THE DEVELOPMENT OF A GENERAL ALGORITHMIC PROCEDURE FOR UNIVERSITY EXAMINATION TIMETABLING

by

WAN ROSMANIRA ISMAIL

B.Sc.(Hons), M.Sc.

A Doctoral Thesis submitted in partial fulfilment of the requirements for the award of the Doctor of Philosophy of Loughborough University

ACKNOWLEDGEMENTS

First and foremost I thank the Almighty Allah for His mercy and guidance throughout my studies and in completion of this thesis.

I would like to express my deepest gratitude to my supervisor, Mr. David Johnson, for his support and endless help in my work and in writing the thesis. Thank you too to Professor M King, my Director of Research and Dr. Alan French for their invaluable comments.

I would also like to thank my sponsor, Universiti Kebangsaan Malaysia and Malaysian Government for funding my work and giving the study leave which has allowed me to pursue the degree.

Last but not least, my sincere thanks to all my family, friends and colleagues for their encouragement and help in so many ways.

ABSTRACT

The problem of scheduling university examinations is becoming difficult for examination officers especially when they have to construct the timetables manually. It is largely due to the increasing number of students and greater freedom in choosing the courses. Examination officers would have to spend a considerable amount of time checking for student conflicts so that no student would have to sit for more than one exam at any one time. There are also other limitations such as the number of examination rooms, the length of the examination period and others. The examination timetabling problem varies between institutions, depending on their particular needs and limited resources. Most of the existing computerised examination timetabling systems found in the literature are developed and used by particular institutions.

Therefore, the aim of the research is to produce a general computerised system for timetabling examinations which can be used by most universities. The research is done in two stages; the first stage involves carrying out a survey on the university examination timetabling systems and the second stage is the construction of a university examination timetabler incorporating the common objectives and constraints found in the survey.

The survey was carried out to determine the extent to which the computerised examination timetabling procedures are used, to identify the objectives and constraints which are commonly considered when constructing examination timetables and to evaluate the effectiveness of the existing examination timetabling systems in achieving the objectives and satisfying the constraints

The construction of the general examination timetabling system is done in two parts. In the first part, a new algorithmic rule is developed to assign exams to the minimum number of sessions without creating conflicts for any student. The rule adopts a clique initialisation strategy as a starting point and a graph colouring approach for assigning the exams. This rule is also quite capable of scheduling exams to the sessions which are as close as to the least number of sessions possible, without having to carry out any

backtracking process. The backtracking process can sometimes be time consuming if there are a lot of exams firstly to be scheduled, and secondly clashing with each other.

The second part of the work involves minimising the total number of students taking two exams on the same day and scheduling large exams early in the examination period subject to a specified time limit on the overall examination period and a maximum number of students that may be examined in any session. A swapping rule was introduced where exams in one of the sessions in any day with large number of same-day exams are interchanged with exams in other sessions which will reduce the total number of same-day exams. The experimentation showed that if the swapping procedures are repeated three times, the total number of same-day exams will be reduced by 50%. The total number of same-day exams will be reduced even more if some extra sessions can be added to the initial minimum number of sessions. A simple rule was devised to schedule large exams early in the examination period.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Introduction To Scheduling Problems

Scheduling problems arise when there are a number of activities to be performed and there are alternative ways of doing them, and also resources or facilities are not always available for performing each activity at the best possible time or in the best possible way. The primary concern is to ensure that the combination of activities and limited resources or facilities produces the best outcome as well as satisfying all the requirements and constraints. Some of the scheduling problems found in a large variety of areas are:

- *machine scheduling in a factory.* An important design problem in manufacturing concerns the optimal allocation of machines to workstations within a manufacturing system. In single machine scheduling problems [Potts and Van Wassenhove, 1991], the objective is to find the sequence of jobs such that the weighted tardiness of a set of jobs on a single machine is minimised.

- *shift scheduling of staff in service organisations.* The problem involves determining the number of employees to be assigned to each shift and specifying the timing of their relief and lunch breaks. An example of this type of problem is personnel scheduling at airline ground stations [Brusco *et* al, 1995]. The objective is to minimise the costs of labour and increase productivity from the efficient utilisation

of the airline ground station staff while taking account of various constraints on the staff.

- *scheduling of sporting activities*. Every year the English cricket authorities must timetable the county fixtures for the teams competing in the major English county cricket tournaments [Wright, 1994]. The objective is to reduce the cost of accommodations and long-distance travelling from one venue to another. Some constraints that are taken into account are the requests to have or not to have home matches on particular dates, requests to have a match against a particular opponent on specific date etc.

- *scheduling of magistrates to courts*. The problem is to assign a pool of magistrates to particular sittings of courts. For a particular sitting of a court, there will be a series of constraints and/or requirements as to who may sit in that court [Wilson, 1981].

- *conference seminar timetabling*. The problem is to schedule a number of seminars in a set of periods, with a maximum number of seminars that can be held during any time period [Eglese and Rand, 1987]. The objective is to satisfy as many as possible of the participants' preferences subject to time and conflict constraints.

- *school timetabling*. This involves scheduling a number of classes or tuples to a fixed number of time slots. Each tuple consists of a set of students, a teacher, a subject and a room. A number of tuples may be scheduled in the same time slot provided that no class, teacher or room appears more than once in any time slot. The problem often occurs as a satisficing rather than an optimising problem for a given set of subject/class requirements [Abramson, 1991].

- *university course timetabling*. The problem is to reduce the number of conflicts, which arises when courses taking place simultaneously involve common students or lecturers, or require the same lecture rooms. The problem becomes difficult since several constraints have to be taken into account such as courses which involve a

large number of students have to be repeated several times during the week, the length of the courses may not be uniform, lecturers may not be available or may prefer to teach at certain times of the week and sufficient time should be provided for students to move from one building to another if necessary [Hertz, 1991].

Each of these problems has been solved using a mathematical formulation such as linear programming, integer programming or a branch-and-bound approach. In recent years however, heuristic algorithms have been introduced and have become an important tool in solving such problems As the size of the problem increases, they become difficult to solve using conventional methods. Heuristic methods are very popular because they are known to cope better with a large number of variables and constraints and with problem complexity. Their appeal also stems from their ability to produce solutions more quickly even though they are not always optimal.

1.2 The University Examination Timetabling Problem

1.2.1 The Problem

The work of this thesis also falls into the category of scheduling problems, in particular one of the three different timetabling problems in educational environments (school timetabling, university course timetabling and university examination timetabling). The examination timetabling problem in general is an attempt to assign activities (exams) to timeslots (sessions) taking account of constraints on the activities and resources (exam rooms, invigilators etc.). The examination timetabling problem in higher education varies between institutions, depending on their particular needs and limited resources.

The problem of scheduling university examinations at the end of each term or semester is becoming increasingly difficult. The complexity of the problem depends largely on the number of students registered and the number of courses taught, as well as the

amounts of freedom students have in choosing their courses. In general, greater freedom of choice for a large number of students increases the difficulty of producing an examination schedule to fit into a limited time interval without creating conflicts for any student. With hundreds of courses and thousands of students to be examined, it is also essential to have a very efficient timetabling system so that the examination timetable produced does satisfy all requirements.

The construction of a university examination timetable usually requires two stages. The first stage involves gathering the information regarding exam details, student details and special requests from the departments. From the data relating to the students and the courses they take, the clash list is produced. The clash list is a list of all pairs of exams that have at least one student in common. In the second stage, the exams are organised into schedules by allocating them to sessions bearing in mind that it is crucial to avoid any clashes for students. The exams are also allocated to their respective accommodation available bearing in mind that the number of students should not exceed the capacity of each examination room. Finally, the requisite number of invigilators will be assigned.

1.2.2 Objectives And Constraints Of The Examination Timetabling Problem

The objectives of the examination timetabling problem are the goals that examination officers are trying to achieve when timetabling examinations. The objectives can be divided into two categories; student oriented objectives and institutional oriented objectives. Some of the objectives that fall into the first category are:

- the timetable must if at all possible be free of conflict, i.e. no student should be required to take more than one exam at any one time. This is the common primary goal of the examination timetabling system.

- ideally, students should not be required to take exams more than one exam in two (or perhaps more) consecutive sessions. In practice, this often translates into students not having to take more than one exam on any day for however many sessions there are in the day. Generally, the exams should be spread out as evenly as possible throughout the examination period.

- to take account of the year of course when scheduling examinations. Some departments assess their final year students by both written exams and project work. Often the written exams are held before the presentation of their project works. Therefore, they must sit their exams as early as possible so that the presentations and other oral examinations can be conducted within the examination period. It also means that final exams can be marked fairly early to enable graduation to take place.

- to avoid having exams of different durations in the same room. Students having a 2 hour exam will usually finish earlier than those having a 3 hour exam. Therefore, students with the shorter exam will leave the examination room earlier and by doing so may cause some disruption to other students.

Common objectives from the second category are:

- to schedule large exams early in the examination period. This is to allow as much time as possible for marking.

- to minimise or fit within the overall duration of the examination period. This is to avoid having the examination period exceeding a predetermined number of days, which often cannot be easily extended if results need to be available by a certain time.

- to avoid splitting any exam between two or more rooms. If an exam is split between two or more rooms, it can be difficult for the lecturer in charge of the exam to

administer the students. The lecturer may have to move from one room to another quite frequently so that any problems that arise can be solved quickly.

Constraints, sometimes referred to as side constraints, are the requirements of the activities and resources which must be satisfied in order to produce a feasible timetable. Some of the constraints that are usually taken into account when constructing examination timetables are:

- all examinations must be scheduled within the specified examination period. The examination period is usually tightly defined, particularly in a semesterised system.

- in any session, there is a maximum number of students that may be examined. The number of students that can be examined depends on the number of seats available in the examination rooms.

- certain specified exams may require special facilities or equipment, such as laboratories.

- certain specified exams must take place during the same session. They are usually considered as one examination by merging them into a composite subject.

- certain specified exams cannot take place during the same session. These exams are usually treated as if they have a student in common.

- certain specified exams must take place during a defined subset of the sessions. This may be done to give priority to large exams so that there will be enough time for marking.

1.2.3 Some Definitions

This thesis uses certain terminology which may need some clarification. This is especially so due to the specialised use of certain terms in the field of examination timetabling and also due to the possibly ambiguous nature of them.

- A *session* usually refers to each of the non-overlapping time slots available in any day.
- An *examination period* is the total time duration during which all examinations are conducted.
- A *conflict-free assignment* is one where no student is scheduled to take more than one examination in any session.
- The number of *same-day exams* is the number of students having to take more than one exam on the same day.
- The *capacity* is the maximum number of students that could be examined at the same time using all of the available examination rooms.
- The *duration* of an exam is the length of time needed to sit the exam. Typically there are exams of 2 hour duration and also 3 hour duration.

1.3 Research Aim

The aim of the research is to produce a general computerised examination timetabling system for universities. The main area of this research is twofold. Firstly, a survey of university examination timetabling systems in universities is carried out and secondly

the construction of the university examination timetabler incorporating the common objectives and constraints found in the survey.

The survey is intended to discover how different universities schedule their examinations. One of the main aims of the survey is to identify the objectives and constraints that are considered when constructing examination timetables, and to examine the extent to which computerised timetabling procedures are used in different university structures. Another reason is to evaluate the effectiveness of different examination timetabling systems in achieving these objectives and satisfying the constraints.

The second part of the research work entails producing a method to timetable examinations which can be used by most university examination officers to schedule the exams at the end of each semester or term. Existing university examination timetabling systems found in the literature have tended to be developed and used by particular institutions. The systems are mainly designed to be appropriate only to that university's specific needs. The proposed general examination timetabling system will be able to handle the commonly used objectives and constraints which are identified in the survey.

Our aim is also to produce a fast and flexible examination timetabling system that provides a choice of the number of sessions to be used. This is different from previous systems where the number of sessions is usually fixed. The proposed system will offer the examination officers a choice from the minimum number of sessions required to assign all non-conflicting exams to the maximum number of sessions available. At the same time, the number of students having to take more than one exam in any one day will be calculated. Ideally we can say that the more the number of sessions that are used, the less the number of same-day exams will be. But we do not know how much better the solution quality will be. This will be a measure of the trade-off between the number of same-day exams and the number of sessions used.

1.4 Proposed Solution To The Problem

A procedure for examination timetabling in universities using a method based on a graph colouring approach is used in this thesis. In the proposed solution, the problem is decomposed into two subproblems; (i) the timetabling problem without side constraints and (ii) the timetabling problem with some side constraints.

In the first subproblem, an algorithmic rule is developed to assign exams to the least number of sessions such that no student has to sit for more than one exam at any one time. The rule employs a graph colouring approach and a clique initialisation strategy. The rule is also designed in such a way as to avoid the need for backtracking. The second subproblem is actually an extension to the first subproblem. The most common constraints discovered in the survey will be incorporated into the first subproblem. A rule-based method is then introduced to solve the underlying problem.

1.5 Structure Of The Thesis

Following this introduction to the university examination timetabling problem, the rest of the thesis is arranged as follows:-

Chapter 2 reviews the literature of the university examination timetabling problem. It reports on the different methods used in solving specific problems at particular institutions, namely graph colouring and heuristic methods. It also lists the objectives and constraints of the particular institution concerned when timetabling examinations.

Chapter 3 describes a survey carried out into the examination timetabling systems used in U.K and Eire universities. First the reasons for carrying out the survey are explained and then the methodology and data collection process are described. Finally the interpretation of the data and the findings from the survey are discussed.

A detailed description of data sets used in the testing is given in Chapter 4. Some techniques used to solve the examination timetabling problem without costs are discussed, namely the initial ordering strategies, the clique initialisation strategy and the backtracking process. The results by Carter *et al.* [1996] are presented and these results are used for comparison in the subsequent work.

Chapter 5 considers a heuristic based solution method to solve the examination timetabling problem without costs and side constraints. The newly developed algorithmic rule can be divided into three parts; firstly to find sets of maximum cliques, secondly to schedule the critical exams and finally to schedule the non-critical exams. The numerical results obtained using this algorithmic rule are then presented and compared with those of Carter *et al.* [1996].

Chapter 6 extends the work done in the previous chapter, by incorporating the results obtained in Chapter 3. It explains how the two side constraints that are mainly considered by examination officers when constructing timetables, are incorporated into the basic algorithmic rule. It then incorporates the procedure to minimise the total number of same-day exams, i.e. the number of students having to take more than one exam on the same day. A method of swapping exams in sessions is introduced to tackle this problem and the results using this method are presented. Finally we describe the method of assigning the exams in each session to available rooms, and sessions to the days in the examination period.

The final chapter, Chapter 7 summarises the whole report and discusses some general conclusions. Finally, we describe some areas where future work in this area could be directed.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Timetabling is used in almost every aspect of human life. We will surely be consulting some sort of timetable or schedule whenever we are travelling, working or just watching television.

Progress in the area of university examination timetabling in the early years was hampered by the lack of computer power to handle large-scale real problems. Much of the early work was theoretical or conceptual with limited implementation and testing on real problems. Most testing was only of small-scale problems with limited realism. Another handicap was the inability to include many soft constraints, particularly concerning students' and lecturers' convenience. With the rapid increase in computing power, work in timetabling has progressed along with the advances that were possible.

Different approaches have been used to tackle university examination timetabling problems. Much early work was based on graph colouring but later heuristic methods were introduced as the problems became larger, and more constraints were incorporated. There have also been many attempts at timetabling using mathematical approaches such as integer linear programming.

This literature review is organised according to the different approached used in timetabling examinations in universities. It begins by looking at graph colouring heuristic approaches, then at local search methods and finally the integer programming approach.

2.2 Graph Colouring Approach

The simple examination timetabling problem, i.e. the problem of finding a clash-free examination timetable where no student has to take more than one exam at any one time is equivalent to the graph colouring problem. For a given examination timetabling problem, an exam is represented by a vertex and an edge connects two exams which have at least one student in common and cannot be scheduled together. The degree of a vertex is the number of edges adjacent to it, i.e. the number of other exams with which there are students in common.

In the graph colouring problem, the principal objective is to determine the minimum number of colours needed to colour the vertices such that no two vertices connected by an edge have the same colour. Therefore, the process of colouring the vertices is similar to assigning exams to sessions in order to produce a clash/conflict free timetable.

The minimum number of colours necessary to colour the vertices of a graph is called the chromatic number of a graph, denoted by χ. The problem of computing χ is known to be NP-Complete where the CPU time required grows exponentially with the number of vertices (Karp [1972]). It is reported in Manvel [1981] that this property causes the graph colouring problem with over 100 vertices to become unmanageable. However, with advances in computers nowadays, it has the capacity to handle much larger problems. Nevertheless, the graph colouring problem only deals with one hard constraint, that is to produce a clash-free timetable, and ignores other secondary or soft

constraints which are essential in practical examination timetabling situations. As a result, several graph colouring heuristics were developed to overcome these obstacles.

A heuristic method is a 'rule of thumb' selected on the basis that it would help in solving complex problems in which it will contribute to the search for a solution in an acceptable amount of time. It consists of simple procedures that are meant to provide good, but not necessarily optimal solutions.

Reeves and Beasley [1993] define a heuristic as

> *A technique which seeks good (i.e. near optimal) solutions at a reasonable computational cost without being able to guarantee either feasibility or optimality, or even in many cases to state how close to optimality a particular feasible solution is.*

2.2.1 Graph Colouring Heuristic Methods

One of the earliest examples of examination timetabling incorporating a graph colouring heuristic is presented by Broder [1964]. The algorithm used to minimise the number of student clashes is based on a "largest degree first" method. The exams are initially sorted in decreasing order of degrees of exams and the scheduling of exams proceeds by selecting an exam from the top of the list and assigning it to the session that creates the fewest number of clashes. Broder used a Monte Carlo approach as a random selection to choose the sessions to which the exams are scheduled in case of ties. The process is repeated several times to find the best results. This method can also be adapted to minimise the number of times a student has 2 exams in the same day, or any other compact sequence of exams.

Cole [1964] introduced an algorithm for producing an examination timetable at Leicester University. His objective was to prepare a timetable such that no student has

to sit for more than one exam in any session subject to some soft constraints. The soft constraints introduced are that certain sets of exams need to be scheduled consecutively, precedence ordering for some exams, space constraints on room sizes and certain examinations to be scheduled only in the morning. Exams which need to be consecutive are scheduled first and then the algorithm uses the "largest degree first: fill from top" rule to schedule the rest of the exams. The rule starts by sorting the vertices by decreasing order of degree, and then the list is scanned to find as many exams as possible to be placed in the lowest numbered non-conflicting session. The process of filling the next lowest numbered session with non-conflicting exams is repeated until all exams have been assigned.

Another timetabling algorithm which uses the "largest degree first" rule was devised by Wood [1968] and was implemented at the University of Manchester. His objectives were to assign exams to rooms without violating their capacities such that student clashes are avoided at all time and the number of students taking more than one exam on the same day is minimised. The algorithm firstly sorts the exams according to the size of room required. Within each size group he used the "largest degree first" rule for assigning the exams to a feasible session where the session has either firstly no adjacent conflict, or secondly no conflict on the same day or finally the minimum number of students with another exam on the same day. In case of ties of feasible sessions, he computes the total number of unscheduled exams that clash with the current exam and the session with the minimum number of clashes is selected. The room with the least acceptable number of places available is then chosen.

Mehta [1981] used the "largest saturation degree first recursive" algorithm to schedule student exams at the Cedar Crest College, Pennsylvania. The algorithm starts by sorting unscheduled exams in decreasing order of degrees. The exam with maximal degree is assigned to the first session, and then the list is scanned to find an exam with a maximal saturation degree. The saturation degree of an exam is defined as the number of different sessions with which it is in conflict. In the case of a tie, any exam with maximal degree is chosen. This chosen exam is then assigned to the lowest numbered non-conflicting

session possible. His objectives are to schedule exams within a limited time period, to minimise the number of conflicts and to minimise the number of occurrences of 3 examinations in a row subject to special requirements for some exams.

Balakrishnan [1991] produced a procedure based on the "largest degree first recursive" algorithm to solve the examination timetabling problem at the Freeman School of Business at Tulane University. This method is similar to the rule of 'largest degree first' except that each exam which has been assigned is removed from the list, the degrees of the remaining unscheduled exams are recalculated and the list is then resorted. His objectives were to avoid having students taking more than one exam in the same session, to minimise the number of students having exams in successive sessions, and also having more than one exam within a day, with a room capacity constraint. Other constraints include certain exams must be scheduled during a defined session, certain pairs of exams must be scheduled together and certain exams need to be scheduled early in the examination period.

2.2.2 Combinations Of Graph Colouring Heuristic Methods And Other Methods

Desroches, Laporte and Rousseau [1978] presented an examination timetabling system called HOREX which was implemented at L'École Polytechnique de Montréal. The system is divided into several stages and at each stage, different methods are used to solve different objectives.

At stage 1, the algorithm based on a "smallest degree last recursive with interchange" rule is used to produce an initial clash-free solution. The rule "smallest degree last recursive" is similar to the "largest degree first recursive" except that this time the exam with the lowest degree is placed at the end of the list and the scheduling process starts from the top of the list as before. The interchange rule applies when the exam to be scheduled next (exam i) is in conflict with all of the available sessions. The rule first finds a session, s_k where there is only one exam, exam j which is in conflict with exam

i. If possible, reassign exam *j* to another non-conflicting session, say s_l and schedule exam *i* to session s_k. If this is not possible, then a bichromatic interchange is carried out. A set of exams in any session which do not conflict with exam *i* and also with the rest of the exams in session s_k will be swapped with exam *j* if exam *j* does not conflict with any exam in that session. If this is not possible, then a new session is created and exam *i* will be assigned to it.

The second stage of HOREX involves combining the sets of exams found in Stage 1 in pairs corresponding to morning and afternoon sessions using the minimum matching algorithm. They then use a branch and bound integer linear programming method to try to minimise the number of students taking two examinations in the same day, i.e. in both morning and afternoon sessions. Further improvements to the number of students taking two examinations in the same day are done by making simple moves of one examination at a time at stage 3.

In stage 4, the objective is to maximise the number of examinations held during the morning sessions. This corresponds to a knapsack problem and again branch and bound integer linear programming is used. In Stage 5, the days are ordered using a travelling salesman heuristic to minimise the number of occasions in which a student must take examinations on consecutive days. Finally, in the last stage, the minimum tour is rotated through possible starting days to determine the minimum number of successions.

White and Chan [1979] produced a system similar to HOREX for use at the University of Ottawa. The initial assignment of exams to a pre-determined number of sessions uses the "weighted largest degree first: fill from top" rule in which the degrees are firstly multiplied by the total number of students taking each exam. This will result in scheduling larger exams earlier in the examination period. For exams which cannot be assigned to any of the available sessions, an interchange procedure is carried out and if this cannot be done, they are left unscheduled.

The sessions are then ordered using a travelling salesman heuristic in order to minimise the number of students taking two exams in consecutive sessions (morning/afternoon). To reduce this second order conflict further, they try to move each exam to an alternate feasible session which has a smaller number of conflicting exams. They also try to move every pair of exams simultaneously to improve on the number of students having consecutive exams.

A different approach to the problem was described by Laporte and Desroches [1984]. Their computerised system, HORHEC, employs an overall cost function to space exams as evenly as possible and to eliminate conflicts. The constraints for the system are that room capacities are never exceeded, some examinations must be held on specified days and that other examinations may not take place in given sessions, or in certain rooms.

The system incorporates a heuristic method where the procedure is divided into four stages: (i) construction of an initial feasible solution, (ii) improvement of the initial schedule, (iii) room allocation and (iv) output. In the first stage, a step by step algorithm produces a conflict-free schedule of E examinations, each of which may only be held in one of R types of rooms, in S sessions. In each iteration, an exam i is taken from the waiting list and all sets of the sessions and rooms (s,r) to which exam i can be assigned are identified, i.e. they would not create any conflict and do not exceed room capacities. If no such sublist can be found, backtracking procedures are carried out, i.e. when an examination to be scheduled is in conflict with every session, some assignments already made will be undone in order to schedule that examination. Any examination removed from the schedule through this process is put back into the waiting list.

The examination is then assigned to the session s which creates the smallest increase in the cost function F where

F = AVERSION COSTS + PROXIMITY COSTS.

The Aversion Cost, p_{ij} is the aversion of exam i to session j which can be specified by the user; the larger the aversion cost, the less desirable it is to assign the examination to that session. The Proximity Cost, w_s is the penalty cost for scheduling two exams s sessions apart; the longer the gap s, the lower is the penalty which is an attempt to spread out the examinations as evenly as possible. This iteration is repeated until the waiting list is empty. If no feasible schedule can be found, a partial schedule will be printed instead.

Four strategies for the initial ordering of examinations are examined; (i) decreasing order of the number of conflicting examinations, (ii) decreasing order of the total number of student-conflicts for each examination, (iii) decreasing order of the number of students entered in the examination, and (iv) random order. The results showed however that the initial ordering strategy has no significant impact on the cost function.

The second stage involves the improvement of the initial schedule. The algorithm inspects all sublists repeatedly, each time identifying the examination which would give the largest decrease in the cost function if moved to a different sublist without creating any infeasibility. This process terminates when no move can further decrease the cost function.

Room allocation is carried out in the third stage. After the optimal schedule has been found, each examination will be assigned to a room. The rules for assigning the examination are as follows:

Rule 1 : If the number of students in the largest examination i is less than the size of the largest room r, then assign exam i to room r. The unused portion of the room is considered as a 'new' room and it is moved to its appropriate position in the list of rooms.

Rule 2 : Otherwise, assign the maximum number of students who can be accommodated in the largest room r. The rest of the students (if any) will

create a 'new' examination which is moved to its appropriate position in the list of examinations.

After all examinations have been assigned to rooms, the students taking examinations which have been split between several rooms will be reallocated between the rooms to roughly equalise the numbers of students per room, taking into account the various room sizes. Finally, the program provides the actual schedule of all examinations in each room, various tables on daily room utilisation, the set of lists to which each examination could be moved without creating any infeasibility, and the set of examinations which could be moved without creating any infeasibility.

Carter, Laporte and Chinneck [1994] then produced EXAMINE, an improved version of HORHEC, which has been implemented at the University of Toronto engineering faculty and at Carleton University. It provides a choice of feasible schedules and solutions in which exams are well spread out for most students and handles requirements regarding the proximity of a student's exams and room and time availability. The objectives of EXAMINE are to obtain a conflict-free schedule and to avoid second order conflict, that is, students having two examinations in two consecutive sessions, or two examinations separated by one, two, three or four sessions. The constraints of EXAMINE are:-

1. examinations must be scheduled within a specified examination period
2. rooms must be available
3. certain sessions may be prohibited or undesirable for some examinations
4. for every session, the maximum number of students and examinations may be specified
5. certain examinations require special facilities, such as laboratories or equipment

A heuristic procedure is used by EXAMINE. The algorithm assigns examinations to sessions in order to optimise the objective function while satisfying all the constraints. EXAMINE contained two important features added since HORHEC:

(i) In the initial assignment, the program assigns the most conflicting examinations first. It constructs an incompatibility graph first and determines a maximum clique, the largest set of mutually conflicting examinations. These examinations are then assigned to different sessions. For the remaining examinations, it considers examinations in increasing order of the number of available sessions remaining.

(ii) The maximum number of examinations per student within a given time frame can be calculated. This is to minimise the number of occurrences of a student writing x or more examinations in y consecutive sessions. EXAMINE evaluates the feasibility and the cost of assigning a particular exam i to a specific session s. Therefore, EXAMINE computes the set of students in common among the exams in any group. When an examination is to be assigned to some session s, the precise number of students who will be affected can be identified and the associated cost calculated.

2.3 Local Neighbourhood Search Methods

Many heuristics are problem-specific and can only be applied to a particular problem. However, there is increasing interest in developing techniques that can be generally applied to many different problems. These techniques include heuristics for local neighbourhood search. With this method, the process starts with finding a sub-optimal solution to a problem and searches a defined neighbourhood of this solution for a better one. If a better solution is found, it will be taken as the new starting solution and the iteration continues to find a further improved solution. The process stops when no more improvement to the current solution can be found.

This is known as a descent strategy and the main weakness of this procedure is the likelihood of it being trapped in a local minimum, and the global minimum might not be reached. Alternative procedures such as Simulated Annealing and Tabu Search are

explicitly designed to avoid such problems and they are commonly used in solving timetabling problems.

2.3.1 Simulated Annealing

It starts with an initial solution, perhaps chosen at random. A neighbour of this solution is generated randomly by some suitable mechanism and the change in cost is calculated. If a reduction in cost is found, the current solution is replaced by the generated neighbour. Otherwise, a neighbourhood solution which causes an increase in cost is accepted with a 'small' probability which is normally set to $\exp(-\delta/T)$ where δ is the change in cost and T is a control parameter which is analogous to temperature in physical annealing. The process starts with a high value of T and proceeds by attempting a certain number of neighbourhood moves at each temperature, while the temperature is gradually decreased. The search stops when a pre-set stopping criterion is reached.

