ORiON, Vol. 11 No. 1/2, pp. 5-17 ISSN 0259-191-X

OPTIMISATION OF RESOURCE UTILISATION AT A UNIVERSITY - AN ALLOCATION PROBLEM

C.J. SCOGINGS
Department of Computer Science

and

P.W. UYS
Department of Mathematics
and Applied Mathematics
University of Natal
Pietermaritzburg
South Africa

ABSTRACT

Student enrolment at the University of Natal has been increasing steadily over the years. Moreover additional new courses are introduced from time to time. Despite this State subsidies are declining in real terms. These factors imply escalating demands on physical resources.

Historically, at this university, lecture rooms have been used only during the mornings and laboratories only during the afternoons. An obvious solution to meet the demand for accommodation is to double up on the number of timetabled periods so that the lecture rooms are in use the whole day. Since there are many classes which are in fact too large to be accommodated in any one room it is also necessary to split these classes into separate lecture groups. Likewise classes have to be divided up into several smaller groups for laboratory and tutorial sessions.

The policy at this University is to encourage students to choose curricula including courses selected from as wide a range as possible. The above timetable strategy apparently facilitates this. In practice, however, to ensure that student numbers are evenly distributed across alternative sessions for a given course and to do this for all courses simultaneously while avoiding clashes is not a simple matter.

To achieve this, a heuristic algorithm to allocate students to lecture, practical and

tutorial sessions was developed. This algorithm has been successfully implemented for each of the past three years.

1. INTRODUCTION

Until about ten years ago the University of Natal enjoyed the luxury of relatively abundant lecture venues compared to the number of courses on offer. Class sizes too did not pose a problem. However cutbacks in State funding have meant that infrastructure has not kept pace with increases in student numbers. Moreover new courses were and still are being introduced and an ethos has been evolving of encouraging students to select courses for their curricula from as wide a range as possible. The time came when many classes were too big for the largest venues and it became necessary to divide such classes into smaller groups for lectures.

Indeed to cope with all these requirements a new style of timetable had to be introduced. Whereas previously lectures were given in the mornings only and tutorials and practicals in the afternoons, the new timetable schedules lectures and practicals throughout the day. Since the total number of periods in a week is 50 and there are hundreds of sessions many lectures, tutorials and practicals of necessity clash. The timetable had to be designed to reduce to a minimum the number of desirable combinations of courses that would be rendered unavailable because of such clashing. The decisions involved here are somewhat subjective and arbitrary and it is not the purpose of this paper to discuss the design of the timetable.

A very serious problem still remains and that is how to allocate students to the various classes. To give some idea of the nature of the problem, consider the typical first year curricula as shown in Tables 1 to 3.

For any given curriculum there are apparently a vast number of possible combinations of sessions. Thus for the curriculum in Table 1 the total number of potential combinations is 2x6x2x11x2x6x2x17x11. This is more than 1.18×10^6.

Course	Number of lecture streams	Number of tutorial groups
Economics 1	2	6
Accounting 1	2	11
Quantitative Methods 1	2	6
Commercial Law 1	2	17
Business Information Systems 1	1	11

Table 1. Specimen first year curriculum in Commerce Faculty

Course	Number of lecture streams	Number of tutorial groups	Number of practical groups
Economics 1	2	6	0
Maths 1	2	6	0
Chemistry 1	2	14	9
Physics 1	1	4	6

Table 2. Specimen first year curriculum in Science Faculty

Course	Number of lecture streams	Number of tutorial groups	Number of practical groups
Economics 1	2	6	0
Maths 1	2	6	0
Psychology 1	2	26	4
Computer Literacy	1	5	0

Table 3. Specimen first year curriculum in Arts Faculty

Clashes will however render many of these impossible. This problem of clashing is aggravated since, as seen from the above tables cross-faculty selection of courses can occur, i.e. some courses are available in several faculties. Two extreme examples are Economics and Mathematics which are each available in five faculties. This phenomenon greatly increases the complexity of not only the timetable but also the problem of allocating students since any one of such courses is associated with many other courses.

