_i ($i \varepsilon M$) and n teachers t_j ($j \varepsilon N$) is specified by an m x n requirements matrix $R = [\rho(i,j)]$ whose (i,j) element is the required number of meetings of class c_i with teacher t_j. Of course R is just the adjacency matrix of the corresponding bipartite multigraph. The minimum number of periods in which a timetable can possibly be implemented is p, where p is the maximum of the row and column sums of R. A solution S to T(R), that is a complete timetable, if

one exists, can be represented as an ordered set of p m x n matrices S_k (ordered by period k ($k\varepsilon P=\{1,2,\ldots,p\}$)). The sum of each row and column of the period matrices S_k is at most 1 so that they are sub-permutation matrices. This expresses the condition specified previously that one teacher can meet at most one class at any one period and vice versa.

This basic problem may be modified by the inclusion of a set of preassignments which may similarly be specified as a set of p sub-permutation matrices S'_k ($k\varepsilon P$) subject to the restriction $\sum_{k=1}^{p} S'_k \leq R$.

Example.

Note: this is the same problem as illustrated in figure 1.

$$R = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 2 \end{bmatrix}$$

$$S'_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad S'_2 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad S'_3 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

A possible solution to the problem defined by the requirements matrix R and the preassignments specified by S'_1, S'_2 and S'_3 is

$$S_1 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad S_2 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad S_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.2 A possible extension

It is possible to extend the scope of the timetable problem by the creation of dummy classes and/or teachers.

For example, suppose a teacher t_j is not available for assignment at a particular period k. A dummy class c_i can be created and the corresponding element $s'_k(i,j)$ of S'_k set to 1, thus making teacher t_j not available to meet any real class at this period.

In a real timetabling situation it is often necessary for a teacher to meet two or more classes together, apparently contravening the single teacher single class rule mentioned in 2.1. This difficulty can be removed with the use of dummy resources, teachers in this case, provided that the corresponding events can be preassigned. Suppose teacher t_{j_1} must meet classes c_{i_1}, c_{i_2} and c_{i_3} at the same period. This can be arranged by creating two dummy teachers t_{j_2} and t_{j_3} and preassigning the following meetings for period k: c_{i_1} meets t_{j_1}, c_{i_2} meets t_{j_2} and c_{i_3} meets t_{j_3}. This will ensure that the three real classes are all available to meet t_{j_1} at period k.

2.3 The timetabling problem

It appears to be impossible to find a rigorous definition of the timetabling problem which is sufficiently general to embrace the special conditions and constraints of every school. However, a definition which covers a broad spectrum of real timetabling problems is as follows.

A set of requirements $R = \{\rho_i\}$, $i=1,2,\ldots,\rho$ is to be implemented over a week of P periods, given a set of resources $Q = \{y_i\}$, $i=1,2,\ldots,Y$. Each requirement ρ_i is specified by its associated resources and

its event pattern, as described below.

Associated resources

With ρ_i is associated a set of resources, $y_{i_1}, y_{i_2}, \ldots, y_{i_n}$.

Each resource may be either unique or selectable (glossary 9).

Event pattern

Requirement ρ_i has a value m(i) (glossary 7) specifying the number of events of each permissible duration to be inserted into the timetable and given by $m(i) = \langle a_{i1}, a_{i2}, \ldots, a_{iZ} \rangle$ where Z is the maximum duration of any event in the problem and a_{ij}, $j = 1,2,\ldots,Z$, is the number of events of duration j for requirement ρ_i.

Let there be associated with R a two dimensional ρ x P timetable matrix $T = [t(i,p)]$ which is updated as timetable construction proceeds and whose (i,p) element can take one of three possible states:

- α if an event of requirement ρ_i has been inserted at period p.
- β if it is still possible to insert an event of requirement ρ_i at period p.
- γ if it is no longer possible to insert an event of requirement ρ_i at period p for any reason.

Also associate with R a two dimensional ρ x ρ clashing matrix $X = [x(i,j)]$ whose (i,j) element is 1 if requirements ρ_i and ρ_j clash (glossary 11) and 0 otherwise.

The initial state of the timetable matrix T reflects the initial conditions of the problem. In particular:

1) If any resource, y_j say, is not available to the timetable at period p then for every requirement ρ_i associated with y_j the appropriate element t(i,p) of the timetable matrix is set to γ.
2) If an event, e(i,z) say, of requirement ρ_i must start at period p then the event is inserted into the timetable so that

$$\left.\begin{aligned} t(i,q) &= \alpha \\ \text{and } t(j,q) &= \gamma \end{aligned}\right\} \quad q = p,p+1,\ldots,p+z-1$$

$$\forall j \text{ such that } x(i,j) = 1.$$

The problem is now to satisfy the requirements specified by R, that is to provide the necessary selectable resources for each requirement ρ_i and to insert every event into the timetable subject to the constraint that an event e(i,z) can only be inserted into the partial timetable starting at period p if $t(i,q) = \beta$, $q = p,p+1,\ldots,p+z-1$, and a number of other constraints of which the following are the most common:

Multiple period events must not start at certain periods; for example, the last period of the day.

Events of a requirement must occur in a normal distribution (glossary 14).

It must be said that the above definition of the timetabling problem does not express the more specialized

and subjective constraints which tend to vary from school to school. It is sufficiently realistic, however, to provide a yardstick against which the timetable problem may be measured.

2.4 The two problems compared

The two problems are compared in table 2.1.

FEATURE	TIMETABLE PROBLEM	TIMETABLING PROBLEM
1. Duration of events	Single period events only	Multi-period events of which the single period event is only a special case
2. Number of resources	Single teacher-single class events except preassignments using dummy resources	Multi-teacher, multi-class, multi-room events in many cases
3. Types of resources	Two resource types, classes and teachers only	At least three resource types: classes, teachers and rooms
4. Selectable resources	No selectable resources - all must be unique	The problem has both selectable and unique resources. Many rooms are selectable resources in practice
5. Initial conditions	Initial non-availability of resources and preassignment of events is possible	As for timetable problem
6. Constraints	Basic timetabling constraint only	Two additional constraints listed in 2.3. In any actual problem there would probably be others

Table 2.1. The timetable and timetabling problems compared.

The shortcomings of the timetable problem as a description of the real situation in a traditionally structured, class based school are not so serious as might be thought from table 2.1. In such a school the great majority of events, perhaps 95%, would be satisfied by the timetable problem in respect of features 1, 2 and 5. The shortcoming in respect of feature 3, lack of provision for rooms, is unlikely to be important, and the absence of selectable resources, feature 4, will almost certainly not be serious. This leaves the question of additional constraints such as the distribution of events over the week, and this can be circumvented as in the Ontario System [27] by some heuristic mechanism to split the problem into days so that an algorithm to solve the timetable problem can treat each day separately. Some means has still to be found for dealing with the comparatively small number of events which do not conform to the simple structure of the timetable problem. Lions has a supplementary algorithm which deals with the assignment of such events and he splits requirements up into two groups, normal requirements which satisfy the timetable problem and special requirements which do not.

Clearly, however, the timetable problem becomes steadily less relevant to the real situation as the structure of the school becomes more complex. In the County Grammar School for Girls at Faringdon, for example, there were 467 events in one timetable of which only 157 satisfied the simple single class, single teacher,

single period constraint of the timetable problem. In Lions's terms there were 157 normal events and 310 special events: thus in such a school the "normal" event is itself unusual.

While the majority of British primary schools and the older secondary modern schools conform reasonably closely to the simple structure of the timetable problem, the situation in grammar and comprehensive schools has diverged increasingly from it over the last decade, and it is these schools with their more complex structure which present the most difficult timetabling problems.

It must be concluded that the timetable problem is not relevant to those real timetabling problems where a computer might be expected to be most valuable. Therefore, the approaches based primarily on the timetable problem have not been pursued, and the heuristic approach, which can be tailored to cope with the timetabling problem as described in this chapter as well as a wide range of variations, has formed the basis of this work.

CHAPTER 3 - STORAGE STRUCTURES

3.1 Data storage and computer timetabling

Much of the literature on computer timetabling centres on the question of solution algorithms. While clearly this is important, the closely related question of how the data for the problem is stored is almost equally so, but has received considerably less emphasis. Certain types of storage structure facilitate one algorithmic approach but make others difficult or impossible; others impose severe restrictions on the size of problem that can be tackled. It is believed that the importance of storage structure in the field of computer timetabling has been underestimated and that an evaluation of the possible methods of data storage is overdue. Cartesian storage structures are pre-eminently suitable for the functions required in the heuristic approach to computer timetabling and emphasis is placed on these structures in the following discussion.

3.2 Essential features

The storage structure must be capable of storing the specification of the problem including the association between requirements and resources, times when certain types of event may or may not start, preassigned events and the initial availability or non availability of resources. Provision must also be made for recording the following information, either explicitly or implicitly, during timetable construction:

1) The assignment of each event to a specific time,

that is to one or more consecutive periods.

2) The possible times for assigning events not yet in the timetable. Lions [28] calls this information the opportunity list.

3) The selectable resources which have been associated with each event. For the purposes of this chapter it will be assumed that only rooms can be selectable resources.

3.3 Desirable features

In addition to the essential features described in 3.2, there are a number of desirable features in a storage structure, as follows.

1) Economy of storage. It is important that the storage structure is economical in storage, particularly those parts of it which are frequently accessed or updated during the construction process and which should therefore be held in core store. It may be possible to overlay some of this data from direct access backing store, but this leads to significant and sometimes unacceptable increases in running time.

2) Simplicity. The storage structure should be simple to understand to facilitate system development and modification.

3) Rapid access and updating. The time taken to construct a timetable for a given problem depends on the number and duration of the operations which have to be carried out. To reduce this number, and hence the time, is clearly a desirable aim.

In certain parts of the construction process the algorithm used and particularly the interaction between the algorithm and the storage structure determine the number of operations of various types that have to be performed for any given problem. In these areas there can be no absolute comparison between data storage methods since a comparison is only valid in relation to a particular algorithm. However, in certain parts of the construction process, notably the insertion of an event into the timetable at a given period, the operations depend entirely on the storage structure and so a valid comparison can be made.

4) Flexibility. The storage structure should be capable of describing as many different variations of the timetabling problem and as many different types of constraint as possible.

5) Transparency. The storage structure should not artificially obscure the characteristics of, and/or the possibilities inherent in, the real problem. For example, suppose an event e(i,z) involves the meeting of class c_1 and teacher t_1. If the storage structure stores the activities of c_1 and t_1 explicitly, an algorithm could, as a last resort, allocate another teacher, t_2 say, to this event if the (c_1,t_1) meeting was found to be impossible to arrange. If, however, the link between e(i,z), c_1 and t_1 is stored implicitly, as in one of the storage structures to be discussed, this possibility does not exist.

3.4 Possible storage structures

It is not practicable to examine all the possible ways of storing all the data for a computer timetabling problem. In this discussion attention is focussed on the parts of the storage structure which are required to store the information mentioned in 3.2 which must be recorded as the construction process proceeds. Four methods of data storage representative of the various possible approaches will be examined, to be known for the remainder of the thesis as storage structures 1 to 4. The first two are based on resources, one holding the class timetable explicitly plus the availability of each resource, and the other, derived from Gotlieb's approach to the timetable problem, stores the availability of each resource and the scheduled meetings of combinations of resources in a single large array corresponding to the matrix of 2.1. In the other two methods the emphasis is transferred from resources to requirements, to a differing extent. Storage structure 3 represents a half way stage in which the timetable is held by requirement, but resource availabilities are also stored explicitly, whereas storage structure 4 holds the timetable, together with the implicit availability of resources, by requirement as a single array.

3.4.1 Storage structure 1 - the class, period array

Variations of this storage structure have been used extensively in past work, for example by Barraclough [4]. The timetable is stored in a two

dimensional CxP array T, where T[j,p] has two items namely the teacher and the room for class c_j at period p. A two-dimensional availability array A is also required, where A[j,p] gives the availability of resource y_j at period p.

3.4.2 Storage structure 2 - the four-dimensional array

In this method, adopted in principle by Gotlieb [16], the timetable is held as a four dimensional C x T x R x P array V, where V[j,k,m,p] is set to 1 when class c_j and teacher t_k meet in room r_m at period p. In fact the demands of this structure on storage are so great that no one has suggested using it exactly as described. The Ontario System [27], based on Gotlieb's work, did not include rooms in the main array and the storage required was further reduced by only considering a single day at a time. Nevertheless, the full structure is considered here in order to make a direct comparison with other storage structures possible.

3.4.3 Storage structure 3 - the requirement list

In this structure, a variant of which was first suggested by Yule [36], the timetable is held as a list of two-item elements giving the period and rooms allocated to each event of every requirement. The most effective way of implementing this list is to hold the timetable as a vector T, with Q, the total number of event periods in the timetable, as an upper bound for the number of elements, and a second vector V, holding for each requirement ρ_i, the element in T at which the events of ρ_i start.

Example

Element no.	1	2	3	4	5					
Vector T	5	27	11	27	36	29	3	13	14	21

Element no.	1	2	3	4
Vector V	1	4	5	

In the example, requirement ρ_1 has three single period events, allocated to period 5 in room 27, period 11 in room 27 and period 36 in room 29. Requirement ρ_2 starts at element 4 of vector T, so that element 2 of vector V is set to 4. In the initial state most of T will be zero indicating no allocations but V can be set up since the number of event periods for each requirement is known.

A resource availability array A is also required as for storage structure 1.

3.4.4 Storage structure 4 - the requirement, period array

The essential feature of this approach, the use of a single array to store the timetable, the opportunity list and (implicitly) resource availability is original, although Johnston used a similarly structured array for a different purpose [22]. Michalsen [31] uses a broadly similar structure with two Boolean arrays, one for the timetable and the other for the opportunity list. The timetable is held as a two dimensional ρ x P array T, where T[i,p] is coded in accordance with the following general principles:

T[i,p] = 100000 - an event of requirement ρ_i may be allocated at period p.

$|T[i,p]| \leq 1000$ - an event of ρ_i has been allocated at p.

The magnitude of T[i,p] may specify the room(s) in which the event is to take place.

Other values of T[i,p] indicate that an event of ρ_i cannot be allocated at p for any reason, for example the allocation of an event of ρ_j at p where ρ_j clashes with ρ_i.

A negative value of T[i,p] indicates that the condition indicated by the magnitude of T[i,p] cannot be altered, while a positive value means that it can.

The availability of classes and teachers, assumed always to be unique resources, is not stored explicitly, nor is the link between these resources and requirements. Instead, these links are stored implicitly by means of a clashing array which holds, for each requirement, those requirements which clash with it. Since rooms may be selectable resources, room availability is stored explicitly as a two dimensional R x P array A, where A[m,p] holds the availability of room r_m at period p in the same way as storage structures 1 and 3 store the availability of all resources.

Multiple rooms in storage structures 1, 3 and 4 can be accessed by a pointer, in the appropriate elements of the timetable array or vector, to a list where the individual rooms are stored.

3.5 Schools used as examples

Data from six establishments, five schools and the Royal Military College of Science, has been used to

illustrate the implications of the four storage structures in terms of storage requirements and the number of operations needed to carry out two frequently performed functions. The six sets of data have been chosen to cover most of the characteristics likely to be found in any timetabling problem, and are briefly described below. Details of the six establishments are given in table 3.1.

No.9 School of Technical Training,R.A.F. Newton,Notts. This school, known hereafter as RAF Newton, is a good example of a short course establishment (see 1.2.5). Courses vary in length from 3 to 46 weeks and the timetable changes every week, so that any set of data used applies to one week only. Unusual elements in the problem for this school include a high proportion of quadruple period events and a relatively large number of event periods per requirement. Spreading events of a requirement evenly over the week is regarded as less important in this school than in most secondary schools.

Aylesbury Grammar School, Bucks

Burnage High School, Manchester

Faringdon Grammar School, Berks (to be known as The Elms) These three schools are secondary schools. Aylesbury is a typical small to medium sized grammar school for boys with about 500 pupils, while the Elms is a school of similar type and size for girls. Burnage is a comparatively large boys school with over 1,200 pupils. The data for Aylesbury and Burnage was

originally prepared for a trial of another computer timetabling system in 1970 and was supplied by the Local Government Operational Research Unit. The data for the Elms was obtained directly from the Headmistress. An important difference between the data for Aylesbury and Burnage on the one hand and that for the Elms on the other is that the VIth form in the former schools is divided into a comparatively small number of classes for time-tabling purposes, 3 for Aylesbury and 5 for Burnage, with most VIth form requirements involving several teachers, indicating provision for several optional subjects to be timetabled simultaneously. The Elms data, by contrast, reflects the traditionally less constrained pattern of choices open to girls in the VIth form and each girl has been treated as a separate class, so that the VIth form accounts for 77 of the 89 classes in the school.

The Royal Military College of Science, Shrivenham (to be known as RMCS)

The data for the RMCS includes the degree courses only so that it indicates an artificially low loading on teachers and rooms. Considering the comparatively small number of students, an unusually large range of alternative options is available in second and third year courses.

The High School, The American School in London (to be known as the American School)

This school is an example of an American High School,

being entirely pupil orientated with no two pupils necessarily taking the same choice of subjects. The problem of timetabling in such a school is quite different from that in British schools and the set of data for the American school has only been included here as an example of how the different storage structures cope with pupil orientation, a trend in British secondary schools, carried to its limit.

School / Quantity	Symbol	RAF Newton	Aylesbury GS	Burnage HS	The Elms	RMCS	American School
Pupils	None	400	500	1240	450	280	560
Classes	C	41	18	43	89	52	560
Teachers	T	60	39	71	29	79	60
Rooms	R	80	16	62	25	29	44
Resources	Y	181	73	176	143	160	664
Requirements	ρ	262	281	424	215	100	200
Events	E	641	476	946	467	285	716
Periods per week	P	40	40	35	38	20	100
Event periods	Q	1523	550	1103	587	290	2000
Teacher periods	B	1389	950	1894	734	440	2271
Averages							
Classes/ requirement	$\bar{c}(\rho)$	1.18	1.21	1.29	3.75	3.41	21.00
Teachers/ requirement	$\bar{t}(\rho)$	1.06	2.25	1.64	1.20	1.25	1.06
Rooms/ requirement	$\bar{r}(\rho)$	1.06	2.20	1.60	1.10	1.25	1.04
Resources/ requirement	$\bar{y}(\rho)$	3.30	5.66	4.53	6.05	5.91	23.10
Clashing requirement	$\bar{x}$	13.26	55.53	29.33	32.30	12.02	68.41

Table 3.1 Details of the six schools used as examples

3.6 Quantitative comparisons

A comparison between the four storage structures applied to the six example schools is given in table 3.2. For each storage structure the following quantities have been evaluated:

1) The storage required in 24-bit computer words.
2) The average number of operations needed to establish whether or not it would be possible to insert a single period event e(i,1) of a typical requirement ρ_i at any period p.
3) The average number of operations required to insert a single period event e(i,1) into the timetable.

The derivation of the formulae is given in Appendix 1. It must be emphasized that the figures given for storage are for comparison only and do not give the complete needs for data storage in a computer timetabling system using any one of the four storage structures discussed in 3.4. Any system needs to store the links between requirements and resources, the periods at which multiple period events may start, and many other items of data for which the figures in table 3.2 do not allow. Nevertheless, the figures given do highlight the main differences in the storage required by the various structures considered.

Two assumptions are made.

1) It must be possible to represent the initial non-availability of resources at two levels, a "preferred" level when the non-availability can be overridden in the last resort and a "definite" level when it cannot. This implies that each element of an array storing the

availability of resources at each period of the week must be capable of existing in at least three states: available, preferably not available and definitely not available.

2) Classes and teachers must always be uniquely associated with requirements. Rooms may be selectable resources and so the storage structure must be capable of storing the room(s) associated with each event.

While the storage figures relate to 24-bit computer words, the figures for the number of operations required to perform a given function do not bear any direct relationship to the number of machine instructions necessary to perform the function since this number can vary widely depending on the machine and programming language being used.

3.7 Discussion

From table 3.2 it would appear that storage structure 2 is not a practical proposition. It can be claimed, however, that by including rooms as an additional dimension to the main array, and by requiring two bits per element in order to conform with the assumption regarding availability states mentioned in 3.6, a fair evaluation of this storage structure is not being given. This is undoubtedly true to some extent, although every structure is being treated exactly alike with regard to problem specifications and assumptions. If rooms are removed and one bit per element is allowed for this structure, the number of words required to store the Burnage data would be reduced from $567{,}858_8$ to 6,106,

Table 3.2 Summary of storage and processing requirements for the four storage structures applied to six example schools

Feature	Storage structure	Formula	RAF Newton	Aylesbury	Burnage	Elms	RMCS	American School
Storage (24 bit words)	1	$2CP+\left\lceil\frac{Y}{12}\right\rceil P$	3920	1720	3535	7220	2360	117,600
	2	$CTR\left\lceil\frac{P}{12}\right\rceil$	787,200	44,928	567,858	258,100	238,264	13,305,600
	3	$2Q+\rho+\left\lceil\frac{Y}{12}\right\rceil P$	3948	1661	3205	1831	992	9800
	4	$\rho P+\left\lceil\frac{R}{12}\right\rceil P$	10,760	11,320	15,050	8284	2060	20,400
Average number of operations to establish whether e(i,1) can be inserted at any period p	1	$\bar{y}(\rho)-1$	2.30	4.66	3.53	5.05	4.91	22.10
	2	$\bar{c}(\rho)TR+\bar{t}(\rho)CR+\bar{r}(\rho)CT$	11,491.0	2525.0	14,910.0	8221.0	14,835.0	116,400.0
	3	$\bar{y}(\rho)-1$	2.30	4.66	3.53	5.05	4.91	22.10
	4	1	1.0	1.0	1.0	1.0	1.0	1.0
Average number of operations to insert an event e(i,1)	1	$\bar{y}(\rho)+2\bar{c}(\rho)$	5.66	8.08	7.11	9.80	9.32	44.10
	2	$\bar{c}(\rho)\ \bar{t}(\rho)\ \bar{r}(\rho)$	1.32	5.99	3.38	11.00	11.29	558.10
	3	$\bar{y}(\rho)+\frac{Q}{2\rho}+2$	8.21	8.64	7.83	9.40	9.44	30.10
	4	$1+\bar{x}+\bar{r}(\rho)$	15.32	58.73	31.93	33.80	14.41	70.45

Note: Meanings are as in table 3.1.

well within the bounds of practicability. On the other hand, the process of establishing whether or not a single period event can be inserted at any period would still require 162 operations, considerably more than any other storage structure.

The structures based on requirements improve compared to those based on resources as the degree to which the data is pupil orientated increases, particularly as regards economy of storage. Table 3.3 illustrates this fact rather more clearly than table 3.2 by normalizing the storage required for structures 1, 3 and 4 for each school in terms of that required for storage structure 3. The parameter $\bar{c}(\rho)$, the number of classes per requirement, is also given as a fair indication of pupil orientation.

Table 3.3 also shows clearly that storage structure 3 is consistently more economical in storage than the other storage structures for the six widely differing sets of data, with one very minor exception. It also compares well with the other structures in the number of operations to insert an event.

It may be concluded from this discussion that, out of the four storage structures examined, no one stands out as being the best in all respects.

Feautre	Storage structure	RAF Newton	Aylesbury	Burnage	Elms	RMCS	American School
Classes/ requirement	-	1.18	1.21	1.29	3.75	3.41	21.00
Storage required (normalized)	1	0.99	1.03	1.10	3.92	2.38	11.92
	3	1.00	1.00	1.00	1.00	1.00	1.00
	4	2.71	6.82	4.69	4.49	2.08	2.08

Table 3.3 Storage and pupil orientation

If economy of storage is of overriding importance then storage structure 3 is clearly to be preferred. If a computer timetabling system is to be expected to deal with a wide range of different sets of data, including heavily pupil oriented data, then storage structures 3 and 4 have a significant advantage over the resource based structures. Storage structure 4 is markedly superior to its competitors in the determination of whether or not an event is available for insertion at any period. The significance of this consideration depends on the algorithm to be used; it is of overriding importance in the displacement algorithm to be discussed in chapter 6.

CHAPTER 4 - ALGORITHMS I - EXISTENCE AND EVENT SELECTION

This chapter and the two following are concerned with different aspects of the process of timetable construction from a theoretical point of view. The practical implementation and assessment of certain of the points discussed is described in Chapters 7, 8, 9 and 10.

4.1 Processes

A simplified view of the problem leads one to propose that ideally there need only be three discrete processes involved in the construction of a timetable: a test for the existence of a solution; problem modification; and time and resource selection. The first step is then to establish whether or not the problem as specified by the set of data has one or more solutions. If it does not, the problem must be modified in some way, either by altering the data or the constraints or both, until a new problem is produced to which a solution does exist. Once the existence of such a solution has been established the construction process starts. This consists of selecting a time for every event in such a way that each selection is compatible with a solution to the problem.

Unfortunately the real timetabling problem is so complex that no way of showing that a solution exists has yet been discovered, and all that has been done so far in this field is to establish a number of necessary conditions for existence. Thus it is often possible to demonstrate that a given problem has no solution; the

only way so far devised to show that a problem has a solution is to find one, that is to construct a timetable. In general, therefore, it is not possible to ensure that the selection of a set of periods for any event is compatible with a solution, and the simplified view of the problem described above is unrealistic.

Nearly all computer systems which have actually produced usable results perform some tests to ensure that certain necessary conditions for existence are satisfied and then carry out the following timetable construction processes repetitively until a solution to the original problem, or some modification of it, is achieved.

1) Event selection. If there are no outstanding events remaining the construction is complete, otherwise select an event e(i,z) for assignment and proceed to 2).
2) Time selection. Attempt to find a set of z consecutive periods {p,p+1,...,p+z-1} for e(i,z). If this is successful insert the event at this set of periods and return to 1); if no suitable period set can be found then a conflict (glossary 13) exists, so proceed to 3).
3) Conflict resolution. Attempt to resolve the conflict by some modification of the partial timetable so as to make one suitable set of periods available for the insertion of e(i,z). If this can be done return to 2) for the event to be inserted otherwise proceed to 4).
4) Problem modification. Modify the problem in some

way, for example by omitting e(i,z) altogether and return to 1).

Process 2) may include the selection of selectable resources for e(i,z) but more often such resources, usually non-specialist rooms, are selected after the main construction process is completed.

This set of processes characterizes the heuristic approach to computer timetabling. However, different systems place different degrees of emphasis on the various processes. Clearly, the more efficient process 2), the less important are the other processes. Lions [28] and Michalsen [31] do not attempt conflict resolution at all on the grounds that the earlier selection processes are sufficiently good as to make it likely that if they fail a solution to the original problem does not exist and it must therefore be modified. Event selection is examined in detail later in this chapter and the other three processes in the following two chapters.

4.2 Existence

The main difficulty over the question of existence in the timetabling problem has already been mentioned in 4.1. A number of conditions necessary for a solution can readily be established; the most obvious of these is that no resource must be timetabled for more periods than it is available. Once these conditions are satisfied, however, there remains in all practical problems a large area of uncertainty as to whether or not a solution to the given problem exists. This

uncertainty can only be resolved completely either by proceeding to find a solution or by exhausting all possible combinations of allocations of times to events, a totally impracticable operation involving many millions of operations for even a relatively small problem. The purpose of this section is to examine ways of reducing this area of uncertainty.

Gotlieb [16] has established a set of necessary conditions for the timetable problem which may be summarised as follows. For all teachers t_j and classes c_i, the total number of times at which at least one of teacher t_j's classes is available must not be less than the total number of times that t_j is required to meet these classes, and the total number of times that at least one of c_i's teachers is available must not be less than the total number of times that c_i is required to meet these teachers. These conditions must not only be satisfied by the problem as initially specified but must also be satisfied at all stages of timetable construction, and can therefore be used to keep a continuous check on whether a conflict has been generated during the construction process. They have formed the basis of much theoretical and experimental work, notably by Duncan [12] and Lazak [25] and can be extended to the timetabling problem as follows.

For every resource y_k the number of periods at which at least one of the events of those requirements associated with y_k is available must not be less than the number of periods for which y_k is to be scheduled. This condition may also be formulated mathematically.

For each resource y_k, $k = 1,2,\ldots,Y$ let H_k be the set of requirements associated with y_k. Then

$$\sum_{p=1}^{P} \emptyset(p) \geq \sum_{\rho_i \varepsilon H_k} \sum_{j=1}^{Z} j.a_{ij} \qquad (4.1)$$

where $\emptyset(p) = 1$ if $T[i,p] = \beta$ for at least one requirement $\rho_i \varepsilon H_k$, and 0 otherwise,

$T[i,p] = \beta$ means that it is possible to insert an event of ρ_i at period p, and

a_{ij} gives the number of events of duration j to be scheduled for requirement ρ_i.

This condition may be further extended by considering the general case of tight sets of requirements (glossary 11). The most obvious example of a tight set is the set of requirements associated with any one given resource, but this is not the only way in which tight sets can arise. For suppose that y_1 is associated with ρ_1, ρ_2, ρ_3, ρ_4, ρ_5, y_2 is associated with ρ_1, ρ_2, ρ_3, ρ_4, ρ_6, and y_3 is associated with ρ_5, ρ_6. Then the set of requirements $\{\rho_1, \rho_2, \rho_3, \rho_4, \rho_5, \rho_6\}$ is tight although no one resource is associated with all the members of this set.

Now inequality (4.1) must be satisfied for every tight set of requirements whether or not it has been generated by a single resource, since the only significant property of the set H_k in this context is that no event of one member of the set can occur simultaneously with an event of any other member. In this respect all tight sets, however generated, are identical.

It is convenient to regard tight sets, which have

not been generated by a single resource as arising from "dummy resources". Then if there are D such dummy resources the inequality (4.1) applies across all real and dummy resources y_k, k = 1,2,...,Y+D. In general D will be a much larger number than Y since it includes the dummy resources for tight sets which are subsets of other tight sets including those generated by real resources. At first sight this appears unnecessary but it is possible for inequality (4.1) to be satisfied for a set of requirements H_k but not for a subset of H_k. For example, suppose that in a three period timetabling problem there is a tight set $H_k = \{\rho_1, \rho_2, \rho_3\}$, where the requirements ρ_i, i = 1,2,3, each have one single period event. If $T[1,p] = \beta$, (see 2.3), p = 1,2,3, then inequality (4.1) is satisfied in respect of H_k whatever the state of the timetable for ρ_2 and ρ_3. If $T[j,p] = \gamma$, j = 2,3, p = 2,3, then the inequality is not satisfied for the subset $\{\rho_2, \rho_3\}$ of H_k and no solution to the problem exists. In practice, however, tight sets which are subsets of other tight sets are very seldom important.

The test for existence specified by inequality (4.1) and amplified subsequently will be known as the extended Gotlieb test. It forms the basis for practical investigations into the existence question and related problems described in Chapter 9.

4.3 Event selection

Given a set of outstanding requirements $R(\rho)$ the problem is to select an event, e(i,z) say, from one of the requirements ρ_i for insertion in the timetable.

The selection of an event for insertion and the selection of a time for the event are closely related processes. Both are apparently easy at the start of the construction process when the timetable itself is comparatively empty and a wide choice exists with seemingly little to choose between many possible courses of action. Yet the early selections set the pattern for all future progress and an inappropriate pattern of assignments in the initial stages of construction can have very adverse effects on the timetable which will only manifest themselves later in the construction process.

The sole objective of event selection is to assist the fundamental operation of time selection, for which no perfect algorithm has been found. This objective can be achieved by selecting "difficult" events before "easy" events so that when times have to be selected for the easy events the algorithms concerned cannot cause "difficult" events to become impossible.

How difficulty should be measured is, unfortunately, by no means clear and there are many possible criteria. Two approaches used in manual timetabling may be mentioned. In the Elms problem the events requiring laboratories and other specialized rooms are timetabled first since there is great pressure on these resources, some of which are in use for every period in the week.

Lewis [26], by contrast, suggests that the VIth form should be timetabled before the rest of the school because the number of options open to VIth form pupils

make the timetable for this form particularly difficult to construct. In general, then, human beings are guided by their own experience of which events are likely to prove difficult in their own particular situation. Computer timetabling programs, on the other hand, must use more formalized and explicit rules for event selection.

4.4 Event selection criteria used in computer timetabling

A number of different criteria have been used for event selection by various workers. Those used by Michalsen [31] and Johnston [22] are briefly discussed here as they are the most complex and powerful yet developed.

Michalsen's degree of freedom function has already been mentioned in 1.7.2. His basic rule is to select the requirement ρ_i having the smallest degree of freedom, that is, the requirement ρ_i with the fewest available periods on any day d to which an event of ρ_i must be assigned. Michalsen modifies this basic rule by ranking requirements on two criteria. The first rank contains those requirements involved in a forcing situation (glossary 13). The second rank contains those requirements specified by the user as having allocation priority. Michalsen suggests that this priority should be accorded to the requirements likely to be most difficult to schedule manually, such as those associated with more than 4 teachers and classes, and requirements with multiple period events. His complete rule involves a three stage selection process: if there are any requirements in the first rank then that with the minimum

degree of freedom is selected; if not the selection is made from those in the second rank by the same criterion. If there are no requirements in either rank the selection is made from the remaining outstanding requirements on the basis of degree of freedom as before. In all circumstances if two or more requirements have the same minimal degree of freedom then that with the most requirements clashing with it is selected.

Johnston, like Michalsen, accords top priority to requirements involved in a forcing situation. If there are no such requirements then a requirement is selected which is involved in a situation that defines all but one of a given set of assignments. If there are no requirements in this second category then a priority parameter is calculated for all outstanding requirements and that with the highest value is selected. This parameter, for each requirement ρ_i, is a function of: teacher and room balance, each a function of the difference between the number of classes associated with ρ_i and the number of teachers and rooms associated with this requirement respectively; the number of resources associated with ρ_i; the loading on these resources; and the duration of the events comprising ρ_i (in Johnston's system every event of a requirement is of equal duration).