An approach to solving an examination timetabling problem based on Simulated Annealing was employed by Johnson [1990] to schedule examinations at the University of the South Pacific (USP). The examination timetabling constraints that must be satisfied are:-

1. The timetable must avoid all conflicts, so that no student has more than one examination at the same time.
2. All examinations should be completed in a specified examination period, in this particular case within two weeks (20 sessions).
3. It must be possible to accommodate all exams in the various rooms available.
4. Those examinations with larger numbers of students should come earlier in the examination period to allow sufficient time for marking.
5. The examinations for any student should be spread out as much as possible throughout the examination period.

The procedure consists of two main stages. The first stage is to find an initial feasible solution by assigning the exams in a predetermined order to the available examination sessions. The exams are ordered according to a 'difficulty' factor, depending on the size of the exam and the extent to which that exam clashes with other exams. Having determined the order in which the exams are to be allocated, each exam is assigned in turn to a particular room and session. For each exam, those sessions which can accommodate that exam are identified, these being the ones where there is sufficient capacity remaining and where those examinations that have already been assigned do not clash with the current one. The session chosen is the one which leads to the smallest increase in the number of same-day examinations, with the first available session being used in the event of a tie. The room chosen is the one with the least number of available seats, subject to its being able to accommodate the exam in question. Each exam is assigned to a single room wherever possible, but if necessary an exam may be split between two or more rooms.

The second stage attempts to improve the initial solution. The sessions are re-ordered in order to reduce the number of same-day exams and to ensure that the larger examinations come early on. The process of re-ordering is carried out using Simulated Annealing. In this case, a neighbouring solution is generated by interchanging a randomly chosen pair of sessions. If this change leads to fewer same-day examinations, then the change is accepted. If the change does not lead to an improvement, it may be accepted provided that the required probability criterion is met. Finally, the days are arranged so that the larger examinations are scheduled first.

TISSUE (The Integrated System for Scheduling University Exams) was developed by Thomson and Dowsland [1995] at Swansea University. The system is a phased method based on Simulated Annealing. The three-phased method allows some improvements to the solution obtained in earlier phases to be made in later phases. The first phase is designed to produce a feasible solution satisfying all the binding constraints:

1. No student should be required to sit more than one exam at any one time.
2. All exams to be scheduled into 24 sessions.
3. No more than 1200 students to sit an exam at any one time.
4. Certain pairs of exams to be scheduled together or at different times.

The second stage involves searching for a solution which minimises the secondary non-binding objective, i.e. the number of instances of a student having exams on consecutive days, while maintaining feasibility. The third phase is designed to ensure that large exams are scheduled as early as possible.

2.3.2 Tabu Search

In this case, a set of neighbourhood solutions is generated by some method and a move is made to the best solution in that neighbourhood. In order to prevent cycling, it is not permitted to go back to a solution which has occurred in the last several moves. This list of forbidden solutions is known as the tabu list and any move going back to one of those solutions is considered a tabu move. The tabu status of a move can be cancelled if the move produces a considerable improvement to the solution. The process stops when a pre-set stopping condition is reached or a maximum number of iterations has been completed.

An application of Tabu Search is reported in Hertz [1991]. The problem is to timetable both written and oral exams in a limited time period for the Swiss Federal Institute of Technology in Zurich. The written exams are scheduled first before the oral exams using an algorithm TATI (tabu search for timetabling). The objective function of TATI is to minimise the number of common students taking two exams at the same time. At each step of TATI, the search is directed towards exam sessions which create at least one conflict in the current timetable, instead of generating a random sample of neighbourhood solutions of the current timetable.

It is also desirable to produce a timetable such that the first oral exam for any student is scheduled after the last written exam and there is at least one day free between the oral and the written exams. This problem is then solved using an algorithm TAG (tabu search for grouping) incorporating two penalties, one for each case. The penalty $\max\{0,t(z)-s+1\}$ is introduced for each oral exam involving student z and scheduled at session s, where $t(z)$ is the session at which student z had his/her last written exam. To satisfy the second requirement, each pair (x,y) of exams is penalised when x and y are scheduled on the same day or on consecutive days.

2.3.3 Genetic Algorithm

A Genetic algorithm (GA) may be described as a mechanism that imitates the genetic evolution of a species. It is a function of combining parents' chromosomes to produce new children that hopefully retain the good characteristics of their parents. GAs deal with 'populations' of solutions rather than with a single solution. Three main operators used to generate and explore the neighbourhood of a population and select a new generation are 'selection' (copies an individual from one generation to the next), 'crossover' (an exchange of sections of the parents' chromosomes) and 'mutation' (a random modification of the chromosome).

In the context of examination timetabling problem, the simplest way to represent a timetable is by referring the chromosome as a vector of length n (the number of exams) where the i-th entry of the vector states which session the exam (gene) is scheduled to. Two different parents (current solutions) can be combined to produce two children (new solutions) by randomly selecting a crossover point X to split the two chromosomes. The offspring produced consists of the pre-X section from one parent followed by the post-X section of the other. The new solutions are evaluated in a 'fitness' function corresponding to the value of objective function for that solution.

Corne, Fang and Mellish [1993] used GA on the exam timetable for the I.A. Department at the University of Edinburgh. They use a population of size 50 where 25 pairs are randomly selected with a bias towards the fittest solutions at each evaluation. The evaluation function included conflicting exams, more than two exams in a day, two consecutive exams and two exams just before and after lunch on the same day. These 50 parents produce the next generation of 50 children, and the procedure was repeated for 300 generations. The whole process was repeated for ten times and the best schedule was selected from all runs.

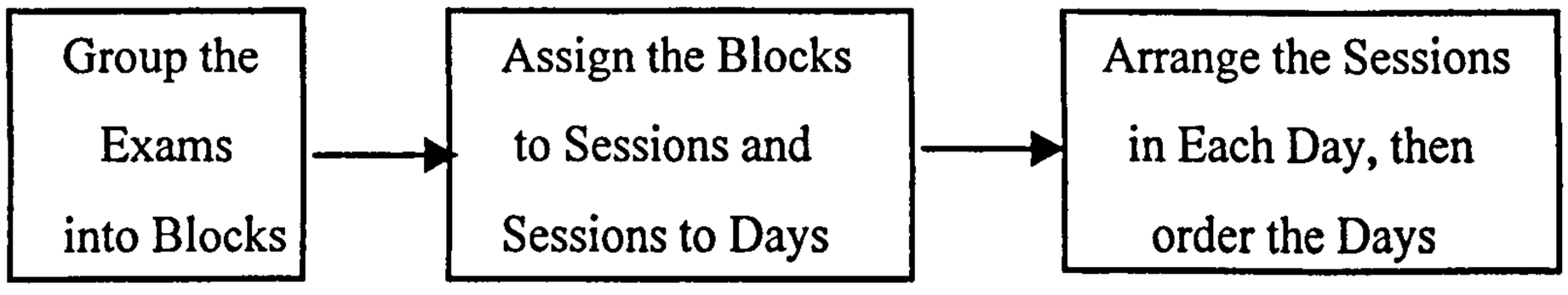
GUNES (Genetic UNiversity Examination Scheduler) has been developed based on a genetic algorithm by Ergül [1996] and has been implemented at the Middle East Technical University (METU). The fitness function involves penalty points for every pair of exams, proportional to the number of common students taking those exams if they are scheduled to the same session, scheduled to consecutive sessions in the same day, scheduled to the last session in one day and the other to the first session of the next day and scheduled within 24 hours. It also gives a higher penalty point to the spread between 'must' (required) exams. In order to discriminate 'must' exams from electives, exams with high conflict density are considered 'elective' exams and those with low conflict density as 'must' exams. GUNES employed 2-point crossover with a rate of 0.75, mutation operators at a rate of 1.00 and 1 chromosome as elite throughout the experimentations. The results, when compared to the manual schedule, yield fewer numbers of student conflicts, both first order and also of higher orders.

2.4 Other Methods

2.4.1 Cluster Approach

Cluster approach that split the problem into several parts and solve each in turn, have been introduced by Arani and Lotfi [1989]. The authors proposed a three-phase

examination scheduling process at State University of New York at Buffalo (SUNYAB). The figure below illustrates the phases of the scheduling process.



The problem in Phase I is to assign *N* examinations to *NS* exam blocks so as to minimise the number of students taking more than one examination in the same exam block. This is a Quadratic Assignment Problem (QAP) without column constraints. Since it was difficult to solve the QAP, an efficient heuristic solution was proposed. In obtaining the initial solution, exams are first sorted in descending order of weighted degree. The weighted degree of an exam *i* is the product of the degree of exam (vertex) *i* and the total number of students taking exam *i* and all other exams in conflict with exam *i*. This will result in assigning 'difficult' exams in earlier sessions.

The procedure starts by choosing the first exam from the top of the ordered list and assigns it to the first unfilled block. It then chooses the next exam from the list and checks to see if it is in conflict with exams already assigned to the first block. If it is not in conflict with any exams in the block, this exam is assigned to that block. Otherwise, the exam is assigned to the next unfilled block. The process is repeated until there is no free block available. If there is an exam which is in conflict with the exams in all of the exam blocks, that exam is assigned to the block which gives the minimum number of students in conflict between the exams. The process stops when all the exams are scheduled. The result obtained is then used as a starting solution for Phase II.

In Phase II, exam blocks are assigned to exam days. The *NS* exam blocks will be assigned to *ND* exam days with *NSD* exam blocks per day so that the number of students with two or more exams per exam day is minimised. The authors presented a

new set-covering formulation to solve the problem and then proposed a Lagrangian Relaxation-based branch and bound solution method to obtain exact solutions for the integer problem. They suggest that the performance of the branch and bound technique can be improved significantly by providing a good initial upper bound. Hence, they developed a heuristic procedure in order to obtain the initial solution.

The algorithm begins by arbitrarily assigning (*NSD-1*) exam blocks to the first (*NSD-1*) sessions of every day. The last session of each of ND days will be empty so *ND* exam blocks are then needed to be assigned to those sessions such that the total additional conflict is minimised. The additional conflict is associated with the additional number of students with two or more exams per day when assigning each of the *ND* exam blocks to the last session of each of *ND* days. Once the optimal solution is found, these blocks are fixed to the last session of each day.

The process of reducing the total conflict within each day is continued by backtracking to the second last sessions and rearranging the exam blocks which have been already arbitrarily assigned to them. The backtracking process is repeated until the assignment problem for the first session in each of *ND* days is solved. Then a forward move is attempted by solving an assignment problem for the second session and so forth. The backward and forward processes of rearranging *NSD* exam blocks within a particular session is repeated until no further reduction is possible. At this point, the heuristic terminates and the last solution is used as the starting point for the branch and bound method.

Phase III involves ordering the exam days in such a way that the number of students having exams on the last session of one day and the first session of the next day is minimised. The problem was formulated as a travelling salesman. Each exam day corresponds to a city and the weight c_{ij} for arc (i,j) is the number of students taking exams in the last session of day i and the first session of day j. The optimal travelling salesman tour then represents the optimal sequence of exam days.

Lotfi and Cerveny [1991] developed further the method of Arani and Lotfi to introduce room allocation. The concept of a cluster approach was still utilised, but differing in two important aspects. Firstly, a fourth phase was added where exams are assigned to rooms so as to minimise the number of split exams. Secondly, the formulations of the first and second phases were somewhat different and more suited to a practical application.

The objective in Phase I is to assign all exams to blocks so as to minimise the number of students with simultaneous exams. The problem is formulated as a QAP with two sets of constraints. The first set of constraints places a limit on the number of exams that could be assigned to an exam block and the second set of constraints prevents two or more exams administrated by the same instructor from being scheduled simultaneously. The authors presented a revised version of Arani and Lotfi's method to solve the Phase I problem.

The exams are first sorted in descending order of weighted degree. The procedure schedules the more difficult exams first. The exams are taken one by one from the ordered list and assigned to unfilled blocks with no conflicting exams or instructors. If no feasible block exists, the exam will be assigned to the block with no instructor conflicts and which has the minimum number of students in conflict. The process is repeated until all the exams are assigned to blocks.

In Phase II, exam blocks are then assigned to exam days while minimising the number of students with two or more exams per day. The problem of assigning the exam blocks to exam days was formulated as a QAP with column constraints. A heuristic solution method was then suggested to solve the Phase II problem.

The algorithm begins by constructing an initial solution by assigning the exam blocks to exam days sequentially. That is, the first block is assigned to the first day, the second block to the second day and so on. Then the current solution is examined for possible improvements through a pairwise exchange of exam blocks. If no further improvement

is possible, the algorithm stops and the current solution is used as an input to the third phase.

Arranging the exam days and exam blocks within days are carried out in Phase III. The exam days and exam blocks within days are arranged so as to minimise the number of students with consecutive exams. The problem of finding an optimal arrangement of exam blocks within a given day can be formulated as a TSP. A heuristic commonly referred to as 'next nearest city' was used to solve the TSP problem. The next-nearest-city solution approach resulted in an arrangement of exam days that approximately minimised the number of students with consecutive exams.

Exams are then assigned to exam rooms so as to maximise the space utilisation in the fourth phase. The problem of assigning the exams scheduled in any given session to available exam rooms so as to minimise the number of split exams can be formulated as a non-linear integer program. A heuristic procedure was developed to solve this problem.

The algorithm begins by sorting the exam rooms in decreasing order of size and all the rooms in descending order of the seating capacity. It then tries to assign each exam to the first available exam room. If the number of participants in the exam is less than or equal to half the capacity of that exam room, it assigns the exam to that room and removes the exam and the exam room from the list. Otherwise, it assigns sufficient students from the exam to fill the room and searches for another room to assign the remainder. This is done by scanning the list of available rooms, from bottom to top, searching for a room with twice the seating capacity of the remaining students, which is located on the same campus. Where the remaining students do not fit into the second remaining largest room, this room is used to contain as many students as possible and the list is searched for a third room.

2.4.2 Constraint Based Approach

The problem in the Constraint Based Approaches is represented by a set of variables to which values must be assigned in order to satisfy a number of constraints and they do not explicitly minimise or maximise an objective function. The search strategies that are used to guide the search are often specific to the application.

A general constraint satisfaction technique has been used by Nuijten, Kunnen, Aarts and Dignum [1994] to solve the examination timetabling problem at the Eindoven University of Technology, Netherlands. The constraints that must be satisfied are the maximum number of students taking exams in any session, time windows on when the exams can take place, no conflict for any student and the spread of exams in groups corresponding to programs. They used a simple random selection algorithm to assign exams to sessions and if an exam cannot be assigned to any session, the backtracking process is carried out. If this does not work, they will restart the random selection process with a different path. They tested on 275 exams with constraints of about 3000.

Boizumault, Delon and Péridy [1994] used Constraint Logic Programming (CLP) at l'Université Catholique de l'Ouest in Angers, France. The constraints used are no conflicting exams, preassignment of exams, no consecutive exams, precedence and room assignments. Multiple exams may be assigned to one of the seven rooms available and this is solved as a bin packing problem. The search procedure used is a 'best fit decreasing' strategy found in the bin packing problem. They tested the model on 308 exams which produced a set of 2600 constraints.

2.5 Summary

In early days, resolving conflicts for every student was the main priority of timetabling university examinations, before satisfying any other constraints. As the work on

timetabling progressed, many constraints were introduced to produce a more practical and realistic timetable. Room capacities, time restrictions on the length of the examination period and special requirements for some exams are some of the constraints considered in the problems. Many authors are also interested in spreading the exams as evenly as possible throughout the examination period for every student.

Timetabling problems can be represented using different mathematical formulations such as graph colouring and integer linear programming, but the successful implementation of these approaches has been limited when involving large scale problems. This is due to the size of the problem generated when using mathematical formulation and the inability and inefficiency to handle complex requirements within the problems. Phased methods and heuristic methods have therefore been introduced to tackle such problems. While graph colouring heuristics have been used extensively, local search methods such as Simulated Annealing, Tabu Search and Genetic Algorithm are now gaining popularity.

Phased method decomposes the whole problem into several small problems, which are then solved sequentially. The advantage of using this method is that the size of each problem will be small and therefore can be solved much easier and also more sophisticated optimization techniques can be used. However, there are several drawbacks of this method. Since the size of the problem in each phase is reduced, so is the size of the solution search space, which will then cause a substantial effect in the potential quality of the final solution. In the literature, simple heuristics have been used to solve the decomposed problem even though more sophisticated and beneficial method can be employed. Moreover, the problem is solved in such a way that it does not keep any reference to individual exams or students so it is not possible to solve the problem of, for example, students having three exams in any four sessions. Another disadvantage of phased method is that the solution obtained in the initial phases cannot be undone in later phases and may affect the overall quality of the final outcome.

However, some approaches using sequential meta-heuristics such as Simulated Annealing and Tabu Search have been used successfully in overcoming the drawbacks of the phased method (Thompson and Dowsland [1996], Hertz [1991]). Nevertheless, there is a significant difference in the processing time between the simpler constructive approach of phased method and the SA and TS. It is reported in Thompson and Dowsland [1996] that a feasible solution using SA employing "kempe chain" can be found in about 18.5 hours on a 486 pc, although this can be reduced by using a faster cooling schedule but with a reduction in the final solution quality. Therefore, there seems to be a gap between a fast approach and an effective system that is capable of achieving high quality solutions over a wide range of objectives and constraints.

CHAPTER 3

SURVEY OF EXAMINATION TIMETABLING IN UNIVERSITIES

3.1 Introduction

This chapter will discuss one of two major pieces of work carried out in this research, namely a survey of examination timetabling systems in universities in the U.K and Northern Ireland. It will firstly highlight the aims of the survey, then it will describe the methodology used and how data were collected and finally the results from the survey will be presented. The results from the survey are very important, particularly the common objectives and constraints of the examination timetabling systems since they will be used in the second stage of the research work; to develop a general examination timetabling procedure for universities. Most work that has been done previously on university examination timetabling systems concentrates on an individual institution, where each has different needs and requirements when constructing examination timetables.

A survey of examination timetabling in British universities has been carried out by Burke *et al.* [1996] identified the sort of criteria a general automated timetabling system must meet. They concluded that a good generalised examination timetabling must produce good quality timetables which can take into account a wide variety of complex constraints and be able to adapt itself to special circumstances which may arise. The system should be complete where it handles the whole process of examination timetabling from collecting registration to producing results. The system must be user

friendly so that even those who are not computer experts can use it and also it should be compatible with present system where it can download data from the current system and produce outputs in a form that this system can accept. Real timetabling test data is required to fully test the system since the data will provide the system with realistic and complex problems.

Since the main part of the work is to produce a feasible timetable while taking account of different objectives and constraints, we are only concentrating on the scheduling part of the timetable process. Therefore, our survey is carried out mainly to identify the most commonly used objectives and constraints by universities to be incorporated in the next part of the work.

3.2 Aims Of The Survey

A survey of University Examination Timetabling Systems was conducted in order to discover how different universities schedule their examinations. There are three main aims of the survey:-

i) to examine the extent to which computerised timetabling procedures are used in both term based and semester based universities

ii) to identify the objectives and constraints that are commonly considered by universities when constructing examination timetables

iii) to evaluate the effectiveness of different university examination timetabling systems in achieving the objectives and satisfying the constraints

3.3 Methodology And Data Collection

The term 'survey' refers to the gathering of data or information from a chosen sample or specific population, usually by either self-administered questionnaires, face-to-face interviews, or telephone interviews. For the purpose of this study, a mail questionnaire was chosen because it can cover a wide geographical area at relatively low cost of data collection as well as data processing. On the other hand, the potential disadvantages of mail questionnaires are low return rates, and no direct contact between the respondents and researchers (Oppenheim [1992]).

Some techniques have been suggested to improve response rates. One of them is to keep the questionnaire as simple and short as possible so that the respondents find it easy to answer the questions. Providing the respondents with a self-addressed, stamped return envelope and offering to provide respondents with a summary of the results often acts as an incentive to co-operation. One other important method is to give advance warning of the survey by contacting prospective respondents and inviting them to participate in the survey. This should therefore guarantee their response to the questionnaire and the costs of postage would not be wasted. A reminder letter to those who have not returned the questionnaire after a specific amount of time can also be used to encourage respondents to complete the questionnaire (Oppenheim [1992]).

3.3.1 Design And Construction Of The Questionnaire

The questions are written in both closed form, involving a set of alternative answers from which the respondents must choose, and an open-ended form where the respondents are allowed to answer in their own words. The meanings of words in the questionnaire must be clear to all respondents. Therefore, definitions to some words are provided. In order to avoid ambiguity, the questions are kept short and direct and as many as possible of the questions are posed in a closed form. A copy of the questionnaire can be seen in Appendix 1.

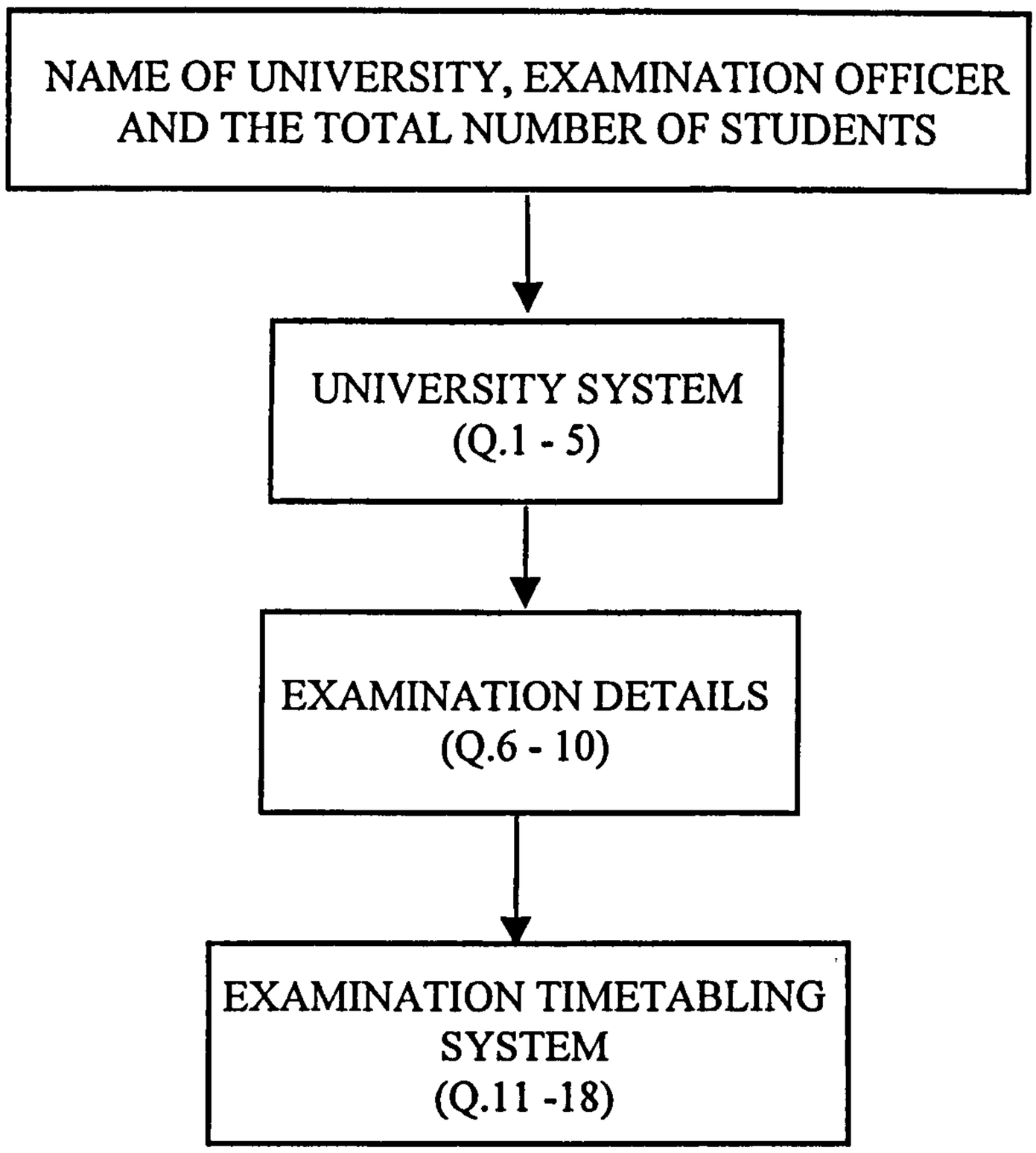


Figure 3.1 : Design of the questionnaire

The design of the questionnaire is shown in Figure 3.1 and is divided into three main parts. The first part deals with the academic structure of the universities being surveyed, either term based or semester based, and the frequency of the major examinations held each year. The second part concerns the details of examinations which were most recently held. The last part examines the systems used by the universities for timetabling the examinations.

3.3.2 Sampling

From a list of 119 universities or university level institutions in the U.K and Northern Ireland, a total of 93 institutions which are likely to have a centralised examination system were identified. Many of the institutions excluded from the survey are small specialised institutions, which are mostly the federations of University of London. The list of the institutions is given in Appendix 2.

3.3.3 Pilot Study

An early version of the questionnaire was reviewed by Mr. J Wilcox, the examination officer in Loughborough University to ensure its validity and clarity. This resulted in a number of important enhancements being incorporated before the questionnaire was sent for a pilot study. The pilot study was tested on four university examination officers; University of Reading, University of Sussex, University of Aberdeen and University of Hull as well as Loughborough University and their comments on the ease of response were noted. Following these, several changes were made particularly in some definitions as they seemed to lead to some misunderstanding.

3.3.4 Data Collection

Before questionnaires were sent to the selected universities in the U.K and Northern Ireland, telephone calls to each of the universities were made in order to find out the person to whom the questionnaires should be addressed. This was usually the person in charge of or responsible for timetabling examinations. Not all of the universities have one specific person in charge, with timetabling being done by a group of people. In this case, the questionnaires were generally addressed to the Examination Officer of the university. A brief introduction to the survey was given in a covering letter and they were then asked whether they would like to participate in the survey. By doing this, it

was hoped that they would answer and return the questionnaires, and hence increase the rate of response.

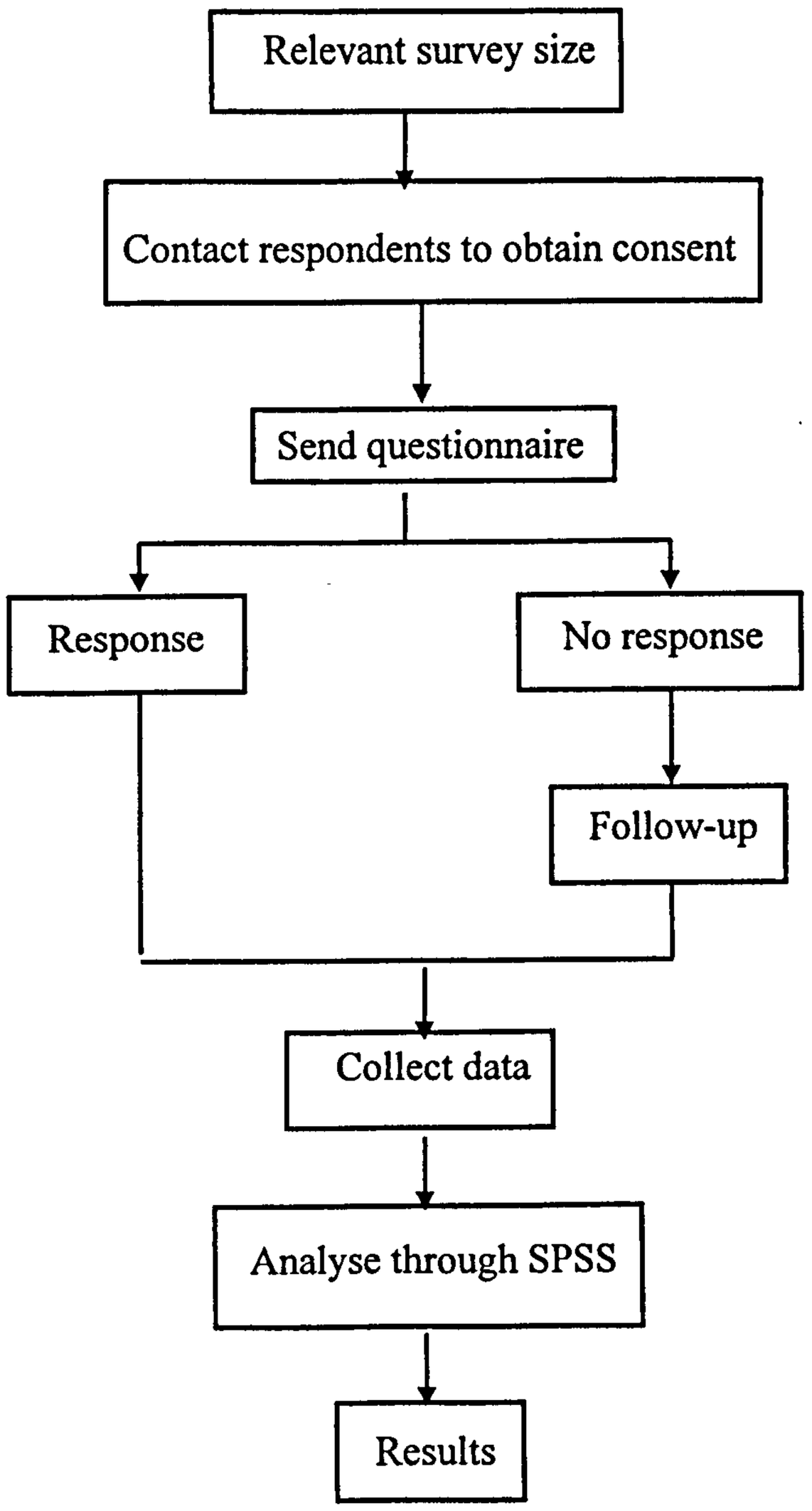


Figure 3.2 : Procedure of data collection

A flow diagram on the procedures of data collection is shown in Figure 3.2. About a month after the first batch of questionnaires was sent out, about 45% of questionnaires

had been returned. As expected, poor response was from the questionnaires which were not specifically addressed to the person in charge of timetabling examinations. To further increase the response rate, follow-up letters containing the questionnaires were sent to those who had not replied. This increased the response rate to 59%.