At the time the timetable is revised for the forthcoming year, venues are chosen for these various sessions. The principle used is to choose the smallest room sufficiently large to cater for the anticipated number of students assuming an uniform distribution within a course across the tutorial sessions and also across the lecture streams. This arrangement optimises usage of the physical facilities and manpower. But it also means that a given venue cannot accommodate more students than were planned for that session.

Consequently, to allow students to choose sessions for themselves would be inviting chaos. Certain times might be particularly popular or unpopular resulting in overcrowding or uneconomically small classes. But more seriously a very unfortunate consequence could arise as follows: Some students, who because of their particular curricula are able to attend only certain sessions, could find these sessions already fully occupied by students who could in fact attend other sessions.

To prevent such situations arising it is essential to impose a centralised allocating system. This system needs to be based on a computerised algorithm. A suitable algorithm was developed here in 1991 and since then this algorithm has been successful in allocating students at the commencement of each academic year.

Of course, owing to their practical importance and difficulty, assignment problems have been extensively studied. Hertz [1] provides a useful survey and describes heuristic procedures based on tabu search techniques. Another allocation problem is discussed by Sinclair and Esterhuysen [2] who also give a useful list of references.

Generally, solution methods that have been described use a heuristic procedure to find a feasible solution, that is, a solution satisfying a suitable subclass of the constraint set. This solution is then improved so as to satisfy a larger subclass of the constraint set. In this regard, as Tripathy [3] has explained, quality of the solution at any stage often becomes an issue requiring the involvement of the decision maker.

The system described in this paper uses an algorithm to generate an initial feasible solution. An iterative process then uses the same algorithm to produce solutions satisfying larger and larger subclasses of the constraint set. Particular advantages of this system include the simplicity of the concept involved and the fact that reports are produced which identify problem areas. These reports facilitate the involvement of the decision maker identified as so important by Tripathy. In contrast to this, a mathematical formulation would lead to a cumbersome large size problem without necessarily achieving a satisfactory level of user involvement.

Although the system described is in the context of allocation of students to groups the principles used can be readily employed in other assignment problems such as timetabling.

2. THE STUDENT ALLOCATION SYSTEM - INTRODUCTION

For clarity of exposition it is necessary to provide some preliminary definitions and to give an outline of the system.

A. Preliminary definitions

A **session** is a set of four or five lectures (as given in one week to a particular course); or one tutorial or one practical which extends over several timetable periods.

One course could have one, two, or more parallel lecture sessions, several tutorial sessions and several practical sessions. These sessions constitute what is called the **lecture session group, the tutorial session group** and the **practical session group** for that course.

The following data concerning the Psychology 1 course illustrate this:

SESSION GROUP NAME	SESSION NAME	TIMETABLE PERIODS
Psychology 1 lecture group	Psychology 110+ Psychology 110++	M1;T5;W2;H4 M6;T2;H3;F1
Psychology 1 practical group	Psycho 1 Prac Mon Psycho 1 Prac Tue Psycho 1 Prac Thu Psycho 1 Prac Fri	M9;M10;M11 T9;T10;T11 H9;H10;H11 F4;F5;F6
Psychology 1 tutorial group	Psycho 1 Tut 1 + 25 others	M10

Table 4. Psychology 1 session groups
(H denotes Thursday)

If a student is registered for a course he has to attend exactly one each of the lecture sessions, tutorial sessions and practical sessions selected from the *lecture session group*, the *tutorial session group* and the *practical session group* respectively for that course. Thus a student taking Psychology 1 could, depending on his other courses, attend the following sessions:
Psychology 110+; Psycho 1 Tut 1; Psycho 1 Prac Tue (but not Psycho 1 Prac Mon).

The **session file** is a list of all the *session records*. A **session record** consists of:

- name of the session (eg: Economics 100 Monday Tutorial)
- name of the course (eg: Economics 100)
- maximum number of students which can be accommodated in the session (determined by the venue selected)
- one to five periods (the timetable periods required by the session)
- **session priority value** (this is an integer in the range 1 to 10 with an initial value of 5. This value may need to be changed as explained later)

The **Student File** is a list of all the *student records*.