4.5 Current and inherent difficulty

The foregoing discussion shows that the criteria used to measure difficulty fall into two main groups, current difficulty and inherent difficulty. Current difficulty is a dynamic measure which, for any requirement,

changes as construction proceeds. Michalsen's degree of freedom function is a measure of current difficulty. Inherent difficulty is a static measure defined by the specification of the problem; the number of requirements clashing with one requirement is an example of an inherent difficulty criterion.

4.5.1 Measures of current difficulty

The work so far carried out on the question of event selection leaves two potentially fruitful areas unexplored: the best way of measuring current difficulty, and the relative importance of current and inherent difficulty. One measure of current difficulty which has not been tried before involves the trial assignment of every event of each outstanding requirement using the same algorithm as for the actual process of assigning the selected requirement into the timetable. After each trial assignment a measure of its difficulty can be derived. One such measure is the slack s_i for each requirement ρ_i. This is the number of blank periods remaining in the timetable for the requirement after the trial assignment is complete. If the trial assignment fails, this indicates that a conflict exists and that the partial timetable must be modified if all the events of ρ_i are to be inserted. This situation is indicated by a negative value of s_i of magnitude equal to the total number of event periods unsatisfied. Storage structure 4, using a requirement-period array for the timetable, is particularly well suited to the trial assignment process.

Slack is similar to Michalsen's degree of freedom. The essential difference is that slack is a more direct measure of the difficulty which would actually be experienced in assigning each requirement. It takes into account all the constraints to which the assignment algorithm will be subject, such as the need to cater for tied requirements (see 5.2).

A disadvantage of slack is that it can provide a misleading impression of current difficulty under some circumstances. For example, two requirements each with one triple period event outstanding might both have a slack of +3. But this is misleading if in one case the slack has been generated by three consecutive periods suitable for the insertion of a triple period event and in the other by three isolated single periods. This shortcoming leads to the concept of "choice". Suppose that the trial assignment of a requirement ρ_i has been successful, and the durations of those events involved in this process are $z_1, z_2, \ldots, z_m$ in descending order. The choice g_{iv} for duration z_v, $v = 1,2,\ldots,m$, is the number of <u>additional</u> events of duration z_v which could be inserted after the trial assignment of ρ_i. Additional events of duration z_{v+1} are only considered after the trial insertion of as many events of duration z_v as can be accommodated. The choice G_i for requirement ρ_i is given by

$$G_i = \sum_{v=1}^{m} g_{iv} \cdot z_v \cdot \qquad (4.2)$$

Example

Suppose that, at the start of a trial assignment, a requirement ρ_i has 1 triple and 2 single period events to be fitted in a problem with a 3 day, 15 period week. The state of the notional timetable for this requirement after a trial assignment is given below.

Period:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	X	X	1	1	1		1	X	X	X	X	1		X	

Note: X - period not available due to an event of a clashing requirement,

blank - period available,

1 - event "inserted" by the trial assignment algorithm at this period.

In this example,

slack $s_i = 3$

choice for duration 3, $g_{i3} = 0$

choice for duration 1, $g_{i1} = 3$

choice for requirement ρ_i, $G_i = 3$,

If the trial assignment fails g_{iz} is the number of events of duration z not satisfied in the trial assignment process, while G_i is the total number of event periods likewise unsatisfied.

In fact, the choice G_i for requirement ρ_i will always be identical to the slack s_i if ρ_i involves any single period events, and the choice g_{iz} for the longest outstanding event is therefore likely to be more valuable as an indicator of current difficulty. The effectiveness of slack and choice as criteria of current difficulty are investigated in chapter 9.

4.5.2 Current and inherent difficulty compared

It would seem reasonable to suppose that the relative importance of current difficulty would increase as the construction process proceeds, since once a forcing situation has been encountered failure to follow the course of action indicated will lead immediately to a conflict. Yet this course of action may be quite different from that indicated by inherent difficulty which is really irrelevant in such a situation. On the other hand one would expect current difficulty to be decidedly suspect as a selection parameter at the start of construction when no values of slack were less than, say, P/4, where P is the number of periods in the week. One would expect inherent difficulty to come into its own at this stage of construction but the best criteria of inherent difficulty is far from clear. This question is investigated in chapter 9.

4.6 Other event selection criteria

While difficulty has been very widely used for event selection, there are two criteria that should be mentioned which relate to difficulty only indirectly. The first is a user specified indication of the importance of a requirement. Clearly important requirements should, other things being equal, be selected before unimportant requirements (Michalsen [31]). The second is the degree to which a requirement can be altered, for example in the breakdown of a multiple period event (Lions [28]).

4.7 Frequency of selection

The strategy used in most of the work discussed

has been to assign all events of the selected requirement before further selection is carried out. This is not necessarily the best strategy. In a sense the requirement is an artificial concept, being a grouping of similar events for convenience. In Michalsen's system the general rule is that selection is carried out before every event is assigned. This may be unnecessarily costly in computing time and a compromise strategy is to select a requirement, ρ_i say, and then find the duration z of the longest event(s) of ρ_i and assign all events e(i,z) of that duration. When this assignment is complete a further selection is made and the process repeated.

CHAPTER 5 - ALGORITHMS II - TIME AND RESOURCE SELECTION

5.1 Interaction between time selection and resource selection

There are two resource selection processes in any timetabling problem: allocation of unique resources to requirements, which is part of curriculum preparation and is outside the scope of this work, and allocation of selectable resources to requirements and events. There are three possible approaches to the latter process in relation to time selection, that is, the allocation of periods to events.

1) Time selection takes no account of selectable resources and resource selection is a completely separate process carried out when the allocation of times to events is completed.
2) Resource selection proceeds in parallel with time selection. When an event is selected, a set of periods and, when necessary, an appropriate set of selectable resources are allocated to it before another event is considered.
3) Time selection ensures that enough selectable resources of the required resource groups are available whenever a period is considered for an event. Actual selection of resources is then carried out when time selection is completed, as in 1) above.

In manual timetabling the first method is usually adopted. Selectable resources are normally limited to non-specialist classrooms, and in most schools a satisfactory class-teacher-specialist room timetable

will, by the constraints governing its construction, ensure that not too many subjects requiring non-specialist classrooms occur simultaneously. Furthermore the sheer labour of keeping track of selectable as well as unique resources militates against the use of the second or third method. The first method, however, clearly involves the risk that when the main timetable is completed it will, at some periods, demand more resources of one or more groups than are available. Considerable effort is often necessary to alter the main timetable when such circumstances arise.

Approaches 2) and 3) are both feasible for computer timetabling. Resource selection in parallel with time selection has the relatively minor advantage that it caters for requirements for which the same selectable resource must be allocated to every event. Under certain circumstances the third approach fails to deal adequately with this constraint, a point discussed further in 5.7. This advantage of the second approach is more than outweighed by the fact that approach 3) makes the main timetabling process simpler, faster and less demanding in storage. Approach 3) is therefore adopted for the discussion of time and resource selection in this chapter.

5.2 Steps in time selection

Time selection may be regarded as a step by step process in which the number of periods remaining as candidates in the selection is reduced at each step.

Suppose that a set of z contiguous periods is being sought for an event e(i,z). The first step is to

eliminate from consideration those periods which are clearly unacceptable due to the properties of e(i,z) in relation to the current partial timetable. These include all periods for which any of the unique resources, or enough selectable resources of the specified groups, are not available and, in the case of multiple period events, sets of periods which span two days or cross the lunch break.

Step 2 involves the removal of periods which are undesirable owing to the relationship between e(i,z) and other events already in the partial timetable. The most important criterion in this category is the strong preference in most schools for a normal distribution for events of the same or similar requirements. Johnston and Wolfenden [21] have described a very satisfactory way of designating different requirements as similar by "tieing" them together. A set of tied requirements is one where all the members are treated as a single requirement for the purposes of event distribution only.

The normal distribution criterion for an event e(i,z) to be inserted on day d is: $|h_d + 1 - h_f| \leq 1$ for each day $f \neq d$ where h_d, h_f are the number of events of requirement ρ_i, or a requirement tied to ρ_i, which have already been assigned on days d and f respectively.

Step 3 attempts to identify those periods now remaining to which allocation of e(i,z) would preclude subsequent completion of the timetable. This involves determining, for each possible set of periods, whether the allocation of e(i,z) to that set would or would not violate any of the conditions necessary for the existence of a solution. The most

powerful of these is the condition specified in inequality (4.1) extended to cover all tight sets of requirements.

If no period set remains after these three steps, the normal distribution constraint may be relaxed by making available, if possible, the least undesirable period set previously eliminated by step 2. If there is no such period set or if such a course of action is unacceptable, then a conflict exists and this must be resolved or the problem must be modified.

If only one period set remains after the third step then it is selected for the assignment of e(i,z). If more than one candidate remains then a selection must be made on grounds which are less clearly defined. The two main criteria are as follows.

1) Interference; that is, the degree to which the chosen assignment is likely to lead to difficulties later in the construction process.
2) Timetable quality.

The characteristics of certain steps and the relation between different steps are now discussed.

5.3 The elimination of unacceptable periods

Step 1 involves checks on resource availability and on multi-period sets in relation to the time structure of the teaching week.

5.3.1 Resource group availability

All the storage structures described in chapter 3 cater adequately for checks on the availability of unique resources. However, the method of storing the availability of each resource individually, used to some

extent in all four storage structures, is not very suitable for an algorithm which takes account of selectable resources during time selection as described in 5.1, approach 3). For this purpose it is not the availability of individual resources that is of interest but rather the aggregated availability of groups of resources. The most direct way of providing for this is to use the concept of "resource group availability". Let AG [k,p] represent the number of resources of some resource group H_k which are available at period p. Suppose a requirement ρ_i is associated with n_k selectable or unique resources belonging to H_k. Then to test the availability of periods p,p+1,...,p+z-1 for an event e(i,z) it is necessary to ensure that $AG[k,q] \geq n_k$, for q = p,p+1,...,p+z-1. If e(i,z) is inserted at this period set then AG[k,q] must be decremented by n_k.

5.3.2 Multi-period sets

The simplest way of ensuring that a multiple period event does not span two days or cross a long break is to restrict the periods at which such an event may start. Let S[z,p] = 1 if an event of duration z may start at period p, and 0 otherwise: then any period set q,q+1,...,q+z-1 can be eliminated from consideration for the allocation of an event e(i,z) if S[z,q] = 0.

5.4 Existence testing in relation to other steps

The logical progression described in 5.2 is not necessarily the most "cost effective" strategy for time selection since the cost of each step, in terms of the number of operations required for its completion, varies

greatly. To illustrate this, assume that storage structure 4 is in use and that a set of periods is to be found for an event e(i,z) which has no selectable resources associated with it. Consider the number of individual data elements which must be accessed (either tested or altered) during each step.

In step 1, 2P such elements must be accessed: every element of row i of the timetable array T, and of row z of array S (see 5.3.2) must be tested. Let Q_1 period sets remain after step 1. In step 2 the maximum number of accesses is t_iP, where t_i is the number of requirements tied to requirement ρ_i. For, if at least one of the Q_1 period sets occurs on each day of the week then every element of the t_i rows of array T must be examined to check for possible violation of the normal distribution constraint. Assume that Q_2 period sets, each of z consecutive periods, remain after step 2.

Step 3 is an altogether more time consuming operation. For each of the Q_2 period sets, e(i,z) must be inserted in array T, the consequences of such an insertion must be evaluated and the event must then be deleted from array T. Suppose that a period set p,p+1,...,p+z-1 is being considered. If x_i is the number of requirements which clash with ρ_i, then insertion into array T requires $z(x_i + 1)$ elements to be accessed assuming that the room availability array does not require updating (see table 3.2). Subsequent deletion requires the same number of operations. To establish whether or not inequality (4.1) is satisfied, both sides

of the inequality must be evaluated for each resource, real or dummy, that could be affected by the assignment of e(i,z) to the period set in question; that is, for all the resources associated with those requirements clashing with ρ_i. For example, suppose that a set of periods is being sought for an event e(3,z) of requirement ρ_3, which clashes with ρ_4, ρ_6 and ρ_8. Let the resources associated with these clashing requirements be: y_1, y_5, y_7 and y_{11} associated with ρ_4; y_1, y_5, y_6 and y_8 with ρ_6; and y_3, y_4 and y_{10} with ρ_8. When e(3,z) is inserted into the timetable it will affect the timetables for ρ_4, ρ_6 and ρ_8 and this may cause the test for the existence of a solution to fail in respect of any one of the resources y_1, y_3, y_4, y_5, y_6, y_7, y_8, y_{10} or y_{11}. Note that the insertion of e(3,z) will also affect the resources associated with ρ_3, but the result of the existence test in respect of these resources cannot be affected since <u>both</u> sides of inequality (4.1) will be decreased by z periods.

Considering the general case of event e(i,z) once again, suppose that x_i requirements clash with ρ_i forming a set $\{\rho_{i_j}\}$, $j = 1,2,\ldots,x_i$, and that associated with ρ_{i_j} are n_{i_j} resources $\{y_{i_{j_k}}\}$, $k = 1,2,\ldots,n_{i_j}$. Let there be $s_{i_{j_k}}$ requirements associated with $y_{i_{j_k}}$. If all the x_i sets of resources associated with the x_i clashing requirements are disjoint, the total number of requirement timetables which must be accessed to carry out an existence test in respect of one set of periods for e(i,z) is

$$r' = \sum_{j=1}^{x_i} \sum_{k=1}^{n_{i_j}} {}_{s_{i_{j_k}}}$$

The x_i sets of resources will not in general be disjoint and for this reason r' will be reduced by a number h (≥ 0) so that the number of requirement timetables r which must be accessed becomes $r = r' - h$.

For every requirement ρ_u whose timetable is examined during this process a total of $(P + 1)$ individual data elements must be accessed. This represents one operation to test the state of $T[u,q]$ for every period q in the week plus one operation to obtain the number of outstanding event periods for ρ_u.

Thus the total number of data elements m_3 that must be accessed during step 3, taking all the Q_2 periods into account, is given by

$$m_3 = Q_2(2z(x_i + 1) + (P + 1)r)$$

$$= Q_2\left(2z(x_i + 1) + (P + 1)\left\{\sum_{j=1}^{x_i} \sum_{k=1}^{n_{i_j}} {}_{s_{i_{j_k}}} - h\right\}\right). \quad (5.1)$$

5.4.1 Numerical example

With the notation used in 5.4 above, let $P = 40$, $z = 2$, $x_i = 30$, $t_i = 2$, $Q_2 = 4$, $\bar{n}_{i_j} = 4$, $\bar{s}_{i_{j_k}} = 10$, $h = 200$, (bars denoting average values). Then the number of data elements m_v, $v = 1,2,3$ to be accessed for each of the three steps is:

$$m_1 = 2.P = 80,$$

$$m_2 \leq t_i P = 80,$$

$$m_3 = 4(124 + 41{,}000) = 164{,}496.$$

Clearly step 3 is a costly operation compared with steps 1 and 2. Economies in computing time are therefore likely to be obtained by altering the order in which the steps are carried out if the number of times that step 3 is executed is significantly decreased thereby.

5.4.2 Alternative strategy

An alternative strategy to that outlined in 5.2 is to select one period set from the Q_2 remaining after step 2 on grounds of interference and timetable quality, and then to test this period set only for necessary conditions for existence. On occasions this test will fail, requiring a different period set to be selected, and some operations will thus have been wasted. This penalty is worth paying for the reduction in the frequency of existence testing provided that the time required to execute the selection on grounds of interference and timetable quality is small compared with that required for an existence test. A further possibility is to dispense with the existence test altogether on the grounds that the benefits it provides in terms of added confidence in the selected assignment are not worth the cost in computing time. Experiments to determine the validity of this argument are described in chapter 9.

5.5 Interference

Suppose that, relative to a given initial situation, a timetabling problem has n possible solutions $S_1, S_2, \ldots, S_n$ and assume that the problem is to be solved by an algorithm which never alters an assignment once it has

been made. As events are inserted into the timetable, the effect of these insertions on the opportunities for assigning events as yet unsatisfied will diminish n. If a solution is to be found n must not vanish during the construction process. In many cases n will reduce to 1 at some point before the last event is inserted. Once this point has been reached every decision made by the algorithm is crucial; if at any point a wrong decision is made, failure to find a solution is certain. Clearly in any given problem the later the onset of this point can be delayed the greater the likelihood of success, and more generally n should be kept as large as possible for as long as possible.

To keep n as large as possible requires that times should be selected for events so as to minimize the decrease in n. The practical value of this is limited since no general method has been found of determining when $n > 0$, still less of computing the actual value of n. It is therefore not possible to determine directly the effect on n of any selections made. But the ideal of minimizing the decrease in n leads to the concept of minimizing interference, that is, of selecting times so as to keep the loss of assignment opportunities for all outstanding events to a minimum. A number of workers have used interference as a time selection criterion, notably Appleby et al [2] and Michalsen [31]. The latter's method is briefly discussed as it is believed to be the most powerful yet devised.

5.5.1 Michalsen's interference criterion

Michalsen's method is based on inequality (4.1)

extended to cover all real and dummy resources. He defines a slack resource as one for which the number of periods available (the left hand side of the inequality) exceeds the number required (the right hand side) by a margin which is defined as the number of surplus available periods. Michalsen selects the set of periods p,p+1,...,p+z-1 for an event e(i,z) which minimizes the sum, across all slack resources y_j, of the total loss of assignment opportunities for the requirements associated with y_j (caused by the allocation of e(i,z) to p,p+1,...,p+z-1), weighted by the reciprocal of the number of surplus available periods for y_j. The weighting tends to favour resources with few surplus available periods at the expense of those with many such periods. Resources which are "tight", that is those with no surplus available periods, are taken into account at an earlier stage, which limits the number of period sets considered by the above process.

All interference criteria represent an attempt to predict the consequences of a current decision on the future course of timetable construction and all suffer from a degree of uncertainty. Their effectiveness in enabling future conflicts to be avoided must be balanced against their cost in computing time. Michalsen's criterion is relatively expensive in this respect; the calculation of the number of surplus available periods, for example, is a similar type of process to the test for existence discussed in 5.4. Could not equal or almost equal effectiveness be obtained at a smaller cost in computing time?

Time selection is particularly important early in the construction process, when the timetable is nearly empty. In the later stages time selection per se becomes less important since the allocation of events to periods will, to an increasing extent, be dictated by the consequences of previous allocations. Now it is in the early stages that "dynamic" criteria, whether for time or event selection, are at their weakest; see also 4.5.2. The reason for this is that these criteria are based on evaluating alternative courses of action in relation to the current partial timetable, which in the early stages of construction is extremely sparse and may be expected to be an unreliable basis for assignment decisions. To take an extreme example, suppose that the first event e(i,z) is to be selected for assignment in a problem where there are no preassignments. By dynamic criteria based on the partial timetable, all the period sets with initial period p for which S[z,p] = 1 (see 5.3.2) are equally acceptable for this event; yet in any given problem certain periods in the day are better than others for the assignment of an event of given duration. The latter assertion is an example of a static or inherent time selection criterion (cf. inherent difficulty in chapter 4). A static interference algorithm based on event duration and nothing else is now described.

5.5.2 A static interference criterion

In the majority of timetabling problems the time structure of each day is identical and this is assumed to be so in the discussion which follows. The following

algorithm places the period sets within a typical day in the order in which they should be selected for an event e(~~i,z~~) so as to keep the potential interference with events of all durations as low as possible. No account is taken of the current partial timetable.

The maximum number of different periods at which an event e(i,z) may possibly start is given by

$$n(z) = \sum_{p=1}^{D} S[z,p],$$

where D is the number of periods in one day. Consider the period p at which an event e(j,v), ρ_j clashing with ρ_i may start in order to minimize $\Delta n(v,z,p)$, the number of opportunities for e(i,z) that the event e(j,v) will destroy if it is assigned to start at this period. $\Delta n(v,z,p)$ is the sum of the opportunities for events of duration z to start at periods p-z+1,p-z+2,...,p and the opportunities for such events to start at p+1,p+2,...,p+v-1 and is given by

$$\Delta n(v,z,p) = \sum_{m=x}^{p} S[z,m] + \sum_{\substack{u=p+1 \\ v>1}}^{y} S[z,u], \qquad (5.2)$$

where x = 1 if p<z, and p-z+1 otherwise, and

y = D if (p+v-1)>D, and p+v-1 otherwise.

Generalizing for all durations z = 1,2,...,Z,

$$\Delta N(v,p) = \sum_{z=1}^{Z} \Delta n(v,z,p). \qquad (5.3)$$

A possibly effective and certainly very fast and simple algorithm selects a starting period p which minimizes ΔN; note that there will be one such period on every day. If no such period is available then the

algorithm selects periods in ascending order of the values of ΔN which they produce.

Example

In the Elms problem a typical day has 8 periods, with a 5 period morning and a 3 period afternoon, and a maximum event duration, Z = 3. The pattern of event starting times is set out in table 5.1 which gives the values of S[z,p].

Duration z	Period, p							
	1	2	3	4	5	6	7	8
1	1	1	1	1	1	1	1	1
2	1	1	1	1	0	1	1	0
3	1	1	1	0	0	1	0	0

Table 5.1 The pattern of permitted event starting times

Table 5.2 sets out the values of $\Delta n(v,z,p)$ and $\Delta N(v,p)$ for all the periods in one day for an event of duration v = 2.

	Duration z	Period, p							
		1	2	3	4	5	6	7	8
Δn	1	2	2	2	2	-	2	2	-
	2	2	3	3	2	-	2	2	-
	3	2	3	3	2	-	1	1	-
ΔN		6	8	8	6	-	5	5	-

Note: An event of duration 2 cannot start at periods 5 or 8 so the appropriate columns are left blank.

Table 5.2 Values of Δn and ΔN for v = 2

From table 5.2 periods within the day for an event of duration should be selected as follows:

1) Most desirable: sixth, seventh.
2) : first, fourth.
3) Least desirable: second, third.

This computation only has to be executed once, at the start of timetable construction. It has thus two advantages over dynamic criteria such as that of Michalsen: it is much less expensive in computing time, and it is likely to produce better results in the early stages of construction when time selection is particularly important and when dynamic criteria are at their weakest. Its disadvantage is that it takes no account of the state of the partial timetable and will tend to become less effective as construction proceeds.

5.6 Timetable quality

The overall quality of a timetable is a difficult concept to define in a general way, particularly in view of the many different ideas held as to the relative importance of the various characteristics. There are, however, a number of such characteristics which are desirable in nearly all timetables. Most of these have been mentioned already either implicitly or explicitly. Many quality considerations can be catered for by pre-assignment: events may be preassigned to specific periods, while requirements and resources may be designated as not available at specific times during the week. The most important criteria which cannot be provided for by preassignment are those concerned with the distribution of events over the week. In this context a test for normal distribution of events of the same requirement or of tied requirements forms part of the process of eliminating period sets from consideration and has already been described.

A straightforward normal distribution with no additional constraints is not entirely satisfactory for requirements with 2 or 3 events in a five day timetable. Most head teachers would prefer a separation between days in these circumstances; Tuesday and Thursday or Monday, Wednesday and Friday for example, rather than having events occurring on consecutive days. Such a day distribution constraint may be implemented as follows. Suppose that a set of periods is being sought for an event e(i,z). Let n_1 be the maximum number of events of requirement ρ_i, or a tied requirement, occurring on any day d in the partial timetable; let n_2 be the maximum and n_3 the minimum number of days between successive days on which there are n_1 events. The criterion to be satisfied, if possible, when e(i,z) is allocated to any day d is that $n_2 - n_3 \leq 1$, after the allocation.

In a final timetable this day distribution quality is less important than the completeness of the solution. The attainment of the latter is enhanced by minimizing interference so that the interference criterion should take precedence over the day distribution criterion. A selection system which reflects the relative priority of these two criteria is as follows. The available period sets for the event e(i,z), for which a time is to be selected, are first ranked in accordance with their desirability on interference grounds. Let N[1,d], d = 1,2,...,D, where D is the number of days in the week, give the "true" period number (that is the number <u>within the week</u>) of the initial period of a period set on day d which is "most desirable" (see 5.5.2, particularly table

5.2 and the following sentence). If there is no such period set available on day d then N[1,d] = 0. Let successively less desirable period sets, available on day d, be given in N[j,d], j = 2,3,...,p. Rows of N are examined sequentially until one is found, row k say, with at least one non zero element. If there is just one such element, N[k,d] = p say, then the period set selected for e(i,z) is p,p+1,...,p+z-1. If there is more than one element in the row then that for day d' is selected for which $n_2 - n_3 \leq 1$ would be satisfied if e(i,z) were allocated on that day. If there is no element for which this condition would be satisfied an arbitrary choice of periods is made from the non-zero elements of row k.

5.7 Resource selection

If the procedure for time selection described in 5.2, or a variant of this discussed in later sections, is followed, the timetable produced by the main construction process will have the following property with regard to resources: all unique resources, and enough selectable resources of the right groups associated with every event will be available at the time allocated to that event. Thus in nearly all cases there is no doubt that a "solution" to the resource selection problem exists and this process is therefore only concerned with choosing the individual selectable resources for each event according to some criteria. Important considerations in resource selection are as follows:

1) The same selectable resources must be allocated

for all periods of an event.

2) For certain requirements, it may be essential that the same selectable resources are allocated to every event of the requirement.

3) User preferences.

The first consideration is almost self evident, for example, it is clearly undesirable to switch rooms during a multiple period lesson, and this is not discussed further. As has been mentioned in 5.1, the timetable's property with regard to the availability of selectable resources does not ensure that the second consideration can always be satisfied. For example, suppose that there are three requirements ρ_i, ρ_j and ρ_k, each for two single period events, and each of which requires one resource from a group containing two resources. Let the main construction program allocate periods to events as follows: p_1, e(i,1), e(j,1); p_2, e(i,1), e(k,1); p_3, e(j,1), e(k,1). Under these circumstances the constraint can be satisfied by at most two of the three requirements. Such circumstances are, however, only likely to arise infrequently. In any case, the importance of this constraint is limited, since if it is necessary that all events of the same requirement are associated with the same selectable resources, then these resources can normally be made unique. In cases where the constraint does apply, the requirements concerned should be selected first in the resource selection process.

5.7.1 User preferences

By far the greatest part of the resource selection

problem concerns rooms and it is necessary to examine the way in which rooms are allocated in British schools. An almost universal rule is that all practical instruction in such subjects as Science, Art and P.E. takes place in specialist rooms dedicated to one function or a narrow range of functions. For the most part such rooms can, and indeed must, be associated uniquely with requirements so that no resource selection problem is involved.

The position is more complicated for non-specialist classrooms. Broadly there are two policies which may be adopted:

Static class. Each class has its own classroom in which as much as possible of its weekly programme of instruction takes place and teachers move as they change classes.

Static teacher. Each teacher is allocated his own classroom in which, as far as possible, he conducts his weekly teaching programme and the pupils move between subjects.

In practice both systems appear to operate to a greater or lesser degree and no overall pattern is discernible in British secondary schools.

5.7.2 Other factors

A theoretically attractive object in room selection is that of "minimum disturbance". This is a compromise between the static class and static teacher policies in which neither classes nor teachers are linked to specific rooms but it is desired to minimize the amount of

movement from room to room by teachers, classes or both.

Minimum disturbance may be regarded as ranging from simply minimizing the number of room changes to minimizing the total distance involved in such changes with regard to each individual class, all classes, each day, the whole week etc. The degree to which this criterion should be applied to these different groupings is very dependant on local circumstances, and however the concept is interpreted its implementation in a resource selection algorithm is likely to be difficult and expensive. In general minimum disturbance is not an important consideration in single site British secondary schools with which this thesis is primarily concerned and is therefore not considered further in this work. In the context of timetables for universities, colleges and dispersed site schools it is most important and would form a valuable topic for future research.

A final consideration is the loading on resources. On the occasions when teachers are selectable resources it is desirable to keep the loading on different teachers as even as possible. This consideration does not apply to non-specialist rooms and as teachers are almost invariably unique resources it is comparatively unimportant.

5.7.3 Implementation

Two valid and practical objectives only have been identified for room selection, static class and static teacher. One of these may apply to the whole of a given timetabling problem or alternatively one objective

may apply to part of the problem and the other to the remainder. How can these objectives be implemented in a selection algorithm? The simplest method is to allocate a user defined "preferred non-specialist classroom" to each class and teacher, and an indicator for each requirement ρ_i specifying whether class or teacher preferences are to take precedence when selecting rooms for events of this requirement. Suppose that a requirement ρ_i is flagged as a "static teacher" requirement and is associated with a teacher with preferred room y_m, a class with preferred room y_n and one room from room group H_k. Let A[j,p] = 0 if resource y_j is available at period p and 1 otherwise. Suppose that an event e(i,z) has been scheduled at periods p,p+1,...,p+z-1 and a room is to be found for this event. The resource selection algorithm proceeds as follows.

1) If $y_m \in H_k$ and A[m,q] = 0, q = p,p+1,...,p+z-1, then room y_m is selected for e(i,z).

2) If 1) is not satisfied then if $y_n \in H_k$ and A[n,q] = 0, q = p,p+1,...,p+z-1, then room y_n is selected for e(i,z).

3) If neither 1) nor 2) is satisfied then any available room in H_k is selected for e(i,z).

5.8 Learning techniques

A person with no previous experience of timetabling learns a great deal about the process during the construction of his first few timetables. Indeed it is

just this ability to learn by experience which makes the human being so effective in this field. No computer timetabling system, however, has used learning techniques to improve its strategy for solving a given problem. Although, intuitively, one feels that this omission is justified by the crudity of machine learning techniques in relation to the complexity of the problem, some investigation of the application of these techniques in the field of computer timetabling is clearly desirable.

Any process which involves selection of one from a number of choices is potentially amenable to improvement by learning techniques. The approach generally used in heuristic programming is to quantify the various features which are believed to affect the decision process. Let there be F such features. A weighting factor w_i is associated with the i^{th} feature, $i = 1,2,\ldots,F$. When a decision has to be made between N alternative courses of action the first step is to calculate the value v_{ij} of the i^{th} feature of the j^{th} course of action $j = 1,2,\ldots,N$. The weighting factors are applied to each feature of each alternative, the weighted values summed and that course V is selected for which

$$V = \max_j \text{ (or } \min_j) \sum_{i=1}^{F} w_i v_{ij}, \quad j = 1,2,\ldots,N, \qquad (5.4)$$

depending on whether the features are desirable or undesirable.

The learning process evaluates the consequences of

the decision and attempts to find a correlation between the success of the decision and the values of the various features affecting it. If a correlation is found the appropriate weighting factors are varied so as to enhance the relative importance of features which lead to successful decisions. The main difficulty in applying learning techniques to timetabling lies in the great complexity of the problem. In any given situation there may well be several hundred alternative choices of assignment. Very seldom are two situations exactly alike and their differences may be in a pattern of interrelations between problem elements so complex as to be almost unmeasurable. These differences may well be far more important to success or failure than the simple measurable features which are all that can practicably be used in a learning algorithm. However, some practical investigation into the possibilities of learning algorithms related to event selection has been done with results described in chapter 9.

CHAPTER 6 - ALGORITHMS III - CONFLICT RESOLUTION AND USER GUIDANCE

6.1 Detection of conflicts

The selection processes described in chapter 5 are not infallible and assignments may be made which are incompatible with a solution. It is a feature of the timetabling problem that once such an error has been made, the more steps that are taken in the construction process before remedial action is taken, the less likely is it that such remedial action will be successful. Thus the ability to detect conflicts soon after they occur is a valuable feature of a computer timetabling system.

Conflict detection has already been fully discussed implicitly since it is a key function of the processes associated with event and time selection, in particular the computation of current difficulty, existence testing and calculating interference. The more powerful these processes are, the fewer the conflicts that will arise and the less time should need to be spent in resolving those that do so since most of them will be detected soon after they occur. Furthermore, as has been pointed out already, completely efficient time selection would render conflict resolution unnecessary, but no perfect algorithm for time selection has yet been discovered so it is necessary to consider conflict resolution.

6.2 Resolution of conflicts

Once a conflict has been detected there are two

ways in which it may be resolved without modifying the original problem. The first approach devised by Johnston [22] involves "retreating"; that is, removing events from the partial timetable until some point is reached and then "advancing" again in a different direction with the hope of avoiding a recurrence of the conflict condition.

If at any point a conflict occurs, Johnston retreats to a point where one or more choices of assignment are available which have not already been tried. One such choice of assignment is then made and the program advances again by inserting events into the timetable using its normal assignment algorithms until either a further conflict is encountered or the timetable is completed. In the former case the retreat procedure described above is repeated unless a prespecified number of retreats has already been carried out in the current run, in which case the program prints out the partial timetable and terminates. Johnston found it necessary to limit the number of retreats in any one run to between 5 and 20 to avoid excessive running times.

The second approach is to change the times allocated to events already in the partial timetable until either the conflict is cleared or failure is accepted at some point. This point may be, for example, the completion of a prespecified number of operations or, in manual timetabling, when the person constructing the timetable gets tired of making alterations.

Barraclough's interchange scheme [4] and the exchange routine outlined in [3] by Berghuis, Van der Heiden and Bakker are examples of this approach in the field of computer timetabling. Both of these, however, have limitations in relation to the types of event catered for. Clearly it is desirable that an interchange or displacement algorithm which does not suffer from such limitations should be developed and its effectiveness compared with the retreat strategy adopted by Johnston.

6.2.1 A displacement algorithm

The displacement algorithm is first illustrated by a simple example. Consider the problem used in 2.1. It is summarised in table 6.1 which, in addition to specifying the requirements for class-teacher meetings gives a list of the requirements clashing with each requirement.