When it was apparent that no more questionnaires would be returned, the data from the responses was coded and keyed into the computer. The data were then analysed using SPSS for windows, leading to the results presented in the following sections.

3.4 Data Interpretation And Results

A total of 55 questionnaires were returned which gives a response rate of 59%. This would usually be regarded as a very high response rate which shows that there is a strong interest in the problem of examination timetabling in universities. It may be due to the fact that with the increasing number of students entering higher education each year and the introduction of modularisation in universities, the task of timetabling examinations is becoming more difficult.

For analysing some of the data obtained, two types of statistical test on SPSS were used:

- a simple factorial ANOVA
- a 2x2 contingency table

When taking into account the objectives of the examination timetabling problems, examination officers were asked to specify the level of importance of each one of them. They were also asked to specify the level of effectiveness of their examination timetabling systems in achieving the objectives and satisfying the constraints. It is interesting to investigate the effects of the examination timetabling systems and the

structure of universities upon the performances (the level of importance of objectives and the level of effectiveness of examination timetabling systems).

A simple factorial ANOVA is used to test for the presence of a main effect of each of two factors considered separately and also for an interaction between the factors. The two factors involved in this case are:

(i) examination timetabling system, either computerised or manual systems
(ii) university structure, with term based and semester based universities

The factor is said to have a main effect if the performance is not the same at all levels. For example (refer to Table 3.1 below), if the marginal row mean of the computerised system ($\bar{X}_1$) is different from the marginal row mean of the manual system ($\bar{X}_2$), then there would be a main effect of the examination timetabling system. Similarly, there is a main effect of the university structure if the marginal column mean for term based universities ($\bar{X}_3$) differs from the marginal column mean of the semester based universities ($\bar{X}_4$).

		University structure		
		term based	**semester based**	row
Examination timetabling system	**computerised**	$\bar{x}_{11}$	$\bar{x}_{12}$	$\bar{X}_1$
	manual	$\bar{x}_{21}$	$\bar{x}_{22}$	$\bar{X}_2$
	column	$\bar{X}_3$	$\bar{X}_4$	

Table 3.1 : Mean scores of the performance

Two factors are said to interact if the effect of either is not the same at all levels of another. For example, if the performance means of the computerised systems do not show the same pattern as the performance means of the manual systems, the university structure factor has different effects at different levels of the examination timetabling system factor. Therefore there is an interaction between the university structure and the examination timetabling system.

The indication whether there is a main effect of the university structure and the examination timetabling system and also an interaction between the two factors are given by the significance level of F in the ANOVA tables. Small values of the significance level of F indicate that there is a significant main effect or interaction as appropriate.

Examination officers were also asked to identify the constraints which were considered when constructing timetables. The plan is to study for each of the constraints being considered whether the structure of the university (term based or semester based) associates with the examination timetabling system (computerised or manual systems). A 2x2 contingency table is used for determining the existence of an association between the two variables. The significance value of the test determines whether there is an association or not. If it is small enough, then the two variables are said to be associated with each other.

		Examination timetabling system	
		computerised	**manual**
University structure	**term based**	n_{11}	n_{12}
	semester based	n_{21}	n_{22}

Table 3.2 : A 2x2 contingency table

In Table 3.2, n_{11},....,n_{22} denote the number of term based and semester based universities which have either computerised or manual timetabling systems considering each constraint. For example, from the contingency table, if n_{11} » n_{12} and n_{22} » n_{21} (the majority of the term based universities with computerised examination timetabling systems have considered the constraint, whereas the majority of the semester based universities which consider the constraint have manual timetabling systems), then it would appear that there is an association between the university structure and the examination timetabling system when considering that particular constraint.

3.4.1 Survey Respondents

To determine whether the nature of university systems has any effect in examination timetabling, comparisons were made between term based and semester based universities. From Figure 3.3, more than half of the sample have a semesterised structure. About half of these universities are new universities which were previously Polytechnics. A few of the universities using a term based system are however planning to change to a semester based system in the near future.

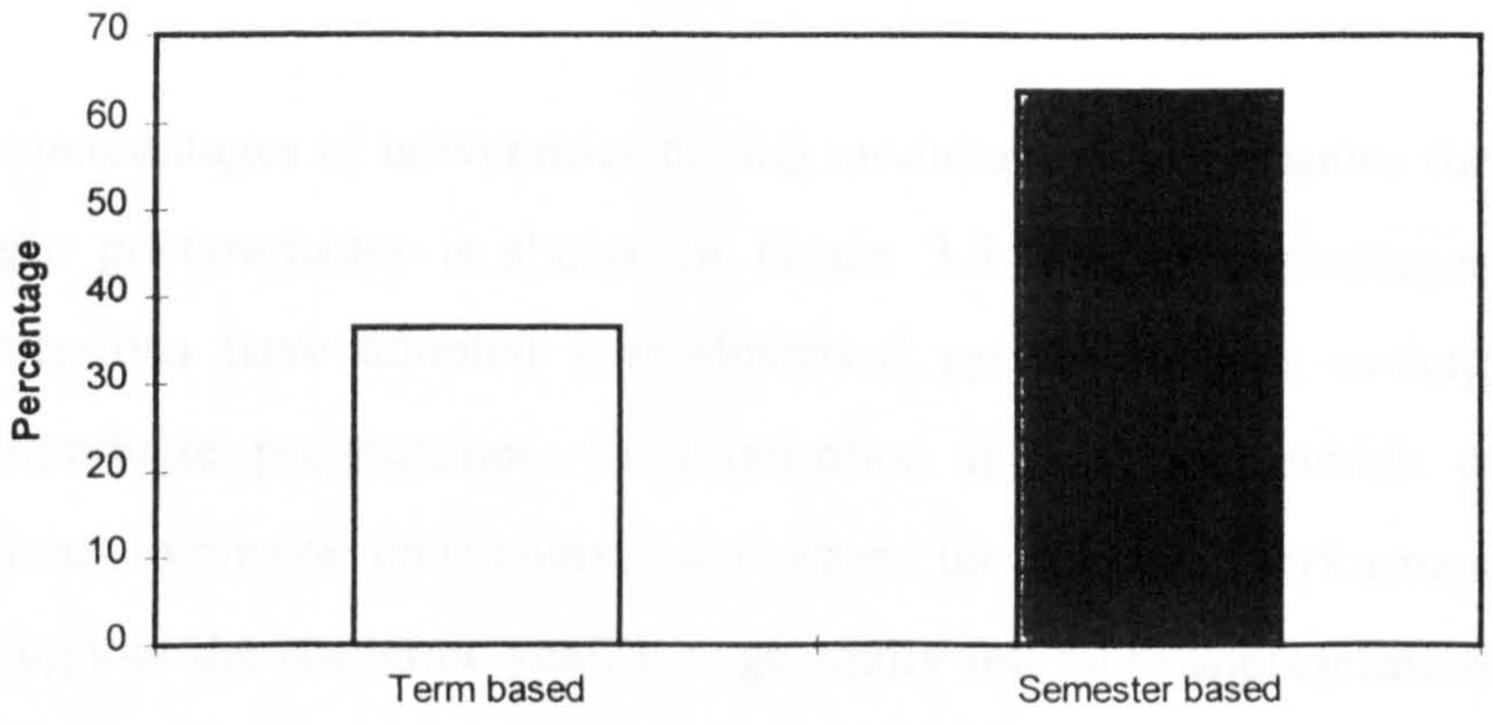


Figure 3.3 : Academic structure of universities

The size of each university is determined by the number of students registered. The average number of students in semester based universities is much larger than in term based universities as shown in Figure 3.4, showing that larger universities tend to have semesterised systems.

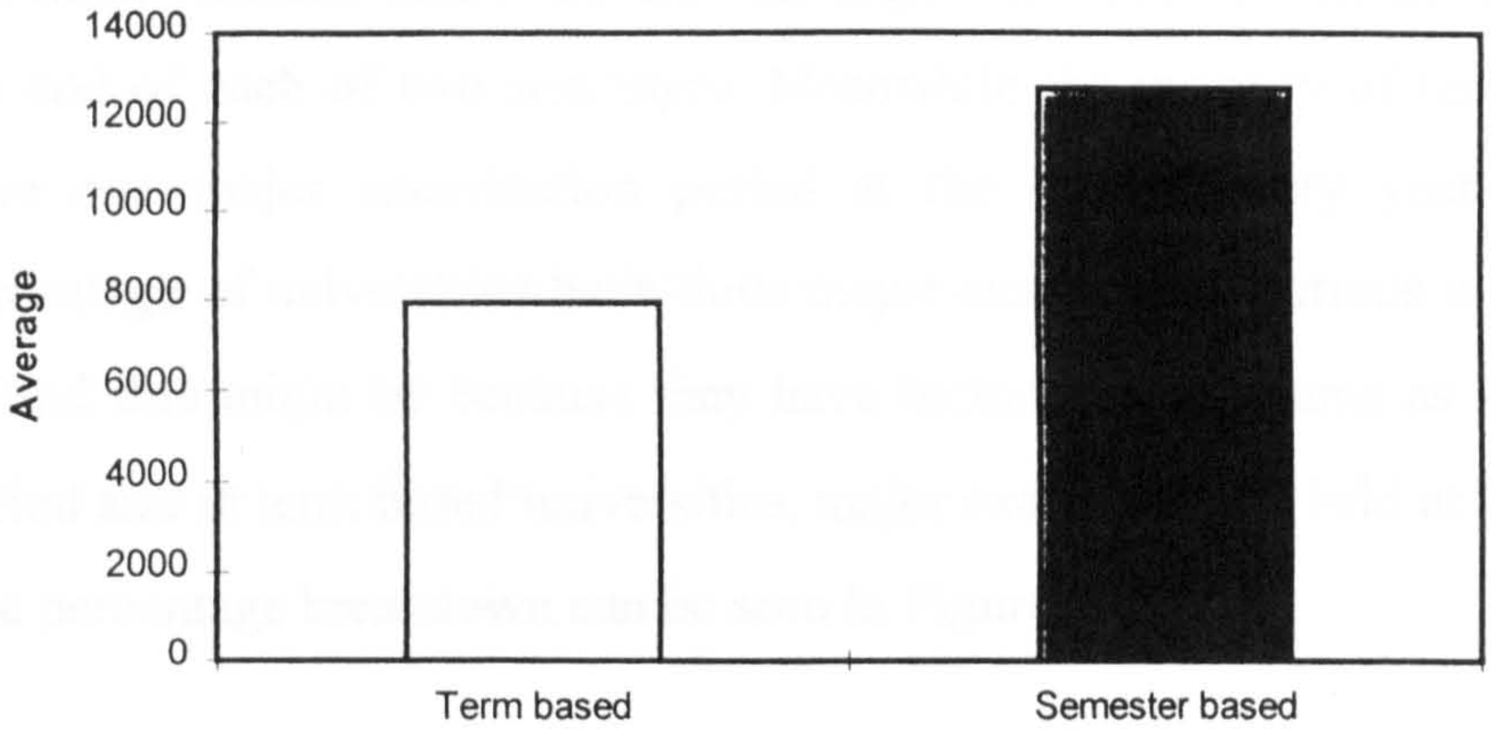


Figure 3.4 : Number of students registered in universities

The percentages of universities having modularised programmes for undergraduates and taught postgraduates is shown in Figure 3.5. Higher percentages of semester based universities have adopted a modularised system in both undergraduate and taught postgraduate programmes. Modularisation is a system which offers flexibility for students to choose their courses and where the students' performances will be assessed throughout the academic year. It is generally felt more appropriate for a modular system to be implemented within a semesterised system where the academic year consists of two semesters, with the teaching and examining of a module usually taking place within one semester.

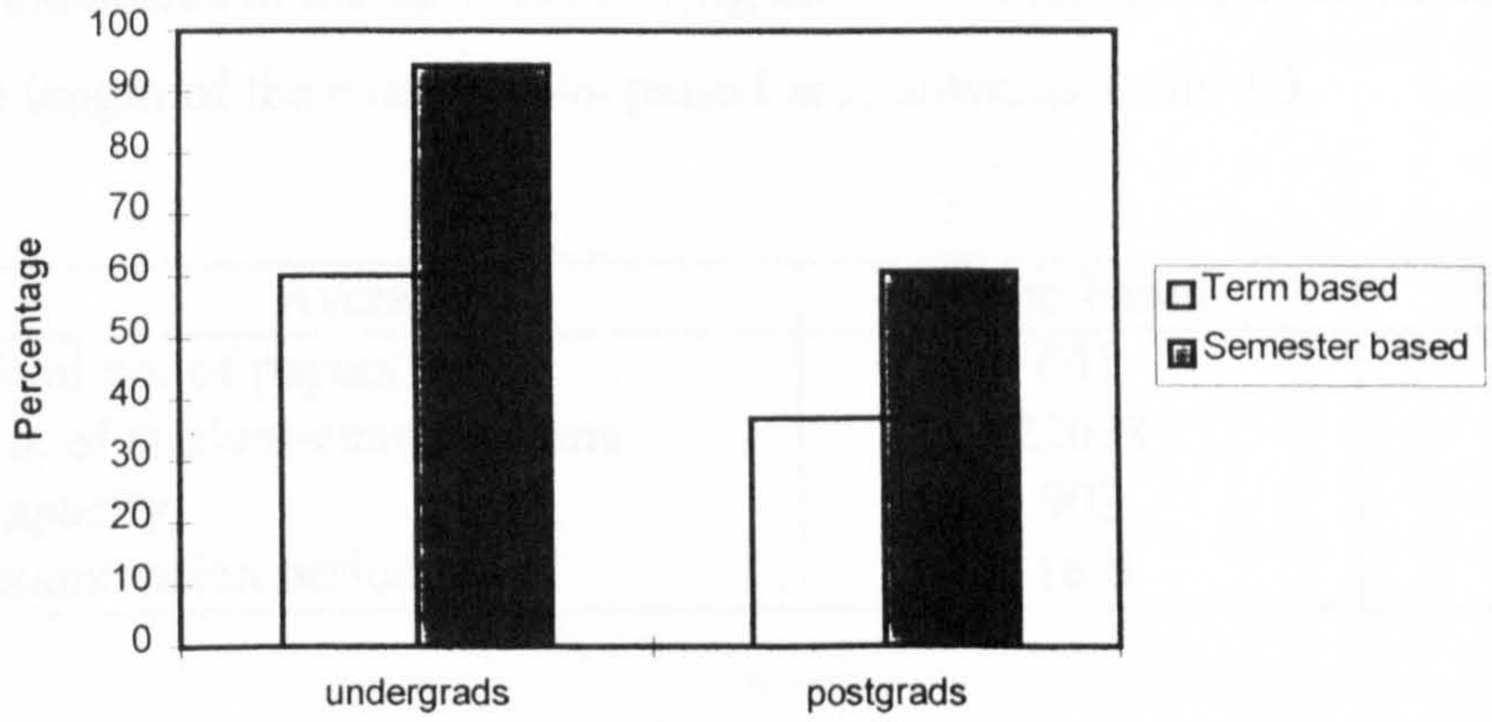


Figure 3.5 : Modularised programmes in universities

For most semester based universities, major examinations are held twice a year, i.e. at the end of each of two semesters. Meanwhile the majority of term based universities have one major examination period at the end of every year. However, a small percentage of universities have three major examination periods in a year. The reasons behind this might be because they have included resit exams as a major examination period and in term based universities, major exams may be held at the end of each term. The percentage breakdown can be seen in Figure 3.6.

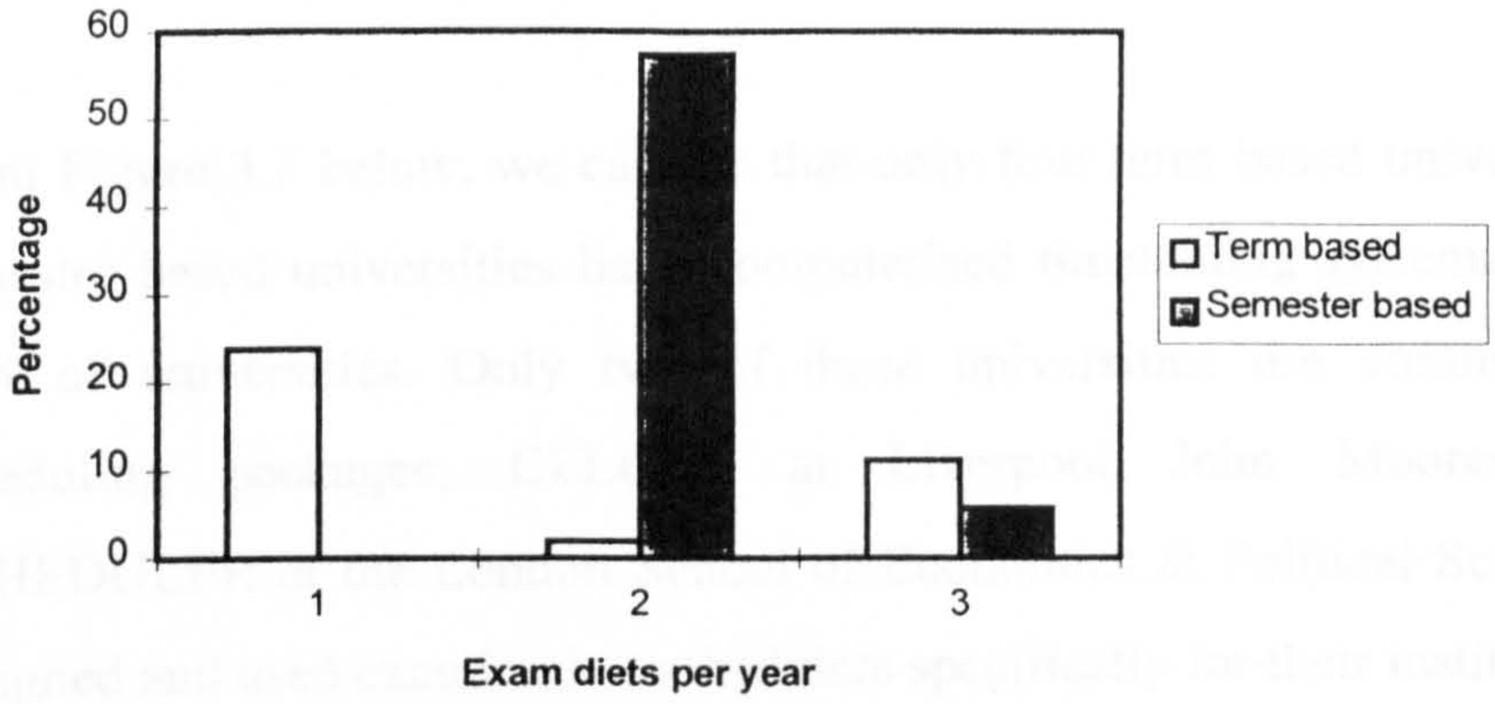


Figure 3.6 : Major exams diets per year in universities

The averages for the total number of examination papers taken by students, the number of student-examinations, the capacity, i.e. the maximum number of students that could be examined at the same time using all of the examination rooms normally available and the length of the examination period are shown in Table 3.3.

Average	Term based	Semester based
Total no. of papers	641	638
No. of student-examinations	22038	23292
Capacity	903	1329
Examination period (day)	16.6	13.9

Table 3.3 : Examination details in universities

Both universities seem to set similar numbers of examination papers. The average number of student-examinations in semester based universities is higher as there are more students in these universities. For the same reason, the average capacity of semester based universities is higher and therefore more students can be examined on any day. The average length of the examination period in semester based universities is shorter because mostly they have two major exam diets per year.

3.4.2 University Examination Timetabling Systems

From Figure 3.7 below, we can see that only four term based universities and thirteen semester based universities have computerised timetabling systems, comprising about 30% of universities. Only two of these universities use commercial examination scheduling packages; CELCAT at Liverpool John Moores University and SCHEDULER at the London School of Economics & Political Sciences. Others have designed and used examination schedulers specifically for their institutions. Many of the universities in the survey appear still to be constructing the timetable manually.

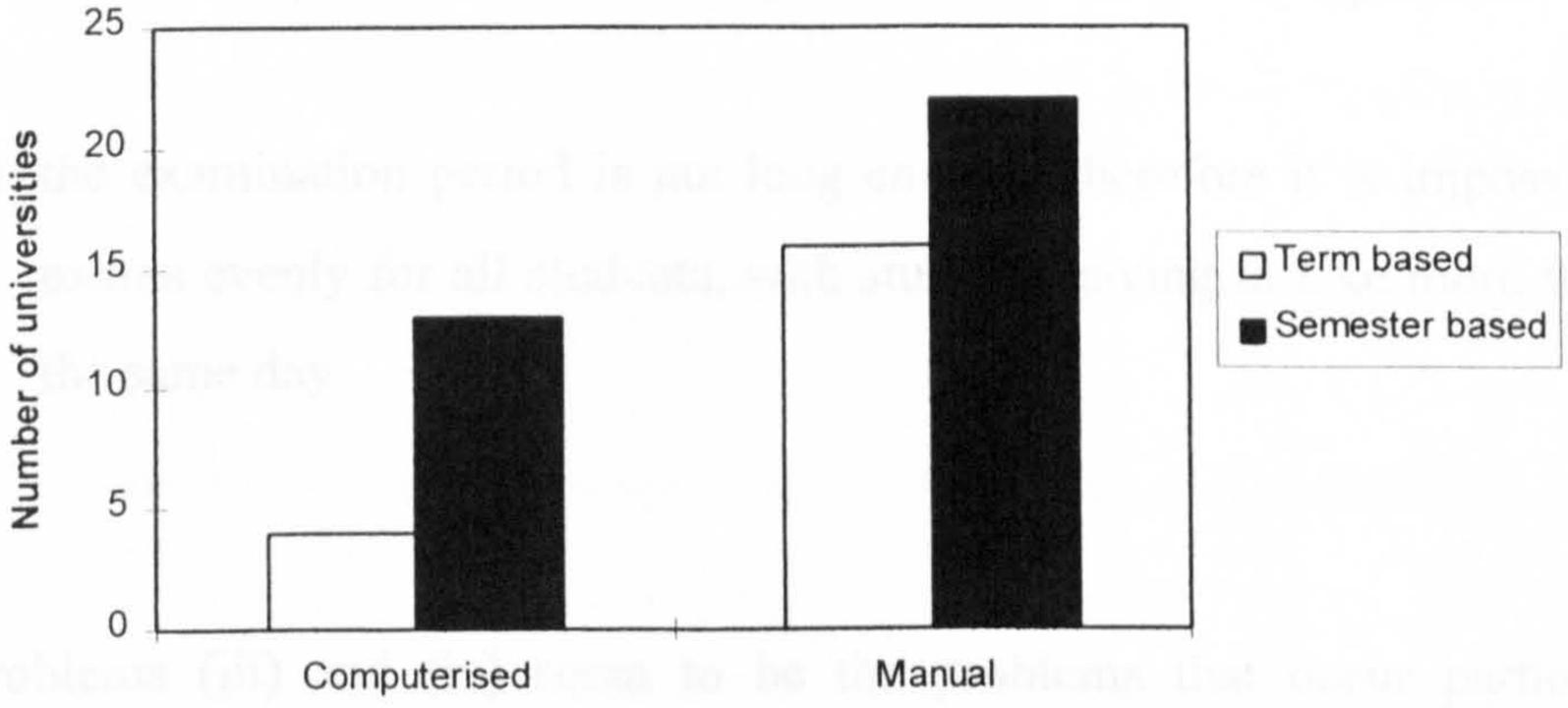


Figure 3.7 : The nature of examination timetabling systems

3.4.3 Problems Encountered When Timetabling Examinations

Almost all universities have encountered some problems when constructing timetables with only 9% of respondents claiming not to have experienced any. The problems mostly cited are the following:

i) there are often not enough rooms available to cope with the different finishing times of exams and for exams with large numbers of students. The students then have to be split between several rooms

ii) inaccuracy in source data or information; either having difficulty obtaining co-operation from departments or from the students themselves

iii) the task of producing the timetable becomes laborious and time consuming when there is a long clash list for many exam sessions and also when errors are detected after the timetable has been completed

iv) it is extremely difficult to avoid clashes for students where they are allowed a wide choice of options, not necessarily restricted to only one department

v) the examination period is not long enough, therefore it is impossible to space out exams evenly for all students, with students having to take more than one exam on the same day

Problems (iii) and (iv) seem to be the problems that occur particularly when the timetable has been done manually.

3.4.4 Objectives Of The Examination Timetabling Problem

To indicate how significant is each of the objectives considered by examination officers when constructing examination timetables, the respondents were asked to specify its level of importance. Various objectives of examination timetabling were proposed in the questionnaire, namely:

Objective 1 : to eliminate conflicts for all students

Objective 2 : to ensure that no student takes more than one examination on the same day

Objective 3 : to minimise the number of exams taken by any student in 2 consecutive days

Objective 4 : to minimise the overall duration of the examination period

Objective 5 : to take account of year of course when scheduling examinations

Objective 6 : to schedule large examinations early in the examination period

Objective 7 : to avoid having exams with different durations in the same room

Objective 8 : to avoid splitting any examination between two or more rooms

The scale of importance ranges from 1 to 5, '1' being unimportant and '5' essential and the results are shown in Figure 3.8 where the boxplot defines fifty percent of those replied. **x** marks the mean value of the level of importance and it presents the significance of each objective. The higher the value of the mean, the more significant the objective.

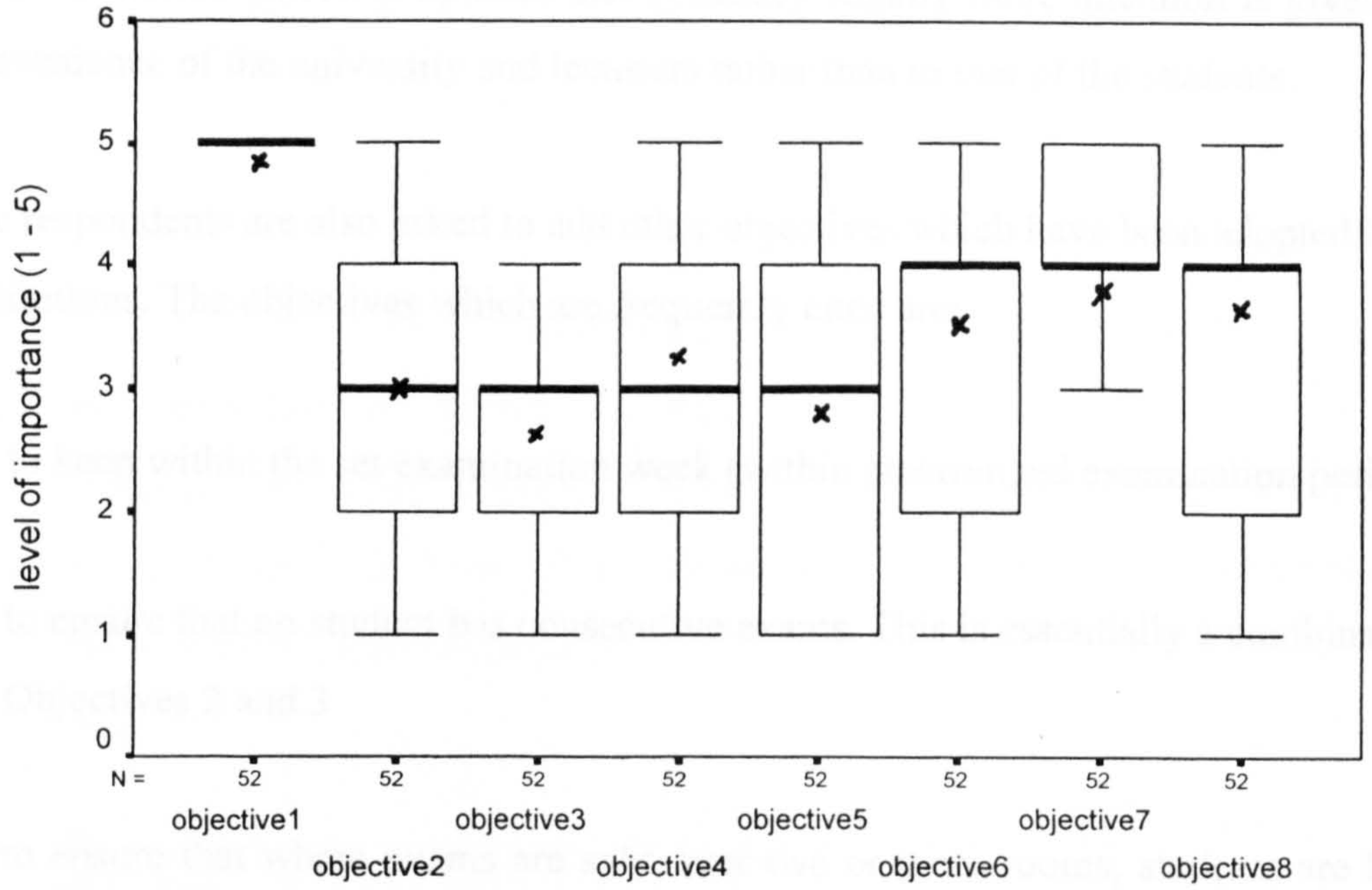


Figure 3.8 : The level of importance of timetabling objectives

It can be seen from Figure 3.8 that Objective 1 has the highest mean level of importance. It is therefore highly significant to produce a clash free timetable such that no students has to take more than one exam at the same time. The second highest scoring objective is to avoid having exams with different durations in the same room (Objective 7). This is to prevent the students who finish early from disrupting the concentration of other students who finish later.