Each **student record** consists of:

- student name
- student number
- student faculty
- one to six courses

B. Outline of system

i. Input requirements

The input for the system is drawn from the *student file* and the *session file.*

ii. Data accumulated during execution

During the running of the program, the following data are gradually accumulated:

- a personal timetable for each student.

  The personal timetable must be produced in such a way that the student is not expected to attend two different lectures or tutorials simultaneously. When the student is allocated to a new session, the appropriate periods are marked in his timetable.
- the current number of students allocated to any particular session - This is termed the **current number of the session.**

  This number must never exceed the maximum number of students allowed in the session. It is also desirable that the numbers allocated to a set of similar sessions should be "balanced". For example, it is desirable to allocate 120 students to each of the six Economics 100 tutorials, even though each tutorial venue in fact has a maximum capacity of 200. Note that this goal is desirable but it is not necessary to achieve precision in this matter.

3. THE ALLOCATION ALGORITHM

The algorithm allocates each student individually. It begins by determining all the *session groups* to which the student needs to be allocated. This yields a list with the courses appearing *in the order in which they occurred when the student file was produced.* This order will generally be random but it will be seen later that this order is in fact immaterial.

The program first allocates the student within those *session groups* which consist of one session. An allocation is made to a session only if the *current number* of the session is less than its maximum value and if the session periods do not clash with any previously allocated periods. It may be impossible to make an allocation at one of these stages for one of two reasons:

i. The student has an incompatible selection of courses in his curriculum i.e. his curriculum cannot work with the existing timetable. This situation often arises when a student is repeating a course so that he has a mixture of courses from different levels. If possible the student should change his curriculum. Otherwise he has to do the best he can with this condition.

ii. The enrolment for that course is larger than anticipated and hence exceeds the accommodation provided for the course. This can be remedied by changing the venue. As this could be very difficult it is better to over-estimate the expected student numbers when selecting venues in the first place.

In either case one is not dealing with an inadequacy of the algorithm as such.

Next the algorithm allocates the student to *session groups* which consist of more than one session. All the sessions in the various *session groups* to which the student has to be allocated are sorted in ascending order according to their *session priority value* (initially 5) and within that ordering according to their *current number.* The program now attempts to allocate the student to sessions in this sorted order. It will assign the student to the first session it finds which does not clash with sessions to which the student has already been assigned.

If it is not possible to perform an allocation to a session in some *session group* (this would be the case if all potential sessions clash with some session to which the student has already been assigned) then the algorithm backtracks to the previous *session group.* It now attempts to allocate another session (in sorted order) in this *session group.*

If a different session is allocated the algorithm progresses to the next *session group*, otherwise it backtracks to the previous *session group*.

If the program is unable to assign a student to a session in any one of his *session groups* the student presents a problem. His details are appended to a **problem file** with a report giving the reason for the failure.

When the program has succeeded in completely assigning a student it adds the details concerning that student and his assignment to the **student list**. It then proceeds to process the next student.

The program also produces a list called **session statistics**. This is a list of all sessions showing the final *current* and maximum numbers in each session and also the current *priority value* for each session.

It will be recalled that the initial run of the program is performed with all the *session priority values* set at 5. A second run of the program is now carried out but with modified *session priority values*. A perusal of the *problem student file* enables one to identify which sessions are producing difficulties. Confirmation can be obtained by consulting the *session statistics file* since this shows which sessions are indeed prematurely filled. The *priority values* for those sessions need to be raised so that in the second run of the program students are assigned to them less frequently.

This is also an opportunity to even out the allocation of student numbers to the various sessions in a given *session group*. Under-utilised sessions should be given numerically smaller *priority values* so that they will be first in the sorted list of sessions in a *session group* to be allocated.

To summarise, it should be particularly noted, that the *lecture session groups*, the *tutorial session groups* and the *practical session groups* are sorted according to the *priority values*, which are found in the *record* for each session in the *session* file. Allocations are then made according to this sorted order with due regard to the current session numbers. Hence a change in some of the *priority values* can lead to a completely different pattern of allocations for **all** students.

The use of *priority values* is an essential part of the system. An initial run of the system is performed and various "problem" sessions are identified. These include:

- sessions which have a low *current number* of students relative to the maximum number for that session. This usually means that similar sessions will not be "balanced".
- sessions that have their maximum number of students allocated. This condition could have been attained **before** all students had been allocated. This often means that students who should have been allocated to these sessions only, have consequently not been allocated at all.