Requirements	No. of events (see note)	Class	Teacher	Clashing requirements	Pre-assignments
1	1	1	1	2, 3, 4	
2	1	1	2	1, 3, 5	to period 2
3	1	1	4	1, 2, 8	
4	1	2	1	1, 5, 6	
5	1	2	2	2, 4, 6	
6	1	2	3	4, 5, 7	
7	1	3	3	6, 8	
8	2	3	4	3, 7	

Note: All events are single period events.

Table 6.1 A summary of the example problem

Suppose that a series of assignments have led to the

construction of the partial timetable shown in table 6.2.

Requirement	Period		
	1	2	3
1	X	X	3
2	X	O	X
3	4	X	X
4		1	X
5		X	
6		X	
7		2	
8	X	X	

Note: Entries in the [i,p] element of the table have the following interpretation:

Positive integer: an event e(i,1) has been assigned to period p. The value of the integer gives the stage at which the assignment took place.

Zero: an event e(i,1) preassigned to period p.

X: a non-available period for requirement ρ_i due to one or more clashing event(s) at period p.

Blank: an available period.

Table 6.2 An example of a conflict situation

Clearly the partial timetable illustrated in table 6.2 contains a conflict since ρ_8 is only available for one period. The displacement algorithm seeks to make one additional period available for ρ_8 by displacing one or more clashing requirements. The two non-available periods are examined to see if such displacement is possible. The operations and tests performed are set

out in table 6.3.

Period	Clashing events	Displacement possible?	Operation
1	e(3,1)	No	Try new period
2	e(7,1)	Yes	Displace e(7,1) to 1 Insert e(8,1) at 2

Table 6.3 Resolution of a conflict by displacement

This displacement algorithm can be defined formally for events of any duration as follows. (Certain points are illustrated by reference to storage structure 4 which, although not essential, is the most suitable storage structure for the implementation of this algorithm). Assume that the state of the current partial timetable is such that a suitable and available set of periods cannot be found for an event e(i,z). The displacement algorithm attempts to find a suitable set of periods $S = \{p,p+1,\ldots,p+z-1\}$ from which events of requirements ρ_j, clashing with ρ_i can be displaced to permit the insertion of e(i,z) at S.

Suppose that m events of durations $z_1,z_2,\ldots,z_m$ of n clashing requirements $\rho_{j_1},\rho_{j_2},\ldots,\rho_{j_n}$ occur during one or more of the periods belonging to S (m and n are not necessarily equal since it is possible for more than one non-contiguous event of one requirement to occur during the z periods of S if $z \geq 3$). The algorithm has to find m suitable and available period sets $Q_k = \{q_k,q_k+1,\ldots,q_k+z_k-1\}$, $k = 1,2,\ldots,m$, into which the m events may be appropriately displaced from their existing period sets $R_k = \{r_k,r_k+1,\ldots,r_k+z_k-1\}$, $k = 1,2,\ldots,m$,

so as to free S for the insertion of e(i,z).

If some of the m clashing events belong to the same requirement or to tied requirements, the corresponding period sets Q selected must be suitable in relation to each other as well as in relation to the remainder of the partialtimetable. If two such events belong to tied requirements, for example, the two period sets Q selected for them must conform to the normal distribution constraint. An analogous consideration applies to available period sets. Two events of clashing requirements may have the same period set available if each is considered separately from the other: yet clearly it is not possible for both events to be assigned to it. It is thus necessary when selecting a period set Q_k for the k^{th} clashing event ($1<k\leq m$) to take into account the consequences of displacing the previous k-1 clashing events. In storage structure 4 this is most conveniently achieved by updating the timetable array T each time a suitable and available period set Q_k is found for the k^{th} clashing event. If no such set can be found for the $(k+1)^{th}$ event then the "status quo ante" must be re-established and the previous k events restored to their original period sets R and a new period set S tried for e(i,z).

A new period set Q_k may overlap R_k provided that Q_k does not overlap S. In storage structure 4 this condition may be enforced by temporarily distorting the appropriate row of the timetable array T. Before examining row j_k of T to seek a new period set Q_k for

event $e(j_k,z_k)$ the following action is taken:

Step 1: $T[j_k,v] = 100000,\ v \in R_k$

Step 2: $T[j_k,w] = -99999,\ w \in S$

These two steps carried out sequentially have the effect of making all the periods within S not available for inclusion in Q_k, but if there are any periods in R_k which do not belong to S these are made available for possible inclusion in Q_k.

Requirement	Period		
	1	2	3
1		X	1
2		O	X
3	X	X	X
4		2	X
5		X	
6		X	
7	X	X	
8	3	3	

Note: The numbers in table elements indicate the stage at which the appropriate assignment was made. X indicates non availability. See note to table 6.2.

Table 6.4 Further example of a conflict situation

For example, consider the problem summarised in table 6.1, modified so that requirement ρ_8 has one double period event instead of two single period events. Suppose that after three assignments in the construction process the partial timetable is as shown in table 6.4. A conflict exists since no periods are available for e(3,1). Event e(8,2), at present assigned to periods 1 and 2, is preventing the insertion of e(3,1) at period 1. Steps in the process for selecting a new set of

periods Q for e(8,2) are as follows.

Step 1: T[8,1] = 100000, T[8,2] = 100000,

Step 2: T[8,1] = -99999.

After step 2 the values of elements T[8,2] and T[8,3] are 100000 indicating that periods 2 and 3 are available for the insertion of e(8,2), thus permitting the assignment of e(3,1) to period 1.

Clearly the operation of the algorithm on real data is likely to be greatly accelerated if the period set S with the best chance of success is tried first after which period sets are examined in descending order of likelihood that they will permit successful displacement. Storage structure 4 very readily yields a criterion for ordering period sets for this purpose. This criterion is the total number of event periods clashing with e(i,z) in the period set S, given by

$$E(S) = \sum_{u=0}^{z-1} (100000 - T[i,p+u]). \qquad (6.1)$$

Candidate period sets S are then examined in ascending order of their E(S) values.

6.2.2 Recursive displacement

The displacement algorithm can be used recursively. Suppose that in the procedure discussed above it is not possible to find a suitable and available period set for the k^{th} clashing event, $e(j_k,z)$ say, then the displacement algorithm can be used to operate on requirement ρ_{j_k} in order to clear some period set Q_k for the insertion of $e(j_k,z_k)$.

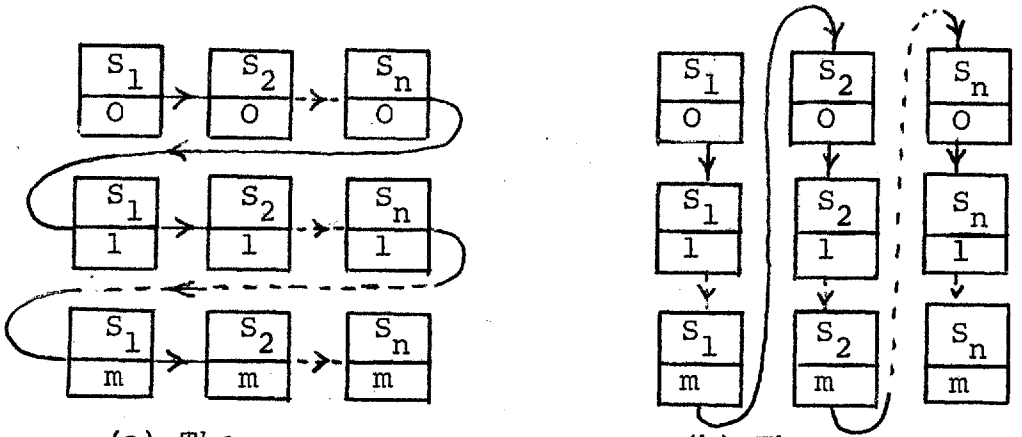
Let $\delta(e(i,z),S)$ represent the complete displacement

operation on any event e(i,z) including the final insertion of the event at the period set S, if this period set is acceptable, and the complete restoration of all changes to the timetable caused by this operation if it is not successful. Let $\mu(e(i,z),R,Q)$ represent the movement of any event e(i,z) between an "old" period set R and a "new" period set Q caused by a displacement operation. This movement may be from R to Q or in the reverse direction depending on whether the displacement algorithm is attempting to find a new period set or restoring a previous situation. Displacement may then be defined recursively as

$$\delta(e(i,z),S) \equiv \mu(e(j_k,z_k),\ R_k,Q_k) + \delta(e(j_h,z_h),Q_h), \quad (6.2)$$

where $e(j_k,z_k)$ belongs to the set of events clashing with e(i,z) at S which can be moved from S to an acceptable period set Q without recursive use of displacement and $e(j_h,z_h)$ belongs to the set which cannot be so moved.

There are two possible strategies which may be adopted if a period set S cannot be cleared for the insertion of e(i,z) with a given level of recursion of δ. The first, to be known as the horizontal strategy, is to try a new period set S, and the second, to be known as the vertical strategy, is to maintain the previous period set and invoke a deeper level of recursion of δ in an attempt to clear it. Figure 6.1 illustrates the "pure" horizontal and "pure" vertical strategies.



(a) The pure horizontal strategy

(b) The pure vertical strategy

Note: It is assumed that there are n suitable period sets $S_1, S_2, \ldots, S_n$ that can be tried. In both strategies the maximum permitted level of recursion is m. The figures in the lower part of a box show the permitted level of recursion when trying the period set in the upper part of the box.

Figure 6.1 Strategies for implementing the recursive use of displacement

The pure strategies illustrated in figure 6.1 represent extremes, between which many different mixed horizontal-vertical strategies are possible.

6.2.3 Processing implications of recursion

The demands for processing time can be expected to rise sharply with each increment in the maximum permitted level of recursion m, a user defined parameter.

Let there be n suitable period sets to be tried at every level at which the displacement algorithm operates and assume a situation where the displacement cannot be successful due to an undetected problem infeasibility. In such circumstances the total number N of period sets to be examined rises with m in geometrical progression as follows:

m	0 (i.e. no recursion)	1	2	3
N	n	n^2	n^3	n^4.

Although n will not be constant in any real problem the increase of N with m in geometrical progression is realistic and clearly m must be chosen carefully in relation to the computing facilities available, and the size and complexity of the problem to be solved.

6.3 Retreat and displacement compared

From a theoretical standpoint retreat has the advantage that it must eventually find a solution to the given problem if one exists. As Johnston has pointed out [22], this has little significance unless the number of retreats is very small compared with the total number of assignments made, which is usually not the case. He found 20 to be an upper limit for the number of retreats to be allowed but with this constraint his system failed either to complete a timetable or to show that no solution existed in the majority of runs using real timetabling data.

The main advantage of retreat over displacement is that whenever requirements are assigned and events inserted the system is in the same basic mode of operation, that is, "advancing". This means that algorithms for time and resource selection and problem modification are always operating in the same "environment". When a displacement strategy is used these three algorithms must operate in two different circumstances, during the normal assignment procedure and during displacement when new period sets have to be found for clashing events.

Displacement has the advantage that all its operations relate directly to the conflict which it is attempting to resolve. For example, when trying to clear a period set S for the insertion of an event e(i,z) the algorithm only attempts to move events which are preventing insertion of e(i,z) at this period set. Retreat does not have this property. It may be necessary to retreat several steps before taking an alternative choice which will affect the timetable for requirement ρ_i, and as a result much processing time can be wasted on actions which stand no chance of resolving the conflict.

Tables 6.5 to 6.8 illustrate the performance of the two algorithms by reference to the example problem originally defined in Chapter 2. Table 6.5 describes a sequence of operations which ends in failure to fit event e(3,1) at stage 7. A simple assignment strategy based only on minimizing loss of choice (see 4.5.1) has been used: the dictates of a forcing situation (glossary 13) are not necessarily followed. Table 6.6 shows the state of the timetable at stage 7 and tables 6.7 and 6.8 show the operations of the retreat and displacement algorithms respectively to resolve the conflict. A pure horizontal strategy has been used for displacement.

Stage	Assignment (see note (i))		Choice for requirement (see note (ii))								Loss of choice (see note (iii))
	Choice	No choice	1	2	3	4	5	6	7	8	
0			1	X	1	2	1	2	2	1	
1	4→2		1	X	1	X	1	1	2	1	1
2	7→2		1	X	1	X	1	1	X	0	1
3	1→3		X	X	0	X	1	1	X	0	1
4	5→3		X	X	0	X	X	0	X	0	1
5		6→1	X	X	-1	X	X	X	X	0	-1
6		8→1,3	X	X	-1	X	X	X	X	X	0
7		3 fails									

Notes:

(i) When an assignment is shown in the "choice" column it indicates that the choice (see 4.5.1) for the requirement was non-zero. Assignments in the "no choice" column indicate that the choice for the requirement was zero. The notation i→p denotes the assignment of e(i,1) to period p.

(ii) The entries in the columns under "choice for requirement" give the choice for the appropriate requirement after the assignment on the same line. A negative choice indicates a conflict.

(iii) The loss of choice column gives the total loss of choice for all outstanding requirements caused by the assignment.

Table 6.5 A sequence of operations in the partial solution of the example problem

Requirement	Period		
	1	2	3
1		X	3
2		O	X
3	X	X	X
4	X	1	X
5	X	X	4
6	5	X	X
7	X	2	X
8	6	X	6

Note: The numbers in table elements indicate the stage at which the appropriate assignment was made. X indicates non availability. See note to table 6.2

Table 6.6 The timetable at stage 7

Insertions	Deletions	Is there a choice of assignment not already tried?	Remarks
4→2 7→2 1→3 5→3 6→1 8→1,3			Initial sequence of assignments as shown in table 6.5 reproduced for convenience only. 3 fails. Retreat.
	8	No	Retreat.
	6	No	Retreat.
	5	Yes	Make alternative assignment.
5→1 6→3 8→1,3			Advance using normal assignment algorithm. 3 fails. Retreat.
	8	No	Retreat.
	6	No	Retreat.
	5	No	Retreat.
	1	Yes	Make alternative assignment.
1→1 5→1 6→3 8→1,3			Advance using normal assignment algortihm. 3 fails. Retreat.
	8	No	Retreat.
	6	No	Retreat.
	5	No	Retreat.
	1	No	Retreat.
	7	Yes	Make alternative assignment.
7→3 1→1 5→3 6→1 8→1,2 3→3			Advance using normal assignment algorithm. Timetable complete.

Table 6.7 Retreat

Level of recursion used	Event e(i,z)	Period set S	Clashing events e(j,z)	Period set Q available?	Action
0	e(3,1)	1	e(8,1)	No	Try new set.
0	e(3,1)	2	e(2,1)	No (preassigned event)	Try new set.
0	e(3,1)	3	e(1,1)	1	Delete 1, 1→1
			e(8,1)	No	Delete 1, 1→3 Increase permitted recursion level.
0	e(3,1)	1	e(8,1)	No	Use displacement on 8. Delete 8 from 1.
1	e(8,1)	2	e(7,1)	No	8→1. Revert to level 0. Try new set.
0	e(3,1)	2	e(2,1)	No (preassigned event)	Try new set
0	e(3,1)	3	e(1,1)	1	Delete 1, 1→1.
			e(8,1)	No	Use displacement on 8. Delete 8 from 3.
1	e(8,1)	2	e(7,1)	3	Delete 7, 7→3, 8→2, 1→3.

Note: "Delete i from p" means remove a single period event of requirement ρ_i from period p. Where there is only one such event in the requirement "delete i" is used to mean the removal of this event from its current period in the timetable.

Table 6.8 Displacement

From tables 6.7 and 6.8 it can be seen that retreat involves 28 insertions and deletions, while displacement involves 13. On the other hand displacement probably

requires more operations of other kinds such as testing the state of timetable array elements, although without going into the exact details of storage structures this cannot be quantified.

The example just given is not in fact "fair" to the retreat strategy since the assignment and conflict detection algorithms used were unduly simplified. If the conflict had been detected soon after the insertion which caused it (e(7,1) to period 2), the retreat algorithm would have cleared it with one retreat/advance sequence, whereas displacement would have still required one level of recursion, albeit with slightly fewer operations. The real significance of the example is that it illustrates the fact that the effectiveness of the retreat strategy is dependent on the quality of the selection algorithm to an extent that displacement is not.

A further advantage of displacement is its ability to move preassigned events as well as those assigned in the course of timetable construction. In many cases, of course, the displacement of preassigned events is unacceptable to the user, but in certain circumstances preassignment can be used to indicate a preference for, rather than a mandatory specification of, the times at which preassigned events should occur and to ensure that such events are present in the final timetable. Provided that the storage structure can differentiate those events which can be moved from those which cannot, the former can be displaced if necessary in the light of developments during timetable construction. In the

above example, had it been permitted to displace event e(2,1), preassigned to period 2 and assumed to be immovable, the displacement algorithm could have resolved the conflict very rapidly without the use of recursion (see table 6.8, second row), since e(2,1) could have been displaced to period 1. The retreat strategy does not lend itself to the alteration of preassignments because preassigned events have no "history".

A possibility is a compromise approach embracing both strategies with displacement used initially and a retreat carried out if this fails. This is discussed further in chapter 10.

6.4 Modification of the problem

How should a computer timetabling system proceed when it encounters a conflict which it cannot resolve? Most of the possible causes of action are listed below.

1) Abandon the run.
2) Omit one (or more) events so as to remove the conflict.
3) Change the characteristics of one or more events or requirements, in particular,
   (i) break down a multiple period event e(i,z) into two events e(i,z-1) and e(i,1). This can be done repeatedly until one event e(i,z) has become z events e(i,1),
   (ii) change the resources associated with a requirement. For example the association between a requirement and a specific room may be removed.
4) Relax one or more time or availability constraints

on the problem, in particular,

(i) Override the initial non-availability of resources or requirements at particular periods or groups of periods.

(ii) Relax the rules governing when events of a given duration may start. For example a double period may be allowed to run over a lunch break.

(iii) Relax the rules governing the time distribution of events, including the spreading of events of the same or similar requirements over the week.

Course 1) is not in general satisfactory for any practical system whether implemented by manual or computer means. A school must have a timetable and one that is, say, 99% complete is clearly more valuable than no timetable. The only circumstances when complete abandonment might be justified is after a large number of failures indicating the need for a major revision of the curriculum. Course 2) is very commonly adopted by computer timetabling systems as the only possible alternative to course 1). Changing the characteristics of events, courses 3) (i) and (ii) is likely to be appropriate for some events and not others. Some English and Mathematics lessons, for example, are normally taught in double periods at the Elms, but these can be broken down into singles if necessary for timetabling. This is not the case with practical Science subjects. Similarly alternative teachers may be acceptable for some requirements but not others.

Course 4) is the most commonly adopted expedient

in manual timetabling. Very few British school timetables do not incorporate some violations of the desired pattern of time and availability. In a computer timetabling context it is most important for the user to influence, and if possible control, what is actually done in any particular set of circumstances.

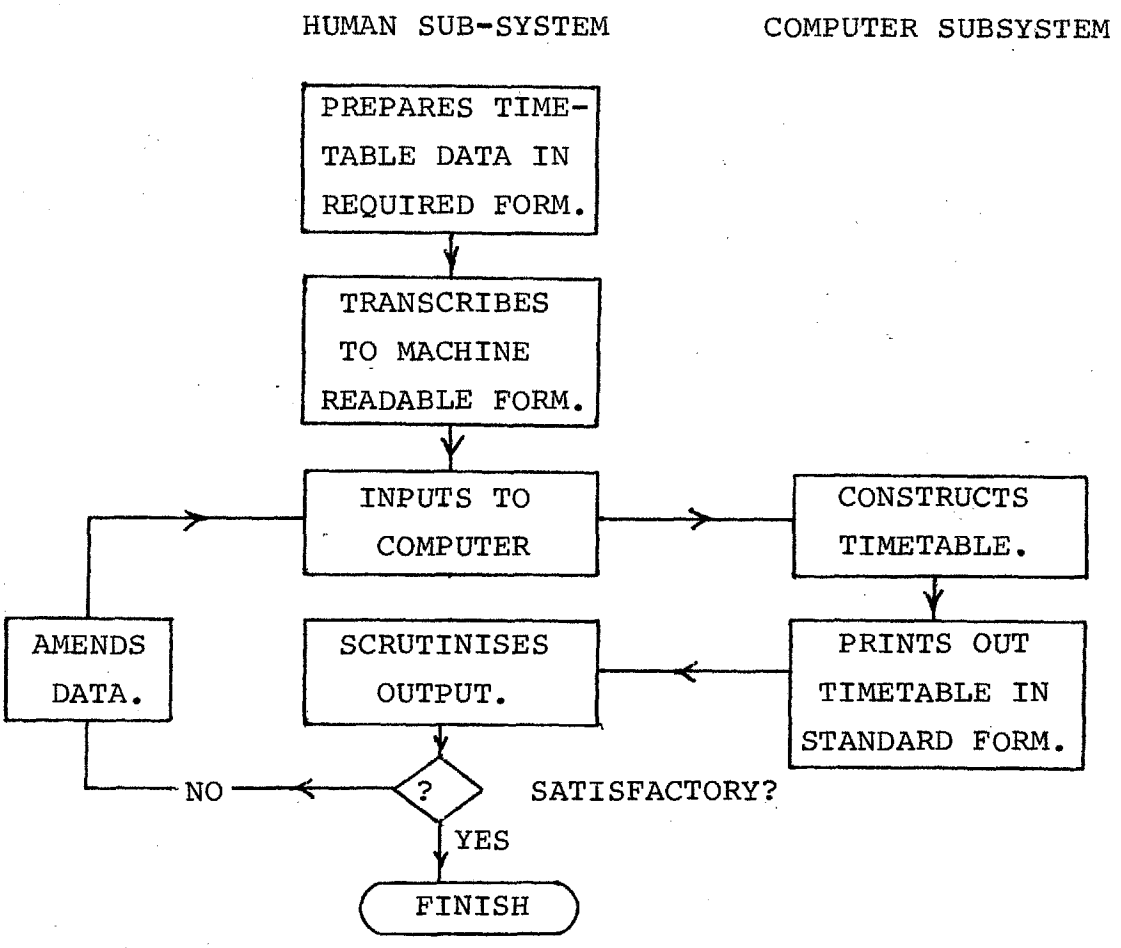
There are three possible mechanisms for deciding on the course of action in a computer timetabling system. The simplest and least satisfactory is for the course of action to be taken when a conflict occurs to be prespecified by program with no provision for any user influence on the course of action. An alternative approach, which has important advantages, is for the user to prespecify in the input data the course of action to be taken when conflicts occur. The third possibility is an interactive system whereby the computer can seek guidance from the user on meeting a conflict. The last two possibilities are part of the larger question of interaction between the user and the computer timetabling system.

6.5 Man-computer interaction

In the field of information processing by computer for business and commercial applications it is well understood that an information processing system comprises a number of subsystems, of which the computer subsystem is only one. Careful attention is, or should be, paid to the various clerical and other non-computer procedures and particularly to the points at which computer and non-computer subsystems interact. There has been a tendency in the literature to play down the

information processing aspects of computer timetabling, partly because very few computer timetabling systems have been used operationally, and partly because of the algorithmic interest and difficulty of the problem itself. Nevertheless the interaction between user and computer subsystems is an important factor in determining whether a computer timetabling system is operationally satisfactory.

The pattern of interaction between user and computer envisaged by early workers is illustrated in figure 6.2



Note: The term "human subsystem" includes various different types of people such as school teachers, computer systems experts, and data preparation and operating personnel. No differentiation is made between these people as the interaction between them is not the present concern.

Figure 6.2 The interaction between human and computer subsystems

It must be said that the concept illustrated in figure 6.2 is very simple and has much to recommend it provided that it produces satisfactory results. When computer timetabling systems started being applied to real schools, however, the importance of very careful validation of the input data became apparent, firstly to remove low level syntax errors, an alphabetic character where a digit was expected for example, and secondly to remove inconsistencies in the data, for example a class being scheduled for more periods than the number of periods in the week. The need for careful data validation has led naturally to the idea of a separate data checking run in which the computer would point out mistakes and inconsistencies. Only when the data was found to be correct would timetable construction, time consuming and expensive if operating on invalid data, be started. A number of methods of specifying the input data itself have been used by different workers and there is a need for a systematic evaluation of these, since the operational success of any computer timetabling operation depends on data supplied by teachers who initially are likely to have no experience in computer work.

It has also been found that improvements can be made at the output end of the timetabling process. Not all schools like their timetables produced in the same way and this has led to the idea, first advanced by Hickman [20], of allowing the user to specify the form in which the output is to be produced.

Even more important than improvements to the

mechanism for the inputs to and outputs from the computer subsystem has been the introduction by Lions [28] of a set of priority indicators for each requirement by which the user can indicate whether any departure from the specification of the requirement is permissible and if so in what order the different kinds of relaxation should take place. This is the first systematic implementation of the second alternative mechanism for problem modification discussed in 6.4. Its disadvantage is that once timetable construction has started the user can play no part in the process. It leaves manual timetabling with the great advantage that the person constructing the timetable, with all his acquired experience of what will and will not be acceptable in the school concerned, can decide on the course of action to be taken as and when conflicts arise in the light of the current state of the partial timetable. In general, the number of possible detailed steps that may be taken in a particular situation in order to resolve a conflict will be far greater than can practicably be specified in the input data. It really is not possible for the user to forecast every conceivable eventuality and provide an appropriate decision in case it arises. Any practical scheme, then, which tries to forecast difficulty and provide for it in advance is likely to be a poor second to a system which allows the user to decide on the course of action at the time when difficulty arises.

Another disadvantage of using the computer in what may be called the "closed" mode of timetable construction

is that when the process is completed the user is presented with a "take it or leave it" situation. If he is very nearly satisfied with the timetable produced but would like some changes made, he has only two choices. He can take the computer produced timetable and amend it by hand. This process is explicitly catered for in the Nordata system [30], which provides output on transparent paper which can be manually amended and then readily copied. Even a small number of amendments, however, can mean a great deal of effort. The alternative to manual amendment is to amend the input data and produce a new timetable on the computer. The disadvantage of this course, apart from the expense, is that there is no guarantee that the new timetable will not have different, and possibly even more serious, shortcomings than the previous timetable.

These arguments lead naturally to the third mechanism for problem modification, the interactive construction process, mentioned in the previous section. This, together with the various improvements in input and output discussed earlier, leads to the concept of timetable construction as a process involving interaction between user and computer at every stage, which is illustrated in figure 6.3. An assessment of this concept forms an important part of the later chapters, particularly chapter 10.

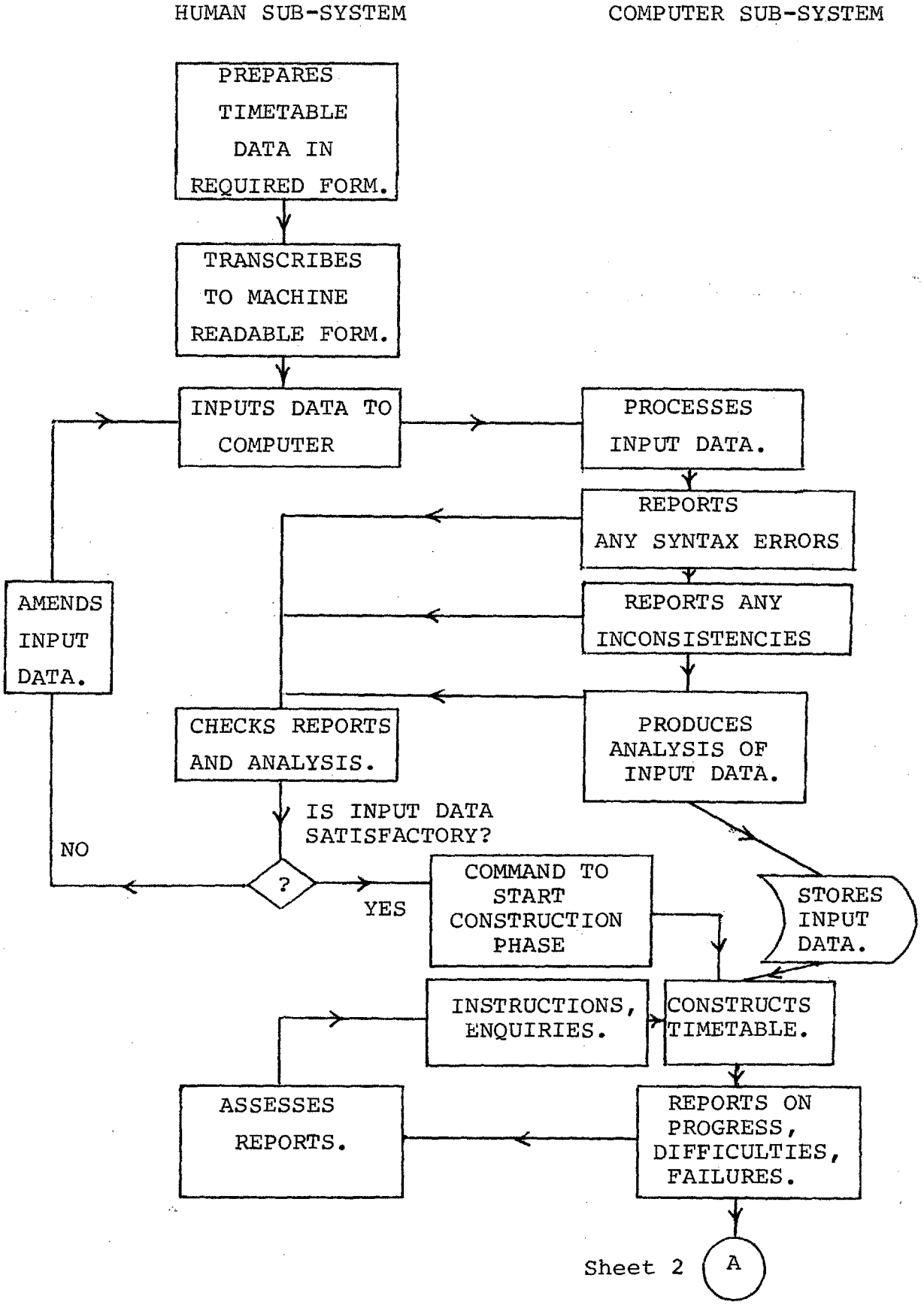


Figure 6.3 Full interaction between human and computer sub-systems

Sheet 1 of 2

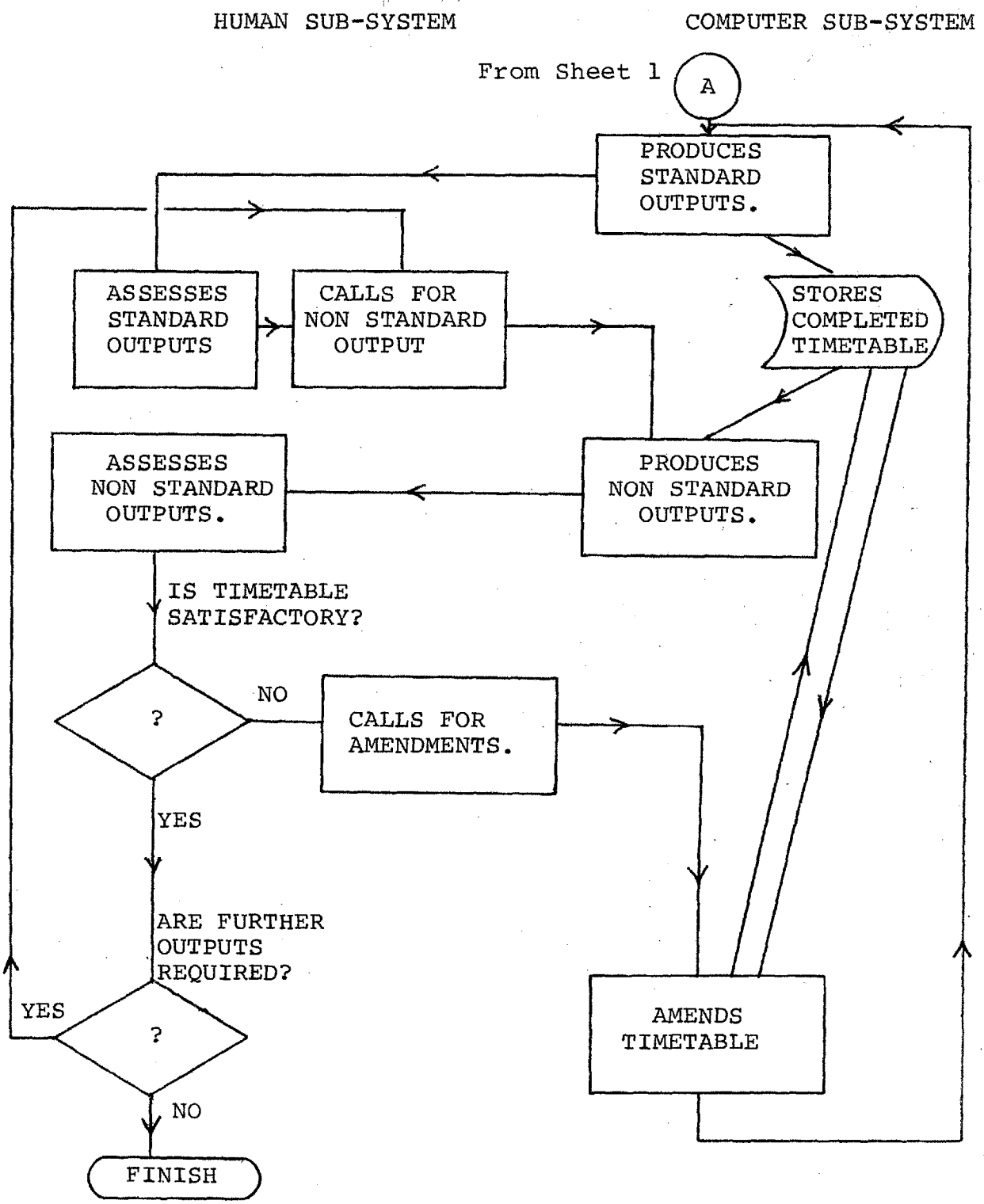


Figure 6.3 Full interaction between human and computer sub-systems

Sheet 2 of 2

CHAPTER 7 - COMPUTER TIMETABLING SYSTEMS

7.1 Early work

In the course of the work described in this thesis, two completely different computer timetabling systems have been developed. The objective of the first system was to enable a thorough understanding of the computer timetabling problem to be gained and in particular to assess whether a storage structure based on resources would be a satisfactory foundation for further development. The system was designed to cater for the Royal Military College of Science degree courses and it therefore had certain features not applicable to the general run of schools and colleges, notably the need to cater for some change of requirements from week to week while maintaining as consistent a timetable as possible for every week of the term.