Next is Objective 6, i.e. to schedule large examinations early in the examination period in order to allow enough time for marking, together with Objective 8 which is to avoid splitting any examination between two or more rooms since it can be difficult for the instructor to administer students sitting the same exam in different rooms. It is then followed by Objective 4, which is to minimise the overall duration of the examination period.

The rest of the objectives (Objectives 2, 3 and 5) seem to have noticeably less importance in the process of examination scheduling with means less or equal to 3.0. Nevertheless, there are small percentages of universities that rated Objectives 2, 3 and 5

to be at least desirable. It appears that generally slightly more attention is given to the convenience of the university and lecturers rather than to that of the students.

The respondents are also asked to add other objectives which have been adopted by their institutions. The objectives which are frequently cited are:

- to keep within the set examination week (within prearranged examination period)
- to ensure that no student has consecutive exams. This is essentially a combination of Objectives 2 and 3
- to ensure that where exams are split over two or more rooms, students are located close together

3.4.5 Constraints Of The Examination Timetabling Problem

To determine which constraints frequently arise in examination timetabling, the respondents were asked to identify the constraints they considered when constructing examination timetables. The constraints proposed in the questionnaire are:

Constraint 1 : specified time limit on the overall examination period

Constraint 2 : maximum number of students that may be examined in any session

Constraint 3 : maximum number of exams that may take place in any session

Constraint 4 : certain specified exams require special facilities or equipment

Constraint 5 : certain specified exams must take place during the same session

Constraint 6 : certain specified exams cannot take place during particular sessions

Constraint 7 : certain specified exams must take place during a defined session

The percentages of universities considering the constraints listed when constructing examination timetables are illustrated in Figure 3.9. The results show that there are three most commonly considered constraints, Constraints 1, 2 and 5. These constraints are considered by more than 75% of the respondents. The rest of the constraints are considered to be less common constraints.

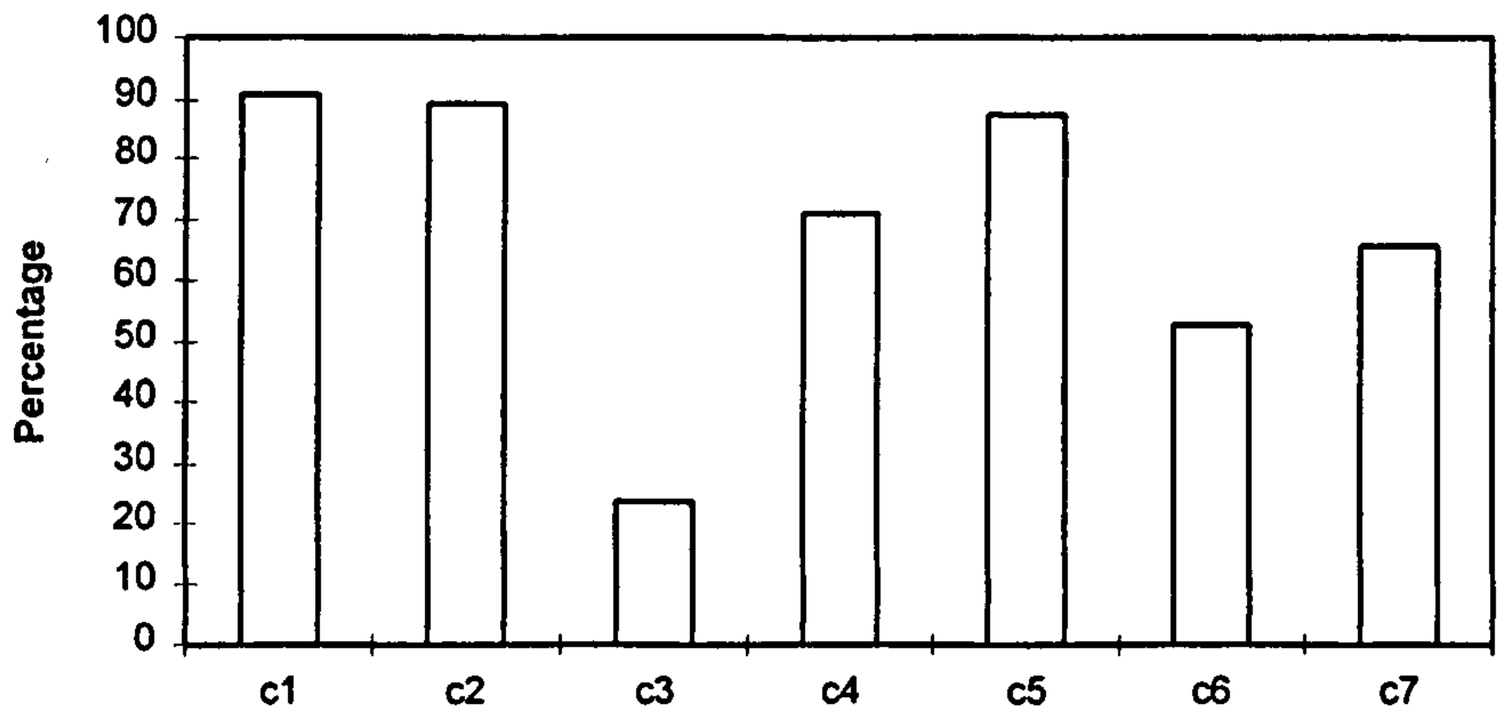


Figure 3.9 : The number of universities considering the constraints

The highest percentage of universities consider having a specified time limit on the overall examination period when constructing the timetables (about 91%). This is an important constraint since the amount of time during which exams can be held is already prearranged by the university, and any increase in the examination period will cause disruption to the whole organisation of the university academic schedule. The next most frequently encountered constraint is Constraint 2, i.e. a maximum number of students that may be examined in any session. Exam rooms are usually the ones which are big enough to hold as many exams as possible. This is to reduce the number of rooms being used as exam rooms so that invigilation costs can be kept as low as possible. About 87% of respondents considered Constraint 5, certain specified exams must take place during

the same session, as a relevant constraint. For example, an exam which is taken by both first year and second year students must be scheduled at the same time.

It is not entirely surprising to see that Constraints 1 and 2 are considered by most respondents since a large number of the studies reported in Chapter 2 have incorporated these two constraints into their examination timetabling problems.

3.4.6 Relationships Between Examination Timetabling System And University Structure When Considering Objectives

The average level of importance for each of the objectives for both term based (T) and semester based (S) with manual or computerised timetabling systems when constructing examination timetables is shown in Figure 3.10. The scale of importance ranges from 1 to 5, '1' being unimportant and '5' essential.

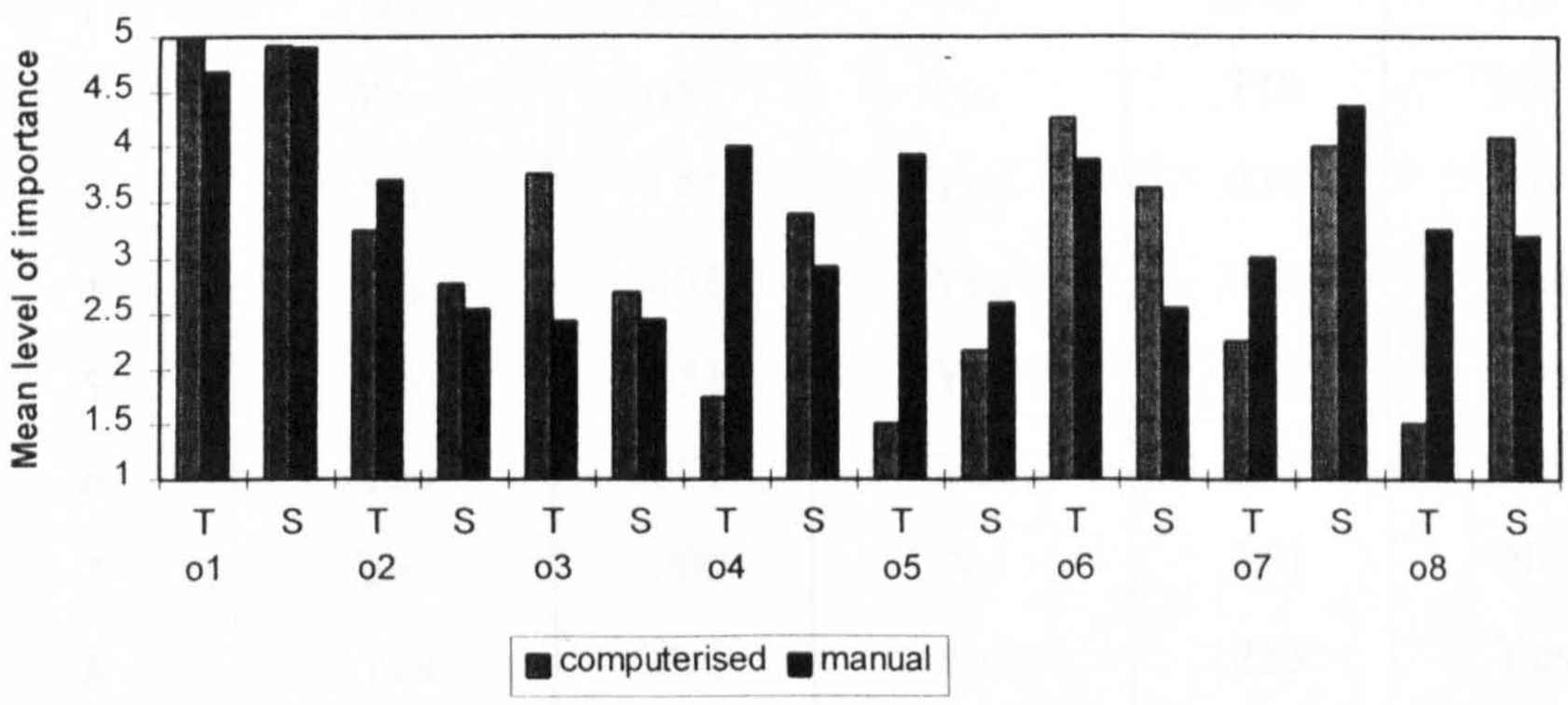


Figure 3.10 : The level of importance of objectives for different university structures having different timetabling systems

The statistical technique used to test for a relationship between the university structure and the examination timetabling system when considering the objectives is a simple factorial ANOVA. The tables of analysis-of-variance for the level of importance of each

of the objectives can be seen in Appendix 3 and the summary of the results obtained for each objective is shown in Table 3.4.

The results from Table 3.4 indicate that there are main effects of the university structure for Objectives 2, 6, 7 and 8 with the observed significance levels of less than 0.10, 0.05, 0.000 and 0.005 respectively. Based on Figure 3.10, scheduling large exams early in the examination period (Objective 6) is rated higher by the term based universities than the semester based universities. Meanwhile, the term based universities felt that it was less significant to avoid having exams with different durations in the same room (Objective 7) and to avoid splitting any exam between two or more rooms (Objective 8) than the semester based universities. The significant result is in the main effect of the university structure for Objective 2, which is to ensure that no student has to take more than one exam on the same day. It makes no real impact since this objective is not regarded as being one of the highly important objectives initially.

Objective	Main effect[1]	Sig. of F	Main effect[2]	Sig. of F	Interaction	Sig. of F
1	No	.625	No	.273	No	.316
2	Yes	.057	No	.778	No	.421
3	No	.141	Yes	.031	No	.129
4	No	.471	Yes	.022	Yes	.001
5	No	.451	Yes	.002	Yes	.028
6	Yes	.028	Yes	.099	No	.431
7	Yes	.000	No	.161	No	.624
8	Yes	.001	No	.253	Yes	.001

[1] Main effect of the university structure factor

[2] Main effect of the examination timetabling system factor

Table 3.4: Summary of the results for the level of importance of objectives

There are main effects of the examination timetabling systems for Objectives 3, 4, 5 and 6 with the observed significance level of less than 0.05, 0.05, 0.005 and 0.10

respectively. From the previous section, Objectives 3 and 5 are both found to be least important so they will be left out in our discussions. However, to minimise the overall duration of the examination period (Objective 4) is quite important and the results show that universities with manual timetabling systems rated its level of importance significantly higher than the universities having computerised systems. The situation is reversed for Objective 6 where universities having computerised timetabling systems rated the level of importance of scheduling large exams early in the examination period higher than the universities with manual timetabling systems.

There are interactions between the university system and the examination timetabling system in determining the level of importance of Objectives 4, 5 and 8 (significance level of less than 0.005, 0.05 and 0.005 respectively) when constructing examination timetables. The graphical representation of an interaction between the university structure and the examination timetabling system is shown in Figure 3.11 for Objective 4 and Figure 3.12 for Objective 8. In both graphs, the semester based universities with computerised timetabling systems rated the level of importance of Objectives 4 or 8 higher than the term based universities with computerised timetabling systems, meanwhile the situation is the opposite in the universities with manual timetabling systems. Therefore, the levels of importance of Objective 4 and Objective 8 for each university structure depend on the examination timetabling systems used.

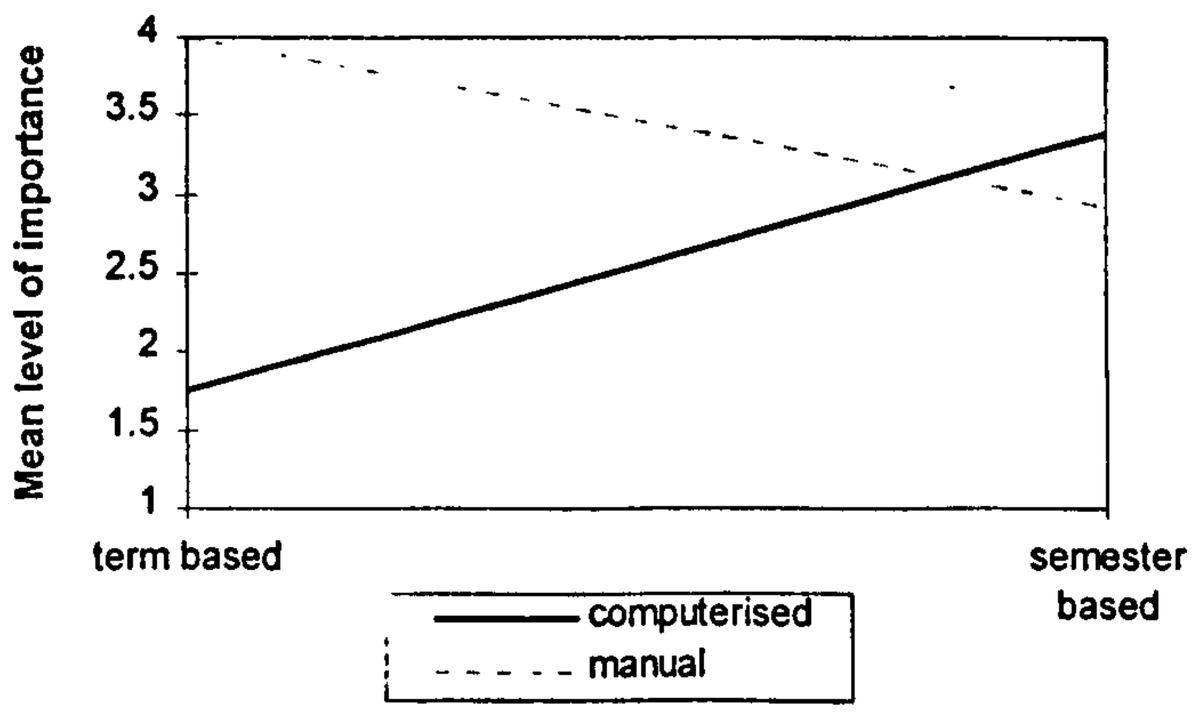


Figure 3.11 : An interaction in the level of importance of Objective 4

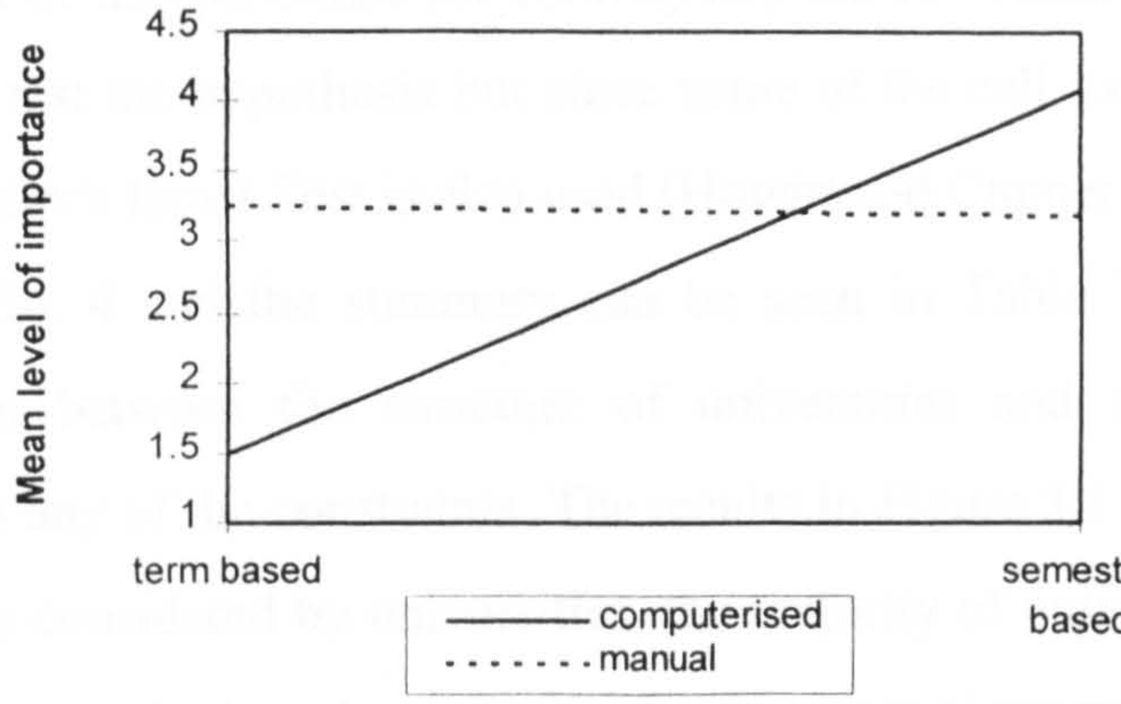


Figure 3.12 : An interaction in the level of importance of Objective 8

3.4.7 Associations Between University Structure And Examination Timetabling System When Considering Constraints

Figure 3.13 illustrates the number of term based (T) and semester based (S) universities having either computerised or manual examination timetabling systems considering each of the constraints when timetabling examinations.

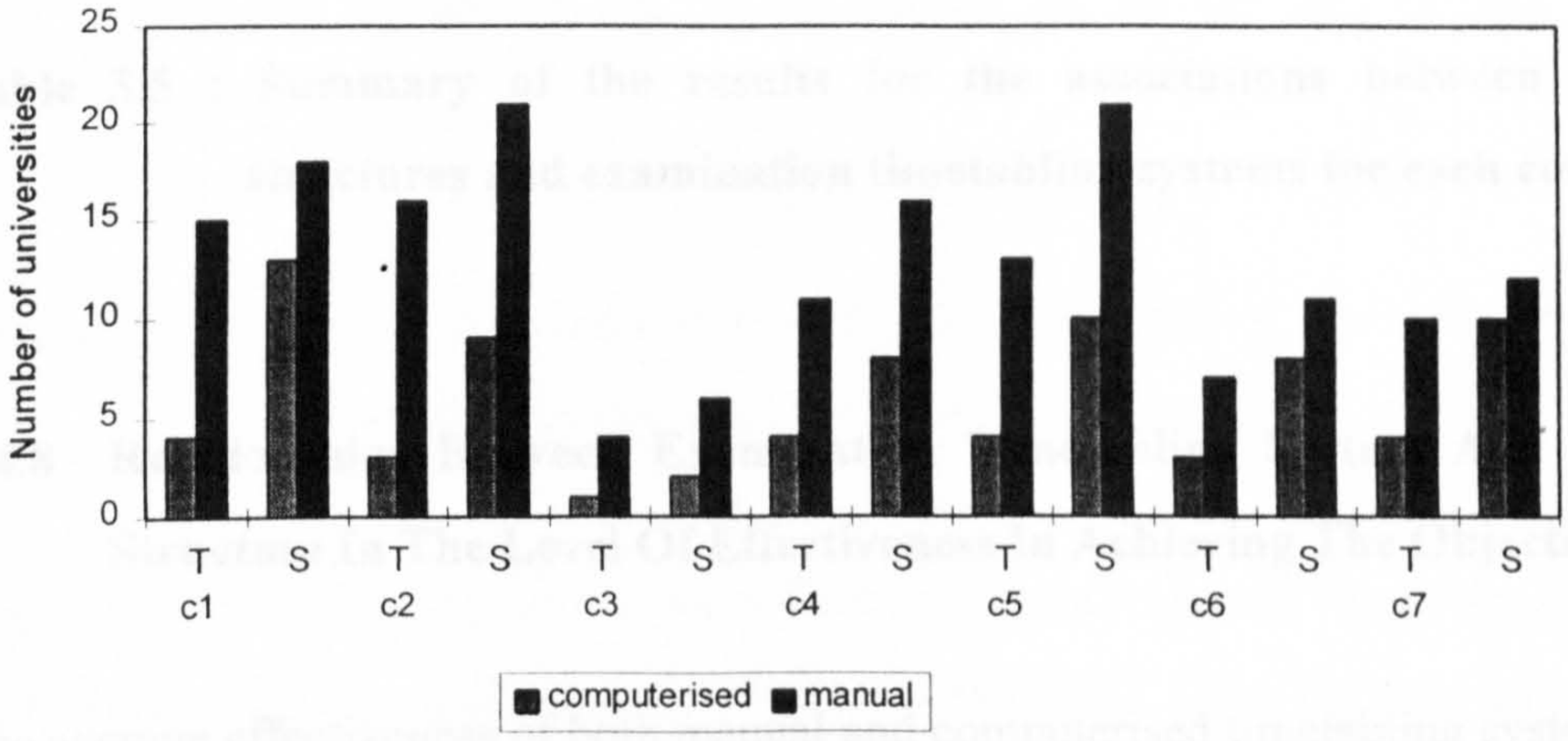


Figure 3.13 : The number of universities with different structures and examination timetabling systems considering the constraints

A statistical test used to determine the association between two variables is the chi-square test of independence for contingency tables. Usually the Pearson chi-square test is used to test the hypothesis but since some of the cell values in the 2x2 table are less than 5, Fisher's Exact Test is also used (Howitt and Cramer [1997]. The results are given in Appendix 4 and the summary can be seen in Table 3.5. There appears to be no association between the structure of universities and the examination timetabling systems in any of the constraints. The results in Figure 3.13 indicate that for each of the constraints considered by universities, the majority of universities come from both term based and semester based universities having manual examination timetabling systems

Constraint	Association between university structure and examination timetabling system	Sig. of F
1	No	.13027
2	No	.21813
3	No	.68531
4	No	.47175
5	No	.38584
6	No	.41119
7	No	.31107

Table 3.5 : Summary of the results for the associations between university structures and examination timetabling systems for each constraint

3.4.8 Relationships Between Examination Timetabling System And University Structure In The Level Of Effectiveness In Achieving The Objectives

The average effectiveness of both manual and computerised timetabling systems in term based (T) and semester based (S) universities is shown in Figure 3.14. A 5 point scale was used to record the level of effectiveness of universities examination timetabling

systems in achieving the objectives when constructing timetables; '1' being not effective at all to '5' being totally effective.

Again simple factorial ANOVA is used to test the main effects of the two factors and the interaction between them (the results are shown in Appendix 5). A summary of the results is shown in Table 3.6. The main effects of the university structure occur in the level of effectiveness in achieving Objectives 5 and 6 with significance level of less than 0.05 and 0.10 respectively. However since Objective 5 is not very popular among the examination officers, the result is not very relevant. The result for Objective 6 is quite important since it is highly rated by examination officers. We can see from the graph in Figure 3.14 that both the timetabling systems in term based universities perform more significantly effective in scheduling large exams early in the examination period than the systems in the semester based universities.

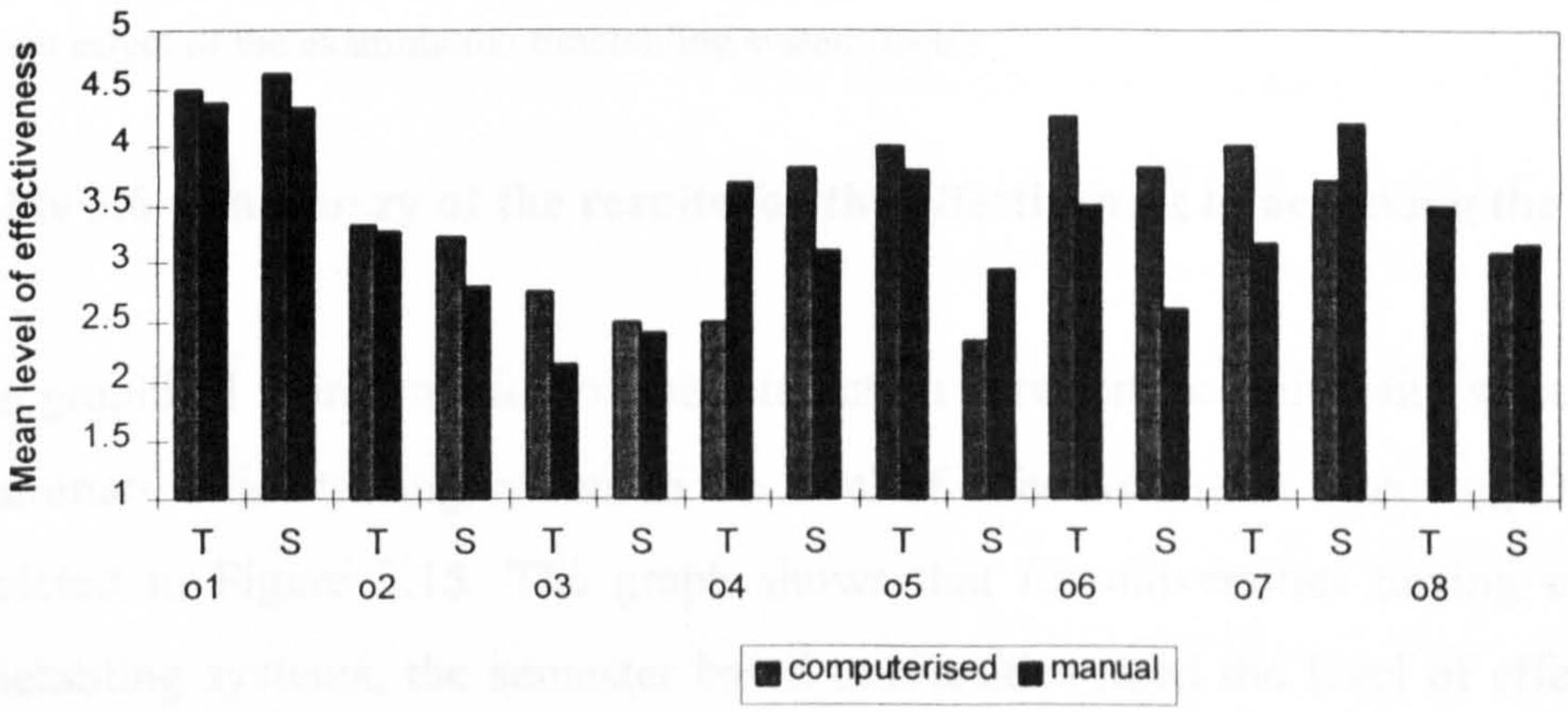


Figure 3.14 : The effectiveness of examination timetabling systems with different university structures in achieving the objectives

There is also only one significant main effect of the examination timetabling system, i.e. for Objective 6 with significance level of less than o0.05. From Figure 3.14, the level of effectiveness in achieving the task of scheduling large exams early in the examination period is higher in the universities with computerised timetabling systems than with manual systems. There seems to be only one interaction between the university

structures and the examination timetabling systems in the level of effectiveness in achieving Objectives 4 (significance level of less than 0.10).