The *priority values* in the session file are then adjusted accordingly. A full session will be allocated a numerically high *priority value,* and thus be last in the sorted list of sessions and an under-utilised session will be allocated a low *priority value* and thus be first in the sorted list of sessions to be allocated.

By changing the *priority values* and studying the output of several runs through the list of students, the number of students that cannot be allocated is minimised, and similar sessions can be reasonably "balanced".

4. OUTPUT OF THE SYSTEM

The program generates several output files. Each file takes the form of a list.

i. Session registers

For each session the system produces a register of student names allocated to that session. This gives, in other words, a list of students in each lecture session, tutorial session and practical session for all the courses. This is important and useful for the various academic departments.

ii. Session statistics

A list of all sessions, the *current* and maximum numbers in each session and the *current priority value*. This is needed to fine-tune the program between runs.

iii. Student list

An alphabetical list of all students, indicating to which lecture session and which tutorial and practical sessions the student has been allocated, for each course that the student is taking. This is the list that is published for the information of the students.

iv. Problem cases

A list of students which could not successfully be allocated. This may be caused by certain sessions reaching their maximum number or a student may simply have chosen a set of incompatible courses.

5. CONCLUSION

Careful consideration of the above algorithm will convince that it is capable of achieving a satisfactory allocation. The reason for this is the iterative approach employed and the possibility of fine tuning using *priority values*. One of the main objectives is to achieve an allocation for which students numbers are uniformly distributed across the sessions in each of the session groups. This goal can be approached as closely as desired by employing sufficiently many iterations. In practice the gains in quality must be weighed up against costs in effort and time. For many session groups it is simply not necessary for the allocation to the sessions to be exactly uniform. This again is the aspect of quality as discussed by Tripathy and calls for the involvement of the decision maker.

Practical experience shows that using a 486 PC the program can, in an initial run, allocate students at the rate of 250 a minute, despite encountering "problem" cases which can take up to 10 seconds each. (Surprisingly such "impossible" problem cases turn out to be infrequent). The speed doubles in the second run since less backtracking is needed. So although the algorithm appears to be rather inefficient, particularly with regard to the backtracking strategy, in practice the speed it achieves is perfectly acceptable. An obvious refinement of the algorithm would be for it to add to a list those problem combinations of courses as they are encountered. Subsequently, during the remaining part of the allocation run, details of students encountered having such combinations are added to the *problem file* immediately. No attempt is made to allocate those students. This strategy saves time since it obviates the algorithm going through fruitless extensive backtracking procedures. Our experience with this algorithm shows, however, that such a refinement is really not necessary when dealing with only a few thousand students.

A very useful by-product of the algorithm is that the *session priority values* can be exploited in future revisions of the timetable. If a session needs a value less than 5 it means that it is relatively difficult to find students having curricula permitting an allocation to that session. That session is therefore scheduled at an unsuitable time and should be moved to some other time.

Before the introduction of the student allocation system the situation at the beginning of the academic year was becoming chaotic. Students were actually unable to get into many lecture rooms because of overcrowding. Allocation of students to tutorial groups and practical groups was done by the individual departments concerned and involved a tremendous amount of labour and time. The necessary coordination among the departments in allocating students was done on an ad-hoc basis and several weeks would elapse before matters were satisfactory.

Since the introduction of this system in 1991 student enrolment has increased by 15% yet despite this, none of the previous problems have appeared. The entire process of allocation is accomplished on the final day of registration and students have the requisite information the next morning before lectures start.

REFERENCES

[1] A. HERTZ, Tabu Search for large scale Timetabling Problems, *European Journal of Operational Research*, **54**(1), 39-47, (1991).

[2] M. SINCLAIR and D.G. ESTERHUYSEN, Allocating Conference Delegates to Workshops: A Special Timetabling Problem, *ORiON*, **7**(2), 95-103, (1991).

[3] A. TRIPATHY, Computerised decision aid for timetabling - a case study, *Discrete Applied Mathematics,* **35**, 313-323, (1992).