The system used a storage structure similar to storage structure 1 as described in chapter 3 although week to week variations required a third dimension to be added to the timetable and availability arrays. Event selection was based on the order in which requirements were input and a simple scheme was used to ensure a normal distribution of periods throughout the week. Rooms were allocated to events as these were inserted into the timetable.

A conflict resolution algorithm dealing only with single period events was incorporated but unlike Barraclough's interchange routine [4], it was used as soon as a conflict was detected. Requirements were input

by "line", and the format of the input data was very similar to that used by Barraclough.

The system was quite successful in operation. It produced timetables with an insertion rate better than 98% for the Michaelmas 1969, Lent 1970 and Summer 1970 terms. The first two were not satisfactory mainly because the spread of events across the week was poor, with free periods concentrated at the end of the week. The program was modified and the Summer 1970 timetable was very much better. The spread of events across the week was now quite satisfactory and the timetables for each week taken in isolation were excellent but the week to week variations in the timetable were more than desirable and indeed greater than were necessary as shown by a comparison with the existing manually produced timetables.

Further modifications could have improved the consistency, but the system had achieved its aim and no further work was done with it. The following conclusions were drawn:

1) Even the simple interchange algorithm was awkward to program and very time consuming with the storage structure used. A more general and powerful conflict resolution algorithm based on the displacement concept would not be practicable with this type of storage structure.
2) The storage structure used is well suited to resource selection during timetable construction. The room allocation algorithm was a highly successful aspect of the system.

3) The whole process of data input requires very careful design.

7.2 SCOCOST - general philosophy

A second computer timetabling system incorporating the lessons learnt from the first has since been designed and developed with the following main objectives:

1) To be capable of constructing timetables for a wide variety of schools and colleges. (The specific Shrivenham problem of week to week variation was not considered important in this context).
2) To act as a vehicle for the investigation of selection and conflict resolution algorithms and of the possibilities of man-computer interaction in the field of computer timetabling.

The system is known as the Shrivenham COmputer system for the COmpilation of School Timetables, SCOCOST. Inevitably the system has changed during development and this chapter describes the current basic system, which is a slightly modified version of that specified in the Users' guide and data manual [6]. The additional experimental procedures and programs developed to investigate specific problem areas are described in chapters 9 and 10.

7.2.1 Computer and programming language

The system could have been developed on any one of the range of powerful machines in the University of London computer complex. It was, however, decided to use the Shrivenham computer mainly because it was a relatively small system very similar to many local authority installations on which programs with a similar

function might be run operationally. One of the subsidiary objectives of the timetabling system was that it should not require a very large computer on which to run. When development started the Shrivenham computer was a 1902A with 32,768 twenty-four bit words of core storage and two 8 million character disc transports. The installation has subsequently been enhanced to a 1903A with 65,536 words of core storage.

The next major decision was the programming language to be used. The initial choice lay between assembler and a high level language. In general, high level languages have the great advantage of focussing attention on the problems to be solved rather than on low level programming detail. In certain parts of complex programs, however, the use of an assembler confers definite advantages notably in economy of storage, in increasing execution speed in frequently recurring computations and by enabling the user to carry out hardware dependent operations such as bit manipulation which may not be readily available in a high level language. Fortunately most current high level language compiling systems provide for mixed language programming and in particular for incorporating segments coded in assembler into programs written in the high level language. It was therefore decided to program the computer timetabling system in such a language using the 1900 assembler PLAN for bit manipulation. The practical alternatives available were FORTRAN and ALGOL 60 and the latter was chosen mainly owing to the author's greater familiarity with ALGOL.

7.2.2 Storage structure

A second major decision concerned storage structure. Storage structure 1 (see 3.4) was rejected because it had been found to be unsuitable for a powerful displacement algorithm. Storage structure 2 is likewise unsuitable and has other shortcomings discussed in 3.7. Moreover both the above storage structures, based on resources, become markedly less efficient as the complexity of the school, as measured by $\bar{c}(\rho)$, increases (see table 3.2). The choice between the two methods based on requirements, storage structures 3 and 4, was more difficult.

The advantage of structure 3, the requirement list, in storage economy is very marked. Furthermore, it retains the important feature of holding the availability of each resource explicitly. The lack of this feature in storage structure 4 makes resource selection during timetable construction impracticable. On the other hand the fact that the availability of each requirement is so readily accessible makes storage structure 4 better suited than storage structure 3 for the implementation of trial assignment and displacement. The evaluation of these algorithms was considered particularly important, and for this reason it was decided to adopt storage structure 4.

This decision was probably the most important taken during the project because while algorithms and peripheral details of the storage structure could be, and frequently have been, readily changed as the project progressed, the basic principles underlying the storage

structure could not be altered without producing a new computer timetabling system.

7.2.3 Overall concept of operations

The third major decision was to adopt the overall concept of operations illustrated in figure 6.3. Following this approach it was decided that the system should consist of four main elements.

1) A data base with which all the programs would interact.
2) Input program.
3) A timetable construction program.
4) An output program.

The timetable construction program in the basic system is not fully interactive but it has been designed to facilitate the implementation of a fully interactive version of this program which is described in chapter 10.

7.2.4 Limitations

The main limitations of the system are as follows.

1) All links between classes, teachers and specialist rooms and requirements must be uniquely specified in the input data.
2) Allocation of non specialist rooms is carried out after the main timetable construction is completed. This has led to the development of a separate room allocation program to be run after the main construction program.
3) No special provision is made for split site schools.
4) All the main arrays in the system have dynamic bounds so that there is no fixed upper limit on the size of problem which can be tackled. This

varies with the computer configuration, and in particular with the core storage available. A 1903A installation with 65,536 words of core storage can just accommodate a problem with 3100 requirements, 1536 resources and a 40 period week with SCOCOST, but this would involve maximum overlaying from disc with consequent very long running time. A practical limit for such a configuration would be 1000 requirements, 1000 resources and 40 periods. Thus the system can cater for all but the very largest schools using a comparatively modest computer configuration.

7.3 Data base

The data base consists of 26 arrays stored on a disc file. Each array starts at a specific logical bucket number. A bucket in this context is an addressable unit of 128 words and a file may for present purposes be regarded as a string of such buckets, each of which is allocated a logical bucket number [38]. The first two arrays, BDATA and NEWDATA, held in logical bucket numbers 1 and 2 respectively, contain the basic information concerned with the problem listed in appendix 2. The third array POINTING stores the logical bucket number at which each of the other 23 arrays start. These arrays are described in appendix 3.

7.4 Input program

The data required by the program is specified in five input forms I1-I5, copies of which are contained in appendix 4. These forms contain a set of data which specifies a problem almost identical to that used in

chapter 2 and subsequent chapters, the only change being the incorporation of rooms. An outline flow chart of the input program is given in figure 7.1.

7.4.1 Main features of the input program

The program includes a comprehensive set of data validation checks the aim of which is to detect all errors on the first run of any set of data. Errors so detected are pointed out by self evident diagnostic messages. Good error detection and diagnostic facilities and "robustness" in the face of data errors are important features of any computer program which has complex data inputs and table 7.1 compares the performance of the latest version of the input program with an earlier version which had a much less comprehensive validation system.

	Earlier version	Latest version
Separate problems (i.e. different sets of data) run	24	32
Runs started	113	55
Runs to completion of program	44	53
Program failures	69	2
Ratio of program failures to runs started	0.60	0.036
Average number of runs started per problem	4.71	1.72

Note: A "run to completion" of the input program does not mean that the data on which the program was run was error free. It simply means that the program did not fail.

Table 7.1 Comparison of two versions of the input program

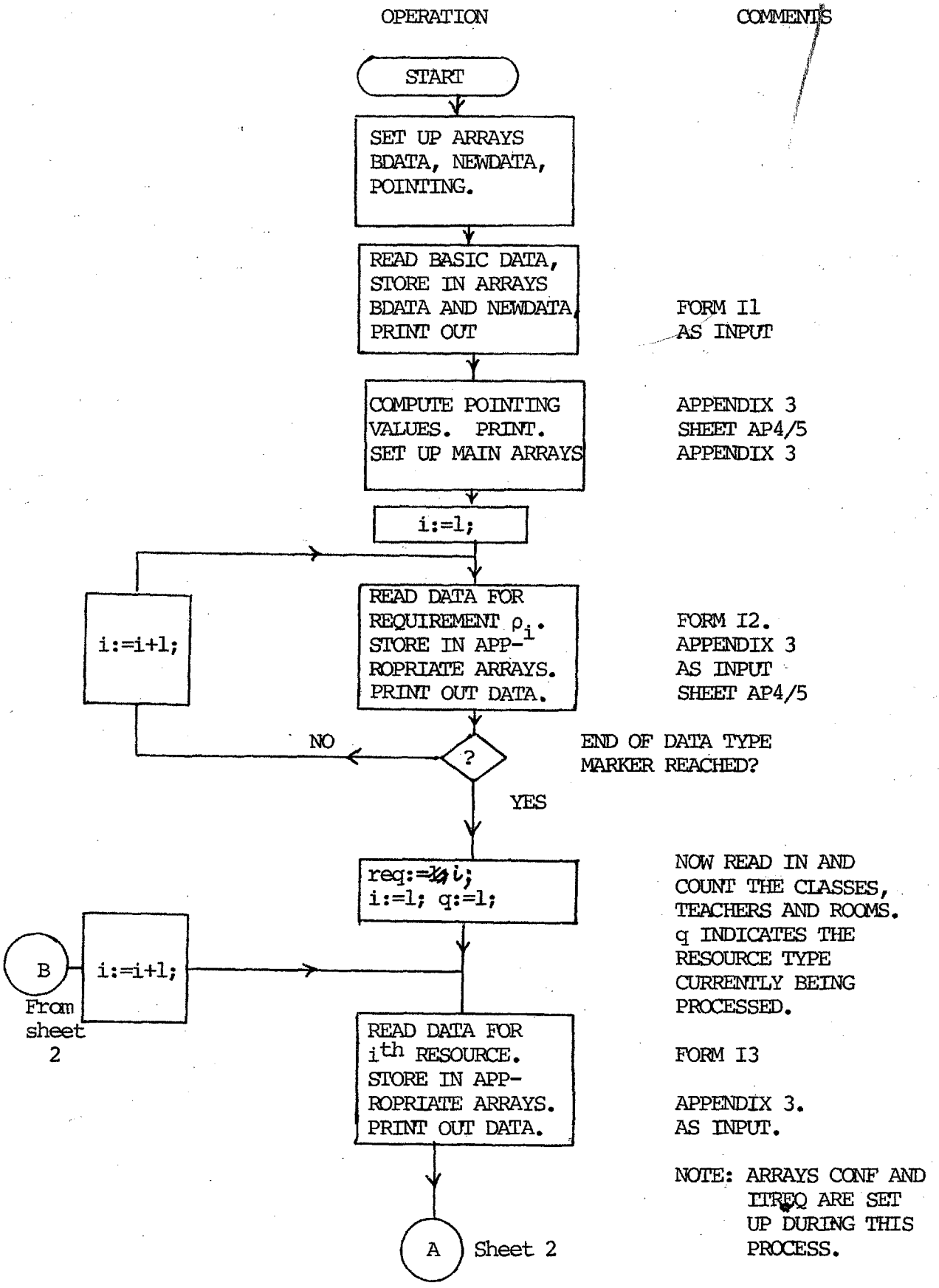


Figure 7.1 Outline flow chart of the input program

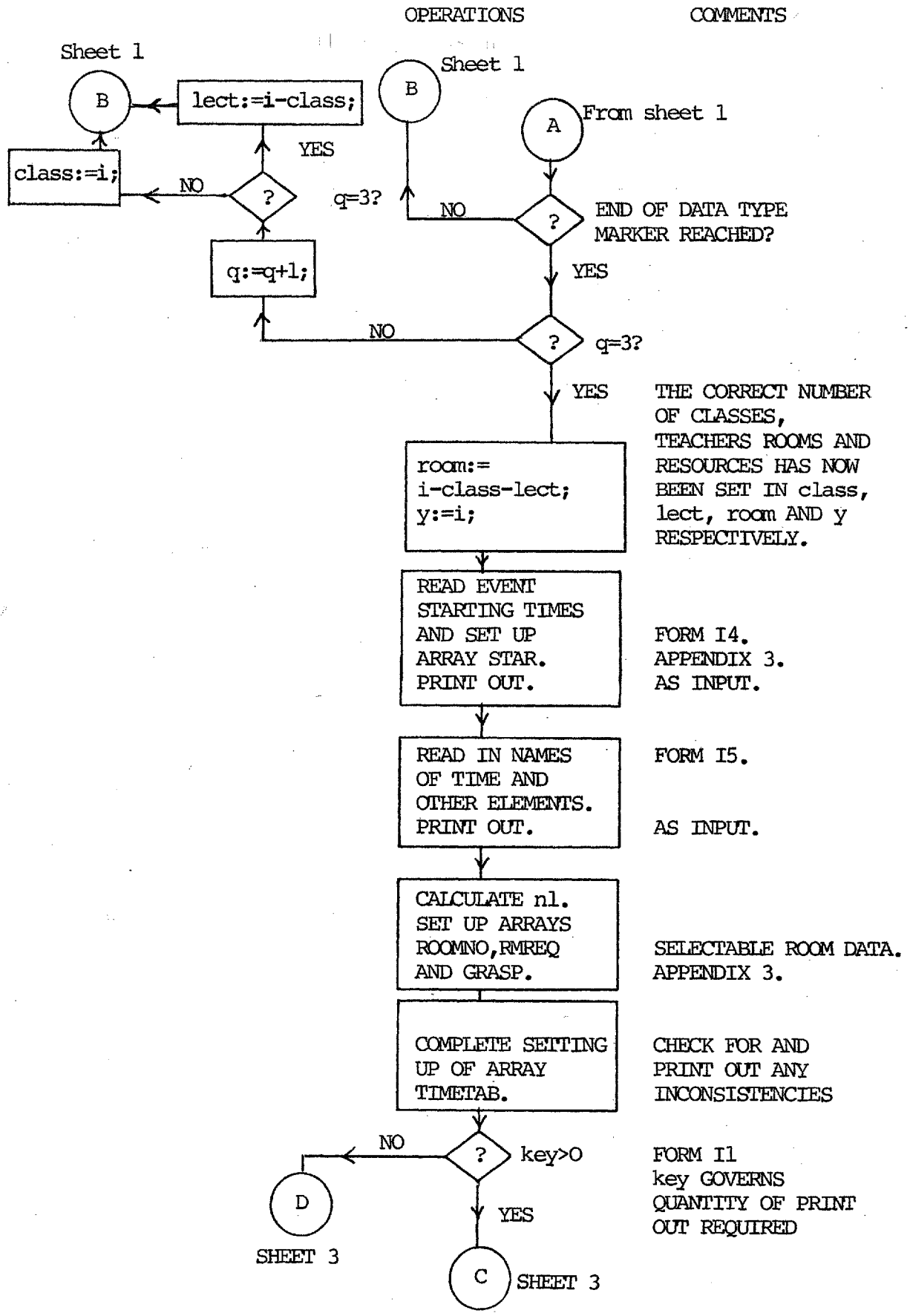


Figure 7.1 Outline flow chart of the input program

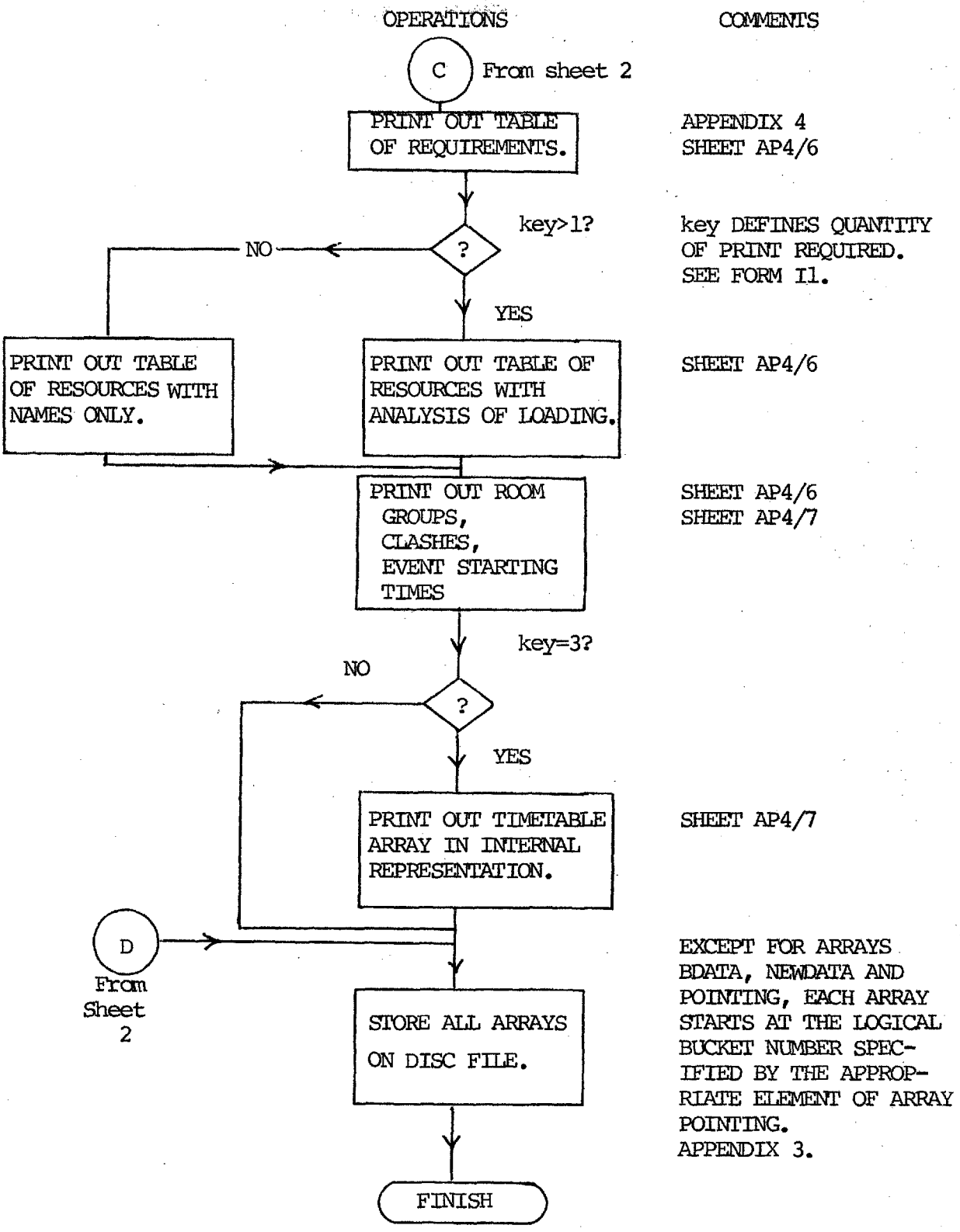


Figure 7.1 Outline flow chart of the input program

In many cases the basic data supplied on form I1 is used only to calculate the appropriate values of the elements of array POINTING, and to set up the various arrays within the input program itself. For example the fourth number on form I1 specifies the value of req, the number of requirements, 10 in the specimen set of data. This number is used to calculate appropriate elements of array POINTING (appendix 3) and to set the bounds of the corresponding arrays, REQPOINT for example, within the program. In fact only 8 requirements were supplied (on form I2); this number is counted by the program and the values of req and the fourth element of array BDATA are amended accordingly. If the number of requirements presented had been 11 or more this would have been pointed out as an error.

Full use is made of the free format facility of the 1900 ALGOL input/output system to input lists of numbers of variable length. There is no need to conform rigidly to the layout of the form as is the case in many computer input documents. Extensive use is made of default settings so that no more information than necessary need be input.

In the great majority of the input data, forms I2, I3 and I5, each line of data, and therefore each card when the data has been punched, contains at least one self evident name.

The quantity of output produced by this program can be varied between wide limits by the data item key, the 14th number on form I1.

Negative numbers are used in a number of input fields to specify information as follows.

Form	Field	Meaning
I2	No. of events	Duration of multiple period events cannot be broken down.
I2	Room group	Rooms are to be selected from the group specified by the magnitude, giving priority to ~~class~~ vendor preference. (A positive number or no number in this field indicates that ~~teacher~~ class preference is to have priority).
I2	No. of rooms	The same selectable rooms are to be allocated for all events of the requirement.
I2	Initial timetable entries	The allocation specified cannot be changed.
I3	Periods not available	Non availability is "absolute".

7.4.2 Discussion

The main novel feature of the data input system is that associations between requirements and resources are established by listing the requirements associated with each resource instead of the usual method of listing resources against requirements. The SCOCOST system is unnecessarily awkward when there are few resources linked with a requirement but it caters well for situations where there is a complex pattern of options with small classes (of one pupil only in extreme cases) with consequent large numbers of resources associated with each requirement.

The "management information" provided by the various output tables, notably that giving resource loadings, has been found to be a valuable feature of

the system.

The main drawback of this scheme for data input is that it is not as easy for the "lay user" to understand as would be desirable and in particular too much use is made of numeric coding.

7.5 Main construction program

As with the input program, the main construction or "compile" program has undergone extensive modification in the light of experience. Several versions of the program exist; namely, the basic version specified in [6], on which the majority of runs have been carried out, and several experimental versions in which the basic structure of the program has been used to investigate the various problem areas discussed theoretically in chapters 4, 5 and 6. The experimental versions are fully discussed in the chapters (9 and 10) dealing with the investigations which they have been designed to support.

The data input to the main construction program is set out in form C1 shown in appendix 4, sheet AP4/8; and an outline flow chart illustrating the operation of the program is given in figure 7.2, with a more detailed flow chart of the assignment routine in figure 7.3.

Requirements are selected for assignment on the basis of slack as defined in 4.5.1, weighted by functions of the maximum outstanding event duration (figure 7.3, sheet 3); the frequency of selection of requirements is governed by the compromise strategy described in 4.7. The selection of a period set for

the event e(i,z) follows the scheme described in 5.6. Procedure DISPLACE (figure 7.3, sheet 2), implements the displacement algorithm described in 6.2.1, using the pure horizontal strategy. The parameter tdepth (form C1) determines the degree of recursion permitted. The value of tdepth can be varied during the run by using test switch 6 (figure 7.2).

The results achieved by this program are described in chapter 8.

7.6 Room allocation program

The process of room allocation is essentially part of the construction process. The present system, however, has a separate room allocation program principally in order to reduce the size of the main construction program. The input data consists of only two numbers specifying the disc file to be used and quantity of print required. The program first allocates rooms which are uniquely associated with requirements to the appropriate events. The process of allocating a room r_m to an event e(i,z) starting at period p involves the following steps:

1) If this is the first room to be allocated to this event, set TIMETAB [i,q] = m for q = p,p+1,...,p+z-1. If it is the second room to be allocated (the first being r_n), then it is necessary to set up a "multiple room". If there are already j-1 multiple rooms, then set TIMETAB [i,q] = j for q = p,p+1,...,p+z-1 indicating that the j^{th} multiple room has been allocated to the event and MULROOM [j,s] = 1 for s = m,n indicating that rooms r_m,r_n belong to the j^{th} multiple room. If the room r_m is the third or