Objective	Main effect[1]	Sig. of F	Main effect[2]	Sig. of F	Interaction	Sig. of F
1	No	.855	No	.316	No	.698
2	No	.557	No	.618	No	.714
3	No	.993	No	.411	No	.554
4	No	.510	No	.649	Yes	.092
5	Yes	.044	No	.748	No	.500
6	Yes	.080	Yes	.013	No	.546
7	No	.542	No	.772	No	.267
8	No	.410	No	.875	Not available*	-

* The interactions have been suppressed due to lack of responses

[1] Main effect of the university structure factor

[2] Main effect of the examination timetabling system factor

Table 3.6 : Summary of the results for the effectiveness in achieving the objectives

The graphical representation of the interaction between the university structure and the examination timetabling system in the level of effectiveness in achieving Objective 4 is depicted in Figure 3.15. The graph shows that for universities having computerised timetabling systems, the semester based universities rated the level of effectiveness of their systems higher than the term based universities. However, for universities with manual timetabling systems, the term based universities felt that their systems are slightly more effective than the semester based universities. In other words, the level of effectiveness of both examination timetabling systems in achieving the objective to minimise the overall duration of the examination period depends on the structure of the universities.

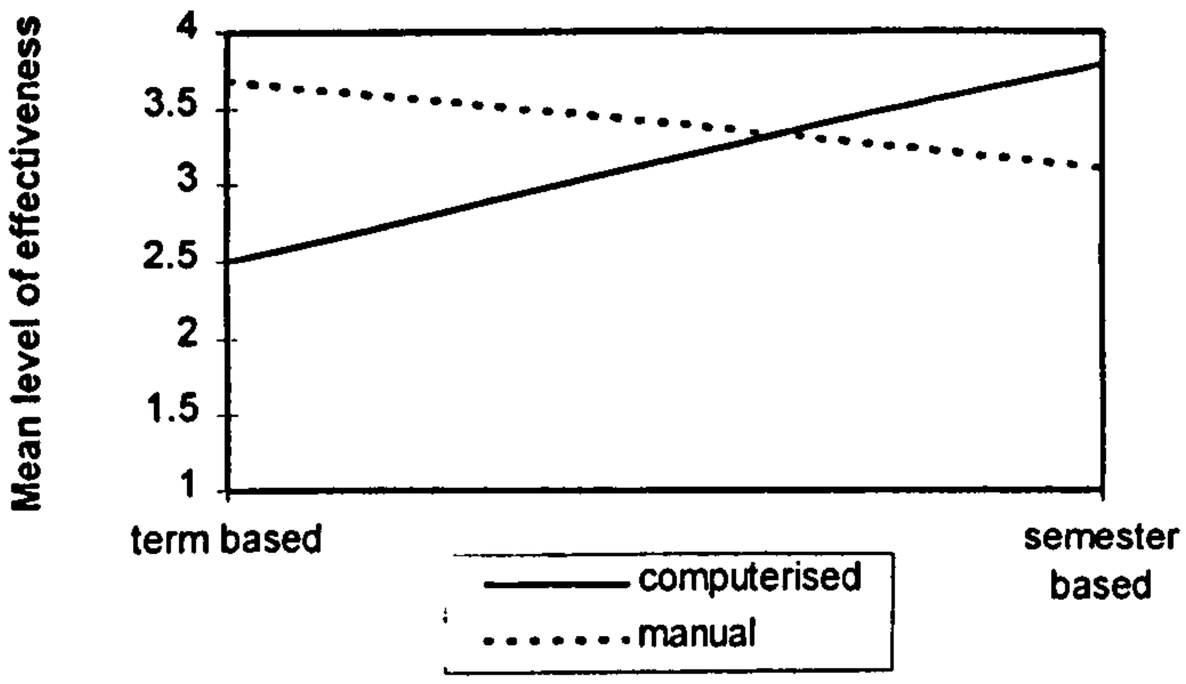


Figure 3.15 : An interaction in the level of effectiveness in achieving Objective 4

3.4.9 Relationships Between Examination Timetabling System And University Structure In The Level Of Effectiveness In Satisfying The Constraints

Figure 3.16 shows the average effectiveness of both computerised and manual examination timetabling systems in term based (T) and semester based (S) universities in satisfying the constraints considered. Similar tests (simple factorial ANOVA) were carried out and the results can be seen in Appendix 6. The results show that there is only one significant result in the entire tests, i.e. there is a significant main effect of the university structure for Constraint 4 with significance level of less than 0.10. Unfortunately, this constraint has been found to be one of the less common ones and therefore, regarded as not relevant. There is no significant main effect of the examination timetabling system factor and there is also no interaction between the two factors (refer to Table 3.7). An obvious conclusion from Figure 3.16 is both types of examination timetabling system in both types of university are quite effective in satisfying all the constraints considered when constructing timetables

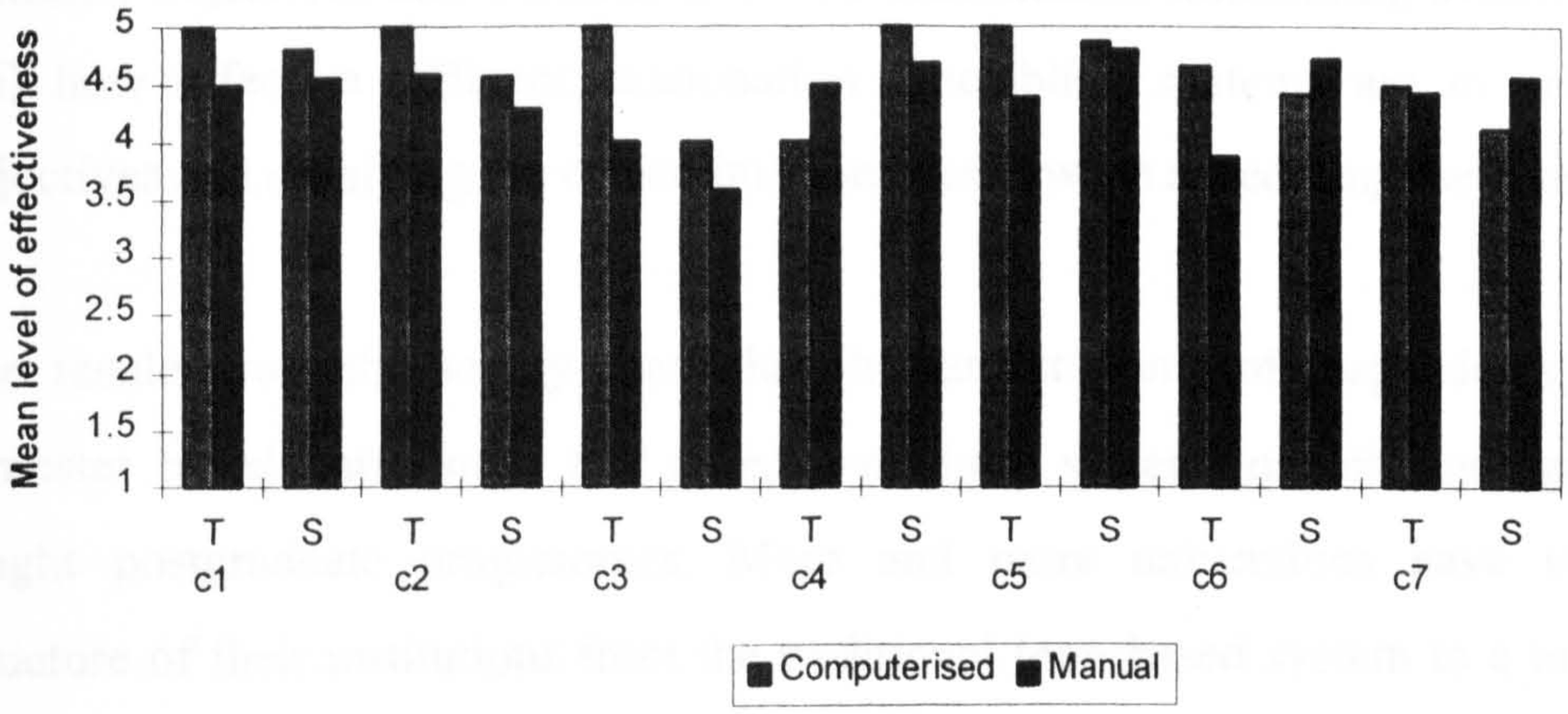


Figure 3.16 : The effectiveness of examination timetabling systems with different university structures in satisfying the constraints

Constraint	Main effect[1]	Sig. of F	Main effect[2]	Sig. of F	Interaction	Sig. of F
1	No	.786	No	.106	No	.610
2	No	.164	No	.199	No	.829
3	No	.221	No	.221	No	.614
4	Yes	.070	No	.770	No	.209
5	No	.571	No	.202	No	.301
6	No	.471	No	.562	No	.206
7	No	.677	No	.488	No	.381

[1] Main effect of the university structure factor

[2] Main effect of the examination timetabling system factor

Table 3.7 : Summary of the results for the level of effectiveness in satisfying the constraints

3.5 Summary

The survey of university examination timetabling systems was carried out to determine (i) how extensively the computer is used when scheduling examinations, (ii) the

common objectives and constraints of the examination timetabling systems used and (iii) how effective different examination timetabling systems are in achieving the objectives and satisfying the constraints considered when scheduling the examinations.

The results from the survey show that the largest group of respondents consists of semester based universities that adopt a modular system in both undergraduate and taught postgraduate programmes. More and more universities have changed the structure of their institutions from the traditional term based system to a more modern semesterised system. The semester based system offers more flexibility for students in that they can choose courses and pick up credits where and when they can. The students can benefit from the system where good results can be obtained for sustained work. It shows that there will be a great need for a computerised examination timetabling in order to cope with the change in the timetabling problem. Students in semesterised universities can choose whatever courses they like and this will create a longer clash list and therefore increase the workload of the examination officer in producing a clash free timetable.

Most of the semester based universities hold their major examinations twice a year, usually at the end of each semester, while most of the term based universities have examinations only once a year. The average examination period in semester based universities is shorter since they have 2 exam diets in a year. This can sometimes cause a problem for examination officers to fit in all exams since the length of the examination period may not be long enough. The results also show that low percentages of both term and semester based universities use computerised timetabling systems even though there are a number of commercial examination scheduling packages available on the market. This, according to Burke *et al.* [1996] might be due to the fact that the timetable does not change significantly from year to year, so they can just do a minor alteration to the old timetable.

The findings from the survey showed that the requirements on the examination timetabling of each university vary but since we are trying to build a general timetabling

system which would satisfy most of the universities, we are only interested in the mostly considered objectives and constraints. The common objectives which are considered by university examination officers in decreasing order of importance are (i) to produce a clash free timetable, (ii) to avoid having exams with different durations in the same room, (iii) to schedule large examinations early in the examination period, (iv) to avoid splitting any exam between two or more rooms and (v) to minimise the overall duration of the examination period. Meanwhile the constraints which have been considered by more than three-quarter of the respondents are (i) a specified time limit on the overall examination period, (ii) a maximum number of students that may be examined in any session and (iii) certain specified exams that must take place during the same session.

In the survey by Burke *et al.*, thirteen common requirements known as timetabling constraint were considered. The results produced in our survey are consistent with the results by Burke *et al.* The first two high ranking constraints in Burke *et al.* are similar to Constraints 2 and 5 in our survey where they were considered by more than 80% of the respondents. Constraints 4 and 5 in Burke *et al.* are in fact Objectives 7 and 6 respectively in our survey and they both have been rated high levels of importance when timetabling examinations.

However, they have not considered any constraints on the duration of the examination period. In our survey, we have taken into account the objective to minimise the overall length of the examination period (Objective 4) and a specified limit on the examination period (Constraint 1) nor did they consider avoiding splitting exams in two or more rooms (Objective 8), which were all found to have significant results.

A simple factorial ANOVA test is carried out to establish the effect of the university structure and the examination timetabling system on the level of importance of objectives, the level of effectiveness in achieving the objectives and also the level of effectiveness in satisfying the constraints. The significant results gathered from the tests are:

- there is a significant main effect of the university structure on the level of importance for Objective 6, i.e. the term based universities rated the level of importance of scheduling large exams early in the examination period higher than the semester based universities

- there is a significant main effect of the university structure on the level of importance for Objectives 7 and 8, i.e. the term based universities rated the level of importance of avoiding having exams with different durations in the same room and to avoid splitting any exam between two or more rooms lower than the semester based universities

- there is a significant main effect of the examination timetabling system on the level of importance for Objective 4, i.e. the universities with manual timetabling systems rated the level of importance of minimising the overall duration of the examination period higher than the universities with computerised timetabling systems

- there is a significant main effect of the examination timetabling system on the level of importance for Objective 6, i.e. the universities with manual timetabling systems rated the level of importance of scheduling large exams early in the examination period lower than the universities with computerised timetabling systems

- there is a significant interaction between the university structure and the examination timetabling system on the level of importance for Objectives 4 and 8, i.e. the level of importance of minimising the overall duration of the examination period and the level of importance of not splitting any exam between two or more rooms in the semester based universities with computerised timetabling systems is higher than in the term based universities with computerised timetabling systems and vice versa

- there is a significant main effect of the university structure on the level of effectiveness in achieving Objective 6, i.e. the term based universities rated the level

of effectiveness in scheduling large exams early in the examination period higher than the semester based universities

- there is a significant main effect of the examination timetabling system on the level of effectiveness in achieving Objective 6, i.e. the universities with manual timetabling systems rated the level of effectiveness of scheduling large exams early in the examination period lower than the universities with computerised timetabling systems

- there is a significant interaction between the university structure and the examination timetabling system on the level of effectiveness for Objectives 4, i.e. the level of effectiveness of minimising the overall duration of the examination period in the semester based universities with computerised timetabling systems is higher than in the term based universities with computerised timetabling systems, and vice versa

Almost all respondents surveyed said that they faced at least one problem when timetabling examinations, either having to do it manually or using a computer. Most problems that occurred however were out of their direct control. One problem is not having rooms big enough to accommodate large numbers of students. The students then have to be split between two or more rooms. Another is a lack of co-operation from the departments and students in getting accurate source data or information about exams. Examination officers who have to construct timetables manually, found it extremely difficult to avoid clashes for all students because the procedure of checking for clashes in each exam session can be very tedious and laborious; and also if errors are detected after the timetable is completed, difficult rescheduling needs to be done. Nevertheless, in spite of all the problems they face when constructing examination timetables, their current systems manage to cope. Examination officers in both systems believed that their procedures were capable of achieving all of the objectives considered as well as satisfying the constraints.

Therefore, one major advantage of using the computer to timetable examinations is that it would prevent examination officers from making mistakes in checking for clashes in every exam session - this could all be done by the computer. Moreover, whenever there are corrections to be made to the timetable after it is completed, it can be easily done using the computer. A computer based timetabling system would therefore seem to be of benefit to most university examination officers.

CHAPTER 4

AN OVERVIEW TO THE TEST PROBLEMS

4.1 Introduction

The test problems used in this research will be described in this chapter. These 13 unconstrained real life problems have been provided by Carter *et al.* [1996] and they can be obtained from the Internet at the University of Toronto web-site www.ie.utoronto.ca. All data files can be found in the directory /mwc/testprob. Real life problems are not widely available for researchers in examination timetabling and most testing is usually limited to randomly generated problems. Testing using real life problems gives us the opportunity to examine the true picture that arises in examination timetabling situations. The approaches in solving the examination timetabling problem are discussed in detail and the results obtained will be presented.

4.2 Description Of Data Sets

For the purpose of this research, eleven out of the thirteen problems from Canada, United Kingdom and the Middle East were selected. We left out the data from Purdue University, Indiana because we felt that it was too big for this work and hence would take a lot of time to run, and is also much larger than is likely to arise in most British

universities. The other data set was not left out deliberately as it was originally missing from the authors' file. The main characteristics of the problems are shown in Table 4.1 below. There are basically two categories of data; medium scale and large scale problems. The first eight problems in the table are of medium scale while the last three CAF, UTA and CAS can be considered as large scale problems. Two unused sets of data are located at the bottom of the table (below the dotted line).

Code	Institution	Number of examinations	Number of students	Number of student examinations	Density *d* of the conflict matrix
HEC	École des Hautes Études Commerciales, Montreal	81	2823	10632	0.42
STA	St. Andrew's Junior High School, Toronto	139	611	5751	0.14
YOR	York Mills Collegiate Institute, Toronto	181	941	6034	0.29
EAR	Earl Haig Collegiate Institute, Toronto	190	1125	8109	0.27
LSE	London School of Economics	381	2726	10918	0.06
UTE	Faculty of Engineering, University of Toronto	184	2749	11793	0.08
TRE	Trent University, Peterborough, Ontario	261	4360	14901	0.18
KFU	King Fahd University of Petroleum and Minerals, Dhahran	461	5349	25113	0.06
CAF	Carleton University, Ottawa (Fall Semester Exam)	543	18419	55522	0.14
UTA	Faculty of Arts and Science, University of Toronto	622	21266	58979	0.13
CAS	Carleton University, Ottawa (Spring Semester Exam)	682	16925	56877	0.13
PUR	Purdue University, Indiana	2419	30032	120681	0.03
RYE	Ryerson University, Toronto	487	11483	45051	0.07

Table 4.1 : Main characteristics of the problems

The density (D) of the conflict matrix (C) represents the proportion of non-zero and non-diagonal entries in the conflict matrix. High density in the conflict matrix shows a

large number of conflicting exams. Let there be E exams and S students, the incidence matrix $X = \{x_{ik}\}$ where x_{ik} is a zero-one variable such that,

$$x_{ik} = 1 \qquad \text{if student } k \text{ takes exam } i,$$
$$= 0 \qquad \text{otherwise.}$$

The conflict matrix $C = \{c_{ij}\}$ where c_{ij} is the number of students taking both exam i and j. Therefore,

$$c_{ij} = \sum_k x_{ik} x_{jk}$$

where $c_{ii} = \sum_k x_{ik}^2 = \sum_k x_{ik} = n_i$ is the number of students taking exam i. Now let d_{ij} be a (0,1) variable such that

$$d_{ij} = 1 \qquad \text{if } c_{ij} > 0$$
$$= 0 \qquad \text{otherwise,}$$

then the density of the conflict matrix can be defined as,

$$D = \frac{\sum_i \sum_{j>i} d_{ij}}{\frac{1}{2} E(E-1)}$$

$$= \frac{\sum_{i=1}^{E} \sum_{j=1}^{E} d_{ij} - E}{E(E-1)}$$

Each of the data sets has two files of the form *.stu and *.crs. The file *.stu contains a set of exams (X), taken by each of S students,

$$*.\text{stu} \Rightarrow \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ : \\ X_S \end{bmatrix}$$

The *.crs file meanwhile contains a record for each of E courses and consists of the total number of students (n) taking each course,

$$*.\text{crs} \Rightarrow \begin{bmatrix} 1 & n_1 \\ 2 & n_2 \\ 3 & n_3 \\ : & : \\ E & n_E \end{bmatrix}$$

Initial testing on the data showed that there were a number of discrepancies in the data sets as reported in the paper. As a result, a letter commenting on the inconsistencies was sent to the authors. The errors were then acknowledged and corrected.

4.3 Techniques Used For Solving Examination Timetabling (Without Costs)

The examination timetabling problem (without costs) involves finding a conflict-free timetable without considering any side constraints. It is also known as the graph colouring problem. This underlying problem is usually solved using graph colouring heuristics, where exams are initially sorted according to some criterion and are then sequentially assigned to sessions without creating conflicts. Carter *et al.* [1996] incorporated a proximity cost w_S where $w_S > w_t$ if $s < t$, whenever a student has to sit for two exams scheduled s sessions apart, to their examination timetabling problem. This proximity cost is used to spread out student exams over the examination period. This problem is referred to the examination timetabling with costs.

This approach however cannot always achieve the best results possible, i.e. the least number of sessions required to schedule non-conflicting exams, where the number of sessions produced can sometimes be excessive. Therefore two extensions to the basic method, namely a clique initialisation strategy (Carraghan and Pardalos [1990], Carter and Gendreau [1992]) and a backtracking process (Laporte and Desroches [1984], Hertz [1991], Carter, Laporte and Chinneck [1994]) have been introduced recently to overcome this problem.

4.3.1 Initial Ordering Strategy

The initial ordering strategy has been proven to play an important part in solving graph colouring problems (Dunstan [1976] and Matula *et al.* [1972]). Most initial ordering strategies are based on the degrees of the vertices (exams). The degree of an exam is the number of 'edges' it has, i.e. the number of other exams with which it is in conflict (having at least one student in common). Carter *et al.* have used four initial ordering strategies, namely:

i) *Largest Degree (LD) :* The exams are sorted in decreasing order of degree. The scheduling process proceeds by selecting exams from the top of the list and assigning them to the lowest numbered non-conflicting session.

ii) *Saturation Degree (SD) :* The exams are sorted in decreasing order of degree. The exam selected next is the one that has the fewest number of non-conflicting sessions. Ties are broken in favour of the largest degree.

iii) *Largest Weighted Degree (LWD) :* The exams are sorted in decreasing order of degree, but this time each edge is weighted by the number of students in conflict.

iv) *Largest Enrolment (LE) :* The exams are ordered by the number of students registered for each examination.

Carter *et al* [1996] also tested their methods using random ordering of the exams (*RO*) as a comparison to the other four initial ordering strategies. There are several other initial ordering strategies that have been used by different authors.

v) *Largest Degree First Recursive (LDFR) :* This method is similar to the *Largest Degree* method except that after an exam has been assigned to the lowest numbered non-conflicting session, the exam is removed from the list, the degrees recalculated and the rest of the exams in the list are resorted.

vi) *Largest Modified 1-Degree First (LMDF) :* The following formula for computing the critical property of an exam is used, $d_1(e_i) = \sum_{j \in Ai} d_0(e_j) \quad i = 1,.....,n$ where A_i is the set of exams in conflict with exam e_i and $d_0(e_j)$ is the degree of exam e_j. Hence $d_1(e_i)$ is the sum of the degrees of the exams adjacent to e_i. The exams are sorted in descending order of $d_1(e_i)$.

vii) *Smallest degree last recursive (SDLR) :* The exam with the lowest degree is placed at the end of the list. The degrees of the remaining exams are recalculated and the process is repeated. When the list is completed, exams are assigned from the top of the list.

viii) *Smallest Degree Last Recursive with Interchange (SDLRI) :* The exams are first ordered by SDLR. The exam at the top of the list of unassigned exams e_i, is assigned to the lowest numbered conflict-free session that has already been used. If exam e_i conflicts with all of the current sessions, find a session s_j for which there is only one conflicting exam e_j. If possible, reassign exam e_j. Otherwise look for a bichromatic interchange. Specifically, for each session s, locate the set of exams E_r in session s that conflicts with exam e_j. If the set E_r does not conflict with exam e_i or any other exam in session s_j, then interchange exam e_j with the set E_r, which allows exam e_i to be assigned to session s_j. If no such interchange can be found, a new session is created and exam e_i is assigned to it. The algorithm continues with the next exam.

Johnson [1990] introduced a 'difficulty' factor for ordering exams at the initial stage before assigning them to the available examination sessions.

ix) *Difficulty Factor (DF) :* Two factors which make an exam difficult to schedule are its size, i.e. the number of students taking the exam, because of the space constraint and its degree, the number of other exams with which it is in conflict. In this case, a weighting element is used to balance the importance between the two factors.

4.3.2 Clique Initialisation Strategy

A clique is a set of mutually conflicting exams and the largest clique represents the most difficult group of exams to be scheduled since they conflict with each other and possibly with other exams as well. By scheduling these exams first, we are left with exams which are likely to have less conflict and which can be scheduled together in the same sessions. This process will reduce the possibilities of having to create extra sessions in order to avoid conflicts. Therefore it is vital to determine the largest clique since the size of a conflict-free schedule is at least as large as the size of the largest clique.

For the determination of the clique, Carter *et* al. have used the Carter and Gendreau [1992] algorithm. The algorithm is based on the assumption that the vertex of minimum degree in a graph is the least likely member of a large clique. On completion of the iterative process of finding a large clique, *CMIN* is the size of the largest clique found and *CMAX* is an upper bound on the largest clique which could have been missed. The recursion will continue until either the bound on the largest clique is less than *CMIN* or the induced subgraph is itself a clique. The values of *CMIN* and *CMAX* will always be equal and each occurrence of *CMAX* can be replaced by *CMIN*. The description of the optimal algorithm is shown below:

CLIQUE1 (*G,CMIN,CMAX,CLIQUE*): Given a graph $G = (V,E)$, the routine computes *CMIN* and *CMAX* and stores the largest clique found in the vector *CLIQUE*.

Step 0 : Set $CMAX = 0$, $CMIN = 0$, $n = |V|$.

Step 1 : Let v_i be the vertex of minimum degree, $deg(v_i)$, in the graph.

Step 2 : If $n \leq CMIN$, STOP. If $deg(v_i) = (n-1)$, go to Step 5.

Step 3 : If $deg(v_i) + 1 \leq CMIN$, then go to Step 4. Otherwise, let V_i be the set of all vertices adjacent to v_i. Let G_i be the graph induced on G by the vertices.

Call **CLIQUE0** ($G_i, CMIN_i, CMAX_i, CLIQUE_i$)

If $CMIN_i + 1 > CMIN$, set $CMIN = CMIN_i + 1$ (and save the new largest clique, the vector $CLIQUE_i$ plus v_i in a vector *CLIQUE*). If $CMAX_i + 1 > CMAX$, set $CMAX = CMAX_i + 1$. If $CMIN > CMAX$, set $CMAX = CMIN$.

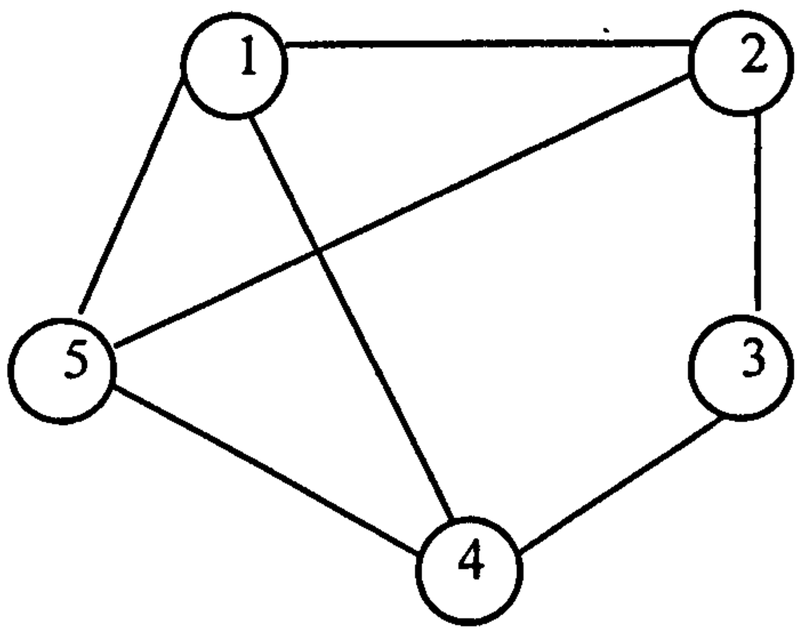
Step 4 : Remove node from the graph *G*, set $n = n - 1$ and go to Step 1.

Step 5 : The remaining vertices form a clique. Set $CMIN = n$, store the vertices in the vector *CLIQUE* and stop.

Due to the difficulty in obtaining the paper written by Carter and Gendreau at the early stage of the research, it was decided to explore other available methods. We then decided to employ an exact algorithm by Carraghan and Pardalos [1990] because of its simplicity and furthermore the FORTRAN 77 code of the algorithm is also given. This is an exact method which would be guaranteed to produce an optimal solution compared to that of Carter and Gendrau which is just a heuristic method and so does not guarantee a maximum clique.

Initially, the algorithm considers a recursive ordering of the vertices of any graph G, say $v_1, v_2,.....,v_n$ where v_1 is a vertex of smallest degree in G, v_2 is a vertex of smallest degree in G-(v_1), and generally v_k is a vertex of smallest degree in G-$\{v_1, v_2,.....,v_{k-1}\}$, $k \leq n-2$. In case of ties, the smallest numbered vertex is chosen. Crucial to the understanding of the algorithm is the notion of depth. Suppose we are dealing with any vertex v_i. At depth 2, the algorithm considers all vertices adjacent to v_i. At depth 3 the algorithm considers those vertices in depth 2 that are adjacent to the chosen vertex in depth 2, and so on. The depths are added until there are no more vertices to be expanded. The size of current best clique (CBC) is equal to the current depth. Let v_{di} be the vertex that is currently being expanded at depth d, i.e. depth d comprises m vertices $v_{d1},.....,v_{di},....,v_{dm}$. Pruning takes place if $d+(m-i) \leq$ CBC since the size of the largest possible clique would be less or equal to the size of CBC. The algorithm will stop if this inequality holds at depth 1 and the maximum clique is equal to the best CBC.

With the help of an example below, the notion of the depth in the algorithm will be explained. Consider the graph,



The steps of the algorithm are as follows.

Iteration 0 : The initial ordering of exams is 3 2 1 4 5.

Iteration 1 : At Depth 1, we have exams {3,2,1,4,5} where $m = 5$. Therefore expand on $v_{11} = 3$.

At Depth 2, $v_{21} = 2$ and $v_{22} = 4$ (where $m = 2$) . We cannot expand on v_{21} further since it does not conflict with any of the other exams v_{11} is in conflict with (in this case v_{22}). Therefore CBC = 2 with exams {3,2}. At Depth 2, we then expand on $v_{22} = 4$, but since $d+(m-i) = 2+(2-2) =$ CBC, then we prune.

Iteration 2 : At Depth 1, we have exams {3,2,1,4,5} where $m = 5$. Therefore expand on $v_{12} = 2$.

At Depth 2, $v_{21} = 1$ and $v_{22} = 5$ (where $m = 2$). We expand on v_{21}.

At Depth 3, $v_{31} = 5$ (where $m = 1$). We cannot expand anymore on v_{31}, so new CBC = 3 with exams {2,1,5}.

At Depth 2, $v_{22} = 5$, but since $d+(m-i) = 2+(2-2) <$ CBC, then we prune.

Iteration 3 : At Depth 1, we have exams {3,2,1,4,5} where $m = 5$. Therefore expand on $v_{13} = 1$, but since $d+(m-i) = 1+(5-3) =$ CBC and also at Depth 1, then we STOP. We have a maximum clique {2,1,5} of size 3.

One other advantage is that the algorithm can be modified to find a set of maximum cliques instead of just one maximum clique in the graph. This method uses the same principle except that the pruning process is carried out when $d+(m-i) <$ size of CBC. The algorithm will produce all possible cliques of maximum size. In the example above, we will find that there exists another maximum clique with exams {1,4,5}. The objective of finding these alternative cliques is to test whether different initial cliques tend to produce different results and if so which one of them gives the least number of sessions (this subject will be discussed further in Chapter 5).

Table 4.2 contains the results from the test data using the modified algorithm, i.e. the size of the maximum clique, the number of maximum cliques and both the times taken to find one maximum clique and all of them. The size of the maximum clique will be a lower bound to the number of sessions required to assign all exams without creating any conflicts. The minimum number of sessions then cannot be less than the size of the

maximum clique. The number of cliques varies from problem to problem; for example there is only one maximum clique of size 17 in HEC whereas there are 156 different maximum cliques of size 23 in CAS.

Data	Size of maximum clique	No. of cliques	Time 1*	Time 2**
HEC	17	1	0.29	0.29
STA	13	60	1.61	39.61
YOR	18	32	3.08	41.27
EAR	21	7	4.50	15.30
LSE	17	2	40.94	57.32
UTE	10	4	3.80	8.36
TRE	20	4	11.16	24.55
KFU	19	2	93.54	130.96
CAF	24	3	130.05	234.09
UTA	26	128	205.05	10621.59
CAS	23	156	252.45	15904.35

[* Time (sec) for 1 maximum clique ** Time (sec) for all maximum cliques]

Table 4.2 : Computational results using the modified algorithm

All of the medium sized problems took less than 2 minutes to find one maximum clique while the larger problems took from 2 - 4 minutes on a Hewlett-Packard HP-UX 9000/755 machine. On the other hand, we can see from Table 4.2 that the times taken to produce 128 maximum cliques for UTA and 156 maximum cliques for CAS are 2.9 and 4.4 hours respectively. For the rest, the times are reasonably small. Therefore, the testing of both UTA and CAS test data can be expected to be quite long.

4.3.3 Backtracking Process

This is a process where, whenever an exam to be scheduled next is in conflict with every session, the conflicting exams in a particular session are removed and put back in the waiting list in order to accommodate the current exam.

Carter *et al.* [1996] have implemented the following backtracking process. Suppose the next exam to be scheduled (exam *i)* is in conflict with every available session. Consider all exams E_j already assigned in session *s* which conflict with exam *i* and identify the session to which each exam in E_j can be moved at least cost M_S where M_S is the cost incurred by moving such exams to different sessions. Each of exam *j* which cannot be moved in this way is bumped back, i.e. put back into the list of unscheduled exams (waiting list) with a cost B_S. The cost B_S of bumping a set of examinations from session *s* is equal to the number of examinations that must be bumped back to the list of unscheduled exams. To avoid cycling, no exam *j* can be bumped by the same exam *i* more than once. If exam *i* has already bumped some exam *j* in session *s*, set $B_S = \infty$, and therefore exam *i* cannot be assigned to session *s*. When it is not necessary to bump any exams back to the waiting list, set $B_S = 0$. At the end of the search, there are three possible outcomes:

1. $\min_{S \in Si} B_S = \infty$. No feasible solution can be found. Exam *i* is removed from the waiting list and reported as unschedulable at the end of the algorithm.

2. $\min_{S \in Si} B_S = 0$. There exists at least one session *s* to which exam *i* can be assigned without bumping any exam on to the waiting list. Select among these sessions the session s^* with the minimum disruption cost, M_S*. Exam *i* is then moved to session s^* and the necessary conflicting exams are moved to other sessions as previously identified.

3. $0 < \min_{S \in Si} B_S < \infty$. Let session s^* be the session yielding the minimum. Break any ties using the minimum value of M_S

4.4 Computational Results For Examination Timetabling (Without Costs)

Carter *et al.* [1996] compared each of the five initial ordering strategies; LD, SD, LWD, LE and RO, in producing the minimum number of sessions required to schedule all exams without creating any conflicts. They also tested the effect of using the clique initialisation strategy as a starting point and the backtracking process in minimising the number of sessions. Their tests showed that in all cases, both the clique initialisation strategy and the backtracking process are very useful in yielding fewer sessions, especially for the larger problems.

Code		LD	SD	LWD	LE	RO
HEC	Sessions	18	17	17	17	17
	Backtracks	5	0	102	155	39
STA	Sessions	13	13	13	13	13
	Backtracks	0	0	0	0	2
YOR	Sessions	20	20	20	19	20
	Backtracks	255	1	378	310	208
EAR	Sessions	22	22	22	22	22
	Backtracks	100	1	35	56	114
LSE	Sessions	17	17	17	17	17
	Backtracks	95	10	25	164	79
UTE	Sessions	10	10	10	10	10
	Backtracks	0	0	2	0	4
TRE	Sessions	22	20	20	22	22
	Backtracks	51	2	63	2	132
KFU	Sessions	19	19	19	20	19
	Backtracks	27	101	124	6	247
CAF	Sessions	31	28	30	31	32
	Backtracks	104	510	432	6	205
UTA	Sessions	33	32	33	33	34
	Backtracks	5	2	92	11	89
CAS	Sessions	32	28	30	32	35
	Backtracks	1	311	210	107	59

Table 4.3 : Computational results obtained by Carter *et al.*

Table 4.3 illustrates the outcomes of the tests on the data; the minimum number of sessions obtained and the number of backtracks required to achieve the minimum number of sessions. The results show that by using the saturation degree strategy (SD) as an initial ordering, the numbers of sessions obtained are the fewest. The minimum number of sessions produced for most of the medium sized problems are indeed the sizes of the maximum clique. For three of the medium sized problems HEC, STA and UTE, the minimum number of sessions possible can be achieved without the need to use the backtracking process. However, for large scale problems, it appears that the backtracking process is very crucial in getting the number of sessions as close as possible to the size of the maximum clique.

4.5 Summary

In this chapter, the method used by Carter *et al.* [1996] to solve several unconstrained real life examination timetabling problems is described. This is to be the basis of the research work, using graph colouring methods and incorporating a clique initialisation strategy and backtracking process to solve examination timetabling problems. The real life problems described will be used in the subsequent research work.

The study by Carter *et al.* [1996] suggested that the saturation degree strategy as an initial starting point together with using a clique initialisation strategy and a backtracking process yields the minimum number of sessions required to assign all exams without creating any conflicts. The clique initialisation strategy is very useful in obtaining the fewest number of sessions since by scheduling difficult exams first, it allows less conflicting exams to be scheduled to the existing sessions. The other conclusion which can be deduced from the study is that the backtracking process helps to reduce the number of sessions for large scale problems. For some of the medium scale problems however, the backtracking process does not seem to be required.

CHAPTER 5

A HEURISTIC APPROACH FOR SOLVING UNCONSTRAINED EXAMINATION TIMETABLING PROBLEMS

5.1 Introduction

The objective in solving examination timetabling problems without any side constraints such as room capacities, time restrictions etc., is to find the minimum number of sessions required to assign all exams so that they do not conflict with each other. This chapter analyses the performance of a new algorithmic rule for each data set for achieving the minimum number of sessions by comparing this rule to the methods used by Carter *et al.* [1996]. They have adopted a combination of a graph colouring approach, a clique initialisation strategy and a backtracking process to solve the problem.

Section 5.2 describes how the new algorithmic rule was developed. The rule incorporates a graph colouring approach as well as a clique initialisation strategy. The rule does not, however, use a backtracking process in finding the minimum number of sessions. The next section, Section 5.3 discusses the results in terms of the number of sessions and the clique number obtained from the tests. A summary of the chapter is given in Section 5.4.

5.2 Development Of The New Algorithmic Rule

The study by Carter *et al.* [1996] concluded that using the saturation degree rule to schedule exams produced the best results in terms of the minimum number of sessions and the required number of backtrackings. In this rule, the exams are first sorted in decreasing order of degree and the exam selected to be scheduled is the one that conflicts with the greatest number of the available sessions. The process is repeated until all exams are assigned and if there exists any tie, the exam with the largest degree is chosen. The concept of the new algorithmic rule is similar to the saturation degree adopted by Carter *et al.* where the exam that is selected next is the one with the fewest number of feasible available sessions remaining. The difference is that instead of assigning one exam at a time, the new rule will be able to schedule several exams to different sessions at the same time. The procedure is based on the idea of critical exams where they are the ones that can only be assigned to one of the sessions. By scheduling these exams first, we are dealing with the 'most difficult' exams found at this stage because they clash with all but one session. The process of finding the critical exams is carried out each time after the assignment process is done. By doing this, we will eventually be left with 'less difficult' exams where they can be fitted into 2 or more sessions. The session will be chosen in such a way that the minimum number of sessions is maintained. This should therefore reduce the possibility of having to do any backtracking process at later stages and also having to unnecessarily increase the number of sessions required to assign all exams without creating any conflict.

The rule has also adopted the clique initialisation strategy as it has proved to be a significant contribution in producing better results. The backtracking process can be laborious at times where there are a lot of exams to be scheduled and also when there are many exams in conflict with each other. Therefore the new rule is designed to avoid having to do any backtracking. The new rule can be divided into three parts; finding maximum cliques of the exams, scheduling the critical exams and scheduling the non-critical exams.

5.2.1 Finding Maximum Cliques

The algorithm proposed by Carraghan and Pardalos [1990] is slightly modified in order to produce all possible maximum cliques. It is interesting to see whether different maximum cliques used as a starting point will yield different results and if so, whether one way of determining a maximum clique is better than the others. There are basically two different types of maximum clique found in a graph; Type I maximum cliques where there is at least one exam in common between different maximum cliques and Type II maximum cliques where the exams in the cliques are totally different.

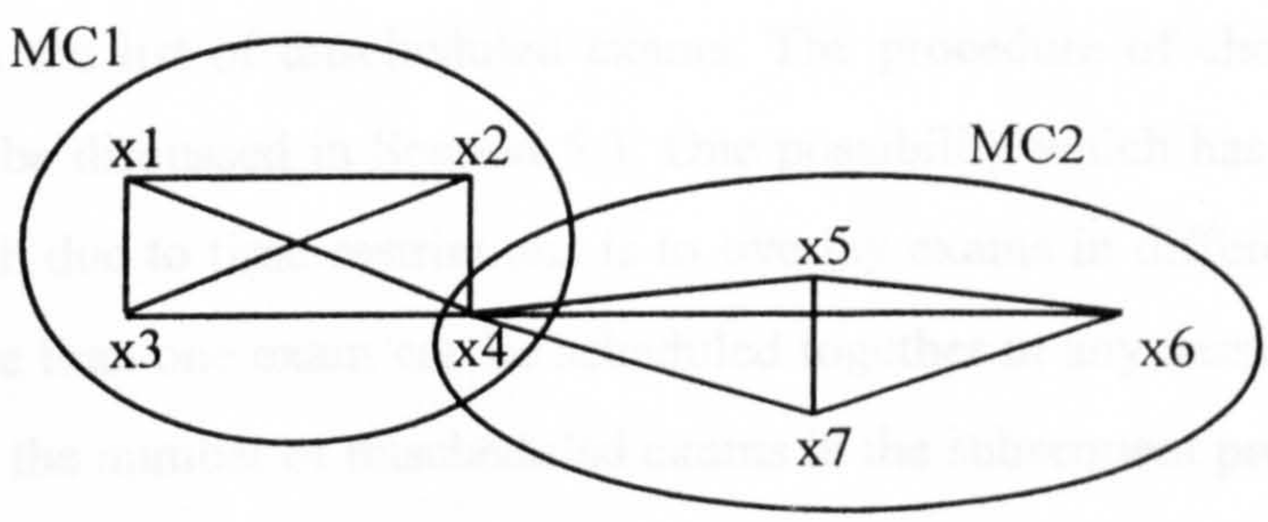


Figure 5.1 : Type I maximum cliques

In Figure 5.1, there are two maximum cliques of Type I. MC1 is the first maximum clique represented by {x1,x2,x3,x4} and the second maximum clique contains {x4,x5,x6,x7}, where x4 is the common exam between MC1 and MC2. Both MC1 and MC2 are of size 4. Figure 5.2 represents the Type II maximum cliques where MC1 and MC2 have different exams, {x1,x2,x3,x4} and {x5,x6,x7,x8} respectively. Therefore, in these two cases, there will be a choice between MC1 and MC2 to be used as a starting point.

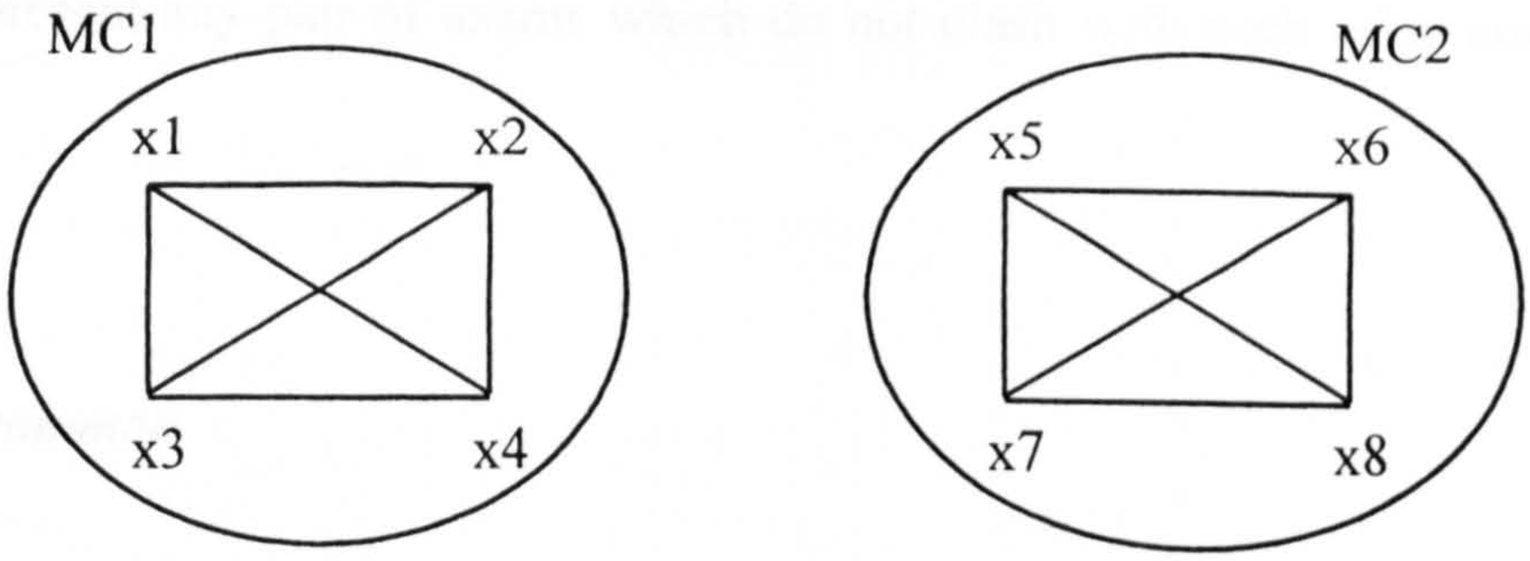


Figure 5.2 : Type II maximum cliques

At this stage of the research, only one of the several maximum cliques found will be used as a starting point. The examinations occurring in other maximum cliques will be put back in the list of unscheduled exams. The procedure of choosing this maximum clique will be discussed in Section 5.3. One possibility which has not been analysed in this research due to time restrictions is to overlay exams in different maximum cliques so that more than one exam can be scheduled together in any session. This would result in reducing the number of unscheduled exams in the subsequent process.

The question now is which of the types of maximum cliques, Type I with 1, 2 or more exams in common or Type II maximum cliques, has the largest chance of matching. If we know which type of maximum cliques can be overlaid, then we can choose the appropriate maximum cliques to be used as a starting point. An example is given to discuss this idea in more detail. Consider two different maximum cliques of size 3 with three possible outcomes:

i) there is no exam in common
ii) there is 1 exam in common
iii) there are 2 exams in common

In each case, let

p = probability that any pair of exams clash

and let '0' represent any pair of exams which do not clash with each other and '1' if they do.

i) 0 exams in common

Figure 5.3 shows the two maximum cliques with no exams in common. Therefore there are 6 possible combinations of exams which do not clash with each other in order to overlay the exams in both maximum cliques. They are:

[x1,y1],[x2,y2],[x3,y3]	[x1,y1],[x2,y3],[x3,y2]
[x1,y2],[x2,y1],[x3,y3]	[x1,y2],[x2,y3],[x3,y1]
[x1,y3],[x2,y2],[x3,y1]	[x1,y3],[x2,y1],[x3,y2]

	y1	y2	y3
x1			
x2			
x3			

Figure 5.3 : Two maximum cliques of size 3 with 0 common exam

Table 5.1 shows the probability of having 0 to 9 pairs of exams which do not clash with each other and the probability of a match, i.e. one of the six combinations of non-conflicting exams which enable the exams in both cliques to be overlaid, associated with each number of non-conflicting exams. Let P_0 be the probability of overlaying exams in two maximum cliques with no exams in common. Therefore,

$$P_0 = {}^9C_3p^6(1-p)^3*6/{}^9C_3 + {}^9C_4p^5(1-p)^4*[6*{}^6C_1]/{}^9C_4 + {}^9C_5p^4(1-p)^5*[6*{}^6C_2-9]/{}^9C_5 + {}^9C_6p^3(1-p)^6*(1-6/{}^9C_3) + {}^9C_7p^2(1-p)^7 + {}^9C_8p(1-p)^8 + (1-p)^9$$

No. of '0'	Prob. of pair of exams which do not clash	Prob. of a match
0	p^9	0
1	${}^9C_1p^8(1-p)$	0
2	${}^9C_2p^7(1-p)^2$	0
3	${}^9C_3p^6(1-p)^3$	$6/{}^9C_3$
4	${}^9C_4p^5(1-p)^4$	$[6*{}^6C_1]/{}^9C_4$
5	${}^9C_5p^4(1-p)^5$	$[6*{}^6C_2-9]/{}^9C_5$
6	${}^9C_6p^3(1-p)^6$	$1-6/{}^9C_3$
7	${}^9C_7p^2(1-p)^7$	1
8	${}^9C_8p(1-p)^8$	1
9	$(1-p)^9$	1

Table 5.1 : Probability of overlaying exams in two different maximum cliques with 0 exams in common

ii) 1 exam in common

In Figure 5.4, exam x1 is the common exam between the two maximum cliques. The two cliques will match if either one of two possible combinations of non-conflicting exams occur. The combinations of the exams are either {[x2,y2],[x3,y3]} or {[x2,y3],[x3,y2]}.

	x1	y2	y3
x1	0		
x2			
x3			

Figure 5.4 : Two maximum cliques of size 3 with 1 common exam

The probabilities of having from none of the pairs which can be scheduled together to all pairs of exams do not clash with each other together with the probability of a match are shown in Table 5.2. Let P_1 be the probability of overlaying exams in two different maximum cliques with 1 exam in common, then

$$P_1 = {}^4C_2p^2(1-p)^2*2/{}^4C_2 + {}^4C_3p(1-p)^3 + (1-p)^4$$

No. of '0'	Prob. of pair of exams which do not clash	Prob. of a match
0	p^4	0
1	${}^4C_1p^3(1-p)$	0
2	${}^4C_2p^2(1-p)^2$	$2/{}^4C_2$
3	${}^4C_3p(1-p)^3$	1
4	$(1-p)^4$	1

Table 5.2 : Probability of overlaying exams in two different maximum cliques with 1 exam in common

iii) 2 exams in common

Figure 5.5 depicts the two maximum cliques having two exams in common, exams x1 and x2. In order to overlay the exams in both cliques, exam x3 must not clash with exam y3. Therefore, the probability of having the pair [x3,y3] scheduled together is

$$P_2 = 1-p$$

	x1	x2	y3
x1	0		
x2		0	
x3			

Figure 5.5 : Two maximum cliques of size 3 with 2 common exams

No. of '0'	Prob. of pair of exams which do not clash	Prob. of a match
0	p	0
1	1-p	1

Table 5.3 : Probability of overlaying exams in two different maximum cliques with 2 exams in common

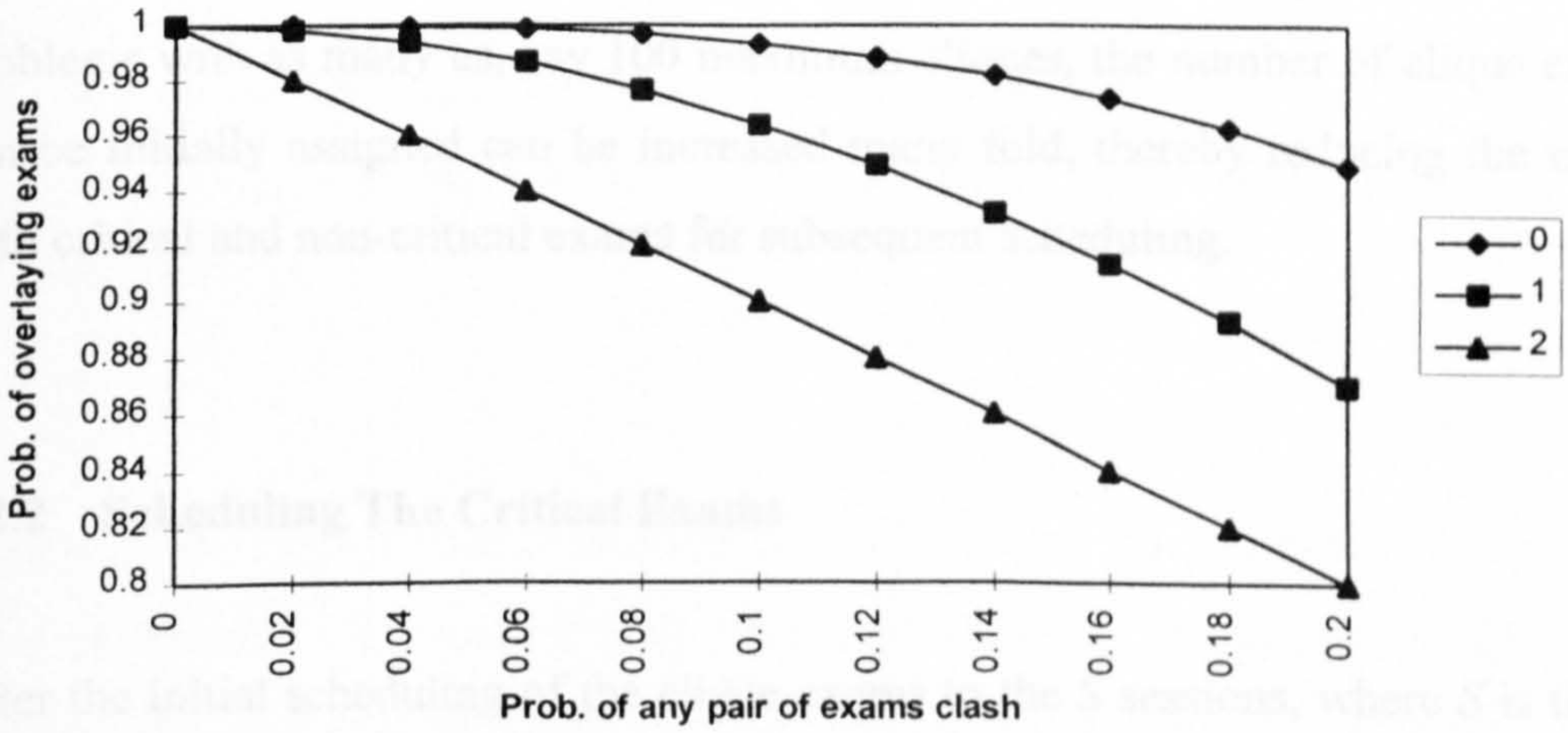


Figure 5.6 : The probability of overlaying exams in two different maximum cliques

The graph of the probability of overlaying exams in two different maximum cliques for various values of 'p' when there are no exams in common, 1 exam in common and 2 exams in common, is shown in Figure 5.6. The graph shows that the higher the number of common exams found in the two maximum cliques, the lower the probability of overlaying the exams in both cliques. Therefore, it is best to have two maximum cliques with totally different sets of exams. In other words, we are more likely to be able to overlay exams in two maximum cliques if we have the Type II maximum cliques. This is however, the case for overlaying exams in only two maximum cliques. The problem becomes more complicated and complex if there are more than two maximum cliques found in the graph.

On the evidence of this example, it seems reasonable to assume that the probability of being able to overlay two maximum cliques is greater if the cliques are larger and if they have fewer exams in common. If there are a large number of reasonably disjoint maximum cliques, it seems that there will be a high probability that two or more of these cliques can be assigned at the outset.

We propose to do more work to examine this situation and to develop algorithms for selecting the cliques that can be overlaid in this way. It is our expectation that for large problems with as many as, say 100 maximum cliques, the number of clique exams that can be initially assigned can be increased many fold, thereby reducing the number of both critical and non-critical exams for subsequent scheduling.

5.2.2 Scheduling The Critical Exams

After the initial scheduling of the clique exams to the S sessions, where S is the size of the maximum clique, the process will proceed to find the critical exams. There would be two possible outcomes from the process of finding the critical exams. The first possibility is that there is only one critical exam which can be assigned to any one of the S sessions, and the second possibility is where there is more than one critical exam in

any session. In the first case, there would be no problem in scheduling these critical exams as each of them can be assigned to their respective sessions.

However, the critical exams found in the second case cannot be directly assigned to their respective sessions since they might be in conflict with each other. To solve this particular problem, we introduce the concept of finding a maximum sub-clique (MAXSC). $MAXSC_i$ is the maximum clique found among the critical exams associated with session i. Again there could be two possible outcomes from the search:

1. $\max_{i \in S} \{MAXSC_i\} = k > 1$, where S is the set of sessions, and $MAXSC_i$ is unique. In this particular session, there would be k exams which are in conflict with each other so we need to create (k-1) new sessions so that one of the critical exams can be assigned to this session and the remaining (k-1) critical exams can be assigned to the new sessions in order to avoid clashes. We then sort these critical exams in decreasing order of degree and assign each of the critical exams to the sessions respectively. The remaining critical exams found are put back in the unscheduled list.