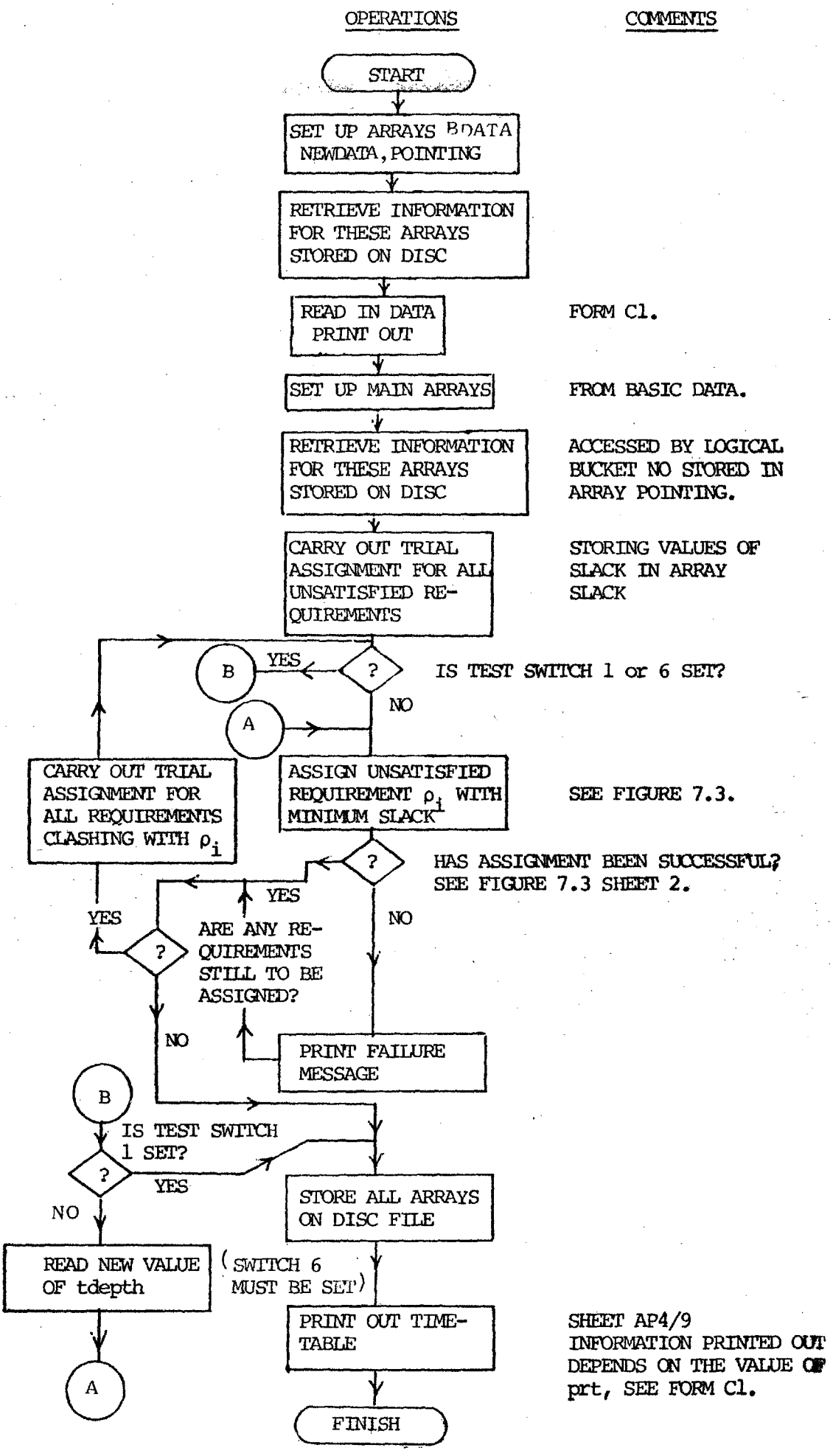


FIGURE 7,2 Outline flow chart of main construction program

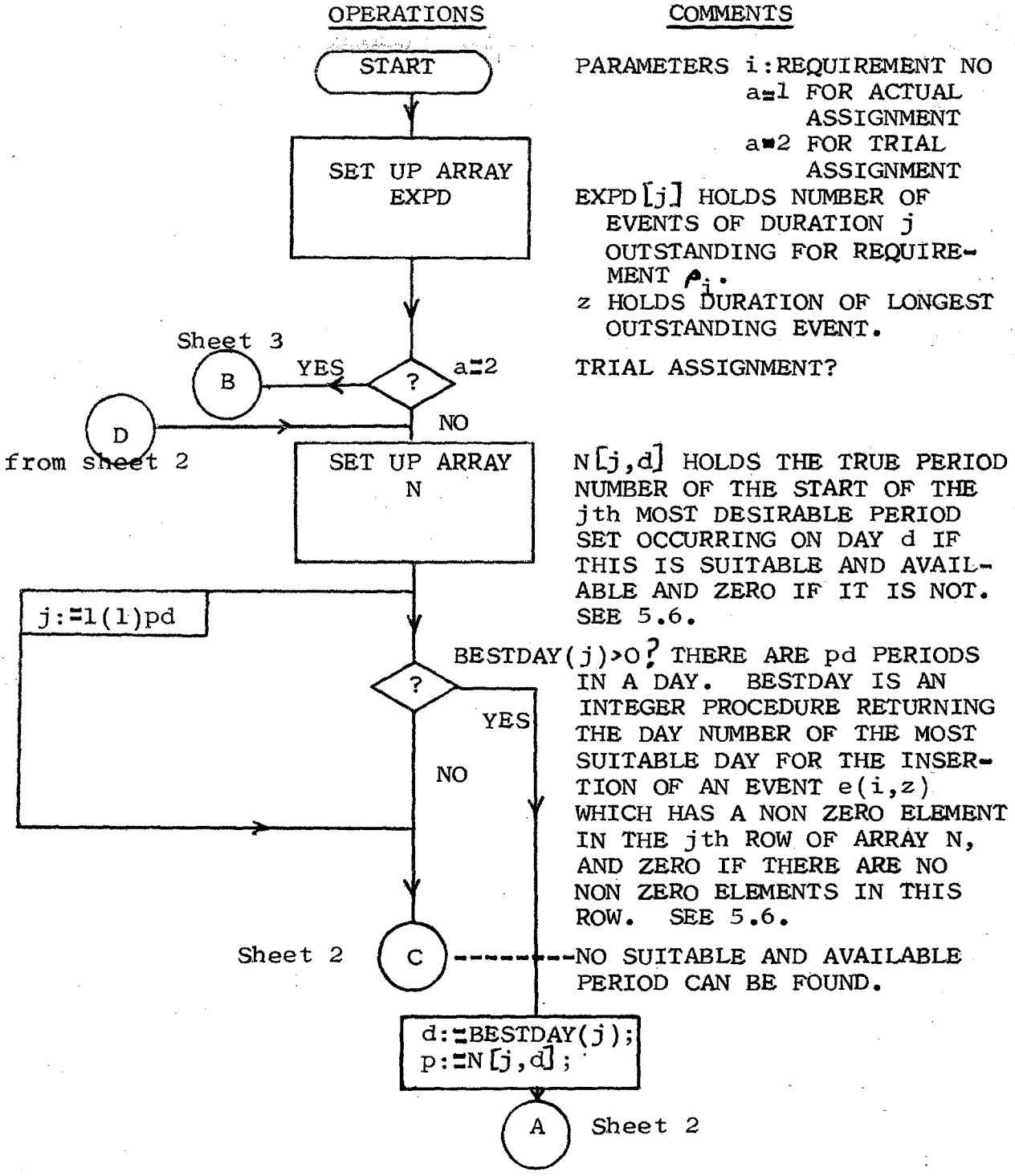


Figure 7.3 Flow chart of procedure ASSIGN

Sheet 1 of 3

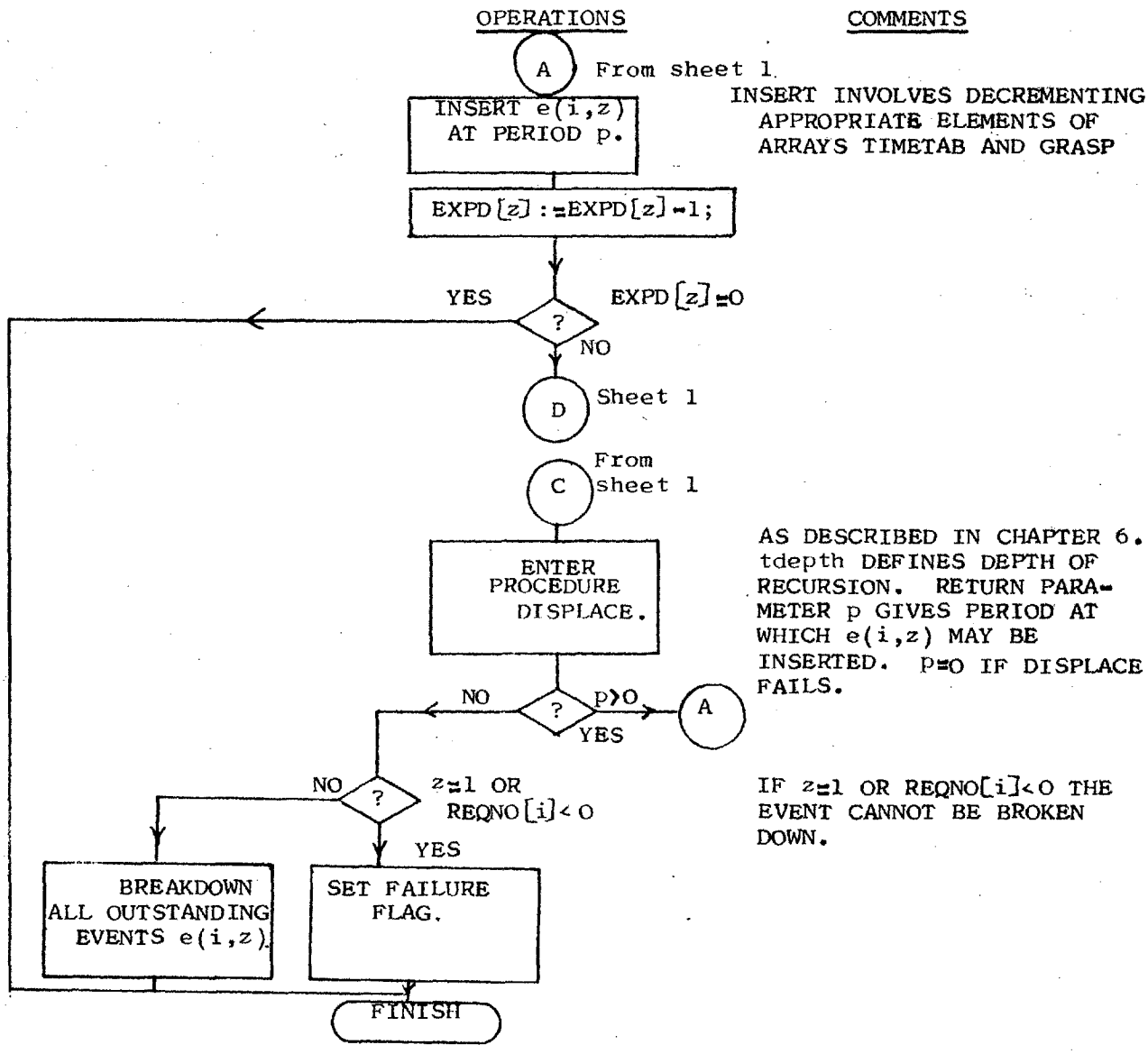


Figure 7.3 Flow chart of procedure ASSIGN

Sheet 2 of 3

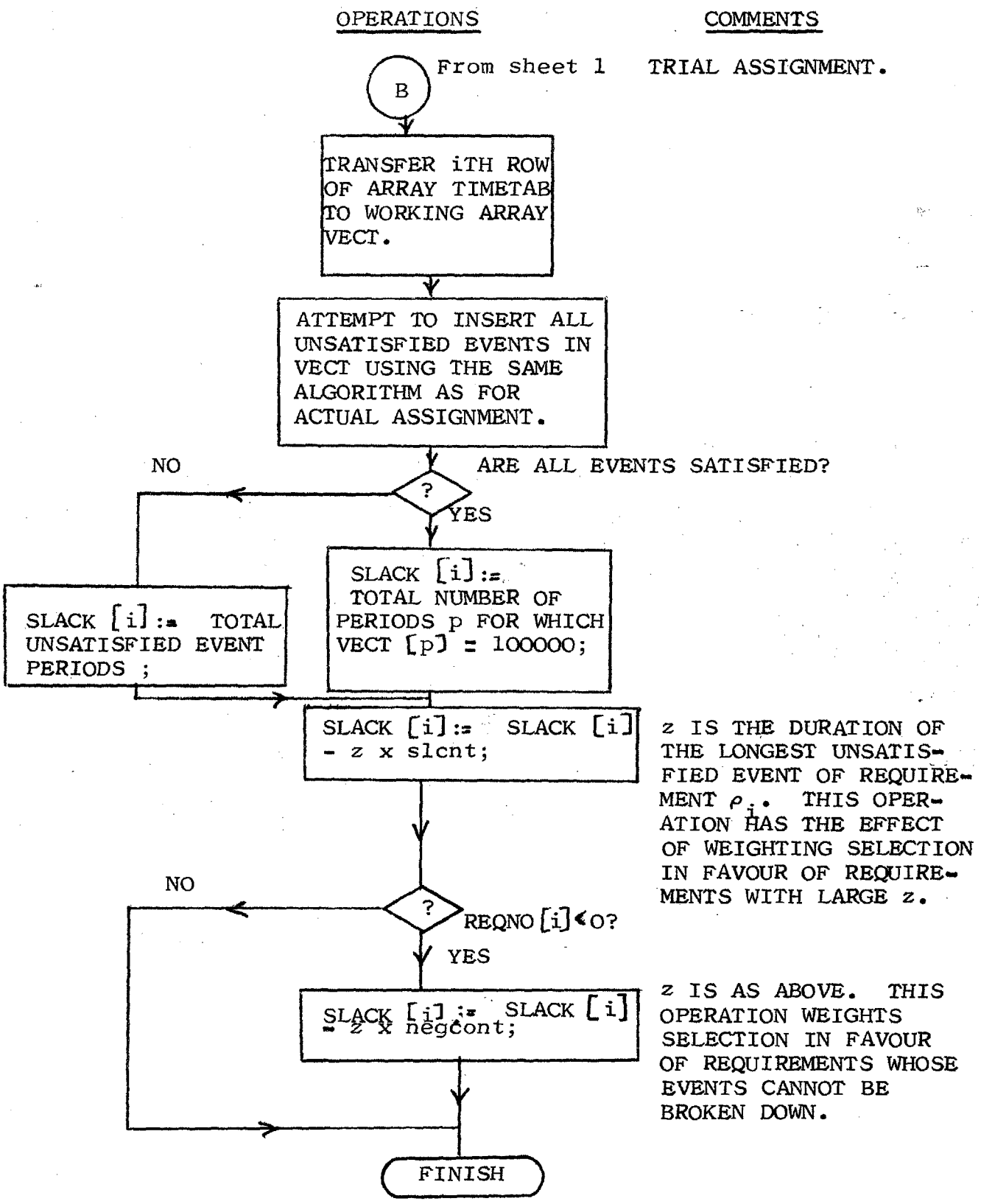


Figure 7.3 Flow chart of procedure ASSIGN

Sheet 3 of 3

subsequent room to be allocated it is only necessary to set MULROOM [j,m] = 1, TIMETAB [i,q] having been set already.

2) Set the bit corresponding to room r_m of columns p,p+1,...,p+z-1 of array RASP to 1 indicating the subsequent non availability of room r_m at these periods.

The elements of a working array NUMROOM [1:req] are now computed for <u>all</u> requirements ρ_i which are incompletely satisfied in respect of rooms by

```
NUMROOM [i]  := m ÷ k;
```

where m is the number of rooms in the room group (the j^{th}, say) for requirement ρ_i and k is the number of rooms required. NUMROOM [i] is thus an indication of the amount of choice of rooms still available for requirement ρ_i.

The requirements for which one or more rooms are still to be allocated are then divided into two categories: those for which all events must have the same selectable rooms (specified on input by a negative value in the "number of rooms" field on form I2), and those for which this condition does not apply. Rooms are then allocated to requirements in the first category in ascending order of their values in array NUMROOM. For each requirement ρ_i in this category, the availability of successive rooms in the room group for that requirement is tested for all periods at which events of ρ_i are scheduled, until the required number of selectable rooms is found. <u>Each</u> of these rooms, r_j, must satisfy RASP [j,q] = 0 for <u>all</u> periods q at which

an event of ρ_i occurs in the timetable. The order in which rooms in the group are tested depends on whether ρ_i is a "static class" or a "static teacher" requirement. For static class requirements (designated on input by a negative number in the room group field on form I2) the order is as follows.

1) Preferred room(s) for the class(es) associated with ρ_i.
2) Preferred room(s) for the teacher(s) associated with ρ_i.
3) Other rooms in the group in ascending order of capacity.

For static teacher requirements the order of 1) and 2) above is reversed.

Rooms are finally selected for requirements in the second category. The order in which these requirements are selected and the order in which rooms within a group are examined are determined by the same criteria as before. However, the availability condition which must be satisfied by a room r_j when being tested for a requirement ρ_i is now less stringent; it is simply RASP[j,q] = 0 for all periods q for which a single event e(i,z) has been scheduled.

When a number of requirements have been specified as needing the same set of selectable rooms for every event the program may fail to satisfy all these requirements in respect of rooms for the reason described in 5.7. In this case an appropriate message is printed out.

7.7 Output program

One of the great advantages of a computer time-

tabling system, particularly when the timetable is retained on backing store, is the ability to present information about the timetable in a wide variety of different forms. The output program provides comprehensive facilities for the user to obtain the information he requires in any one of eight different formats.

The program can operate in one of two modes, standard mode or select mode, which determine how the information to be output is to be defined. The mode is specified by the second number presented to the program (see form 01, sheet AP4/10). In standard mode the timetable for all classes or all teachers or all rooms is printed out in the desired format. The advantage of this mode is that only two numbers are required to be input. Select mode requires more effort in data preparation but gives far greater flexibility than standard mode.

In select mode a set of groups are defined (see form 02, sheet AP4/11). A group may consist of any combination of resources or of requirements, but a mixture of resources and requirements is not allowed. The timetable for individual groups or sets of groups may then be obtained in any desired format using form 03 (see sheet AP4/12). Examples of possible groups for use in select mode are:

All science laboratories.

All rooms in a building.

All teachers and/or rooms in a department.

All Vth and VIth form Science subjects.

The lower VIth form: this may involve many classes. Examples of two timetable outputs are given in appendix 4, sheet AP4/13.

When the program was first designed it was thought that the great flexibility provided by select mode would be most useful. Indeed standard mode was not incorporated initially. Operational experience has shown that select mode has hardly ever been used. This may not necessarily mean that the concept of great flexibility in output groupings is invalid but rather that users are not yet sufficiently experienced in using computer timetabling systems to think in these terms.

CHAPTER 8 - TRIALS AND RESULTS

This chapter describes the more significant trials carried out using the "basic" SCOCOST system and the results achieved. Work in cooperation with Research branch, RAF Training Command including a live trial with RAF Newton is first described after which the results achieved from a large number of runs with several different sets of data are analysed.

8.1 RAF Newton

In early 1971 the Research branch RAF Training Command, in collaboration with the author, carried out a comparative trial of a computer timetabling system developed by Research branch and an early version of SCOCOST. The results of this trial are given in [5], but as objective and thorough comparative trials in this field are rare, the more significant comparisons are reproduced in table 8.1.

Data configuration	Percentage of event periods inserted	
	RAF	SCOCOST
Basic data (RAF Newton)	84.4	95.7
Variation 1 (Newton)	90.4	96.9
Variation 2 (Newton)	91.4	96.5
Variation 3 (Newton)	93.4	97.6
Basic data (RAF Locking)	92.4	99.0

Note: Variation 1 - quadruple and triple period events reduced to doubles and singles.
Variation 2 - restrictions on initial availability of requirements lifted.
Variation 3 - restrictions on instructor availability lifted.

Table 8.1 Comparison of results

As a result of this trial the Research branch decided to abandon work on the computer timetabling system which they had developed and use the Shrivenham system.

After further experiments a live trial was carried out in which six one week timetables for RAF Newton were produced on the Shrivenham computer and used by the school. Each week's timetable was, of course, quite different from its predecessor, reflecting the changed pattern of courses from week to week. The input data for a week's timetable was liable to change up to a few days before the start of the week concerned. There were thus important time constraints in the trial. The data for each week was prepared and validated, the timetable produced, subjected to extensive manual checks, altered where necessary and issued within the space of one week.

The more important results of the trial are given in table 8.2.

Week commencing	No. of classes	Number of event periods not fitted	Insertion rate	Human effort required in completion and amendment of final timetable hours
			%	hours
17.11.71	41	36	97.8	12
25.11.71	41	-	-	Timetable not produced
2.12.71	39	17	98.9	Not used
9.12.71	41	12	99.3	6
6. 1.72	40	0	100	2
13. 1.72	37	8	99.5	3
20. 1.72	40	7	99.6	3

Table 8.2 Details of live computer timetabling trial

The first week's timetable had a number of shortcomings, partly owing to incorrect preparation of the data but mainly because the computer system did not permit the user to specify which multiple period events could be broken down to events of shorter duration. This failing was corrected by alterations to the appropriate programs but these delayed the runs for the second and third weeks, the timetables for which were not used. The program amendments allowed requirements whose multiple period events could not be broken down to be specified by a negative number in the events column (form I2, sheet AP4/2). These amendments dramatically improved the timetables produced in the third and subsequent weeks.

The trial was a success in that the computer produced timetables were accepted and operated by teachers and students with very little trouble. In a few cases, however, practical lessons were placed before the theory lessons on which they were based. This shortcoming reveals a minor limitation of the system. The need to specify that a given event must take place before another is not catered for as it is not important in the majority of secondary schools.

A comparison of costs in [5], taking into account the notional costs of computer time, human effort and travelling, and stationery, shows that the average cost of a computer timetable was about £200, of which £140 was for a total of 4¼ hours computer time, while that for a manual timetable was £50, representing about 35 hours human effort. This does not, however, give a

fair comparison between the potential of the two methods since the manual system is well established, while the computer system is only in the research and development stage. Clearly if such a system were fully operational, the need for the extensive checks carried out during the trial would be greatly reduced with increased user familiarity. An interactive system which permitted the user to amend data on line would be most helpful in this context (see chapter 10). Furthermore program run times can be greatly improved: better organisation of disc files and an improved system for retrieving names in the output program have already doubled the speed of this program since the trial.

Making all reasonable allowances for potential economies in the method for producing timetables by computer, the cost might be reduced to £60, still above the cost of a manually produced timetable. Thus the use of a computer to construct each week's timetable "from scratch" for a "short course" school such as RAF Newton is unlikely to be cost effective. The reason for this is mainly that the construction of the weekly timetable is only part of the overall timetabling problem.

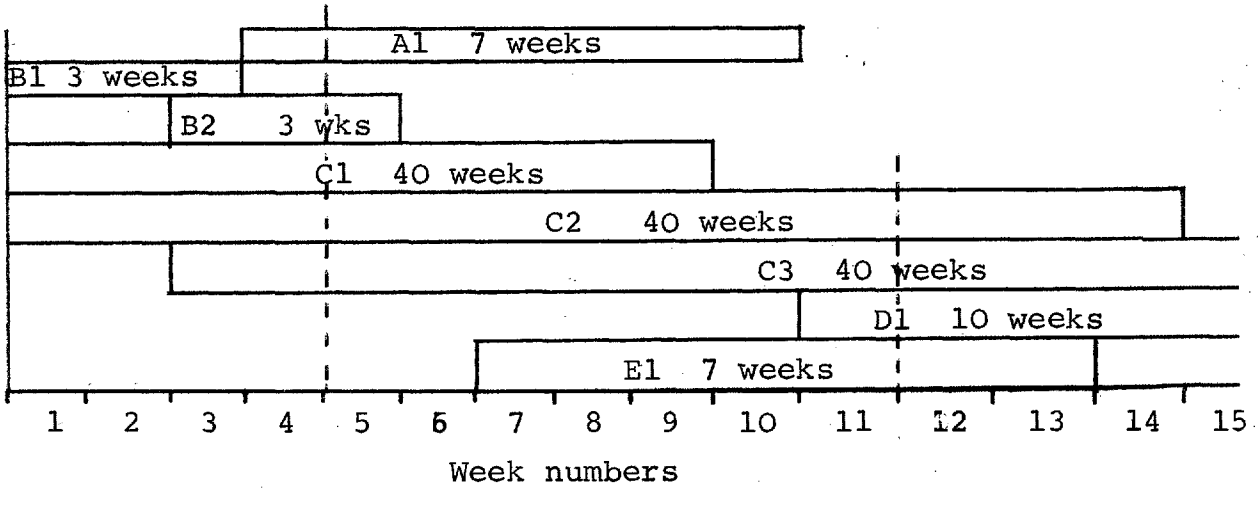


Figure 8.1 Pattern of courses at a short course school

Figure 8.1 shows five different types of course, A, B, C, D and E. Each time a particular type of course is run the same pattern of instruction is followed. Thus in week 3, course B2 is undergoing the same instruction as B1 underwent in week 1. Thus the timetable for week 5 must provide for courses A1 (second week), B2 (third week), C1 (36th week), C2 (31st week), C3 (third week) while that for week 12 must provide for course C2 (38th week), C3 (10th week), D1 (second week) and E1 (sixth week). A computer can be used effectively for such a school to store the profiles of each type of course in terms of its requirements for each week. The system can then be provided with the starting dates of every course that is run so that a pattern of the school's activity such as shown in figure 8.1 is stored. This can be used firstly as an aid to long term planning to indicate when particular resources (or groups of resources) are over or under-used and secondly to generate the input to the weekly timetabling program. This would not only remove the need for human data collection and punching every week but would make the task of planning the school's future activities and its need for resources much less laborious and more efficient. Some work has been done in this area but as it is not directly relevant to the timetabling problem in a British secondary school it is not pursued further in this thesis; readers interested in this field are referred to [37].

8.2 Data sets, difficulty factors and system performance

The computer timetabling system described in chapter 7 has been used to process data from the RMCS and four

schools, Burnage High School, Aylesbury Grammar School, The Elms and RAF Newton. The data for the first two schools was used in a field trial of the University of Birmingham timetabling system and was supplied to the author by the Local Government Operational Research Unit for testing purposes. Data from The Elms has been obtained directly from the school but the timetables produced have not been used. Timetables produced for RAF Newton were used during the trial described in 8.1. Table 8.3 gives details of the sets of data and the results achieved by the system in each case. An explanation of the three difficulty factors used is given below.

The first two difficulty factors in the table were used by Barraclough [4]. Factor a is:
(the number of teacher periods required)/(the number of teacher periods available),
while factor b is:
(the number of teacher periods in sets or double periods)/(the number of teacher periods available).

Difficulty factor c, which is original, is given by

$$D_t = \frac{\sum_{i=1}^{\rho} (x_i \sum_{z=1}^{Z} z.a_{iz})}{P.\rho - Q},$$

where ρ, P and Q are the number of requirements, periods in the week and event periods respectively, Z is the longest duration of any event in the problem, x_i is the number of requirements clashing with requirement ρ_i and a_{iz} is the number of events of ρ_i of duration z.

D_t is a more complex factor than either of the previous factors and attempts to measure the "tightness"

School/College	RMCS	Burnage HS	Aylesbury GS	The Elms GS	RAF Newton
Requirements	100	424	281	215	262
Events	285	946	476	467	641
Event periods	290	1103	550	587	1523
Resources					
Classes	52	43	18	89	41
Teachers	79	71	39	29	60
Rooms	29	62	16	25	80
Teacher periods					
Available	1560	2515	1507	980	2400
Required	440	1894	950	734	1389
Number in sets or doubles	416	1091	661	541	1295
Clashes (see note (i))					
$\bar{x}$	12.02	29.33	55.53	32.3	13.26
Difficulty factors (see explanation on pages 151 & 153)					
a	.282	.752	.610	.750	.587
b	.267	.436	.420	.552	.540
c	1.821	2.713	2.750	2.736	2.047
a+b	.549	1.188	1.030	1.302	1.127
System performance (see note (ii))					
Event periods not fitted	0	17	4	23	2
Percentage insertion rate	100	98.5	99.2	96.1	99.9
Time (min) (see note (iii))	8	104	50	62	18

Notes (i): $\bar{x}$ is the average number of requirements clashing with any one requirement.

(ii): The insertion rates quoted are the best achieved by the basic system with one level of recursive displacement (tdepth = 2, see 7.5).

(iii): Printing time is not included.

Table 8.3 Details of sets of data and system performance

of a hypothetical solution to the given problem. The justification for D_t is as follows. Suppose that storage structure 4 is in use and that a solution to the problem exists. In the solution timetable the $k_i = \sum_{z=1}^{Z} za_{iz}$ event periods of requirement ρ_i will generate $x_i k_i$ clashes, so that the total number of clashes will be $K = \sum_{i=1}^{P} x_i k_i$. Thus there will be a total of K decrements from the elements of the timetable array T and these must be "spread" over the total number of array elements less the number representing the event periods which have been assigned. Therefore the K decrements must be "spread" over $P\rho - Q$ elements, so that D_t is a measure of the number of decrements which a typical unassigned element T[i,p] would undergo during the construction of the solution timetable. It is thus an indication of how many events would have to be displaced in order to free the period p for an event of requirement ρ_i and is therefore an indication of how difficult this process would be.

There is no consistent relationship between any of the difficulty factors and the insertion rate achieved by the system. On the limited evidence afforded by table 8.3, however, Barraclough's measure, a, and the linear addition of measures a and b provide a significantly better prediction of system performance than measure c devised by the author. Measure c makes Aylesbury markedly more difficult than any of the other data sets and yet the system performance was better for this set of data than for two sets of the other four sets.

The reason for the high value of difficulty factor c for Aylesbury was the large number of clashing requirements (as shown by $\bar{x}$), due mainly to the fact that the VIth form is only divided into three separate classes, with each VIth form requirement involving several members of staff so that a large number of optional subjects must take place simultaneously. This contrasts with the Elms where every one of the 77 girls in the VIth form is regarded as a separate class. However, McIlroy [29] appears to contradict the superiority of Barraclough's measures indicated by table 8.3. In runs with the same program with 64 small sets of synthetic data, each with 32 requirements and a variable number of resources, McIlroy found that measure c gave a good prediction of the insertion rate achieved while there was virtually no relationship between the insertion rate and the two Barraclough measures. While results based on synthetic data must be treated with caution, McIlroy's work has cast considerable doubt on any conclusions regarding the relative merits of different measures of difficulty which can be drawn from table 8.3. It must therefore be admitted that no reliable predictor of likely system performance against a given set of timetable data has been found.

The number of event periods appears to be a reasonable predictor of running time in the case of the three secondary school timetables.

8.3 Insertion rate, time and tdepth

A feature of SCOCOST is that the user can determine how hard the main construction program "tries" to find a solution, by the parameter tdepth. Variation of tdepth

alters the degree to which use is made of the procedure DISPLACE, as follows:

tdepth =	use made of DISPLACE
0	not used.
1	Used but without recursion.
2	Used with one level of recursion permitted.
.....	
n	Used with n-1 levels of recursion permitted.

Table 8.4 gives the insertion rates and running times for 12 sets of runs. In each set of runs the data and all control parameters are identical, the only difference between the runs in a set being the value of tdepth.

School	Values of tdepth							
	0		1		2		3	
	Insertion rate %	Time min	Insertion rate %	Time min	Insertion rate %	Time min	Insertion rate %	Time min
Burnage	91.9	14	94.2	32	98.5	104		
Burnage	91.2	18	93.9	36	97.9	110		
Burnage	92.1	16	94.1	36	98.3	108		
Aylesbury	90.6	10	94.1	18	96.1	51		
Aylesbury	93.5	10	96.1	19	98.8	51		
Aylesbury	93.3	10	96.6	17	98.8	50		
Aylesbury	93.9	11	97.0	18	99.2	50		
Aylesbury	93.5	12	96.7	28	98.6	90		
The Elms	89.7	12	94.0	21	96.1	62	97.4	213
The Elms	89.5	12	93.4	26	95.9	76		
The Elms	89.5	12	93.4	28	96.0	77		
RMCS	96.5	5	99.4	7	100.0	8		

Note: Due to the very long computer running time involved only 1 run at tdepth = 3 has been carried out.

Table 8.4 Insertion rate and running time for different values of tdepth

From table 8.4 it can be seen that, as expected, both

running time and insertion rate increase with each increment in tdepth. Table 8.5 gives the ratio of running times and the difference in insertion rates for variations of tdepth.

School	t_0/t_1	t_1/t_2	t_0/t_2	i_1-i_0	i_2-i_1	i_2-i_0
Burnage	.435	.308	.134	2.3	4.3	6.6
Burnage	.5	.346	.163	2.7	4.0	6.7
Burnage	.444	.333	.149	2.0	4.2	6.2
Aylesbury	.555	.352	.196	3.5	2.0	5.5
Aylesbury	.526	.372	.196	2.6	2.7	5.3
Aylesbury	.588	.34	.2	3.3	2.2	5.5
Aylesbury	.612	.36	.22	3.1	2.2	5.3
Aylesbury	.428	.311	.133	3.2	1.9	5.1
The Elms	.572	.339	.193	4.3	2.1	6.4
The Elms	.461	.342	.158	3.9	2.5	6.4
The Elms	.428	.363	.156	3.9	2.4	6.3
RMCS	.715	.875	.626	2.9	0.6	3.5

Note: t_n, i_n are the running time and percentage insertion rate at tdepth = n respectively.

Table 8.5 Ratios of running times and differences in insertion rates

Table 8.5 shows that, apart from the RMCS results where the very high insertion rate with tdepth = 1 led to distortion, the insertion rate achieved at tdepth = 0 appears to be a reasonable predictor of performance at tdepth = 2, and on this evidence it is an appreciably better predictor than the insertion rate at tdepth = 1. It is also a much better predictor than any of the difficulty factors used in table 8.3.

Performance at tdepth = 0 is not such a good predictor of running time as insertion rate: running time at tdepth = 1 is significantly better in this respect.

Thus for a cost of between 1/5th and 1/8th of the cost of a run with tdepth = 2 it is possible to obtain a reasonable prediction of the insertion rate on such a run. This could be valuable for educational planning and curriculum development, to be discussed in chapter 11.

8.4 Quality of timetables

User reactions to the timetable used during the live trial at RAF Newton have already been discussed. None of the other timetables have been put into operation, and therefore two quantitative measures have been used to assess the quality of the timetables produced from the five sets of data. These measures are modifications of those used by the Local Government Operational Research Unit [7] in assessing the results produced by the Newcastle upon Tyne University system.

The first measure of quality is the spread of events of the same requirement across the week. Table 8.6 gives the information on the quality of the spread achieved by the system on the five sets of data. The second measure is the degree to which multiple period events cross long or short breaks: relevant figures for this measure for the five sets of data are given in table 8.7.

Table 8.6 shows that the system achieved higher quality timetables on the two easier sets of data, RMCS and RAF Newton. The reason for this is that the spread algorithm's insistence on a normal distribution is relaxed to a limited extent when the DISPLACE procedure calls itself recursively. This clearly occurs more frequently in a more difficult problem. The relatively

Number of events	Distribution see note (i)	Points awarded see note (ii)	RMCS		BURNAGE		AYLESBURY		ELMS		RAF		Totals	
			No	Pts	No	Pts	No	Pts	No	Pts	No	Pts	No	Pts
2	10100	2	42	84	63	126	24	48	35	70	54	108	218	436
	11000	1	17	17	30	30	24	24	21	21	25	25	117	117
	20000	0	0	0	2	0	0	0	1	0	1	0	4	0
	Totals see note (iii)		59	101 / 118	95	156 / 190	48	72 / 96	57	91 / 114	80	133 / 160	339	553 / 678
3	10110	3	2	6	14	42	5	15	10	30	9	27	40	120
	11100	2	1	2	10	20	3	6	5	10	1	2	20	40
	20100	1	0	0	4	4	0	0	3	3	0	0	7	7
	21000	0	0	0	6	0	0	0	1	0	2	0	9	0
	Totals		3	8 / 9	34	66 / 102	8	21 / 24	19	43 / 57	12	29 / 36	76	167 / 228
4	11110	4	27	108	29	116	10	40	20	80	2	8	88	352
	21010	3	0	0	7	21	4	12	0	0	1	3	12	36
	21100	2	1	2	3	6	1	2	1	2	0	0	6	12
	30100	1	0	0	0	0	1	1	0	0	0	0	1	1
	31000	0	0	0	0	0	0	0	0	0	0	0	0	0
	Totals		28	110 / 112	39	143 / 156	16	55 / 64	21	82 / 84	3	11 / 12	107	401 / 428
5 and more see note (iv)	11111	5	1	5	15	75	4	20	6	30	19	95	45	225
	21110	4	1	4	24	96	14	56	12	48	10	40	61	244
	22010	3	0	0	5	15	2	6	1	3	0	0	8	24
	22100	2	0	0	1	2	0	0	0	0	0	0	1	2
	31010	1	0	0	1	1	2	2	1	1	0	0	4	4
	31100	0	0	0	0	0	0	0	0	0	0	0	0	0
	Totals		2	9 / 10	46	189 / 230	22	84 / 110	20	82 / 100	29	135 / 145	119	499 / 595
	Grand totals		92	228	214	554	94	232	117	298	124	308	641	1620
	Possible points			249		678		294		355		353		1929
	Percentage			91.6		81.7		79.0		84.0		87.4		83.6

Table 8.6 Spread of events of the same requirement across the week

Notes:

(i) The codes given in the distributions column are illustrated by the following examples:

10100 means one event on two days of the week separated by one blank day.

11000 means one event on two consecutive days of the week (Friday and Monday are regarded as consecutive).

20000 means two events on one day.

(ii) Points have been awarded for different distribution types as shown, on an empirical basis, in order that a numerical assessment of the distributions achieved compared with the "ideal" distribution may be obtained.

(iii) The two figures separated by a diagonal in the total points positions give total points actually achieved and the maximum possible points.

(iv) In the few cases where there are more than 5 events in a requirement, points have been awarded as if there were 5 events. Thus a 6 event 22110 distribution is awarded 4 points.

Table 8.6 Spread of events of the same requirement across the week

Event duration	Crossing	RMCS	BURNAGE	AYLESBURY	ELMS	RAF	TOTAL
Double period	no breaks	2	127	65	81	240	515
	short break	0	0	0	8	3	13
	long break	0	0	0	0	0	0
Triple period	no breaks	0	6	1	13	59	79
	short break	0	2	0	0	0	2
	long break	0	1	0	0	0	1

Table 8.7 Multiple period events crossing breaks

poor performance with 5 event requirements was due to the fact that in all the runs the value of SPREAD (see appendix 4, form C1, sheet AP4/8) was 1, permitting the construction program to schedule events in a 21110 distribution without attempting displacement if suitable clear periods could not be found for a 11111 distribution. As might be expected there is a trade-off between quality and insertion rate as table 8.8 shows.

Set of data	Value of spread	Quality	Insertion rate %	Time minutes
RMCS	0	94.7	99.6	10
	1	91.6	100	8
Burnage	0	90.0	97.9	110
	1	81.7	98.5	104
Aylesbury	0	88.0	98.6	90
	1	79.0	99.2	50
Elms	0	86.8	95.9	76
	1	84.0	96.1	62
RAF Newton	0	88.5	99.9	23
	1	87.4	99.9	18

Table 8.8 Trade-off between quality, insertion rate and time

The almost total absence of multiple period events crossing long breaks shown by table 8.7 is due to the use of permitted event starting times (see figure 7.3, sheet 1) to prevent this occurring. The only exception to this was in the Burnage data when triple period events were allowed to start at any period of the day from 1 to 5 inclusive. Double period events were prevented from crossing short breaks in this way for all sets of data except the Elms and RAF Newton: in these data sets no constraints were placed on doubles crossing short breaks and it is interesting that this occurred in only a small percentage of cases.

Unfortunately the two comparatively simple measures described do not give a complete picture of timetable quality. Other factors, such as the quality of the timetable in respect of each individual teacher and class, will also play an important part in determining the degree to which the timetable satisfies the whole school and indeed whether it is acceptable for use. Again the question of spread of subjects over the week is not really a simple quantifiable parameter as has been indicated. For subjects involving homework an adequate spread is probably vital, particularly if there are only two lessons in the week. On the other hand with certain non-homework subjects one head teacher expressed the view that she would be quite satisfied not only with two lessons on one day but with the only two single period lessons in a week run into one double period lesson. Caution is thus necessary in drawing conclusions from the figures on timetable quality given in this

chapter, but it is believed that these figures do provide a reasonable overall assessment of performance in this area.

8.5 Comparison of manual and computer timetables

It is surprisingly difficult to obtain exact comparisons between computer produced timetables and those produced manually. The main reason for this is that the final version of a manually produced timetable invariably represents a complete solution but often the problem that has been solved is different in certain respects from that originally defined, and the departures from the original problem have been made in a series of decisions during the lengthy timetable construction process. When the timetable is complete it is often difficult to assess the exact implications of these departures in relation to what would have been achieved if the original problem had been adhered to exactly. Furthermore it is not easy to arrange an exactly parallel exercise in which computer and human being start with identical data. Fortunately, however, this was done during the live trial at RAF Newton. A manually produced timetable for the Elms from a set of data similar but not identical to that used for the computer runs has also been analysed to obtain an indication of how computer and manually produced timetables compare in a secondary school. Results of these two comparisons are given in table 8.9.

Quality criterion	RAF Newton		Elms	
	Computer	Manual	Computer	Manual
Distribution				
2 events per week	83.1	86.1	87.6	77.0
3 events per week	80.5	84.2	75.5	80.5
4 events per week	91.6	94.0	97.6	98.1
5 events per week	93.1	95.5	82.0	92.9
Overall	87.4	90.3	84.0	89.5
Percentage of multiple period events crossing breaks				
doubles/short break	1.2	0	9.0	0
doubles/long break	0	0	0	0
triples/short break	triples must cross a short break		0	23.1
triples/long break	0	0	0	0

Table 8.9 Comparison of computer and manually produced timetables

It can be seen that the manually produced timetables for these two schools are slightly better than those produced by computer.

The superiority of the manual system is hardly surprising since nearly all manually produced timetables have evolved over a period of years, so that the construction procedures used by the person constructing the timetable have reached a high degree of refinement in that the timetables produced satisfy most of the important criteria for acceptability in the school concerned. But not only do the construction procedures evolve: the acceptability criteria evolve also, with staff and pupils coming to accept (or possibly becoming resigned to) the kind of timetables which are produced. This process of evolution of course gives the manual system a great initial

advantage over a computer system which, when it is introduced in a given school, inevitably starts from scratch. After a computer system had been introduced, however, the process of evolution would undoubtedly start working in its favour.

CHAPTER 9 - INVESTIGATIONS INTO ALGORITHMS

This chapter describes investigations into algorithms for testing for the existence of solutions and for event and time selection.

9.1 Existence

The extended Gotlieb test, as specified in inequality (4.1) and applied to real resources only, was incorporated in the input program. This was applied to the five sets of data described in table 8.3. Four sets satisfied the test; the Burnage data failed in respect of two resources as shown in table 9.1.