2. $\max_{i \in S} \{MAXSC_i\} = k > 1$ where S is the set of sessions, and $MAXSC_i$ is not unique. Here we choose from the sessions with maximum MAXSC the one with the largest sum of degree among the critical exams and assign the critical exams in the same way as in (1).

In case 2, the largest sum of degrees is used to break the ties between critical exams. This is because after testing with other rules such as the recursive largest sum of degrees, the least sum of degrees and the recursive least sum of degrees, we found that the recursive sum of degrees yields the best final results, i.e. in terms of the minimum number of sessions required to assign all exams without creating any conflicts.

The flowchart of finding the critical exams in the graph is shown in Figure 5.7.

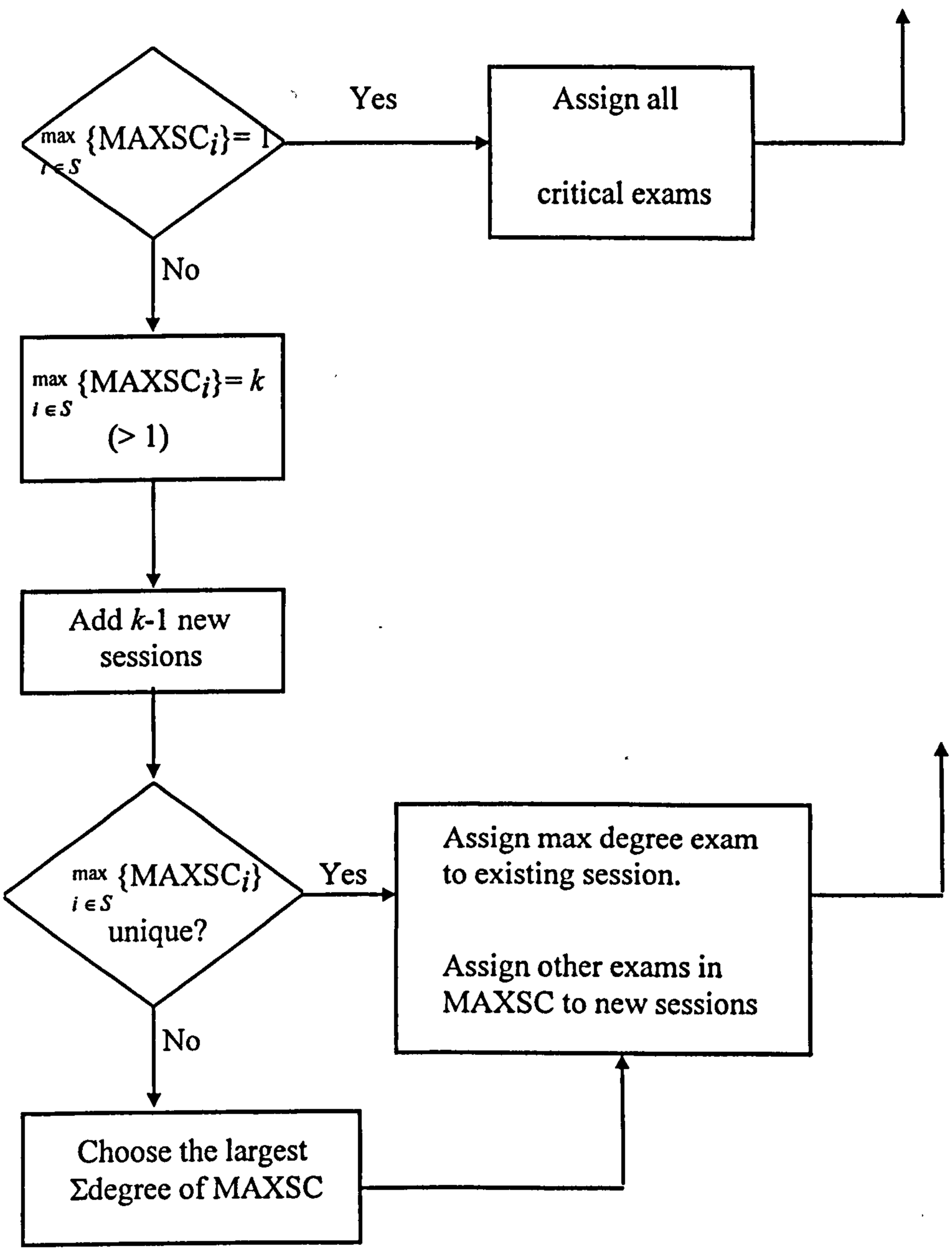


Figure 5.7 : Flowchart of scheduling critical exams

5.2.3 Scheduling The Non-critical Exams

If we cannot find any critical exam, then the rest of the exams are said to be non-critical. Non-critical exams are those which can be assigned to more than one of the available sessions. These exams are firstly sorted according to some criterion and subsequently assigned to the most appropriate session. We have tested seven different strategies for assigning the non-critical exams.

Strategy 1 - The unscheduled exams are ordered according to largest degree. Choose the first exam from the list which clashes with most sessions. Assign this exam to the lowest numbered clash free session. If it clashes with all of the available sessions, then assign the exam to a new session.

Strategy 2 - The unscheduled exams are ordered according to recursive largest degree. Choose the exam at the top of the list. Assign this exam to the lowest numbered clash free session. If it clashes with all of the available sessions, then assign it to a new session.

Strategy 3 - The unscheduled exams are ordered according to recursive largest degree. Choose the first exam from the list which clashes with most sessions. Assign this exam to the lowest numbered clash free session. If it clashes with all of the available sessions, then assign it to a new session.

Strategy 4 - The unscheduled exams are ordered according to recursive largest degree. Choose the first exam from the list which clashes with most sessions. Assign this exam to the lowest numbered clash free session which clashes with most of the remaining unscheduled exams. If it clashes with all of the available sessions, then assign it to a new session.

Strategy 5 - The unscheduled exams are ordered according to recursive largest degree. Choose the first exam from the list which clashes with most sessions.

Assign this exam to the lowest numbered clash free session which clashes with the fewest number of the remaining unscheduled exams. If it clashes with all of the available sessions, then assign it to a new session.

Strategy 6 - The unscheduled exams are ordered according to recursive largest degree. Choose the first exam from the list which clashes with most sessions. Assign this exam to the highest numbered clash free session which clashes with most of the remaining unscheduled exams. If it clashes with all of the available sessions, then assign it to a new session.

Strategy 7 - The unscheduled exams are ordered according to recursive largest degree. Choose the first exam from the list which clashes with most sessions. Assign this exam to the highest numbered clash free session which clashes with the fewest number of the remaining unscheduled exams. If it clashes with all of the available sessions, then assign it to a new session.

The flowchart of the whole process can be seen in Figure 5.8.

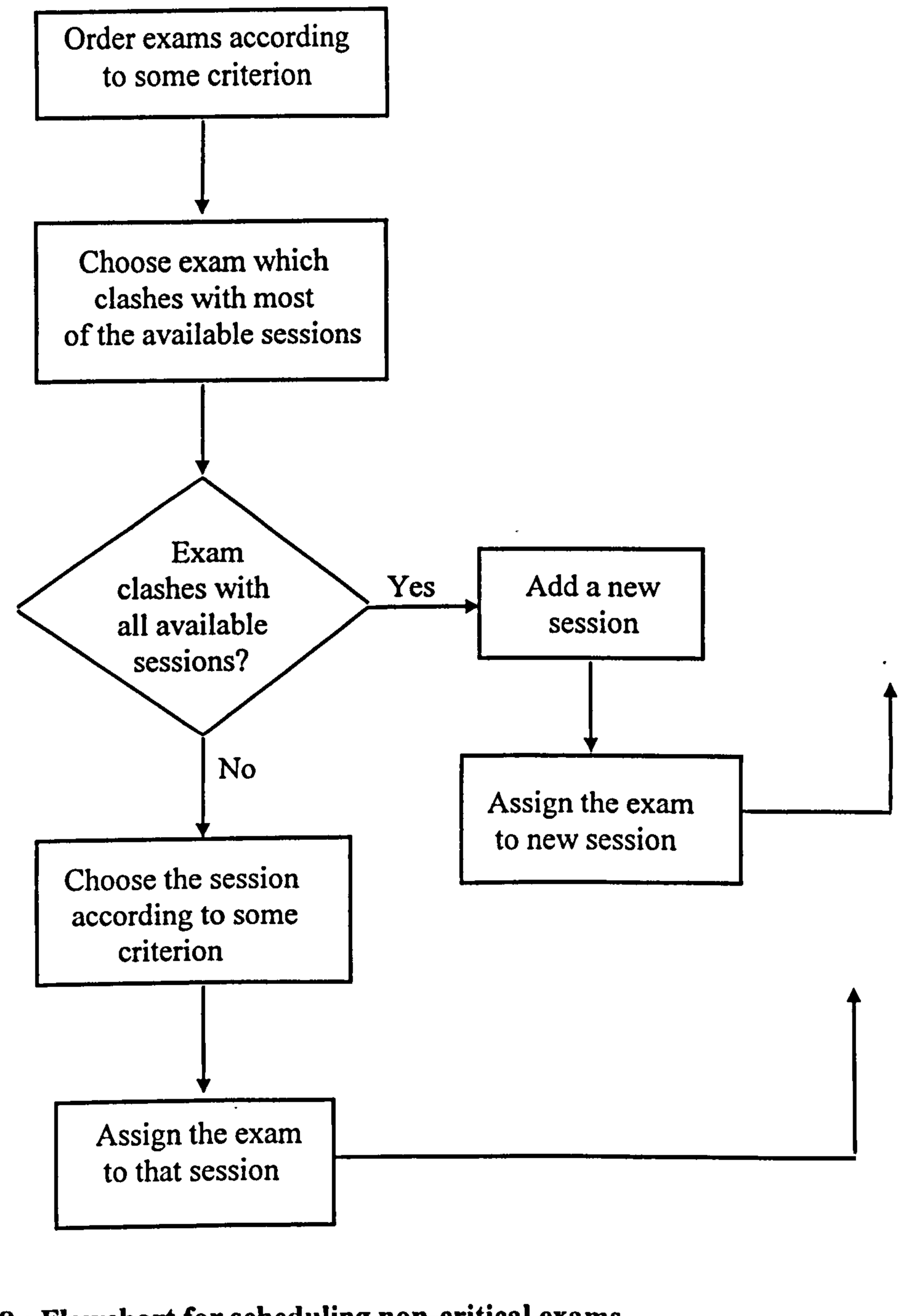


Figure 5.8 : Flowchart for scheduling non-critical exams

5.3 Numerical Results

Tests were carried out using the new algorithmic rule with the seven strategies for assigning non-critical exams. Table 5.4 shows the number of sessions obtained and the clique number used as a starting point for each of the eleven data sets.

		S1	S2	S3	S4	S5	S6	S7
HEC (1)	Session	17	18	17	18	18	17	18
	Clique No.	1	1	1	1	1	1	1
STA (60)	Session	13	13	13	13	13	13	13
	Clique No.	1	13	1	1	1	1	1
YOR (32)	Session	19	19	19	20	21	20	21
	Clique No.	3	20	2	1	1	1	1
EAR (7)	Session	23	22	23	23	23	23	23
	Clique No.	1	2	1	1	1	1	1
LSE (2)	Session	17	18	17	17	17	17	17
	Clique No.	1	1	1	1	1	1	1
UTE (4)	Session	10	10	10	10	10	10	10
	Clique No.	1	1	1	1	1	1	1
TRE (4)	Session	20	20	20	20	20	20	20
	Clique No.	2	1	2	1	1	1	1
KFU (2)	Session	21	21	22	19	23	19	23
	Clique No.	1	1	1	1	1	1	1
CAF (3)	Session	29	29	29	29	29	29	29
	Clique No.	1	1	1	1	1	1	1
UTA (128)	Session	31	31	31	32	32	32	32
	Clique No.	16	53	5	1	3	1	3
CAS (156)	Session	30	30	31	30	32	30	32
	Clique No.	1	7	7	1	1	1	1

(Note : The number in the bracket is the number of maximum cliques found).

Table 5.4 : Computational results using the new algorithmic rule

The general conclusion from these results is that in most instances if the first maximum clique is used as a starting point, it produces the least number of sessions. The first clique is that found by the Carraghan and Pardalos algorithm where the exams are initially ordered recursively according to increasing number of degree. They are also found to have the lowest total degree. In cases such as YOR, EAR and UTA, the best results are obtained by other than the first maximum clique. At this stage, there does not

appear to be any explanation why this particular clique produced the minimum number of sessions. They do not seem to have any special characteristics for identifying them. Therefore, it is decided to use the first maximum clique will be used for all subsequent work. A summary of the different strategies which obtained the minimum number of sessions for each data set is shown in Table 5.5.

Data	Size of maximum clique	Minimum number of sessions obtained	Achieved by
HEC	17	17	S1,S3,S6
STA	13	13	S1,S2,S3,S4,S5,S6,S7
YOR	18	19	S1,S2,S3
EAR	21	22	S2
LSE	17	17	S1,S3,S4,S5,S6,S7
UTE	10	10	S1,S2,S3,S4,S5,S6,S7
TRE	20	20	S1,S2,S3,S4,S5,S6,S7
KFU	19	19	S4,S6
CAF	24	29	S1,S2,S3,S4,S5,S6,S7
UTA	26	31	S1,S2,S3
CAS	23	30	S1,S2,S4,S6

Table 5.5 : Strategies used for obtaining minimum number of sessions

Comparing the minimum number of sessions with the size of the maximum clique for each data set, we can see that the number of sessions obtained reached the lowest numbers possible, i.e. the lower bound, for most medium sized problems. However, it is generally not possible to achieve this lower bound for the minimum number of sessions in large scale problems. To achieve the lowest number of sessions possible, Strategy 1 seems to dominate the others with nine best results while Strategy 2, Strategy 3 and Strategy 6 each have eight (refer to Table 5.6). However, since we have decided to use the first maximum clique as our starting point, then the most suitable strategy to adopt would seem to be Strategy 6. In Strategy 6, non-critical exams are initially ordered according to recursive largest degree and the exam chosen from the list is the one that

clashes with most sessions. From the sessions to which this exam can be assigned, choose the highest numbered session whose exams clash with most of the remaining unscheduled exams. In this way, we are giving the remaining unscheduled exams plenty of leeway to be assigned to the available sessions.

Strategy	No. of best results	No. of best results starting from the first maximum clique
S1	9	6
S2	8	3
S3	8	5
S4	7	7
S5	5	5
S6	8	8
S7	5	5

Table 5.6 : Summary of the results for each strategy

To analyse the performance of the new algorithmic rule (incorporating Strategy 6), we compared these findings with the results obtained by Carter *et al.* (shown in Table 4.3 in Section 4.4). Figure 5.9 below gives a graphical representation of the performances of Carter's method and the new method in producing the minimum number of sessions. The new algorithmic rule with Strategy 6 managed to produce the same minimum number of sessions in 8 cases, seven in medium scale problems and just one of the large scale problems (UTA). In these three cases where the number of sessions was different, the number of sessions was increased by at most 2 sessions (in CAS). Therefore, it can be concluded that using the clique with the lowest sum of degrees as a starting point and incorporating Strategy 6 for assigning non-critical exams can usually produce the minimum number of sessions without having to use a backtracking process.

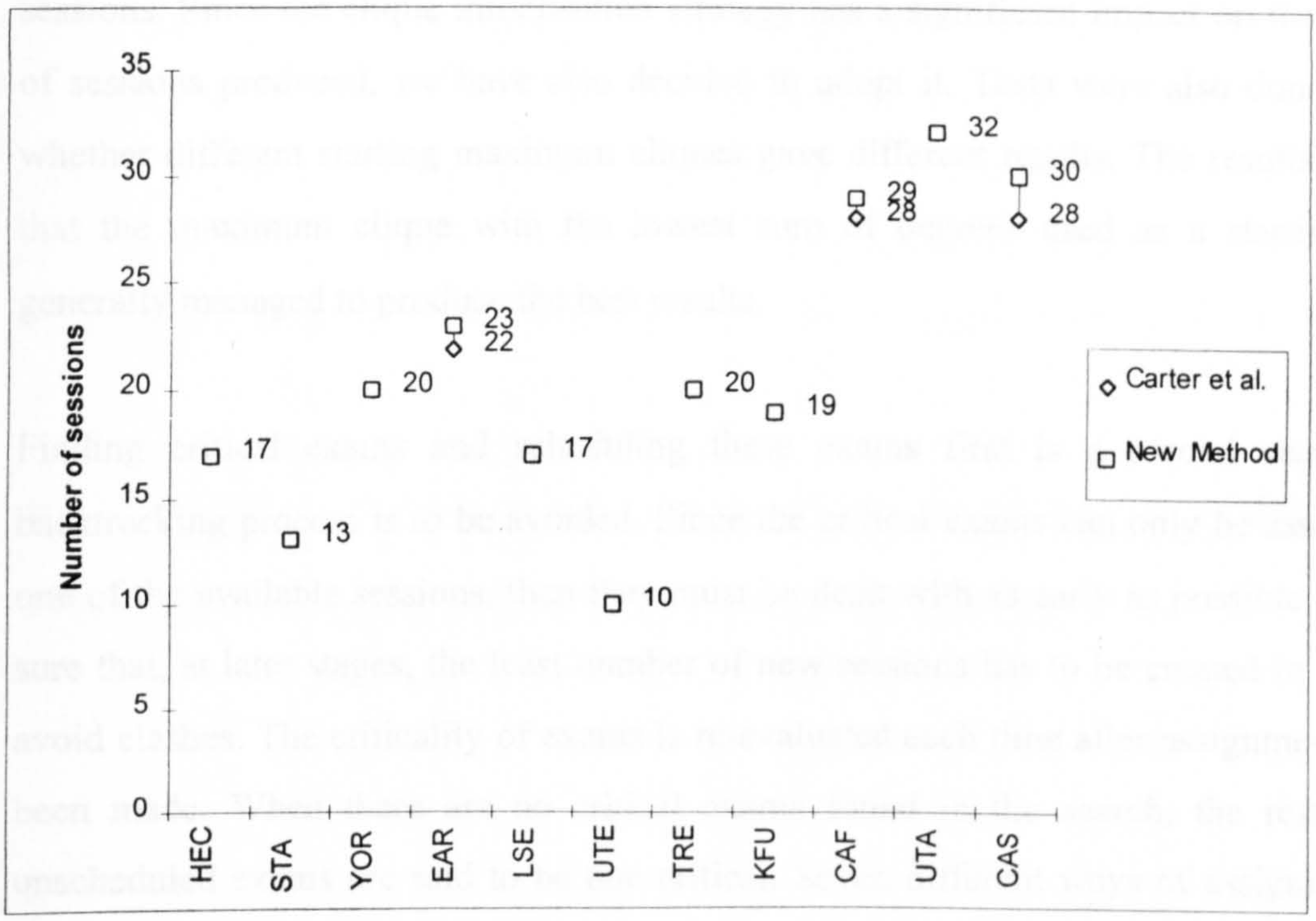


Figure 5.9 : Performances of Carter's method and the new method

5.4 Summary

The objective of the work in this chapter has been to investigate the performance of a new algorithmic rule compared to the methods used by Carter *et al.* [1996] which incorporates graph colouring methods for the initial ordering of exams, a clique initialisation strategy as a starting point and finally a backtracking process. In their study, the Saturation Degree strategy (SD) performed very well in producing the minimum number of sessions possible (equal to the size of the maximum clique) for about half of the problems. In most cases, however, this was after a backtracking process had been used.

Therefore, the new algorithmic rule was developed in order to examine whether the backtracking process could be eliminated but still produce the minimum number of

sessions. Since the clique initialisation strategy has a significant impact on the number of sessions produced, we have also decided to adopt it. Tests were also done to find whether different starting maximum cliques gave different results. The results showed that the maximum clique with the lowest sum of degrees used as a starting point generally managed to produce the best results.

Finding critical exams and scheduling these exams first is a crucial step if the backtracking process is to be avoided. Since the critical exams can only be assigned to one of the available sessions, then they must be dealt with as early as possible to make sure that, at later stages, the least number of new sessions has to be created in order to avoid clashes. The criticality of exams is re-evaluated each time after assignments have been made. When there are no critical exams found in the search, the rest of the unscheduled exams are said to be non-critical. Seven different ways of assigning non-critical exams were explored in this work. The results showed that the strategy which produced the best overall performance is Strategy 6 where the non-critical exams are initially ordered according to recursive largest degree and the exam chosen from the list is the one that clashes with most sessions. Among the sessions to which this exam can be assigned, choose the highest numbered session whose exams clash with most of the remaining unscheduled exams.

The results show that the number of sessions obtained using the new algorithmic rule for medium scale problems is equal or close to the size of the maximum clique (the lower bound to the number of sessions). Nevertheless, the results for large scale problems are not far behind. Therefore, the new algorithmic rule is quite capable of assigning examinations to something close to the minimum number of sessions without employing a time consuming backtracking process.

CHAPTER 6

PROPOSED GENERAL COMPUTERISED UNIVERSITY EXAMINATION TIMETABLING SYSTEM

6.1 Introduction

The basic algorithmic rule discussed in the previous chapter assigns exams to a minimum number of sessions in such a way that no student has to sit more than one exam at any one time. No side constraints are considered at all at this stage. The next stage concentrates on developing a more practical examination timetabling system in universities. The general computerised university examination timetabling system will incorporate some objectives and constraints which have been considered to be important when constructing examination timetables (see the survey in Chapter 4).

The chronological order of the work done on improving the basic algorithmic rule is outlined in this section. Section 6.2 explains how the two side constraints that are mostly considered by examination officers when constructing timetables, namely a time limit on the overall examination period and a maximum number of students that may be examined in any session, are incorporated into the basic algorithmic rule. The third constraint, certain specified exams must take place during the same session, is not considered because there is no information on this in the test data sets, merely the number of students taking each exam. Also, as with previous studies, this constraint can be simply incorporated by considering all exams constrained in this way as forming a

single examination for purposes of scheduling. Section 6.3 presents the process used to minimise the total number of same-day exams, i.e. the number of students having to take more than one exam on the same day. This is however not a very important objective in the survey but we felt that this objective should be included in the general examination timetabling system in common with most previous authors. A method of swapping exams in sessions is introduced to tackle this problem and the results using this method are presented. The last two sections involve post scheduling where exams in each session are assigned to available rooms where larger rooms are filled up as much as possible and then sessions are assigned to the days in the examination period such that sessions containing large exams are scheduled early in the examination period.

We decided not to include two significant objectives; to avoid having exams with different durations in the same room and not to split any exam between two or more rooms. The reason for not including these two objectives in the general examination timetabling system is that we want to maximise the use of all the examination rooms available such that the number of sessions required can be as few as possible.

6.2 Incorporating Side Constraints

Two side constraints, Constraints 1 and 2 as presented in Chapter 4, are incorporated into the basic algorithmic rule. Constraint 1 is where there is a time limit on the overall examination period, i.e. an upper bound to the number of examination sessions, and Constraint 2 is a maximum number of students that may be examined in any session, i.e. the number of students sitting for exams in any session must not exceed the room capacities.

The maximum number of sessions available for assigning the exams can be specified by users. This number, which is the upper bound to the number of sessions must be larger than the size of the maximum clique, i.e. the lower bound to the number of sessions. It

must also be large enough to accommodate all of the non-conflicting exams. If this number is found to be less than the minimum number of sessions, the users will be advised to increase the number of sessions allocated.

To satisfy the second constraint, any exam which can only be assigned to a particular session and which does not exceed the capacity of the session is now considered as a critical exam. The capacity of any session is the total number of seats available in all of the examination rooms. Therefore, if an exam is found to be critical but its size is larger than the remaining capacity of the session, it will be put back into the unscheduled exam list. This will inevitably lead to an increase in the minimum number of sessions needed to schedule all exams.

6.3 Minimising The Total Number Of Same-Day Exams

For the purpose of this research, we have assumed that there are two examination sessions in each day (morning and afternoon sessions). Therefore, students could possibly have two exams in a pair of sessions on the same day, one in the morning session and another in the afternoon session. The number of occurrences of any student having two exams on the same day is the number of same-day exams (SD). The basic algorithmic rule does not take account of the fact that there may be a number of students with this secondary conflict between any pair of sessions.

In our survey, Objective 2 which is not to have students taking more than one exam on the same day does not seem to be significantly important to universities. Out of the 8 objectives that could be taken into account when timetabling examinations that were listed in the survey, this objective only came out sixth. A probable reason for its unpopularity is that the systems currently being used might not be capable of achieving this objective realistically. Most of the universities surveyed are still using manual systems for constructing examination timetables and it was reported to be extremely

difficult and time consuming for examination officers just to avoid first order clashes for students when there was a long clash list without having to consider same-day exams. Quite naturally, many examination officers feel that it is more appropriate to be focused primarily on institutional objectives and constraints rather than student centred ones. On the other hand, we felt that it is only appropriate to try to reduce the number of students having to sit for two exams on the same day, as this is clearly feasible within a computerised system.

The process of minimising the total number of same-day exams is done in two stages. The first stage is the reassignment of exams within the minimum number of sessions and the second stage is the reassignment of exams to available additional sessions.

6.3.1 Reassignment Of Exams Within The Minimum Number Of Sessions

The swapping method proposed will be carried out following the assignment of each exam to its chosen session. To try to reduce the number of students having two exams on the same day, exams in the pair session, i.e. the session on the same day as the chosen session, will be swapped to other sessions. At each stage of the swapping process, it is only necessary to consider those sets of exams which have not previously been paired with the chosen session. This is analogous to a sort of tabu list (see for example Glover [1989]) although the purpose of the list in this case is merely to avoid considering again those sets of exams which cannot lead to any improvement. The iteration process will be carried out a number of times until no more improvements to the number of same-day exams can be found. The swapping rule is as follows:

Step1. Assign the exam c to the chosen session, s_i.

Step2. Identify the pair session s_j, i.e. the session on the same day as the chosen session. Calculate the increase in the number of same-day exams between s_i and s_j (ΔSD_{ij}) when the exam is scheduled.

Step3. Calculate the increase in the number of same-day exams between s_i and s_k (ΔSD_{ik}) when the set of exams in session s_k ($k \neq i,j$) are swapped with the set of exams in the pair session s_j. Calculate also the increase in the number of same-day exams between s_j and s_l (ΔSD_{jl}) where s_l is the pair session of s_k. Negative values denote a decrease in the number of same-day exams.

Step4. If ($\Delta SD_{ik} + \Delta SD_{jl}$) < ΔSD_{ij}, and session s_k produces the least ($\Delta SD_{ik} + \Delta SD_{jl}$) then swap the set of exams x_j with the set of exams x_k, otherwise STOP because no more improvement can be made. Update SD_{ij} and SD_{kl}.

Step5. If the number of iterations has been reached then STOP, otherwise go to Step 2.

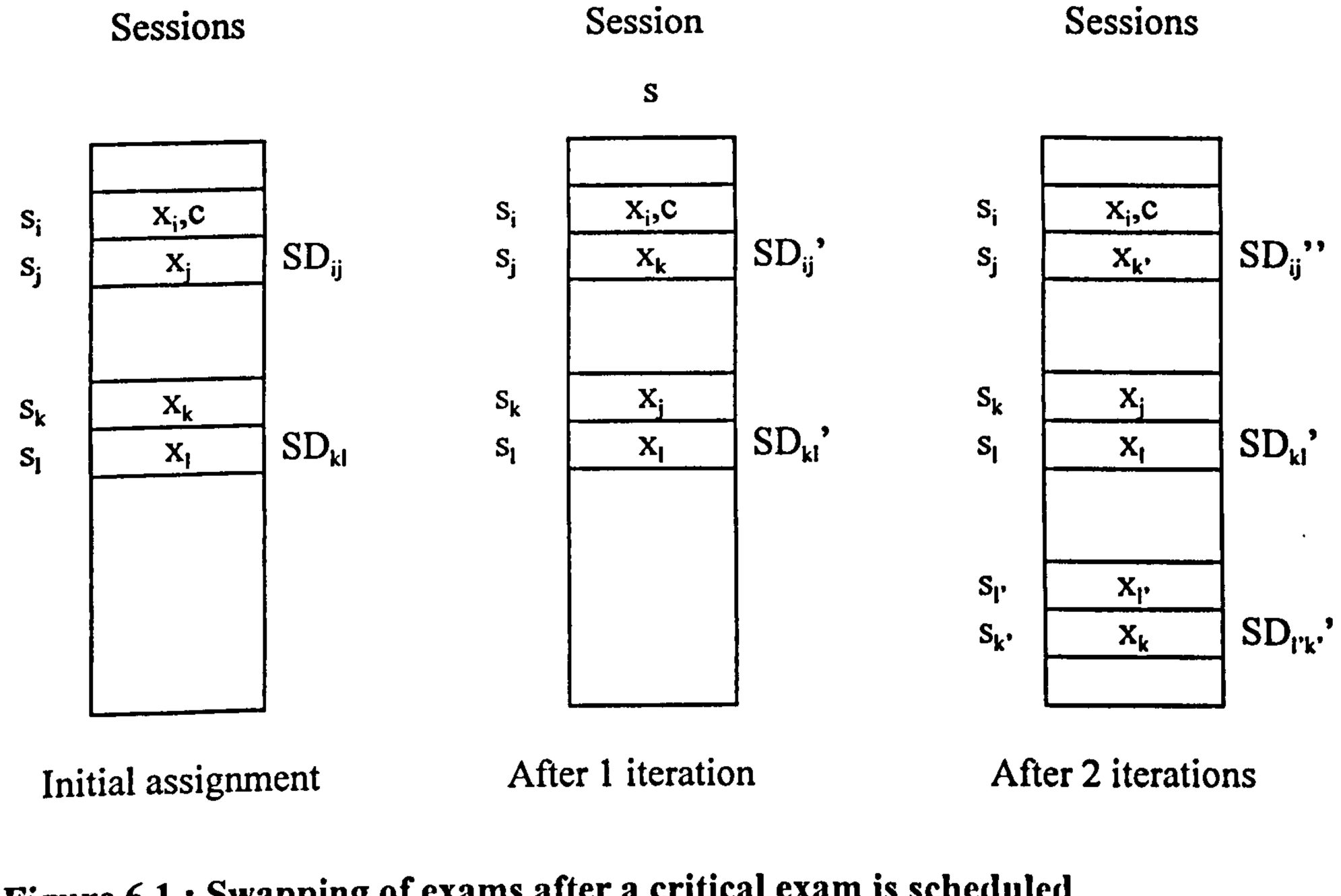


Figure 6.1 : Swapping of exams after a critical exam is scheduled

An illustration of how the swapping rule works is shown in Figure 6.1 above. In this case, sets of exams x_i, x_j, x_k and x_l are initially assigned to sessions s_i, s_j, s_k and s_l respectively. Suppose that a critical exam c, is scheduled to session s_i (chosen session),

together with x_i. If the critical exam c clashes with at least one of the exams in x_j in the pair session s_j, then SD_{ij} will be increased and subsequently the total number of same-day exams.