Resource	Number of event periods required	Number of event periods for which at least one associated requirement is available
Vth form class	35	34
Room	17	15

Table 9.1 Existence test failures in Burnage data

The reason for the failure in respect of the Vth form class was that the teachers and rooms associated with that class were preset as not available for a high proportion of the week. The room was initially not available for 18 out of the 35 periods and none of the requirements associated with this room was available for two out of the remaining 17 periods. It would appear, however, that the extended Gotlieb test applied to real resources only is not powerful enough to reveal all the infeasibilities in a given set of data. For example, although the Elms

data passed this existence test, there are good grounds for believing that a complete solution to the problem as defined by this set of data does not exist. Two ways of strengthening the test have been investigated and are described below.

9.1.1 The creation of dummy resources

A limitation of the extended Gotlieb test applied to real resources only is its inability to take account of tight sets generated by combinations of such resources. The first way of improving the effectiveness of the extended Gotlieb test is to create "dummy resources" corresponding to these tight sets and then to apply the test to all resources, real and dummy. Earlier discussion of this strategy in 4.2 envisaged making use of <u>all</u> tight sets whether or not they are subsets of other tight sets. In practice this is quite impracticable since it gives rise to a very large number of dummy resources; a resource associated with a set of n requirements generates $\sum_{m=2}^{n-1} {}_nC_m = 2^n-n-2$ additional tight sets, all of which are subsets of the original set.

The following algorithm overcomes the above problem by enabling the user artificially to limit the number of tight sets created.

For every pair of resources, y_j, y_k say, the following sets of requirements are established: S_1 common to y_j and y_k, S_2 associated with y_j but not y_k and S_3 associated with y_k but not y_j. If S_1 contains v or more members, where v is a user defined parameter, then dummy resources are created for every requirement $\rho_{m_1}, \rho_{m_2}, \ldots, \rho_{m_n}$ in S_2 with which at least one requirement in S_3 clashes.

Suppose $S_1 = \{\rho_{i_1}, \rho_{i_2}, \ldots, \rho_{i_r}\}$ and a requirement ρ_{m_j} in S_2 clashes with ρ_s, ρ_t and ρ_u in S_3 then an additional tight set (dummy resource) S_T is created where $S_T = \{\rho_{i_1}, \rho_{i_2}, \ldots, \rho_{i_r}, \rho_{m_j}, \rho_s, \rho_t, \rho_u\}$, since the set of requirements $\{\rho_{m_j}, \rho_s, \rho_t, \rho_u\}$ itself forms a tight set each member of which clashes with every member of S_1. The algorithm makes no attempt to distinguish the tight sets from each other and can therefore generate many identical dummy resources. Surplus dummy resources may be removed either by a further computer process or manually before the extended Gotlieb test is carried out.

An example from the RMCS data illustrates the operation of this algorithm. Three classes in the Highway and Bridge Engineering Honours stream are associated with requirements as shown in table 9.2

Resource number	Resource name	Associated requirements
1	23 HBE H A	170 31 150 151 101 103
2	23 HBE H B	170 128 150 151 101 103
3	23 HBE H C	170 21 150 151 101 103

Table 9.2 Requirements associated with three classes

The sequence of operations carried out by the algorithm is illustrated in table 9.3.

Resource pair	S_1	S_2	S_3	Additional tight set	Dummy resource number (see note)
1,2	170,150,151, 101,103	31	128	Nil - 31 and 128 do not clash	-
1,3	170,150,151, 101,103	31	21	170,150,151,101, 103,31,21	161
2,3	170,150,151, 101,103	128	21	170,150,151,101, 103,128,21	162

Note: There are 160 real resources in the RMCS data, therefore dummy resources start at 161.

Table 9.3 The creation of additional tight sets

The two dummy resources, numbers 161 and 162, are treated from the moment of their creation as if they were real resources and their combination may produce further dummy resources. Each is associated with 7 requirements, one more than the number associated with each of the three real resources which generated them. The practical value of this algorithm is limited by the large number of operations required, which is approximately proportional to Y^2P where Y is the number of resources and P the number of requirements considered. It was run on the RMCS data with v = 4, taking into account only the 52 classes. The computer running time was 60 minutes. No infeasibilities were revealed, of course, since the construction program had demonstrated the feasibility of the problem by achieving 100% insertion rate.

The experience with the RMCS data emphasized the need for careful user control of the algorithm if excessive computer running times are to be avoided. The strategy adopted was to divide up the resources and

requirements into groups and apply the algorithm to a group at a time. Of course this precludes comparison of resources or requirements from different groups, but this limitation can be lessened by good grouping which in practice is not difficult to achieve. This procedure was applied to the Elms data, groups being selected from a scrutiny of the input data. Generally, resources with five or more requirements in common were regarded as candidates for grouping. This led to five groups each with about 30 resources and processing time averaged 15 minutes per group. A value of $v = 8$ was used throughout and 211 dummy resources were created of which 19 were distinct.

The extended Gotlieb test was applied to the dummy resources and failure was registered for one resource which had been generated by a group of VIth form girls. This showed that a more powerful test is produced where dummy as well as real resources are involved. The process was not repeated for the other three sets of data as these seemed much less likely to yield "valuable" dummy resources than the RMCS or the Elms, both of which have groups of classes with several associated requirements in common.

9.1.2 Existence in relation to the solution algorithm

A weakness of the extended Gotlieb existence test, even when applied to all possible tight sets in the problem, is that it is a simple count of periods and pays no regard to the need for certain events to occupy a set of consecutive periods, nor to the need to spread events of the same or tied requirements evenly across

the week.

A further test which follows the same general line as the extended Gotlieb test is to attempt to construct a timetable for each real and dummy resource separately. Failure in respect of any one resource does not prove the problem to be inherently insoluble since this may have been due to deficiencies in the construction algorithm. On the other hand success for every resource only means that a number of subsets of the problem are solvable individually, and there is no guarantee of solvability when all these subsets are combined. Thus this test cannot be held to establish either necessary or sufficient conditions for the inherent solvability of any timetabling problem. But it does effectively establish necessary conditions for solution by the construction algorithm in use since failure to solve a subset of a problem almost certainly indicates that the algorithm will be unable to solve the complete problem. It is also far more powerful than the extended Gotlieb test in that it does take into account additional constraints and may therefore be expected to reveal more apparent infeasibilities. These draw attention to particular areas of difficulty in the problem which can then be examined manually.

The test was applied to the Elms data and four apparent infeasibilities were revealed. One of these concerned a real resource, a part time teacher whose initial pattern of available periods made it impossible to satisfy the normal distribution constraint for one 4-event requirement. The other three infeasibilities

were produced by dummy resources. Exhaustive trial revealed that a timetable could not be constructed for any of these resources so that the apparent infeasibilities generated must be regarded as genuine.

The essential difficulty in testing for the existence of a solution is that while simple numerical comparisons such as the extended Gotlieb test have the advantage of establishing necessary conditions unambiguously, they take no account of many of the factors and constraints which make the timetabling problem so complex. Tests similar to that described, which do take account of such factors, introduce a degree of ambiguity in their results. The purpose of the present work has been to reduce the area of uncertainty about the existence of a solution once a set of timetabling data has satisfied all the simple validity conditions. It may fairly be claimed that this purpose has been achieved even if no completely satisfactory test has been found.

9.2 Event selection

9.2.1 Slack versus choice

The relative effectiveness of the three measures of current difficulty mentioned in 4.5.1, slack s_i, choice for a requirement G_i, and choice for the longest outstanding event of a requirement g_{iz} was investigated by a series of 10 runs, the results of which are summarised in table 9.4.

Data set	tdepth	Slack s_i		Choice for a requirement G_i		Choice for the longest outstanding event g_{iz}	
		Insertion rate %	Time Mins	Insertion rate %	Time Mins	Insertion rate %	Time Mins
Elms	0	89.5	12	89.7	13	90.0	13
	1	93.4	26	93.3	29	93.7	28
	2	95.9	76	96.3	90	96.2	84
Burnage	0	91.2	18	91.2	20	91.4	20
	1	93.9	36	94.1	41	94.6	39
	2	97.9	110	97.4	122	98.6	119
Aylesbury	0	93.5	10	93.2	10	93.6	10
	1	96.1	19	96.5	21	96.7	21
	2	98.8	51	99.0	58	99.1	55
RMCS	0	96.5	5	96.7	5	96.7	5

Table 9.4 Effectiveness of different measures of current difficulty

Table 9.4 shows that g_{iz} produces results which are consistently better than the other two measures, but the superiority is marginal, an average of 0.32% over G_i and 0.39% over s_i. As might be expected the simplest measure, slack, consistently produced the fastest running times with an 8.5% average advantage over g_{iz} and a 12.7% advantage of G_i. It may be concluded from this evidence that there is little advantage to be gained by using choice rather than slack as a measure of current difficulty.

9.2.2 Weighted slack

In the early version of the SCOCOST construction program, the only parameter used for event selection was slack. In runs using various sets of data it was found that of the events not fitted a large proportion were multiple period events. Considered in conjunction with the theoretical limitations of slack in relation to such events discussed in 4.5.1, this fact led to inclusion of a weighting factor slcnt (appendix 4, form C1) to weight selection in

favour of requirements with outstanding events of long duration. It was hoped that for any one set of data there would be a value of slcnt yielding the optimum insertion rate. This has proved not to be the case. In all about 160 runs have been carried out with different sets of data and values of slcnt in the range -1.0 to 5.0, but no consistent relation between slcnt and insertion rate was found. A set of controlled runs summarised in table 9.5 may be taken as representative.

slcnt	Event periods not fitted	Triple period events not fitted	Double period events not fitted
0.0	24	1	4
0.5	23	0	4
1.0	23	0	4
1.5	25	1	3
2.0	26	0	3
2.5	23	1	2
3.0	24	0	2

Table 9.5 Variation of insertion rate with slcnt using Elms data with tdepth = 2

As might be expected, the contribution which multiple period events make to the number of event periods not fitted falls with increasing slcnt.

9.2.3 Inherent difficulty

Chung [8] has examined a number of inherent difficulty criteria using an early version of SCOCOST. The data used for most of his investigations was a set of RMCS data made artificially more difficult by adding a number of extra events to the requirements. He concluded that none of the criteria examined taken singly was as effective as

slack. The number of requirements x_i clashing with a requirement ρ_i, and the number of resources associated with a requirement were found to be more effective criteria than the others examined. A suitably weighted combination of either or both of these criteria and slack produced marginally better insertion rates than that using slack alone. Chung found that both criteria were about equally effective and it was therefore decided in the present work to consider the effect of one in more detail. The criterion chosen was the number of clashing requirements, x_i. Chung's conclusions were tested by comparing the criteria x_i and s_i (slack) when applied to the five sets of data mentioned in table 8.3. To remove any distortions caused by conflict resolution algorithms, tdepth was zero for all runs. The results are summarised in table 9.6, and support Chung's conclusions on the relative merits of s_i and x_i taken singly.

Criterion	Symbol	RMCS	Burnage	Aylesbury	The Elms	RAF Newton
Slack	s_i	95.8	91.2	93.5	88.9	94.1
Number of clashing requirements	x_i	92.0	88.6	87.1	85.5	92.2

Table 9.6 Slack and number of clashing requirements compared

The possibility of using a simple learning algorithm to find the optimum relative weighting of s_i and x_i was investigated next. The requirement selection criterion was a weighted slack r_i, given by

$$r_i = s_i - \frac{w.x_i}{\bar{x}} \qquad (9.1)$$

where w is the weight to be attached to clashing requirements, and

$\bar{x}$ is the average number of requirements clashing with any one requirement.

The learning algorithm varied w whenever a requirement ρ_i could not be satisfied according to the relation,

$$ {}_{n}w = {}_{o}w + at \qquad (9.2) $$

where $a = +1$ if $\frac{x_i}{\bar{x}} - 1 \geq$ a preset tolerance value v,

$a = -1$ if $1 - \frac{x_i}{\bar{x}} \geq v$,

$a = 0$ if $\left|\frac{x_i}{\bar{x}} - 1\right| < v$,

t is an increment,

${}_{n}w$ and ${}_{o}w$ are the "new" and "old" values of w respectively.

A series of runs on the Elms data with tdepth = 0 were carried out in sets of 5 with constant values of v and t in each set but with values of w transferred from run to run. All combinations of the following values of v and t were tried:

v 0.1, 0.2, 0.4

t 0.1, 0.2, 0.5

Insertion rates from the 45 runs varied from 86.5, slightly worse than using slack alone, to 89.6, slightly better. There was, however, too little stability or consistency in the runs to enable firm conclusions to be drawn and in none of the sets of runs did w show any sign of settling to a particular value. A further parameter, b, was introduced to enhance stability, with

equation (9.2) modified to

$$ {}_{n}w = {}_{o}w + b\ a\ t \qquad (9.3) $$

where b = 1 if a has been either +1 or -1 consistently over the last m occasions, and 0 otherwise, m being a user defined parameter.

With m = 2, there was a significant increase in stability. The best insertion rate was achieved under the following conditions: v = 0.2, t = 0.1 and m = 2. Under these conditions an insertion rate of 89.8% was achieved with tdepth = 0. Unfortunately the insertion rate using these parameters with tdepth = 2 was 96.0, slightly worse than that achieved using the basic system!

A final trial using no requirement selection criteria other than order of input was run on the Elms data. The insertion rate was 80.1% with tdepth = 0, and 90.4% with tdepth = 2, significantly worse than any previous results.

9.2.4 Frequency of selection

A series of tests were run to investigate the effect of varying the strategy governing the frequency with which events are selected. The three strategies discussed in 4.7 were examined using two sets of data, with results given in table 9.7.

The insertion rates achieved with the two values of tdepth follow the same pattern with both sets of data; strategies 1 and 2 produce very similar insertion rates with the insertion rate achieved by strategy 3 slightly worse in all cases. The pattern of running time is rather different. With tdepth = 0, the running times increase

Strategy	The Elms				Aylesbury			
	tdepth = 0		tdepth = 2		tdepth = 0		tdepth = 2	
	Ins.%	Time min.	Ins. %	Time min.	Ins.%	Time min.	Ins.%	Time min.
1. Selection after every event.	90.0	15	95.8	71	93.9	14	98.4	54
2. Selection after all the longest outstanding events of a requirement have been satisfied.	89.7	12	96.1	63	93.9	11	99.2	50
3. Selection after every complete requirement.	88.5	10	95.2	76	91.4	9	97.9	68

Table 9.7 Comparison of selection frequency strategies

with increasing selection frequency as might be expected. With tdepth = 2, the running time of strategy 3, with the lowest frequency of selection, is significantly longer than that for either of the other two strategies. A poorer event selection strategy has led to considerably more demand for conflict resolution leading in turn to longer running times (and slightly worse insertion rates).

The results in table 9.7 broadly confirm expectations; the compromise strategy 2, which is used in the basic system, is significantly better than strategy 3 but nothing is gained by increasing the selection frequency still further.

9.3 Time selection

The basic system selects times for events without attempting to establish that the assignment of an event to a set of periods is compatible with a solution to the problem. The main reason for this is that any one of the existence tests discussed so far involves a high cost in

computer time if applied frequently in the course of time-table construction, for example after the insertion of every event. When the basic assignment strategy was designed it was considered that this cost was too high for the benefit likely to be obtained. The validity of this assertion was tested by applying the extended Gotlieb test, inequality (4.1), after the assignment of every event e(i,z), to all resources associated with any of the requirements clashing with requirement ρ_i. Dummy resources generated by the algorithm described in 9.1.1 were included in the tests. The basic assignment strategy specified in figure 7.3 was modified to:

1) Select a set of periods p,p+1,...,p+z-1 for e(i,z) in accordance with the criteria shown in figure 7.3.
2) Insert the event into the timetable at these periods.
3) Carry out the existence test. If this test is satisfied, continue as for the basic system strategy: if not, delete the event, return to 1) and select a different set of periods.

If no set of periods can be found which satisfies the existence test then failure is acknowledged and the event e(i,z) is omitted.

The effectiveness of this strategy was compared with that of the basic system with two sets of data, and the results are given in table 9.8.

Row	Performance measure	The Elms			Aylesbury		
		Basic system		Existence test	Basic system		Existence test
		tdepth=0	tdepth=2		tdepth=0	tdepth=2	
1	Insertion rate, %	89.7	96.1	91.2	93.9	99.2	94.9
2	Time, mins.	12	62	104	11	50	78
3	Number of times existence test caused a change to an insertion	-	-	29	-	-	21
4	Number of times the test failed to reveal a conflict subsequently detected by the normal assignment procedure	-	-	14	-	-	8
5	Number of insertions attempted	-	-	431	-	-	444

Table 9.8 The effectiveness of a post insertion existence test

The figures given for Aylesbury are in reasonably good agreement with the performance achieved on the same data by the University of Birmingham system [18] which also uses an existence test as part of its time selection strategy. This system achieved an insertion rate of 95.8% after a running time of 74 minutes on a KDF 9 computer.

The evidence of table 9.8 suggests that existence testing is a relatively inefficient mechanism for time selection. For a high price in computer running time it has produced insertion rates which are significantly worse than those achieved by the basic system with tdepth

= 2.

9.4 Discussion

The investigations into algorithms described in this chapter have on the whole confirmed expectations. The work on existence testing has shown that the uncertainty as to the solvability of a given problem can be reduced somewhat by the use of tests applied to a number of subsets of the problem individually. These tests are more valuable practically for sets of data in which groups of resources exist each containing resources with a significant proportion of associated requirements in common, giving rise to additional, "dummy",resources.

Investigations into the effect of varying event selection strategies have shown that strategies using current difficulty as the sole event selection criterion produce higher insertion rates than those using inherent difficulty. A marginal improvement is obtained by using a suitable inherent difficulty parameter in conjunction with a measure of current difficulty. Any of the measures of difficulty tried as event selection criteria produce better insertion rates than selection by order of input only. Selection after all the longest outstanding events of a requirement have been satisfied appears to be the best frequency of selection.

The work on time selection suggests that associating an existence test with the time selection process is of doubtful value. It improves on the insertion rate achieved by the basic system without displacement at a heavy cost in computing time, but compared with the basic system with displacement it produces a poorer insertion rate and longer running time.

CHAPTER 10 - TIMETABLE CONSTRUCTION BY MAN-COMPUTER TEAM

The justification for investigating a team approach to timetable construction has been given in 6.5, and the possibility of interactive timetabling has also been briefly discussed by Lions [28] and Johnston [22]. This chapter describes the first implementation of this concept, and the investigations carried out using it.

The interactive version of the timetable construction program uses a visual display unit (VDU) connected to the ICL 1903A computer at the RMCS, and makes use of VDU software routines written by Farley [14]. It consists of the following main functional components which are described in 10.1 to 10.5:

1) Setting up facilities.
2) The commentary.
3) The command language INTERACT.
4) Additional facilities available when a conflict is detected.
5) Closing down facilities.

10.1 Setting up facilities

Before the program can be used operationally, the user must specify, via the VDU Keyboard, the disc file on which the data for the current problem is stored and other parameters which control the operation of the program. These initial parameters are similar to those required for the basic system (appendix 4, form C1) except that an additional inherent difficulty parameter clashcont has been introduced. This is used to weight slack so as to cause the event selection process to

favour requirements which have many requirements clashing with them compared with those with few such requirements. Its effect is similar to that of the weighting factor w in equation (9.1). All the slack weighting factors, slcnt, negcont and clashcont are input as integers for convenience and then divided by 100 having been read in. The initial data required for the program is given in table 10.1.

Data Name	Example Value	Remarks
file	-1	As form C1
tdepth	2	As form C1
spread	0	As form C1
prt	1	As form C1
slcnt	240	Values are divided
negcont	100	by 100 when number
clashcont	100	has been read in.

Table 10.1 Initial data

When the initial data has been read in, the various arrays required by the program are set up and certain basic information about the problem and certain of the control parameters are displayed on the top 4 lines of the screen, which are described in 10.2. Timetable construction then begins.

10.2 The commentary

The commentary is designed for a number of purposes. In theory the most important justification for it is that it enables the user to monitor the timetable construction process and to intervene and alter the program's dispositions if he is not satisfied. In practice the commentary has been found to have other uses, notably

in keeping the user "involved" with the construction process even when he is not taking an active part in it. It also provides a most illuminating insight into the operation of the program itself.

The information provided by the commentary depends on the operations currently being performed by the program, for example ASSIGN, TRIAL ASSIGN, DISPLACE, and also on a parameter "display" which is set up by the user via the command language INTERACT.

Plates 10.1 and 10.2 illustrate the information provided by the commentary.

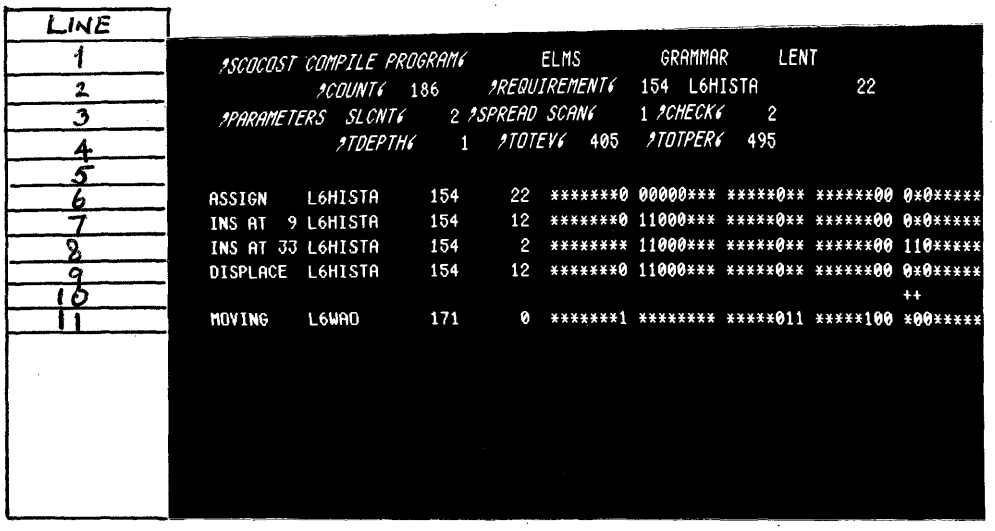


Plate 10.1 The Commentary

An explanation of each line of plate 10.1 follows.

Line 1. The information in this line is self evident: the name of the school and the year (or term) to which the timetable relates.

Line 2. COUNT gives the number of requirements still to be fitted, 186 in this case. Then after REQUIREMENT the

code number, name and number of events outstanding of the requirement currently being processed are given.

Line 3. After PARAMETERS the slack control parameter is displayed together with two parameters concerned with the spread of events over the week, scan and check. The parameter scan is the minimum separation in periods between any two events of the same requirement and check is the departure from a normal distribution which is acceptable. The value of 2 for check indicates that as a last resort a second event of a requirement may be inserted on a day when there are two other days with no events of that requirement.

Line 4. The figure following TDEPTH gives the value of tdepth, the parameter which controls the depth of recursion to which the displacement procedure is used (see 8.3), and TOTEV and TOTPER give the number of events and event periods respectively still to be fitted.

Line 5. Blank.

Line 6. This and subsequent lines illustrate the actual process of timetable construction. In plate 10.1 a requirement named LHISTA, code number 154 is being assigned. The number 22 shows that there are two double and two single period events outstanding. On the right of this are five sets of eight characters indicating the state of the timetable for this requirement for every one of the eight periods of each of the five days in the week according to the following legend:

0 available, 1 allocated, * not available.

This line only reflects the state of the appropriate row of the timetable array; it does not take account of

periods not being available for other reasons. For example, it can be seen that periods 31 and 32 (the last two periods on Thursday) are shown as being available. In this set of data these periods have been set aside for "all school" activities by the use of the event starting time facility in the input data (appendix 4, form I4). Thus in this problem single period events are not permitted to start at periods 31 and 32, double period events may not start at periods 30, 31 and 32 and so on. This non-availability is not reflected in the timetable array and therefore the display does not take account of it.

Line 7. This shows the insertion of one double period event at period 9.

Lines 8, 9, 10 and 11. These lines are less readily understandable on a static photograph than when viewed dynamically on the screen during the running of the program. The operations involved are explained on the flow chart in figure 10.1.

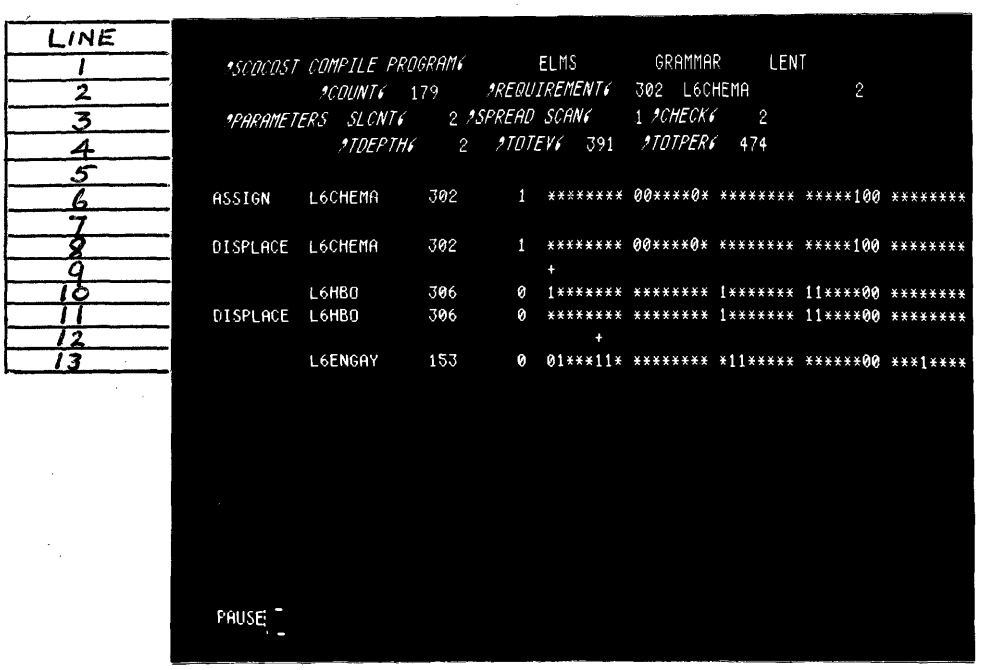


Plate 10.2 The recursive use of DISPLACE

OPERATION	LINE	REMARKS
-	7	A second double period event cannot be fitted in anywhere in the time-table, (except Thursday afternoon which cannot be used, see above).
ENTER PROCEDURE DISPLACE	9	
TRY THE PERIOD SET 33, 34	10	The marker "++" indicates the period set being tried. Note that LHISTA is only not available at period 34. A single period event of L6WAO caused this.
MOVE AN EVENT OF REQUIREMENT L6WAO CLEAR OF PERIOD 34	11	
INSERT DOUBLE PERIOD EVENT OF LHISTA AT PERIOD 33	8	

Figure 10.1 Displacement operations on plate 10.1

Plate 10.2 shows the recursive use procedure DISPLACE. Here the program is attempting to insert one single period event of L6CHEMA. Three periods on Tuesday appear to be available but an event of a tied requirement has been inserted on this day so these periods cannot be used. Period 1 is being tried (see the "+" marker on line 9) and a single period event of L6HBO is preventing insertion

at the period. DISPLACE is being used recursively on this requirement and period 6 is being tried (see the "+" marker on line 12). A double period event of L6ENGAY is preventing the displacement of the single period event of L6HBO from period 1 to period 6.

A suitable value of a parameter called display can cause the slack figures after every trial assignment to be displayed as an additional part of the commentary but this facility is seldom used as it significantly increases running time, and the information is normally of little operational use although it is valuable for program checking and tuning. Another value of display suppresses all the commentary except lines 1 to 4. This parameter is initially set to a default value which causes the information shown on plates 10.1 and 10.2 to be displayed but it may be altered by the user at any time with the VARY command, see table 10.2 command number 16.

10.3 The Command language INTERACT

At any time during timetable construction, the user can interrupt the program and make use of the parameterised command language INTERACT, which gives him a wide range of facilities. The language is summarised in table 10.2

Command number	Command	Parameters	Meaning
1	ASSIGN	r	Assign requirement whose code number is r.
2	TRIAL ASSIGN	r	Trial assign requirement r.
3	REMOVE	r	Remove all events of requirement r from the timetable.
4	DELETE	r p z	Delete an event of duration z of requirement r starting at period p.
5	INSERT	r p z	Insert an event of duration z of requirement r starting at period p.
6	NOLINK	r RESNAME	Dissociate resource RESNAME from requirement r.
7	ADDLINK	r RESNAME	Associate resource RESNAME with requirement r.
8	CLASHEV	r p z	Display events clashing with an event of requirement r of duration z starting at period p.
9	CLASHREQ	r m	Display requirements clashing with requirement r. m=1 for requirement names. m=2 for requirement numbers.
10	DISPRES	r	Display the resources associated with requirement r.
11	TIMEREQ	r n	Display the timetable for requirement r. The parameter n governs the symbols used in displaying the timetable (see note (i)).
12	NEWREQNO	r q	Change the events requested for requirement r to q (see note (ii)).
13	CHANGERES	r	Change the resources associated with requirement r (see note (iii)).
14	TIMERES	RESNAME n m	Display the timetable for resource RESNAME, n and m govern the format of the timetable (see note (iv)).
15	DISPREQ	RESNAME m	Display the requirements associated with resource RESNAME. m=1 for requirement names. m=2 for requirement code numbers.
16	VARY	NAME x	Change the data variable NAME to x.
17	INVERT	s	Invert program switch setting, (see note (v)).
18	LIST		Display a summary of command and variable names.
19	CHANGENAME NEWNAME	n OLDNAME	Change OLDNAME of type n to NEWNAME. n=1 for requirement name. n=2 for resource name. n=3 for miscellaneous name. n=4 for command name.
20	GO		Continue with the program.

Table 10.2 The Command language INTERACT
Sheet 1 of 2

Notes:

(i) The display of a requirement timetable for the TIMEREQ command uses one of two sets of symbols depending on n. For n=1 "stars form" is adopted, in which the symbol for each period can be 0 for clear period, 1 for event allocated and * for period not available. For n=2 "non stars form" is used giving more information with the symbol for each period being one of the following alternatives:

0	Clear period.
@	An event has been allocated at this period but it can be moved.
※	An event has been allocated at this period which cannot be moved.
£	This period is not available for allocation under any circumstances.
1-9	This period is not available due either to events of clashing requirements having been allocated at this period, in which case the number of such events is given or to initial non-availability of resources, in which case the symbol is always 9.

(ii) The parameter q for NEWREQNO must be specified in coded form. Thus if it were desired to allocate one triple and three single period events to requirement 157, the Command "NEWREQNO 157 103" would be input.

(iii) CHANGERES enables several resources to be dissociated and several others associated with a requirement in one command, (compare with ADDLINK and NOLINK).

(iv) If n=1 TIMERES provides the availability of the resource for every period of the week. If n=2, the requirement name or number (depending on m) associated with RESNAME is given for every period.

(v) Program switches in the range 0-8 can be used to control the operation of the program. The most important switch is 1 which when set causes the program to enter the closing down routine (see 10.5).

Table 10.2 The Command language INTERACT
Sheet 2 of 2

It has been found essential to provide the user with a definite response to each command on the display. For example on execution of NOLINK or ADDLINK, the "new" set of resources associated with requirement r is displayed.

The way in which most of the commands can be used individually is self evident. Combinations of commands, however, provide facilities which are less obvious. For example a complete timetable, or the parts of it requiring really close human control, can be built up by "hand" using commands 1 to 5 with the display commands such as TIMERES where necessary. Another potentially useful facility is the creation of new requirements. A number of dummy requirements may be included in the input data and not associated with any resources. During timetable construction it may be necessary to 'split a requirement into two'. For example suppose there is a requirement for 2 double and 4 single period events, and that the doubles must take place in a specialized room while the singles should do so if possible. In the event of serious trouble in satisfying this requirement it might be acceptable to create a new requirement not associated with the specialist room for the 4 single periods, leaving only the 2 double period events associated with this room. One of the dummy requirements mentioned above could be used as the new requirement and the commands CHANGERES, NEWREQNO and CHANGENAME can accomplish the required alterations smoothly and rapidly.

The command VARY is a very powerful facility although it requires the user to have a thorough knowledge of the program. All the parameters concerned with requirement

selection and timetable quality can be varied in this way. The parameters tdepth controlling the depth of recursion available to procedure DISPLACE, and display controlling the amount of information provided by the commentary can also be altered using VARY.

10.4 Additional facilities

The facilities provided by INTERACT give the user virtually unlimited scope for varying the data in any way he pleases in order to obtain an acceptable timetable. It was quickly found, however, that when the computer program fails to satisfy a requirement, the user must have more help in deciding what should be done than is provided by INTERACT. When the program fails to fit one or more events of a requirement, the user's first reaction is to ask "why?". With INTERACT alone a number of commands would be necessary simply to obtain the availabilities of the resources associated with the requirement. Once the user has obtained this information he may be no further forward because it may well be far from clear what possible courses of action might be appropriate and helpful. A series of trial and error operations using INTERACT in these circumstances can easily take 20 minutes and still fail to resolve the difficulty. It must be remembered that the user is trying to solve a problem which has defied a very powerful computer algorithm; furthermore it is just this type of difficulty which makes the construction of timetables by purely manual means such a lengthy process. Clearly in these circumstances the computer must give the user all possible help if "long periods of thinking time at the terminal"

(to use Johnston's words [22]) are to be avoided.

This philosophy has been adopted. As soon as the program fails to fit a requirement, the fact is reported on the display together with the following information, which is illustrated in plate 10.3.

1) The timetable for the requirement in "non stars" form.
2) The resources associated with the requirement.
3) The resources preventing the insertion of the unsatisfied event at each period.

```
 %SCOCOST COMPILE PROGRAM%        ELMS       GRAMMAR     LENT
          %COUNT%  188     %REQUIREMENT%  167  L6BIOLABX      -100
 %PARAMETERS  SLCNT%     2 %SPREAD SCAN%     1 %CHECK%     2
              %TDEPTH%     1   %TOTEV%  411   %TOTPER%  508
L6BIOLABX      167  NOT FITTED REMAINING    1 EVENTS OF DURATION    3
TIMETABLE   L6BIOLAB    167   -100  00111111 00111101 01111101 00111000 11111110

RESOURCES   BUTTAR      CASE        HUMPHREYS   KIMBER      TASKER
MARSHALL
ST.PRD  PRD       RESOURCES
   1      1
          2
          3   KIMBER
   9      9
         10
         11   BUTTAR      HUMPHREYS   TASKER
  25     25
         26
         27   BUTTAR      CASE        HUMPHREYS   KIMBER      TASKER
MARSHALL
   2      2
DO YOU WANT MORE PERIODS PLEASE INPUT 1 FOR YES 0 FOR NO

INPUT COMMAND
```

Plate 10.3 Information display in the event of failure

The display often cannot contain all the information under 3) above and the user is asked, near the bottom of the display, if he wishes to see the same information for more periods. In the case of an unsatisfied multiple period event, of duration z say, the "ST. PRD" column gives the starting period of a set of z consecutive

periods which would be suitable for the insertion of the event. The individual period numbers within this set are given in the PRD column with the resources preventing the insertion of the event at this period. Period sets are given in ascending order of difficulty as measured by their $E(\mathcal{S}_p)$ values, see equation (6.1).

The aim of the information displayed automatically in the event of failure is designed to provide as much basic information about the requirement as possible, together with a reasonably comprehensive answer to the question "Why has failure occurred?". The next step is up to the user: it may be that he can see what can be done to resolve the difficulty straight away, for example the insertion of a double period event spanning a break. In this case he can drop into INTERACT, insert the event and cause the program to continue. If such a simple solution to the problem does not present itself however, the user is given three options illustrated in plate 10.4.

1) To have the alternative courses of action presented to him.
2) To "try again". The selection of this option implies that one or more changes to parameters have been made either as a result of one of the courses of action in 1) or by using INTERACT.
3) To omit the unsatisfied event or events and proceed with timetable construction.

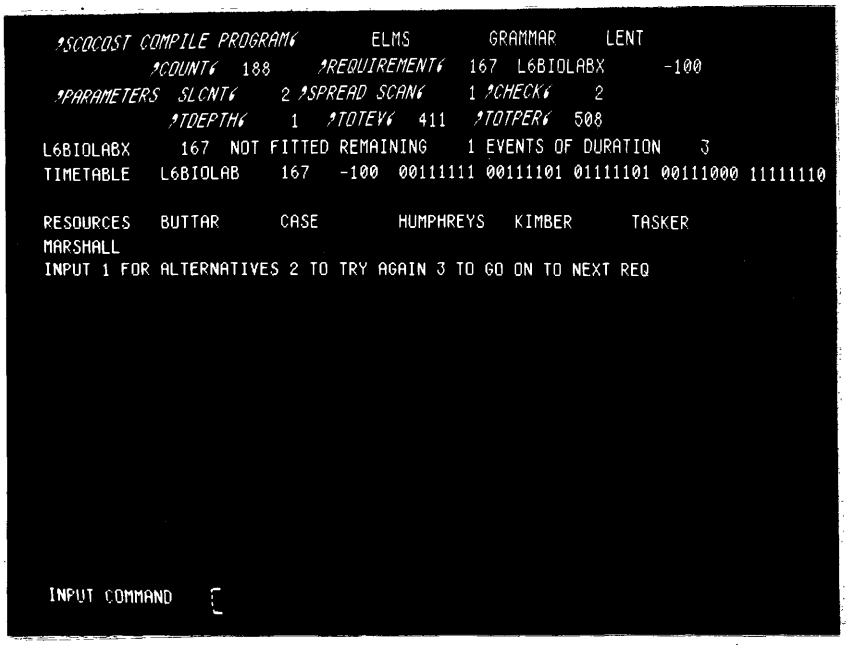


Plate 10.4 Options

Eleven possible courses of action illustrated in plate 10.5 are displayed if the first option is selected, and the meaning of these is described below.

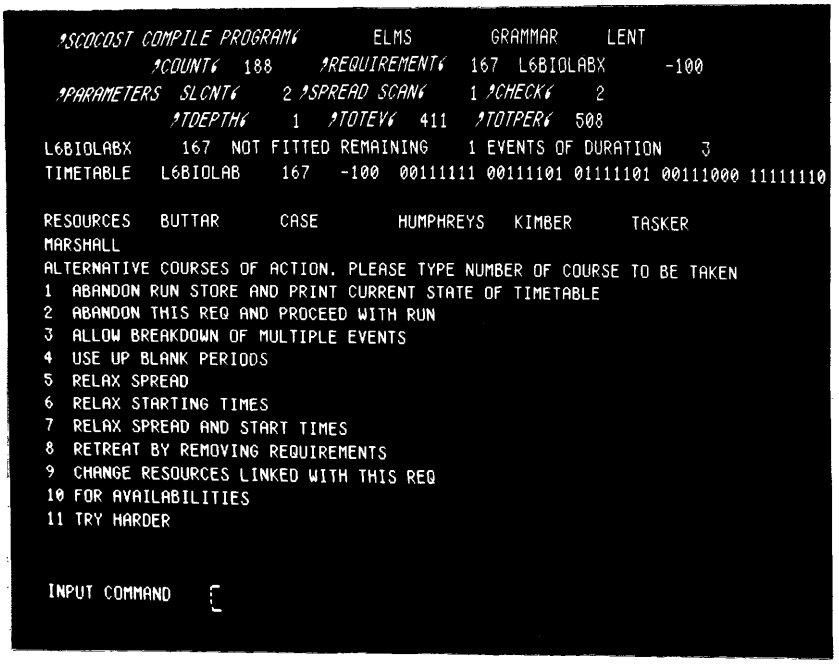


Plate 10.5 Alternative courses of action

1 ABANDON RUN STORE AND PRINT CURRENT STATE OF TIMETABLE

This course is useful if a really difficult problem occurs requiring detailed study of a printed output away from the computer.

2 ABANDON THIS REQ AND PROCEED WITH RUN

This is an admission of defeat but is sometimes useful for requirements which are classified as "desirable" rather than "essential".

3 ALLOW BREAKDOWN OF MULTIPLE PERIOD EVENTS

This course of action is only appropriate when the event causing failure is a multiple period event. It overrides a specification in the input data that multiple period events of the requirement concerned cannot be broken down (otherwise breakdown would have been tried automatically).

4 USE UP BLANK PERIODS

This produces a similar but not identical effect to 3. Suppose that a triple period event of a requirement has caused the failure and there are three non-contiguous blank periods available in the timetable for this requirement. Use of the breakdown facility would cause the program to split the event up into one double and one single. It would then try to satisfy the double using procedure DISPLACE. If this process failed the double would be broken down again into two singles. Recourse to DISPLACE might still be necessary if the three isolated blank periods in the timetable were so positioned as to cause the distribution rules to be violated if single period events were inserted in each of them. Course 4 simply causes single period events to

be inserted into blank periods until no further event periods remain for the requirement in question. No regard is paid to the fact that these single period events originated as multiple period events or to the distribution rules.

5 RELAX SPREAD

If this course is selected, the user is asked to type in a new value of the parameter spread, which determines the degree to which departure from the normal distribution rules is acceptable.

6 RELAX STARTING TIMES

On selecting this course the user is asked to specify one of five policies for event starting times illustrated in plate 10.6.

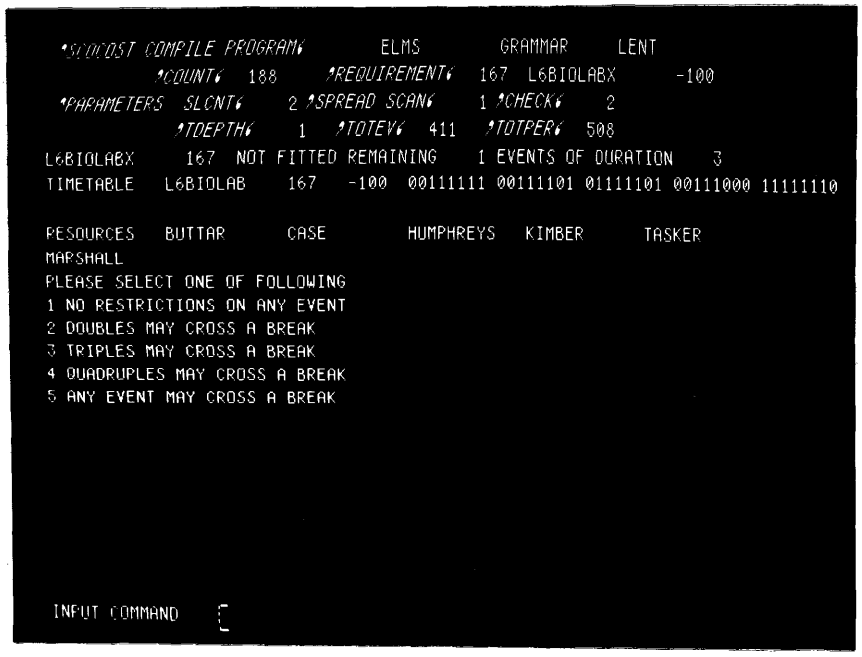


Plate 10.6 Alternative policies for event starting times



Plate 10.7 Implications of the selected policy displayed to the user for confirmation

The detailed implications of the selected policy are then displayed (see plate 10.7). If the user wishes to change any details he can do so by typing in the appropriate parameters.

7 RELAX SPREAD AND START TIMES

This combines courses 5 and 6.

8 RETREAT BY REMOVING REQUIREMENTS

This course enables the user to remove selected requirements and by so doing permit the insertion of the unsatisfied event. There is no guarantee, of course, that all the requirements removed can be completely satisfied subsequently. When this course is selected the user is given the following information to help him decide which requirements should be removed (see plate 10.8):

The last eight requirements to have been completed.

The requirements clashing with that which the program has failed to satisfy.

The user then types in the requirements to be removed.

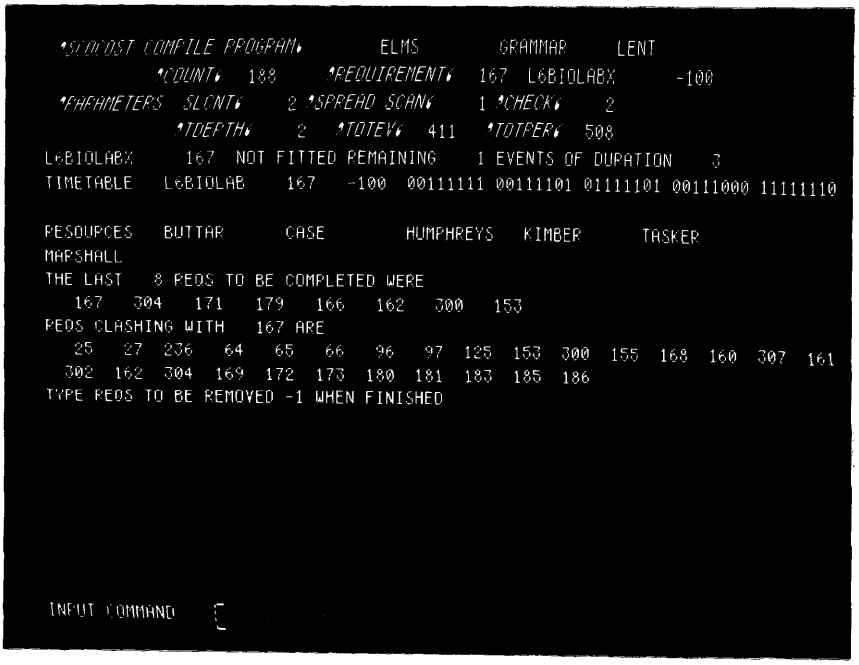


Plate 10.8 Retreat information

9 CHANGE RESOURCES LINKED WITH THIS REQ

The selection of this course causes the user to be asked which resources are to be dissociated from the requirement (see plate 10.9).

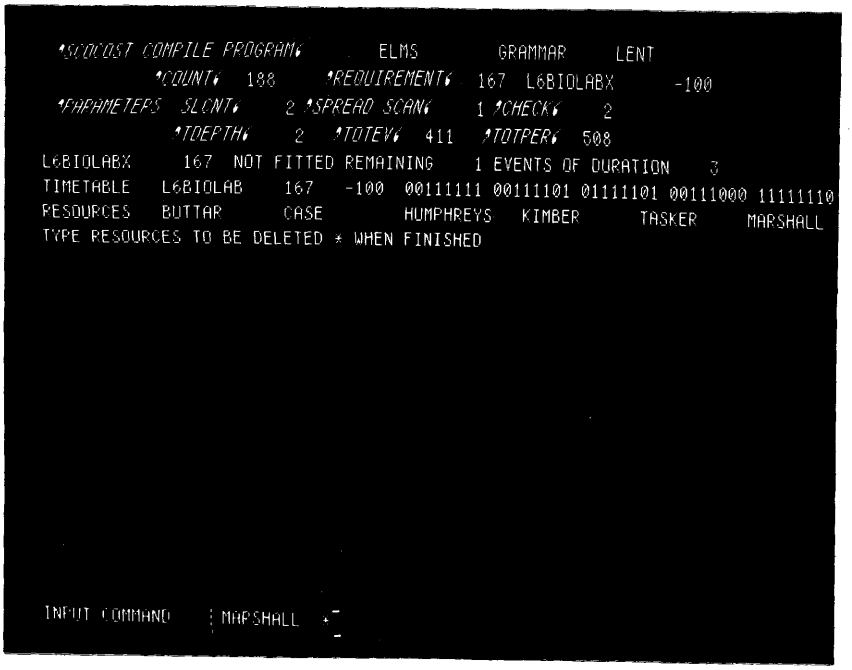


Plate 10.9 Changing resources - 1. Resources to be dissociated

He is then asked if he wishes to try other resources. If so the program calls for the resources which would be acceptable for this requirement. When these have been typed in, the program reports the resources which are available at the periods when events of this requirement have already been inserted.

Finally the user decides which resources are to be associated with the requirement and types in their names. The program replies with the same type of information as was originally given when the requirement failed. This time, however, one period set at least should be completely blank owing to the change of resources linked with the requirement. In the example illustrated in plate 10.10, Dolman has been substituted for Marshall with the result that two period sets are now clear, (compare with plate 10.3).

```
 %SCOCOST COMPILE PROGRAM%        ELMS        GRAMMAR     LENT
          %COUNT%  188     %REQUIREMENT%  167  L6BIOLABX      -100
 %PARAMETERS  SLCNT%     2 %SPREAD SCAN%     1 %CHECK%     2
              %TDEPTH%     2   %TOTEV%  411   %TOTPER%  508
L6BIOLABX      167  NOT FITTED REMAINING    1 EVENTS OF DURATION    3
TIMETABLE   L6BIOLAB    167   -100  00111111 00111101 01111101 00111000 11111110
RESOURCES   BUTTAR      CASE        HUMPHREYS   KIMBER      TASKER
DOLMAN
ST.PRD  PRD       RESOURCES
  30     30
         31
         32
  33     33
         34
         35
   9      9
         10
         11   BUTTAR      HUMPHREYS   TASKER
  25     25
         26
         27   BUTTAR      CASE        HUMPHREYS   KIMBER      TASKER
DO YOU WANT MORE PERIODS PLEASE INPUT 1 FOR YES 0 FOR NO

INPUT COMMAND
```

Plate 10.10 Changing resources - 2. Final check on periods

10 FOR AVAILABILITIES

By selecting this course the user obtains the availabilities of all resources associated with the requirement, illustrated in plate 10.11. The legend is as follows:

O available, * not available.

```
 %SCOCOST COMPILE PROGRAM%        ELMS        GRAMMAR     LENT
          %COUNT%  179     %REQUIREMENT%  302  L6CHEMA          2
 %PARAMETERS  SLCNT%     2 %SPREAD SCAN%     1 %CHECK%     2
              %TDEPTH%     1   %TOTEV%  391   %TOTPER%  474
L6CHEMA        302  NOT FITTED REMAINING    1 EVENTS OF DURATION    1
TIMETABLE   L6CHEMA     302      1  11122111 00111201 32111111 11112000 22211331

RESOURCES   BUTTAR      CAPELIN     GUNN        HUMPHREYS   RADBAND
TASKER      SQUIRE
BUTTAR        0000000*    00***00*    00000***    00**0*00    ***0****
CAPELIN       0000000*    00***00*    *0000***    00**0*00    ***0****
GUNN          **00****    00***00*    ***00***    ****0*00    *******0
HUMPHREYS     0*00****    00***00*    0**00***    00**0*00    *******0
RADBAND       00**000*    00****0*    *00*****    00****00    ***0***0
TASKER        0000*00*    00***00*    00000***    00**0*00    ***0*000
SQUIRE        000**000    00****00    **000**0    0000**00    00000**0
INPUT 1 FOR ALTERNATIVES 2 TO TRY AGAIN 3 TO GO ON TO NEXT REQ




INPUT COMMAND
```

Plate 10.11 Availability of resources

11 TRY HARDER

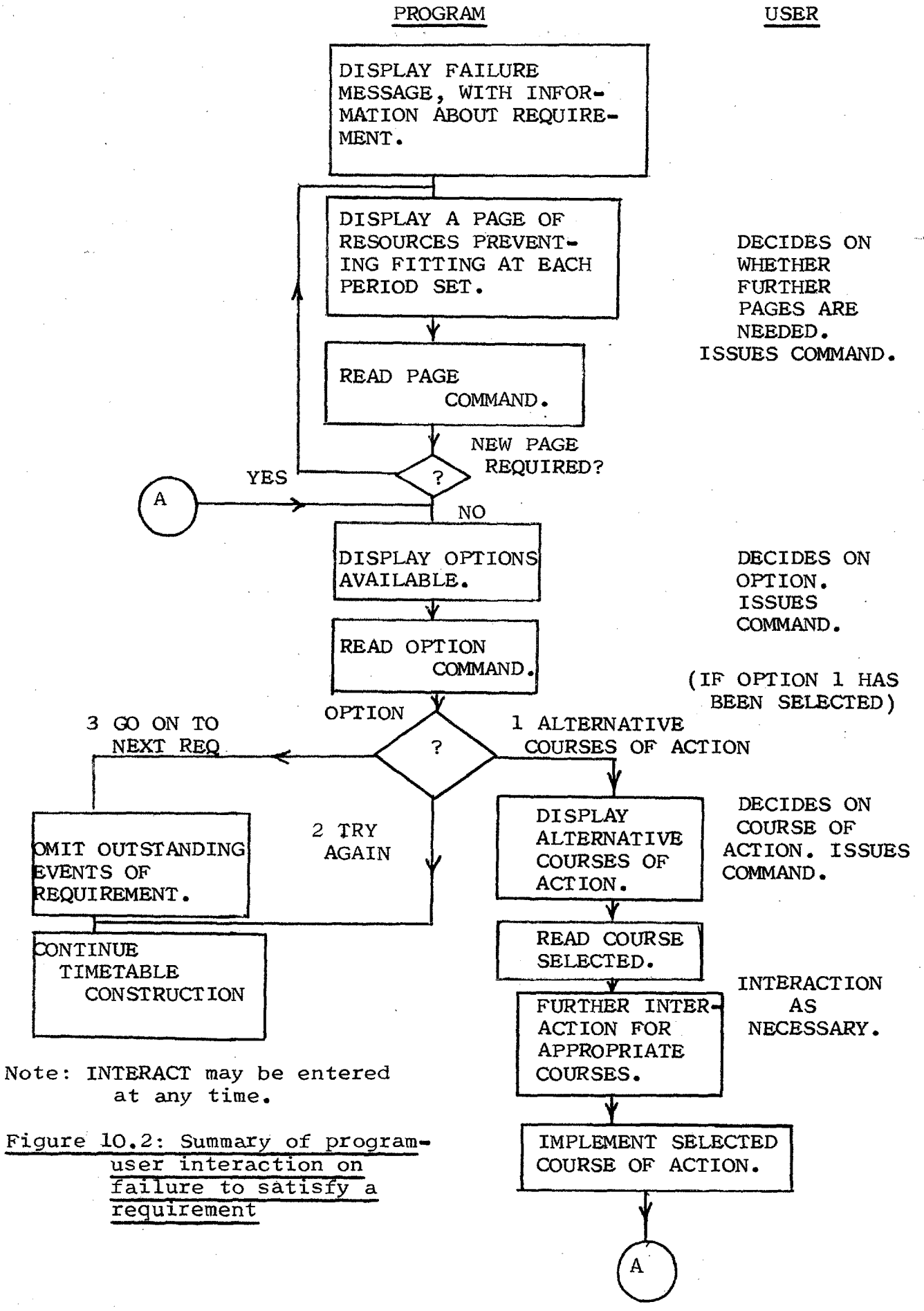
This course causes the value of tdepth to be incremented by 1 and the program to try again to fit the requirement with DISPLACE used to a greater depth of recursion.

When the alternative courses of action facility was first tried out only one course of action could be selected. This was unnecessarily restrictive and prevented combinations of courses being tried. The present scheme returns the user to the "option level" after selecting a course of action, so that the user may select several courses of action before going for the "try again" option.

An overall sequence of operations on failure to satisfy a requirement is given in figure 10.2.

10.5 Closing down facilities

When timetable construction has finished or when the user has set switch 1 with the INVERT command or when course of action 1 has been selected, the program pauses in INTERACT mode in case the user wishes to make any adjustments to the timetable. This facility is designed mainly to enable the user, having studied printouts at his leisure away from the terminal, to call up a completed timetable from the disc file and make amendments to it. On receiving the GO command the closing down phase proper starts with the display of the timetable in internal representation. Several "pages" are normally necessary to obtain the complete timetable in this way. After each page the user can opt to go on to the next form of display which is the timetable in stars form



Note: INTERACT may be entered at any time.

Figure 10.2: Summary of program-user interaction on failure to satisfy a requirement

(see note (i) to table 10.2). This display is paged, with the user able to go on to the next stage as before. The next display is the compile history, with the same information as printed out in the basic system (appendix 4, sheet AP4/9). Finally the user is asked whether or not the timetable should be stored on the disc file.

10.6 Experience in man-computer timetable construction

While the interactive system that has been described is not perfect its development has reached a state where important lessons may be learned about timetable construction by man and computer working as a team.

Four timetables have been constructed by the author using the team approach, with the results given in table 10.3 which also gives the best results achieved by the non-interactive system for comparison.

Data set	MAN-COMPUTER TEAM				COMPUTER ONLY	
	Failures reported by program	Insertion rate %	Elapsed time min.	CPU time min.	Insertion rate %	CPU time min.
Burnage	15	100	195	115	98.5	104
Elms	20	99.7	170	81	(97.4 ((96.1 (	213 tdepth=3 62 tdepth=2
Aylesbury	4	100	88	61	99.2	50
RAF Newton	1	100	42	24	99.9	18

Table 10.3 A comparison between man-computer team and computer only

For Aylesbury and RAF Newton the adjustments which had to be made were trivial. The adjustments made in the runs on the Elms and Burnage data are shown in table 10.4.

Adjustment	Number of occasions used	
	The Elms	Burnage
Breakdown of multiple period events	1	1
Relaxing spread constraints	4	1
Relaxing starting times of events	4	3
Retreat	5	6
Changing resources	10	8
Event omitted	1	0

Table 10.4 Adjustments to problem data

It is not known whether all the adjustments made would have been acceptable to the school in the case of the Burnage problem. A great deal more was known about the Elms data, particularly as the Headmistress had taken part in a previous interactive session during which over half the timetable was constructed. Her guidance as to what should be done in particular cases of difficulty during this session was a vital preparation for the run during which the complete timetable was constructed, and most of the decisions made during this run would probably have been acceptable to the school.

It can be seen from table 10.3 that the total elapsed time to construct a timetable is heavily dependant on the number of occasions on which the program has to ask for help. The interactive concept would become unworkable if the user were asked to cope with too many program failures, and it is believed that 20 to 25 is the practical limit. Certainly an interactive session lasting longer than 3 hours is very tiring for the user; 1 to 2 hour sessions would seem to be acceptable. Thus the interactive system is not an effective means of rescuing a set of

unsatisfactory algorithms or of achieving a complete timetable from a set of utterly infeasible data.

There is no doubt, however, that the man-computer team can produce better results than the computer program by itself. The long elapsed times are due largely to "thinking time" and provided that other programs can use the central processor during the times when the timetabling program is awaiting human reaction, the length of the elapsed time is not too important. It is almost certainly a price worth paying if a completely satisfactory timetable can be produced with no need for subsequent amendment.

There are many different ways of using the interactive system, which can only be adequately explored by the practical use of the system by those actually concerned with the timetabling problem in schools. One possibility is to use the interactive facilities only to amend a completed timetable stored on disc. This has been tried with the Elms data and is certainly a practical proposition with requirements involving few resources. With sixth form requirements associated with many resources the completed timetable is usually too "tight" to permit changes to be made readily. Another possibility, already mentioned, is to build up the really critical parts of the timetable by hand using INTERACT and then storing the partially completed timetable on the disc file to be finished by a non-interactive version of the program.

All the facilities provided by the interactive system have been used, and there do not appear to be any

serious omissions in the range of facilities provided. A minor fault is that when changing resources (course of action 9) the program reports on the availability of alternative resources with regard only to the events of the unsatisfied requirement which have already been inserted into the timetable. It would be more valuable if this report took into account the potential insertion of the unsatisfied event.

10.7 Retreat and displacement

The relative theoretical merits of retreat and displacement were briefly discussed in 6.3. During experiments with the interactive system, retreat was tried many times in different circumstances after failure to satisfy a requirement. On several occasions the strategy was successful in that not only was the event of the unsatisfied requirement ρ_i fitted but so were all the other requirements ρ_j, which had been removed in the retreat process. However, the strategy never succeeded without modifying the problem in some way, in nearly all cases by breaking down a multiple period event of one of the requirements ρ_j. Usually, indeed, the requirements ρ_j were selected with this in mind. A combination of the two algorithms is thus more effective than DISPLACE alone but a degree of selectivity is necessary in deciding which requirements to remove in the retreat process.

10.8 Discussion

The Commentary has been found to be far more valuable than expected and in surprising ways. Firstly it gives a very clear idea of the way in which the program operates. This is most useful when explaining it to the uninitiated,

and it also gives the programmer a far clearer idea of which are the time consuming operations in the program. The VDU commentary is thus useful as a program tuning aid. Two important time saving modifications to the program have been made as a direct result of information gained from the commentary, one concerned with how tied requirements are dealt with and the other with reducing the number of times that trial assignments are carried out.

The right level of commentary also provides the user with a psychological link with the program, which has been found most valuable. The right level in this context seems to be at the default value of display giving details of assignments, insertions and displacements but not trial assignments. More information than this is simply confusing, whereas that provided at display = 0 changes too infrequently to hold the user's interest. At the default value of display the user seems to feel an involvement in the timetable construction process which is totally lacking in the non-interactive situation. It is believed that this is an advantage, albeit unquantifiable. The ability of the user to intervene in the construction process, which was the original justification for the commentary, has been very frequently used but mainly to change global parameters such as display and tdepth. Detailed intervention to correct shortcomings in individual timetables has been less valuable than expected, partly because the information changes very rapidly and partly owing to the lack of really intimate knowledge of the particular timetable

under construction. This facility might be used much more frequently if those responsible for constructing timetables in schools started to use the system operationally.

Finally it is believed that the interactive approach to computer timetabling has great potential. It may be used at many different levels of man-computer relationship, depending on the problem, the temperament of the user and the confidence he has in the program. Above all, perhaps, it breaks down the psychological barrier between user and computer. One headmaster with experience of a commercial computer timetabling system wrote "...the computer does not take account of human factors". The interactive facility offers the human being the means to make it do so.

CHAPTER 11 - CONCLUSIONS

The results of this work lead to certain conclusions regarding the validity of the "timetable" problem, the overall practicability of computer assisted timetabling for schools and the characteristics of computer timetabling systems in relation to storage structures, solution algorithms and the user interface. This chapter presents and discusses these conclusions.

11.1 The timetable problem

What has come to be known as the timetable problem with its very simple structure is hardly relevant to the highly complex problem presented by British secondary schools which have the greatest timetabling difficulties. Methods so far devised for the solution of the timetable problem are of little value to the real timetabling problem for which the heuristic approach offers the best chance of success.

11.2 Overall practicability

As far as is known no British school had used a computer produced timetable when this work started. Such computer timetabling systems as had been tried had not yielded insertion rates significantly better than 90% and as a result were not practically useful. Now the position is very different. The work described in this thesis and also that of Michalsen have shown that timetables of comparable quality to those constructed manually can be produced by a computer, and some of these have been used in practice. Thus it can be claimed that computer assisted timetabling in British secondary schools is technically feasible.

The major obstacle to the widespread use of computers for timetabling in British schools is cost, likely to be between £100 and £200 per timetable. There is no prospect of offsetting this cost by direct monetary savings but it is believed that it can be justified by two important benefits which should accrue from computer assisted timetabling. The first of these is that members of the teaching staff will be released from the more laborious and mechanistic tasks in the preparation of timetables. The second is that a computer timetabling system can be a valuable tool for those concerned with various aspects of educational administration and planning.

11.3 Storage structures

For the heuristic approach to computer timetabling simple cartesian storage structures are believed to be quite adequate: no advantage is seen in adopting other forms of structure. There are many ways in which cartesian structures can be used to store timetabling data but the storage structure must be carefully chosen as it is a key element of any timetabling system. The three main criteria for choosing a storage structure are the ability to describe an adequate range of problem variations, economy of core store and the necessity for the storage structure and construction algorithms to be carefully matched. Storage structures based primarily on requirements have a definite advantage over those based on resources for schools with a complex pattern of options and highly variable pupil groupings.

11.4 Algorithms

11.4.1 Existence testing

While some progress has been made in reducing the

uncertainty as to the solvability of the problem posed by a set of apparently consistent timetabling data, no satisfactory existence test has been found. It does not appear to be possible to devise such a test and one is forced to return to the intuitive view that the only way to prove that a solution to a given problem exists is to find such a solution.

11.4.2 Event selection

If an infallible method of selecting times for events could be found, the order in which events are selected for insertion in the timetable would be unimportant. But as time selection is not an infallible process this order is important and the main criterion for event selection is that "difficult" events should be selected before "easy" events. The problem is to find measures of difficulty which are satisfactory in all cases. In this work it is concluded that current difficulty alone is a better criterion than inherent difficulty alone but that a combination of the two criteria can give better results than either of them singly. Slack or choice, both derived from the trial assignment procedure, have been found to be satisfactory measures of current difficulty.

11.4.3 Time selection and conflict resolution

Similar arguments apply to conflict resolution in relation to time selection. If time selection is an infallible process there is clearly no need to resolve conflicts since none will arise. To be certain that a given choice of time for an event will be satisfactory it is necessary to establish that such a step is compat-

ible with a solution to the problem and thus requires an infallible existence test, which is unlikely to be found. For this reason it is not likely that an infallible time selection process will be devised and the evidence of this work is that a conflict resolution algorithm in conjunction with a time selection algorithm based on static interference criteria only produces at least as good results with less computing time as a time selection algorithm using an imperfect existence test. The most effective mechanism for conflict resolution has been found to be a recursive displacement algorithm, with "retreat" under user control as a last resort.

11.5 The user interface

One of the main themes of this work has been the need to enable the user to play a more active part in the process of timetable construction than simply preparing a set of data for presentation to a computer and then, later, analysing and amending the computer output. The interactive system that has been developed has shown that it is possible for the user to work with the computer in a number of different relationships ranging from that just described, where the construction of the timetable belongs exclusively to the computer, to a relationship where the user plays the dominant role. The interactive system has enabled better timetables to be produced than was possible using the computer alone and it is concluded that interactive timetabling adds an important new dimension to the concept of computer assisted timetabling.

Another aspect of the user interface is the way in which the input data is presented to the computer. In

this and other work the emphasis has so far been placed, correctly, on devising systems which are capable of producing timetables of adequate completeness and quality. Now that some success has been achieved in this direction, there is a need to evaluate different methods of data input with a view to devising the best method. It may be possible to establish a standard input specification to which any computer timetabling system for use in British schools should conform. This would allow for the introduction of new systems, and for the improvement of the internal characteristics of existing systems "behind" a reasonably stable user interface.

11.6 Future development

While the problem of automatic construction of school timetables is certainly not closed, computer timetabling has now reached the stage when its operational implementation in British schools has been shown to be technically possible. While fundamental research on allocation and scheduling problems must certainly continue and this may provide fuller insight into the school timetabling problem, it appears that, in respect of this problem the time has come to move from experiments to the development of at least one system for operational implementation. While certainly not the only possible approach, it is believed that the present work forms the basis for such a system.

APPENDIX 1 - DATA STORAGE CALCULATIONS

This appendix contains the calculations required to support table 3.2.

Storage structure 1 - the class, period array

1) Storage (in 24 bit computer words)

   (i) Timetable array: 2CP.

   (ii) Availability array: $\left\lceil \frac{Y}{12} \right\rceil P$.

2) To determine whether a single period event e(i,1) can be inserted at any period p.

   A series of pure "and" operations on the elements of the availability array A, with each bit taken separately, has to be carried out to establish the availability of all resources linked with requirement ρ_i, at period p. Let X(p) be the availability of e(i,1) at period p and let requirement ρ_i be associated with resources $y_{i_1}, y_{i_2}, \ldots, y_{i_m}$.

   Then

   $$X(p) = A[i_1,p] \;\&\; A[i_2,p], \ldots, \& A[i_m,p].$$

   If m quantities "anded" together involve (m-1) operations then the average number of operations required to establish X(p) is $\bar{y}(\rho) - 1$.

3) To insert event e(i,1) at period p.

   Two different types of updating operation are required.

   (i) Availability array:

   Set $A[j,p] = 0$, $j = i_1, i_2, \ldots, i_m$.

   The average number of operations is $\bar{y}(\rho)$.

   (ii) Timetable array:

   Let t_k, r_s be the teacher and room associated

with ρ_i. If there is more than one teacher and/or room let k and/or s point to vectors where the actual teachers or rooms are stored. If the classes associated with ρ_i are $c_{j_1}, c_{j_2}, \ldots, c_{j_n}$, then the algorithm must set $T[j_q, p] = k, s, q = 1, 2, \ldots, n$.

The average number of operations (assuming that updating one element involves two operations) is $2\bar{c}(\rho)$.

Storage structure 2 - the four dimensional array

1) Storage (in 24 bit computer words).
Timetable array: $CTR\left\lceil \frac{P}{12} \right\rceil$.

2) To determine whether a single period event e(i,1) can be inserted at any period p
In principle the only way to determine the availability of a resource at any period is to search the timetable array for a meeting of that resource with all possible pairs of resources of the other two types at that period. Thus to determine whether a class c_j is available at period p requires TR operations. On average a requirement ρ_i is associated with $\bar{c}(\rho)$ classes, $\bar{t}(\rho)$ teachers and $\bar{r}(\rho)$ rooms so that the average number of operations required to establish whether an event e(i,1) can be inserted at period p is $(\bar{c}(\rho)TR + \bar{t}(\rho)CR + \bar{r}(\rho)CT)$.

3) To insert a single period event e(i,1) at period p
If the resources associated with e(i,1) are
$c_{j_1}, c_{j_2}, \ldots, c_{j_u}, t_{k_1}, t_{k_2}, \ldots, t_{k_v}, r_{s_1}, r_{s_2}, \ldots, r_{s_w}$
then the insertion algorithm must set
$T[j_f, k_g, s_h, p] = 1, f = 1, 2, \ldots, u, g = 1, 2, \ldots, v, h = 1, 2, \ldots, w$.

The average number of operations is $\bar{c}(\rho)\bar{t}(\rho)\bar{r}(\rho)$.

Storage structure 3 - the requirement list

1) Storage (in 24 bit computer words)

(i) Vector T: 2Q.

(ii) Vector V: ρ.

(iii) Availability array A: $\left\lceil \frac{Y}{12} \right\rceil P$.

2) To determine whether a single period event e(i,1) can be inserted at any period p

As for storage structure 1.

The average number of operations is $\bar{y}(\rho) - 1$.

3) To insert a single period event e(i,1) at period p

Two different types of operation are required.

(i) Update availability array A as for storage structure 1.

The average number of operations is $\bar{y}(\rho)$.

(ii) A search starting at V(i) must be carried out for the first ~~now~~ zero element, u say, of T. Then set T[u] to p,m where m is the number of the room associated with e(i,1) or a pointer to a vector containing a set of such numbers if there is more than one such room.

The average number of operations is $\frac{Q}{2\rho}$ approximately for the search plus 2 operations for the insertion in T.

Storage structure 4 - the requirement array

1) Storage (in 24 bit computer words)

(i) Timetable array T: $P\rho$

(ii) Room availability array A: $\left\lceil \frac{R}{12} \right\rceil P$.

2) To determine whether a single period event e(i,1) can be inserted at any period p

As the opportunity list for all requirements is

kept continuously up to date only one operation is required.

3) To insert a single period event e(i,1) at period p Three different types of operation must be carried out.

(i) Set T[i,p] = m,
where m is the number of the room in which the event is to take place, or a pointer to a vector in the case of multiple rooms. 1 operation is required.

(ii) Decrease T[j,p] by 1 for each requirement ρ_j which clashes with ρ_i. The average number of operations is $\bar{x}$.

(iii) Set A[m,p] = 0 for all rooms r_m associated with requirement ρ_i. The average number of operations is $\bar{r}(\rho)$.

APPENDIX 2 - DATA VARIABLES USED IN SCOCOST

Array	Array element	Meaning	How obtained (see note (i))	Program Name (see note (ii))	Remarks
BDATA	1	Number of days in the week	U	dw	
	2	" " periods in the day	U	pd	
	3	" " periods in the week	U	pw	
	4	" " requirements	UC	req	
	5	" " teachers	UC	lect	
	6	" " classes	UC	class	
	7	" " rooms	UC	room	
	8	" " miscellaneous elements	U	misc	This includes the name of the school and names of days and periods.
	9	" " room groups	UC	rgp	A room group is a set of rooms which are equally acceptable for at least one requirement.
	10	File to be used	U	file	Redundant. Stored to assist in diagnosis only.
	11	Duration of longest event	U	mul	
	12	Maximum number of multiple rooms	U	mlr	This is a users estimate of the maximum number of different combinations of rooms that will be allocated to events in the completed timetable.
	13	Spread factor	U	sp	This is used for spreading events over the week.
	14	Page size for clashing and requirement-resource arrays, CONF and ITREQ respectively	U	itcon	Determines the number of columns of these arrays to be held in core at any one time.
	15	Number of characters in a name	U	nn	nn is a maximum. Names with fewer characters are perfectly acceptable.
	16	Total number of ties	U	scope	A users estimate of the total number of times one requirement is specified as being tied to another.
	17	Separation factor	U	sep	Works in conjunction with sp (see 13). Meaning is as follows. 0 events of a requirement must be spread out over the week in a normal distribution and two such events occurring on the same day must be separated by at least one period. 1 normal distribution but if two events do occur on the same day they need not be separated. 2 no constraints on the spread of events of a requirement across the week.
	18	Number of names to be held in core at any one time	U	nonames	Normally arranged so that one bucket of 128 words will contain this number of names.
	19	Number of words required to hold one column of clashing and requirement-resource arrays, CONF and ITREQ respectively	C	nf	These arrays are packed 24 elements to a word.
	20	Number of resources	C	y	
	21	Number of words required to hold one column of room availability array, RASP	C	v	Array RASP is packed 24 elements to a word.
	22	Number of words required to hold one column of room group array, RGP	C	u	Array RGP is packed 24 elements to a word.
	23	Number of multiple teachers	C	ml	Total number of different combinations of teachers associated with requirements in the input data.
	24	Number of elements in array RMREQ	C	nl	
NEWDATA	1-3	Spare			
	4	Page size for timetable array	U	size	Number of requirements for which the timetable is held in core.
	5	Number of pages for timetable array	C	nopge	
	6	Number of buckets required for timetable array	C	timebucs	
	7	Number of buckets required for clashing and requirement-resource array	C	confbucs	
	8	Number of buckets required for one group of names	C	namebucs	Usually 1, see BDATA [18].
	9-29	Spare			

Notes: (i) Key for the "How obtained column".

U = user supplies this information as part of the input data.
UC = user supplies an estimate as part of the input data. This estimate is used for setting up arrays only. True value is computed and stored by input program.
C = computed information.

(ii) The "Program name" column gives the name by which the quantity is known in all the programs in the system.

APPENDIX 3 - SCOCOST DATA ARRAYS

A specification of the arrays used in the system is given in table 1.

POINTING element	Array (see note (i)) name	dimensions	Use
1	None	-	Spare.
2	REQPOINT	0 : req	ith element holds element of array REQITE (see 14) at which the first requirement tied to ρ_i is stored.
3	HEAD	0 : req	used by the output program to hold headings.
4	REQNO	0 : req	to be assigned for requirement ρ_i coded as in note (ii).
5	SLACK	0 : req	Slack as described in 4.5.1.
6	HIST	0 : req	If requirement ρ_i was the nth requirement to be completed then the ith element holds the value n. Elements corresponding to uncompleted requirements have values 0 or -1.
7	LATEST	0 : req	The ith element holds the period at which the last event of ρ_i was inserted.
8	FITTED	0 : req	The ith element holds the number of events of ρ_i fitted so far.
9	None	-	Spare.
10	REQSPEC	0 : req	Code numbers of requirements. Each requirement is allocated a code number as well as a name for ease of communication between the user and the system. Inside the system requirements are known by the order in which they were input. Thus in this table ρ_i refers to the ith requirement to be input.
11	RMREQ	1 : nl	Room groups associated with each requirement, and numbers of rooms required from each group. Accessed via ROOMNO.

Table 1 Data arrays

Sheet 1 of 3

POINTING element	Array name	(see note (i)) dimensions	Use
12	ROOMNO	0 : req+1	ROOMNO[i] holds the starting element of RMREQ for requirement ρ_i.
13	GRAME	0 : req	Names of groups of requirements and resources: used in the output program.
14	REQTIE	1 : scope	Holds tied requirements. If n requirements are tied to requirement ρ_i these are held in REQTIE [REQPOINT[i]], REQTIE [REQPOINT[i]+1],..., REQTIE[REQPOINT[i]+n-1].
15	None		Names. Programs interact with this part of the data base via a working array NAME [1:nn *nonames] used for all names.
16	CONF	1 : nf 1 : itcon	Clashing array, see note (iii).
17,18	None	-	Spare.
19	TIMETAB	1 : size 1 : pw	Main timetable array: coding as in table 2. Overlaid from disc in "pages", with size (appendix 2, NEWDATA[4]), giving the number of requirement timetables in a page.
20	RASP	1 : v 1 : pw	Room availability array. 24 elements to a word. 0 for available 1 for not available.
21	ITNO	1 : py	Preferred room for teachers and classes; capacity of a room.
22	ITREQ	1 : nf 1 : itcon	Requirements-resources array, see note (iii).
23	STAR	1 : mul 1 : pw	Event starting times array. If STAR [i,j]=0 then an event of duration i may start at period j: if 1 it may not.
24	PREF	1 : mul	PREF [z,j] holds the jth period within the day to be selected for an event of duration z.
25	GRASP	0 : rgp 1 : pw	GRASP [j,p] holds the number of rooms of the jth room group that are available at period p.
26	RMGRP	1 : u 1 : rgp	Room groups. The jth column holds the numbers of the rooms in the jth room group.

Table 2 Data arrays. Sheet 2 of 3

POINTING element	Array (see note (i)) name	dimensions	Use
27	MULROOM	1 : mlr 1 : room	Multiple rooms. If MULROOM [i,j]=1 then room r_j is part of the ith multiple room: if 0 it does not.

Notes: (i) The array names and dimensions given are as used in the programs.

(ii) The coding system for the number of events in array REQNO is as follows:
digits - number of single period events,
tens - number of doubles,
hundreds - number of triples and so on.
A positive number means that multiple period events can be broken down as a last resort, a negative number means that they cannot.
Example: 2517 means 2 quadruples, 5 triples, 1 double and 7 singles. The quadruples, triples and double can be broken down.

(iii) Arrays CONF and ITREQ are packed 24 elements to a word. Both arrays can be overlayed from backing storage.

Table 1. Data arrays

Sheet 3 of 3

Table 2 gives the possible values and meanings of a typical element TIMETAB [i,p] of the main timetable array. This element refers to requirement ρ_i at period p. For convenience let n = TIMETAB [i,p].

Value of n		Meaning
1	100000	Available.
2	100000>n>10000	Not available due to the occurrence of clashing events at period p. The quantity 100000-n gives the number of such events and hence an indication of the difficulty of making this period available.
3	10000>n>1000	Not available due to initial non availability of associated resources.
4	1000≥n>1000-mul	An event of this requirement is scheduled at this period. No room has yet been allocated. The value of n indicates the period relative to the start of the event: thus n=1000 indicates the first period of an event, 999 the second and so on. mul is the duration of the longest event in the timetable.
5	900≥n>room	An event of this requirement has been scheduled at this period. More than one room has been allocated to it. n-room points to the row of the array, MULROOM, which holds the actual rooms allocated.
6	room≥n>0	An event of this requirement has been scheduled in this period in the room r_n.
7	0>n≥-room	As for 6 but this assignment cannot be changed.
8	-room>n≥-900	As for 5 but this assignment cannot be changed.
9	-900>n≥-1000	As for 4 but this assignment cannot be changed.
10	-1000>n	Period not available under any circumstances.

Table 2. Coding for the timetable array

APPENDIX 4

EXAMPLES OF INPUTS TO AND OUTPUTS FROM SCOCOST.

SCOCOST - INPUT PROGRAM - BASIC DATA

Note to punch operator: Please punch the numbers in the fourth column only with 2 spaces between each number and 10 numbers to a card.

School or College.....Example problem

Ref. No	Program name	Meaning	Number (Punch these numbers)	Element where stored on EDS	Remarks
1	DW	Days in week	1	B1	
2	PD	Periods in day	3	B2	
3	PW	Periods in week	3	B3	
4	REQ	Number of requirements	10	B4	
5	CLASS	Number of classes	4	B6	
6	LECT	Number of teachers	5	B5	
7	ROOM	Number of rooms	7	B7	
8	MISC	Number of time elements	7	B8	
9	RGP	Number of room groups	2	B9	
10	FILE	File to be used. 1 for "JBRIT"	0	B10	
11	MUL	Length of longest event	2	B11	
12	MLR	Max no of multiple rooms	1	B12	
13	SP	Spread factor	3	B13	
14	KEY	Controls the quantity of output produced by the input program	3	-	
15	NN	No of characters in name	10		Restrict to 6 if Linear format required for output
16	SCOPE	Total number of ties	1	B16	
17	SEP	Controls separation	2	B17	
18	NONAMES	Number of names held in core	12	B18))	
19	ITCON	Page size for conflict and other arrays	10	) B14))	For use by
20	SIZE	Page size for timetable	10	N4)	by the analyst

FORM I2

SCOCOST - INPUT PROGRAM - REQUIREMENTS

School or College Example problem

Notes: (i) Each record must be terminated by "*"
(ii) At least 2 spaces after requirement name. At least 2 spaces or a comma after a number.
(iii) Terminate with ↑↑↑↑

	Compulsory fields			Optional fields			
	Code Fixed	Name Variable	No of events Fixed	Room group Fixed	No of rooms Fixed	Tied Requirements Variable	Initial time-table entries Variable
Ex	124 52	LATIN CHEMISTRY	5 * -100	-2	1	127, 129, ;	-12, *
	1	MATHS	1	*			
	2	FRENCH	0	2	1	;	-2, *
	3	ENGLISH	1	*			
	4	GEOMETRY	1	*			
	5	GERMAN	1	2	*		
	6	GREEK	1	*			
	7	R.K.	1	*			
	8	SCIENCE	10	1	2	*	
	↑↑↑↑						

Note: The default values for room group and number of rooms are 1 in each case.

FORM I3

SCOCOST - INPUT PROGRAM - RESOURCES

School or College Example problem.....

Notes:
(i) Present resources in the order classes, teachers, rooms, terminating each sub group with
(ii) Continuation cards may be used as necessary.
(iii) Each record must be terminated with "*".
(iv) At least 2 spaces after each name.
(v) Fill in room capacity for rooms in "preferred room" column.

	Compulsory fields		Optional fields		
	Name Variable	Preferred room	Associated requirements Variable	Periods not available Variable	Room groups Variable
Ex	JONES SCI LAB	1 50	127, 129, 135, ; 141, ;	-28, * ;	 3, 5, *
	FORM 1	5	1, 2, 3, *		
	FORM 2	4	4, 5, 6, *		
	FORM 3	3	7, 8, *		
	↑↑↑↑				
	SMITH	1	1, 4, *		
	JONES	1	2, 5, *		
	ROBINSON	1	6, 7, *		
	EDWARDS	1	3, 8, *		
	↑↑↑↑				
	ROOM 1	17	;	;	1, *
	ROOM 2	26	;	;	1, *
	ROOM 3	30	;	;	1, 2, *
	ROOM 4	25	;	;	2, *
	ROOM 5	24	8, *		
	ROOM 6	24	8, *		
	↑↑↑↑				

FORM I4

SCOCOST-INPUT PROGRAM - EVENT STARTING TIMES

Terminate with ↑↑↑↑. School or College Example problem

	Event duration	Periods when an event of this duration cannot start
Ex	1	*
	2	4, 8, 12, 16, 20, 24, 28, 32, 36, 40, *
	2	3, *
	↑↑↑↑	

FORM I5

SCOCOST - INPUT PROGRAMS- NAMES OF TIME ELEMENTS

School or College Example problem

Remarks	Names of Elements Term name, then period names then day names, then name of school (two separate names).
Example	EASTER PRD 1 PRD 2 PRD 3 PRD 4 PRD 5 MON TUES WED THUR FRI GROVE SCHOOL
	LENT PERIOD 1 PERIOD 2 PERIOD 3 MONDAY
	EXAMPLE PROBLEM

Sheet AP4/4 - Forms I4 and I5

```
SCOCOST INPUT AND VALIDATION PROGRAM  -  25/8/72

 1     3     3     10     4     5     7     7     2     1     2     1     3
 3     10    1     2      12    10    10
 4     5     6     7      8     9     10    11    12    13    14    15    16    18    81    82    84
86    161   171   172    173   175   176   177   177   177   177   178   179   179   179   179

 1     MATHS                 1     *
 2     FRENCH                0           2           1     ;     -2    *
 3     ENGLISH               1     *
 4     GEOMETRY              1     *
 5     GERMAN                1           2     *
 6     GREEK                 1     *
 7     R.K.                  1     *
 8     SCIENCE               10          1           2     *

 1     FORM 1                15    1     2     3     *
 2     FORM 2                24    4     5     6     *
 3     FORM 3                23    7     8     *
 4     SMITH                 1     1     4     *
 5     JONES                 1     2     5     *
 6     ROBINSON              1     6     7     *
 7     EDWARDS               1     3     8     *
 8     ROOM 1                17 ;;       1     *
 9     ROOM 2                26 ;;       1     *
10     ROOM 3                30 ;;       1     2     *
11     ROOM 4                25 ;;       2     *
12     ROOM 5                24    8     *
13     ROOM 6                24    8     *

2      3      *

REQS.    CLASSES    TEACHERS    ROOMS    ROOM GROUPS.
 8         3           4          6          2

LENT     PERIOD 1    PERIOD 2    PERIOD 3    MONDAY      EXAMPLE      PROBLEM
```

Sheet AP4/5 - Output from input program - I
Print out of data as input

SCOCOST

EXAMPLE PROBLEM

TABLE OF REQUIREMENTS

REQUIREMENT	CODE NO	EVENTS	GROUP	ROOMS	ASSOCIATED RESOURCES			
MATHS	1	1	1	1	FORM 1	SMITH		
FRENCH	2	0	2	1	FORM 1	JONES		
ENGLISH	3	1	1	1	FORM 1	EDWARDS		
GEOMETRY	4	1	1	1	FORM 2	SMITH		
GERMAN	5	1	2	1	FORM 2	JONES		
GREEK	6	1	1	1	FORM 2	ROBINSON		
R.K.	7	1	1	1	FORM 3	ROBINSON		
SCIENCE	8	10	1	0	FORM 3	EDWARDS	ROOM 5	ROOM 6

TABLE OF RESOURCES

NAME	NUMBER	NO. OF EVENTS		TOTAL PERIODS	CODE NUMBERS OF ASSOCIATED REQUIREMENTS		
		1	2				
FORM 1	15	3	0	3	1	2	3
FORM 2	14	3	0	3	4	5	6
FORM 3	13	1	1	3	7	8	
SMITH	1	2	0	2	1	4	
JONES	1	2	0	2	2	5	
ROBINSON	1	2	0	2	6	7	
EDWARDS	1	1	1	3	3	8	
ROOM 1	17	0	0	0			
ROOM 2	26	0	0	0			
ROOM 3	30	0	0	0			
ROOM 4	25	0	0	0			
ROOM 5	24	0	1	2	8		
ROOM 6	24	0	1	2	8		

TABLE OF ROOM GROUPS

GROUP NO	ROOMS		
1	ROOM 1	ROOM 2	ROOM 3
2	ROOM 3	ROOM 4	

Sheet AP4/6 - Output from input program - II
Tables of Requirements, resources and room groups

TABLE OF CLASHES

REQUIREMENT	CODE NO:	CODE NUMBERS OF CLASHING REQUIREMENTS		
MATHS	1	2	3	4
FRENCH	2	1	3	5
ENGLISH	3	1	2	8
GEOMETRY	4	1	5	6
GERMAN	5	2	4	6
GREEK	6	4	5	7
R.K.	7	6	8	
SCIENCE	8	3	7	

EVENT STARTING TIMES

EVENT DURATION	
1	000
2	001

NO: OF MULTIPLE LECTS/CLASSES = 0

INITIAL TIMETABLE IN INTERNAL REPRESENTATION

MATHS	1	0	-99999	0
FRENCH	2	0	-1000	0
ENGLISH	3	0	-99999	0
GEOMETRY	4	0	0	0
GERMAN	5	0	-99999	0
GREEK	6	0	0	0
R.K.	7	0	0	0
SCIENCE	8	0	0	0

Note: When the actual value of an element of array TIMETAB is 100,000 it is printed out at 0 to make for easier reading.

Sheet AP4/7 - Output from input program - III
Clashes, event starting times and initial timetable

SCOCOST - COMPILE PROGRAM - DATA

Note to punch operator: please punch the numbers in the fourth column only with 2 spaces between each number

School or College Example problem.....

Reference Number	Program Name	Meaning	Number (Punch these numbers)	Remarks
1	FILE	EDS file to be used: 1 for JBRIT otherwise JBRITSTORE 1.	0	Integer
2	SLCNT	Slack Control: large values weight selection in favour of long events.	1.4	Real
3	TDEPTH	Depth to which DISPLACE can be used recursively: ie how hard the program is to "try".	2	Integer 1 or 2 are suggested.
4	NEGCONT	Weight to be given to events which cannot be broken down: large values favour such events.	0	
5	SPREAD	Amount by which program may depart from normal distribution of events in a requirement.	1	
6	PRT	Controls quantity of print output. 0 for none except failure and breakdown messages, 1 as 0 plus timetable and compile history, 2 as 1 plus timetable in internal representation.	2	

SCOCOST COMPILE PROGRAM 25/8/72

TIMETABLE IN INTERNAL REPRESENTATION

MATHS	1	1000	-100000	99999
FRENCH	2	99999	-1000	99998
ENGLISH	3	99998	-100000	1000
GEOMETRY	4	99998	1000	99999
GERMAN	5	99999	-100000	1000
GREEK	6	1000	99999	99998
R.K.	7	99998	99999	1000
SCIENCE	8	1000	999	99998

LENT TERM TIMETABLE BY COMPUTER

KEY: 1 ALLOCATED 0 AVAILABLE * NOT AVAILABLE

MATHS	1	1**
FRENCH	2	*1*
ENGLISH	3	**1
GEOMETRY	4	*1*
GERMAN	5	**1
GREEK	6	1**
R.K.	7	**1
SCIENCE	8	11*

COMPILE HISTORY

MATHS	1	4	-1	1	0	1
FRENCH	2	1	1	0	0	1
ENGLISH	3	3	-1	1	0	1
GEOMETRY	4	6	0	1	0	1
GERMAN	5	8	-1	1	0	1
GREEK	6	7	-1	1	0	1
R.K.	7	5	-1	1	0	1
SCIENCE	8	2	-1	10	0	1

Sheet AP4/9 - Output from Compile program

FORM R1

SCOCOST - ROOM ALLOCATION PROGRAM - DATA

Note to punch operator: please punch the numbers in the fourth column only with 2 spaces between each number.

School or College Example problem.....

Reference No	NProgram name	Meaning	Number (Punch these numbers)	Remarks
1	FILE	EDS file to be used. 1 for JBRIT otherwise JBRITSTORE 1.	0	
2	PRT	Controls quantity of print output. 0 for none except failure message, 1 as 0 plus certain tables, 2 as 1 plus the timetable in internal representation.	1	

FORM O1

SCOCOST - OUTPUT PROGRAM - BASIC DATA

Note to punch operator: please punch the numbers in the fourth column only with 2 spaces between each number.

School or College Example problem.....

Reference No	Program Name	Meaning	Number (Punch these numbers)	Remarks
1	FILE	EDS file to be used. 1 for JBRIT otherwise JBRITSTORE 1.	0	
2	NEW	Greater than 0 - Standard mode, no further data required, otherwise select mode. See below.	0	

Significance of NEW in standard mode.

Digit	Condition	Meaning
5th	1	Linear format otherwise non linear format.
4th	1	PP=4, otherwise PP depends on 3rd digit.
3rd	1	PP=3, " " " " 2nd digit.
2nd	1	PP=2, otherwise PP=1.
1st	= 2	Standard print out for classes.
	= 3	" " " " teachers.
	= 4	" " " " rooms.

Examples

10002 Linear format, PP=1, standard print out classes.
104 Non linear format, PP=3, standard print out rooms.

SCOCOST - OUTPUT PROGRAM - DEFINITION OF GROUPS

School or College Example problem

Note: (i) If a group of requirements is to be defined, the letter R is to be inserted on the "R or I" column, otherwise the letter I.
(ii) Do not allow a name to be split between two cards. Continuation cards may be used as required.
(iii) At least two spaces after each name or number.

	Group number	"D"	R or I	Group Name	Requirement or resource name. (These must tie up with the names presented to the input program name).
Ex	1	D	R	MATHS ENG DEPT	MATHS 4 MATHS 5 MATHS 6 * SMITH JONES BLOGGS ENGLAB WILKS FOUNDRY *
	1	D	I	CLASSES	FORM 1 FORM 2 FORM 3 *
	2	D	I	TEACHERS	SMITH JONES ROBINSON EDWARDS *
	3	D	I	SCI LABS	ROOM 5 ROOM 6 *
	4	D	R	LANGUAGES	FRENCH GERMAN GREEK *

SCOCOST - OUTPUT PROGRAM - CALL FOR PRINT OUT

School or College Example problem.....

Note:
(i) FORM stands for format control parameter.
(ii) The meaning of FORM is similar to that given in form 01.
(iii) Do not allow a name to be split between two cards. Continuation cards may be used as required.
(iv) At least 2 spaces after each name or number.
(v) Terminate with single card ↑↑↑↑.

	Card	"P"	FORM	Groups (These must previously have been dinfed on Form 02)
Ex	1	P	1002	MATHS ENG DEPT *
	1	P	1002	CLASSES *
	2	P	103	TEACHERS *
	↑↑↑↑			

Sheet AP4/12 - Form 03

LENT TERM TIMETABLE BY COMPUTER

SCOCOST DATE 25/8/72

CLASSES

MONDAY

PERIOD 1	!MATHS
	!SMITH
	!ROOM 3
	!GREEK
	!ROBINSON
	!ROOM 5
	!SCIENCE
	!EDWARDS
	! R 1
PERIOD 2	!FRENCH
	!JONES
	!ROOM 3
	!GEOMETRY
	!SMITH
	!ROOM 5
	!SCIENCE
	!EDWARDS
	! R 1
PERIOD 3	!ENGLISH
	!EDWARDS
	!ROOM 1
	!GERMAN
	!JONES
	!ROOM 4
	! R.K.
	!ROBINSON
	!ROOM 3

MULTIPLE ROOMS

MR NO	ROOMS	
1	ROOM 5	ROOM 6

LENT TERM TIMETABLE BY COMPUTER

SCOCOST DATE 25/8/72

TEACHERS	PERIOD 1	PERIOD 2	PERIOD 3
MONDAY	!MATHS	!FRENCH	!ENGLISH
	!FORM 1	!FORM 1	!FORM 1
	!ROOM 3	!ROOM 3	!ROOM 3
	!GREEK	!GEOMETRY	!GERMAN
	!FORM 2	!FORM 2	!FORM 2
	!ROOM 5	!ROOM 5	!ROOM 4
	!SCIENCE	!SCIENCE	!R.K.
	!FORM 3	!FORM 3	!FORM 3
	! R 1	! R 1	!ROOM 3

MULTIPLE ROOMS

MR NO	ROOMS	
1	ROOM 5	ROOM 6

Sheet AP4/13 - Print out of Timetable in two non standard formats.

REFERENCES

Note: A list of abbreviations of organisations is given after the last reference.

1. Almond, M. An algorithm for constructing university timetables. The Computer Journal Vol 8 (1966) pp 331-340.
2. Appleby, J.A., Blake, D.V., and Newman, E.A. Techniques for producing school timetables on a computer and their application to other problems. The Computer Journal Vol 3 (1961) pp 237-245.
3. Berghuis, J., Van der Heiden, A.J., and Bakker, R. The preparation of school timetables by computer. Behandling Informations- Tidskrift Vol 4 (1964) pp 106-114.
4. Barraclough, E.D. The application of a digital computer to the construction of timetables. The Computer Journal Vol 7 (1965) pp 136-146.
5. Brittan, J.N.G., Kay, L., McIlroy, W.A., Whitehead, H., and Williams, D.H. R.M.C.S./H.Q. T.C. computer timetable project - pilot study at No 9 School of Technical Training. Report TC/296065/5/RES Research branch R.A.F. Training Command, March 1972.
6. Brittan, J.N.G. Shrivenham computer system for the compilation of school timetables - SCOCOST. R.M.C.S. technical note C.S.10 issue 2, Jan. 1972.
7. Britten, A.A., and Groom, K.N. The use of computers for school timetabling. Report C19. L.G.O.R.U. Reading, June 1967.
8. Chung, C.M. Heuristic programming and the school timetabling problem. M.Sc. project report. U.L.I.C.S., Sept. 1971.

9. Davies, T.I. School organisation. Pergamon Press (Oxford) 1969.

10. Dempster, M.A.H. Two algorithms for the timetable problem. Combinatorial mathematics and its applications, D.J.A. Welsh ed., p 63-85. Academic Press (London) 1971.

11. De Werra, D. Résolution de problèmes d'horaire par la théorie des graphes. Doctoral thesis École Polytechnique Fédérale de Lausanne, 1969.

12. Duncan, A.K. Further results on computer construction of school timetables. Communications of the Association for Computing Machinery, Vol 8 (1965) p 72.

13. Egner, W.E., and Dixon, C. The first operational time-table by Computer. Computer Education No 10 (1972), pp 16-17.

14. Farley, P.R. ALGOL/CORAL/FORTRAN library procedures - mark II visual display units. R.M.C.S. technical note C.S. 13, May 1972.

15. Folkers, J.S. Timetable construction by computer. Report THLR 68. Department of Mathematics, Technical University of Delft, Aug. 1968.

16. Gotlieb, C.G. The construction of class teacher time-tables. Information Processing 1962, C.M. Popplewell ed., pp 73-76. North Holland Publishing Co. (Amsterdam) 1963.

17. Groom, K.N. The New Zealand computer timetabling system - a preliminary application. Report St.1, S.T.A.G. Reading, March 1972.

18. Groom, K.N., and Gwynne, D.W. School timetabling by computer - the report of a field trial. Report C82, L.G.O.R.U., Reading, Sept. 1970.

19. Hall, P. On representatives of subsets. Journal of the London Mathematical Society Vol 10 (1935) pp 2-30.

20. Hickman, D. A phrase structure language for the compilation of timetables. Report ICSI 224. U.L.I.C.S., Sept. 1969.

21. Johnston, H.C., and Wolfenden, K. The computer aided construction of timetables. Information Processing 1968, A.J.H. Morrell ed., Vol 2, pp 1368-1376. North Holland Publishing Co. (Amsterdam) 1969.

22. Johnston, H.C. The computer aided construction of timetables. Ph.D. thesis, University of London, 1970.

23. Lawrie, N.L. Linear programming as an aid to school timetabling. British Joint Computer Conference 1966, pp 16-22. I.E.E. conference publication No. 19, 1966.

24. Lawrie, N.L. School timetabling by computer: methods and problems. Bulletin of the Institute of Mathematics and its Applications. Vol 7 (1971) pp 210-215.

25. Lazak, D. Die beziehungen des Kollisionsindexverfahrens zum Gotlieb - verfahrens Electronische Datenverarbeitung Vol 10 (1968) pp 26-37 and 58-61.

26. Lewis, C.F. The school timetable. Cambridge University Press (London) 1961.

27. Lions, J. The Ontario school scheduling program. The Computer Journal Vol 10 (1967) pp 14-20.

28. Lions, J. A generalization of the method for the construction of class teacher timetables. Information Processing 1968, A.J.H. Morrell ed., Vol 2 pp 1363-1367. North Holland Publishing Co. (Amsterdam) 1969.

29. McIlroy, W.A. An investigation of difficulty factors in computer timetabling. R.A.F. Training Command. (to be published).

30. Michalsen, H. Timeplanlegging. Tapir Forlag (Trondheim) 1971.

31. Michalsen, H. A working strategy for general school scheduling. Tapir Forlag (Trondheim) 1972.

32. Norman, J.M. Heuristic procedures in dynamic programming. Manchester University Press (Manchester), 1972.

33. Sherman, C.R. A combinatorial problem arising from scheduling university classes. Journal of the Tennessee Academy of Science, Vol 38 (1963) pp 115-117.

34. Welsh, D.J.A., and Powell, M.B. Upper bounds for the chromatic number of graphs and its application to time-tabling problems. The Computer Journal, Vol 10 (1967) pp 85-86.

35. Woodward, P.M., Wetherall, P.R., and Gorman, B. Official definition of CORAL 66. H.M.S.O., 1970.

36. Yule, A.P. Extensions to the heuristic algorithm for university timetables. The Computer Journal, Vol 10 (1968) pp 360-364.

37. Brittan, J.N.G. (ed.). Papers presented at a conference on course planning, scheduling and timetabling by computer, June 1972. Obtainable from the editor at R.M.C.S.

38. International Computers Ltd. Direct Access Manual. I.C.L. (Reading), manual no 4195, 1970.

LIST OF ABBREVIATIONS

U.L.I.C.S. University of London Institute of Computer Science, 44 Gordon Square, LONDON W.C.1.

I.F.I.P. International Federation for Information Processing.

L.G.O.R.U. Local Government Operational Research Unit, 201 King's Road, READING, Berks.

R.M.C.S. Royal Military College of Science, Shrivenham, SWINDON, Wilts.

S.T.A.G. School Timetabling Application's Group, 201 King's Road, READING, Berks.