To try to reduce the increase in the total number of same-day exams, we will explore swapping the set of exams x_j in the pair session s_j, with another set of exams x_k in session s_k. After this swap, suppose that the sum of the increases in SD_{ik} and SD_{jl} is less than the increase in SD_{ij} and also this swap is found to give the least value, swap the set of exams x_j with the set of exams x_k in session s_k. We now have $SD_{ij}' = SD_{ij} + \Delta SD_{ik}$ and $SD_{kl}' = SD_{kl} + \Delta SD_{jl}$. The process is repeated for a number of further iterations until a pre-set maximum number of iterations is reached or no further improvement can be found.

6.3.1.1 Numerical Results

Table 6.1 shows the total number of same-day exams obtained for each of the data sets when using the swapping procedures while assigning exams. The first column of zero iterations is where the swapping procedures are not implemented. This is used as a benchmark to study the effect of swapping procedures on minimising the number of instances of students having to take more than one exam on the same day.

The results clearly show the effect of using the swapping procedures, where the number of same-day exams can be reduced by more than 50% in every case. Experimentation showed that swapping exams in the pair session up to 3 times gives the best results for most data. Therefore, this option will be adopted in the subsequent work.

Data	Number of iterations								
	0	1	2	**3**	4	5	6	7	8
HEC	1001	421	358	**360**	358	358	357	357	362
STA	2621	524	523	**523**	523	523			
YOR	1008	647	636	**636**	647	636	636	636	
EAR	1388	560	581	**560**	578	594	583	566	566
LSE	1240	422	383	**383**	383	383			
UTE	2215	1412	1417	**1412**	1412	1412			
TRE	1190	813	813	**813**	813	813			
KFU	3411	991	991	**983**	983	983			
CAF	3375	1334	1389	**1334**	1338	1338	1369	1338	1338
UTA	2757	1408	1411	**1411**	1411	1411			
CAS	2738	1884	1876	**1876**	1876	1876			

Table 6.1 : Number of same-day exams

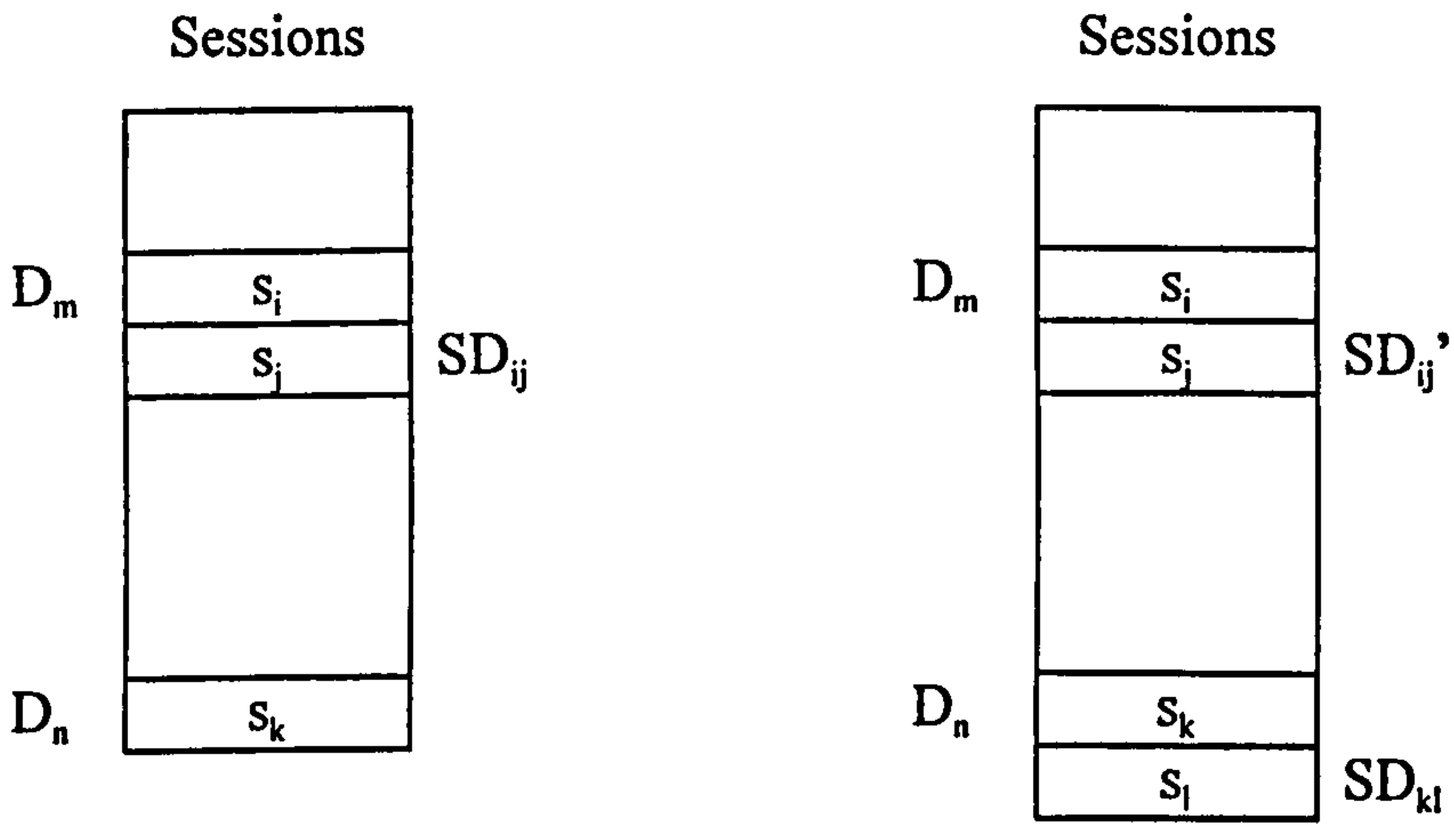
6.3.2 Reassignment Of Exams To Additional Sessions

If the minimum number of sessions obtained using the algorithmic rule is less than the maximum number of sessions specified by the user, then there are, in theory, extra sessions available. In this case, two different approaches for assigning exams to extra sessions are adopted. One is when the minimum number of sessions obtained is odd and the other is when it is even. An odd minimum number of sessions means that the afternoon session on the last day of the examination period is empty. When the minimum number of sessions is even, an additional session would mean increasing the number of exam days.

6.3.2.1 Odd Minimum Number Of Sessions

To make full use of the empty afternoon session and at the same time reduce the total number of same-day exams, exams already assigned to sessions on other days and which do not conflict with each other, are reassigned to the new session. This is an iterative process where exams chosen to be reassigned to the new session are those which have fewer students in conflict with exams in the morning session in the last day of the examination period compared to their existing pair session. These exams must also not exceed the room capacity of the new session. An illustration of the process is shown in Figure 6.2.

In this illustration, D_m is any day of the examination period with the morning session s_i and the afternoon session s_j. Suppose that at the initial schedule, session s_k is the morning session of D_n, the last day of the examination period, where k is an odd number. At this stage, the same-day exam for D_n is zero because the afternoon session is empty. Supposing that an afternoon session, s_l can be added to D_n, by reassigning some of the exams in sessions $s_1,s_2,...,s_i,s_j,...,s_{k-1}$ to the new session, SD_{kl} will be generally increased. To decrease the total same-day exams, we have to make sure that the decrease in the same-day exams where exams have been removed is larger than the increase on the final day. Suppose that when an exam in s_i is reassigned to s_l, $\Delta SD_{ij} > \Delta SD_{kl}$ and $(\Delta SD_{kl} - \Delta SD_{ij})$ is found to be the largest negative value. We now have a reduced value of SD_{ij}' and a new SD_{kl} $(=\Delta SD_{kl})$. Exams which do not exceed the capacity of the session are moved into s_l in this way until no further decrease in the number of same-day exams can be found.



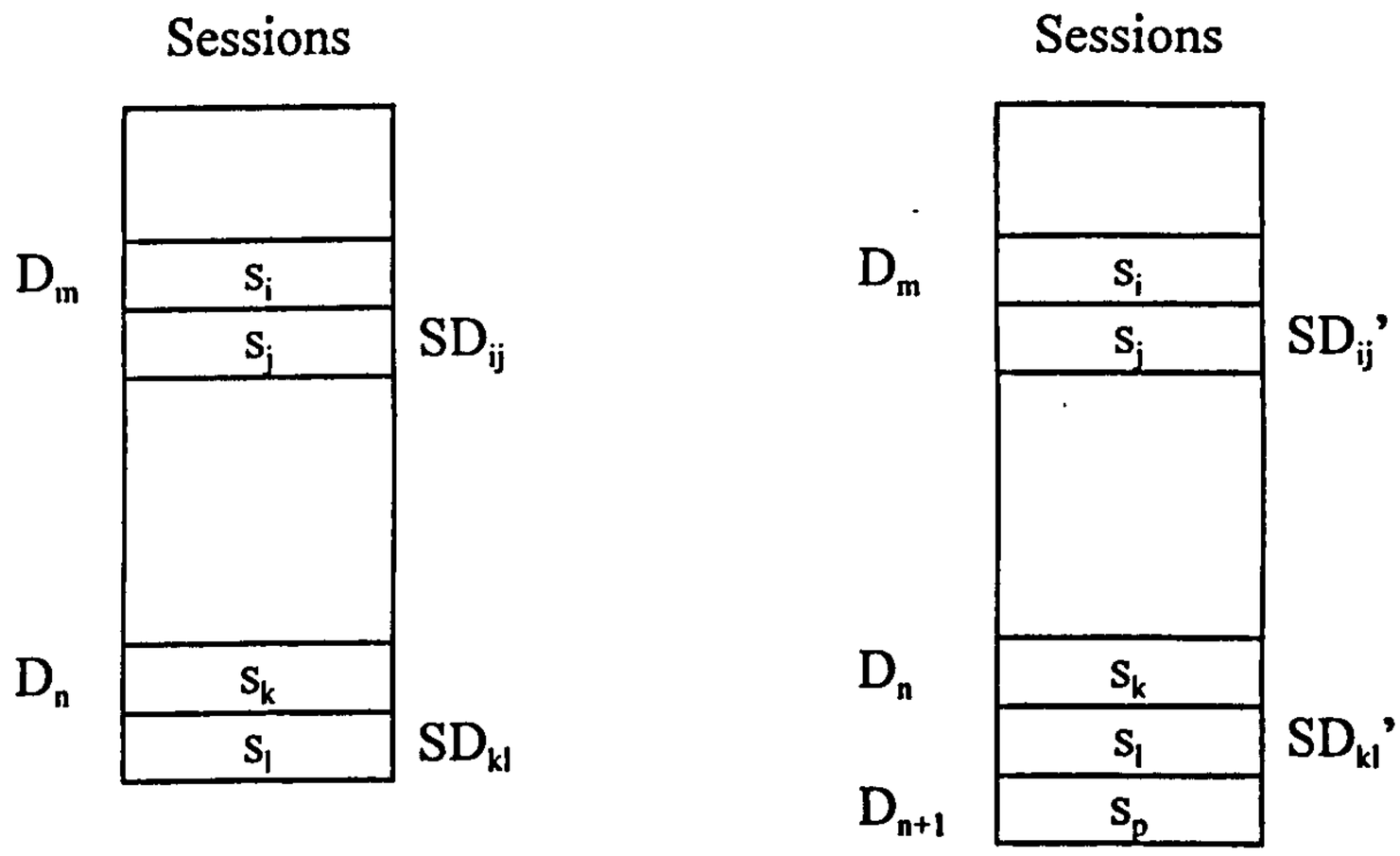
Initial schedule After adding one session

Figure 6.2 : Reassignment of exams to an extra session

6.3.2.2 Even Minimum Number Of Sessions

If we have one more day to spare, the same-day exams could be reduced even more. By adding another session, i.e. a morning session on an extra day, the total number of same-day exams will be reduced by at least the biggest number of same-day exams in any day of the examination period.

The process starts by choosing the day with the largest number of same-day exams. Then all of the exams in one of its sessions will be reassigned to the new session. To choose between the two sessions, we opt for the one with lower total degree. This session will now be empty, so we reassign exams in other sessions which do not clash with each other to the empty session. The exams which can be reassigned to the empty session must produce a maximum net decrease in the total number of same-day exams, and also they must not violate the room capacity constraint on the empty session. There is now an odd number of sessions in total and the procedure described in the previous sub-section is invoked to transfer exams into the afternoon session of the final day.



Initial schedule

After adding one extra day

Figure 6.3 : Reassignment of exams to an extra session in an additional day

The illustration depicted in Figure 6.3 explains the process of reassignment of exams to an extra session on an additional day. We have an even number session s_l on the last day of the examination period, D_n at the initial schedule. Suppose that we have an extra session on D_{n+1} where some exams already assigned to sessions $s_1,s_2,...,s_i,s_j,...,s_k,s_l$ can be rescheduled to the new session s_p. Since the extra session is on D_{n+1}, the total number of same-day exams will not be increased by reassigning these exams. Therefore, to maximise the decrease in the total number of same-day exams, exams in the session with the largest number of same-day exams are reassigned to the new session. Suppose that SD_{ij} is the largest and s_i has a lower total degree than s_j. Therefore, reassign all of the exams in s_i to the new session s_p. The total same-day exams will be reduced by SD_{ij}. Now session s_i is empty, so reassign exams in sessions other than s_j and s_p which produce the maximum net decrease in the total number of same-day exams. The exams which are to be reassigned to session s_i must not clash with each other and also must not exceed the room capacity of session s_i.

6.3.2.3 Numerical Results

Both of the rule-based methods were tested on the eleven data sets. The process of gradually decreasing the total number of same-day exams by adding extra sessions one by one is carried out. The graphical representations of the decrease in the total number of same-day exams when extra sessions are added to the minimum sessions obtained for each of the data sets, are shown in Figures 6.4 to Figure 6.14. Unfortunately, the data provided by Carter *et al.* [1996] do not include any information about the maximum number of sessions allowable. Therefore, we increased the maximum number of sessions up to twice the minimum number of sessions at which point the total number of same-day exams should be reduced to zero.

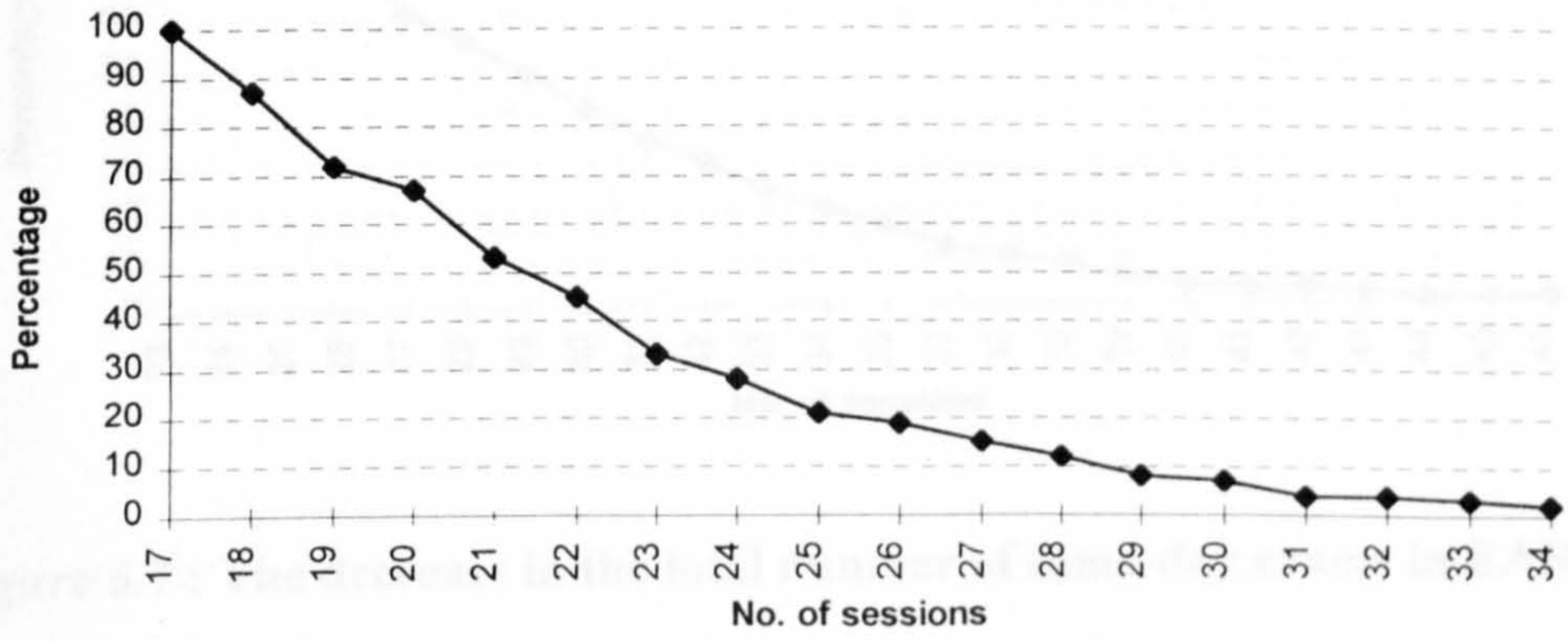


Figure 6.4 : The decrease in the total number of same-day exams in HEC

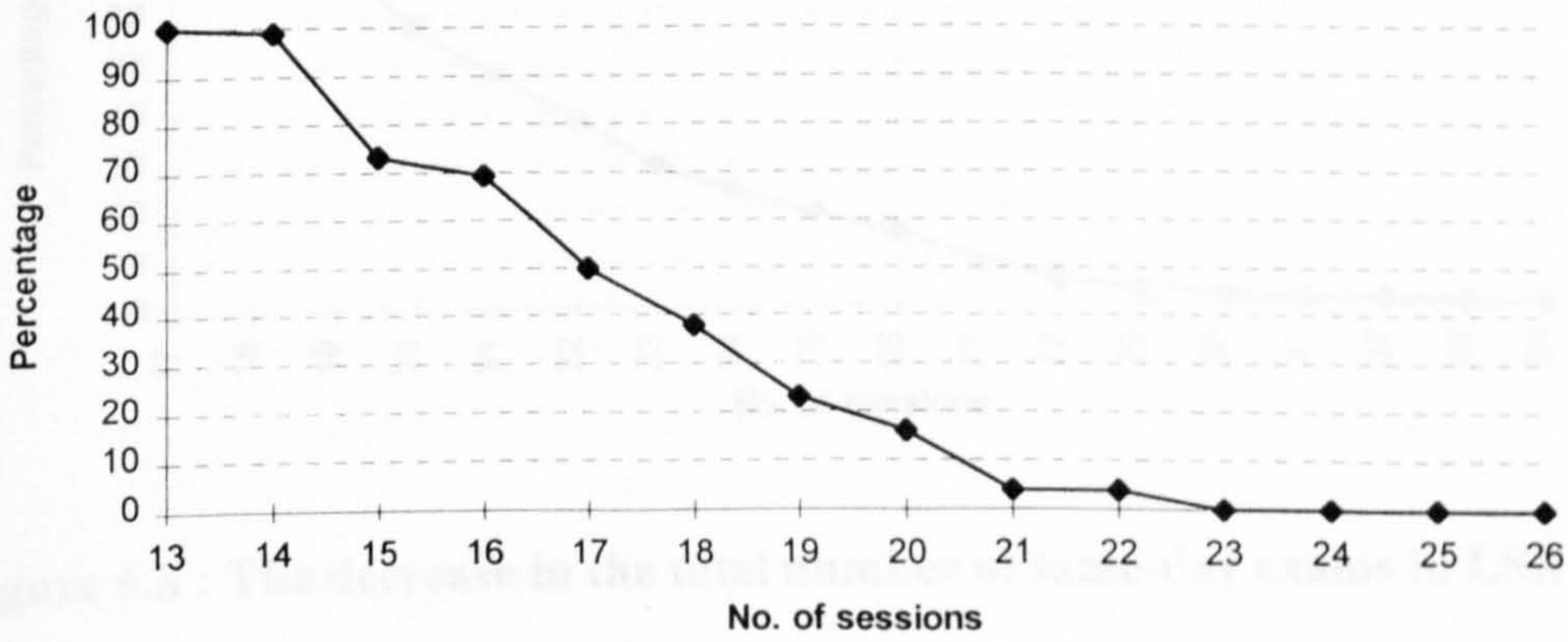


Figure 6.5 : The decrease in the total number of same-day exams in STA

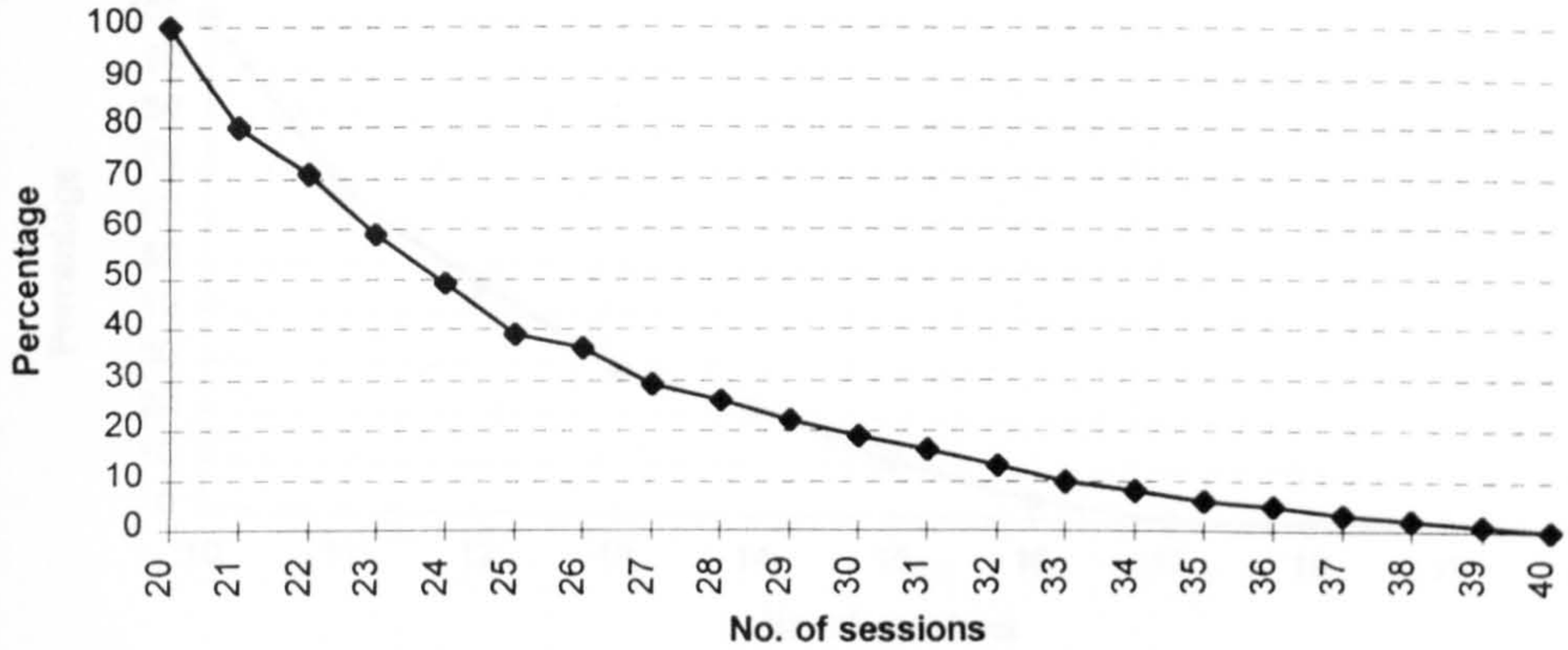


Figure 6.6 : The decrease in the total number of same-day exams in YOR

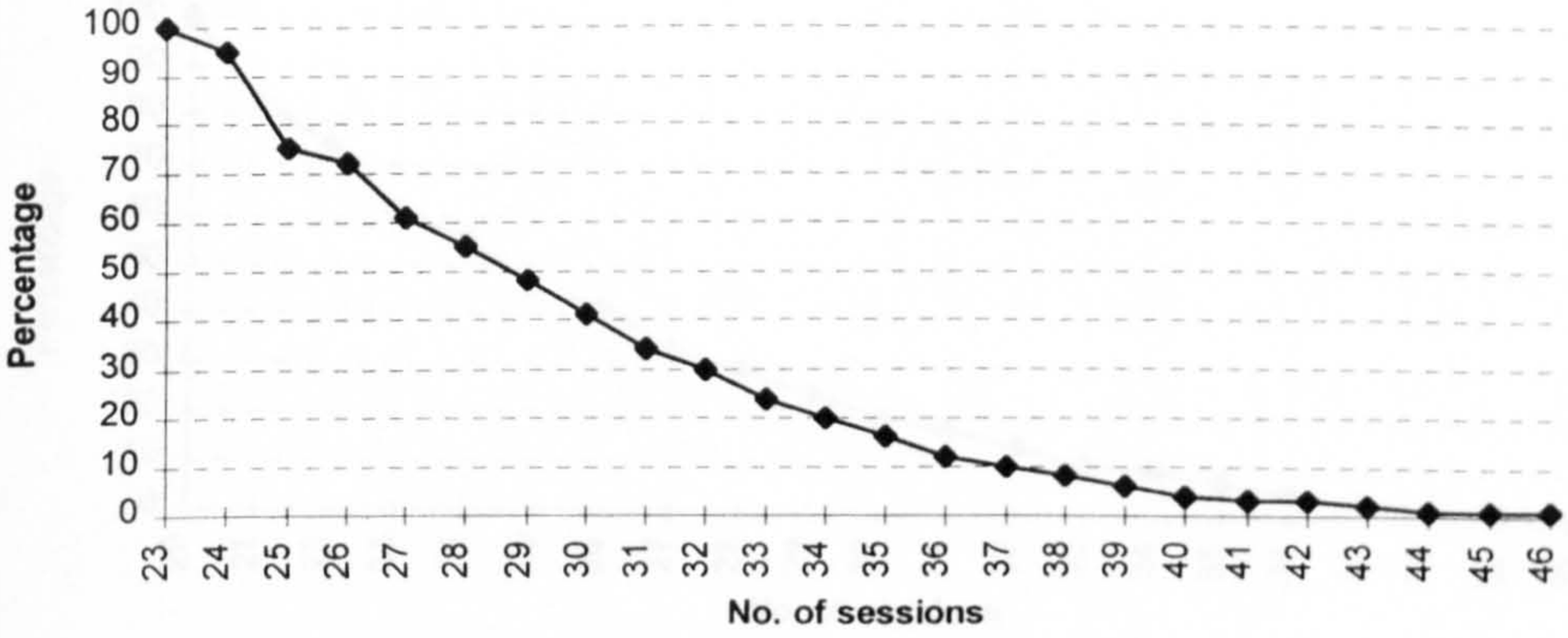


Figure 6.7 : The decrease in the total number of same-day exams in EAR

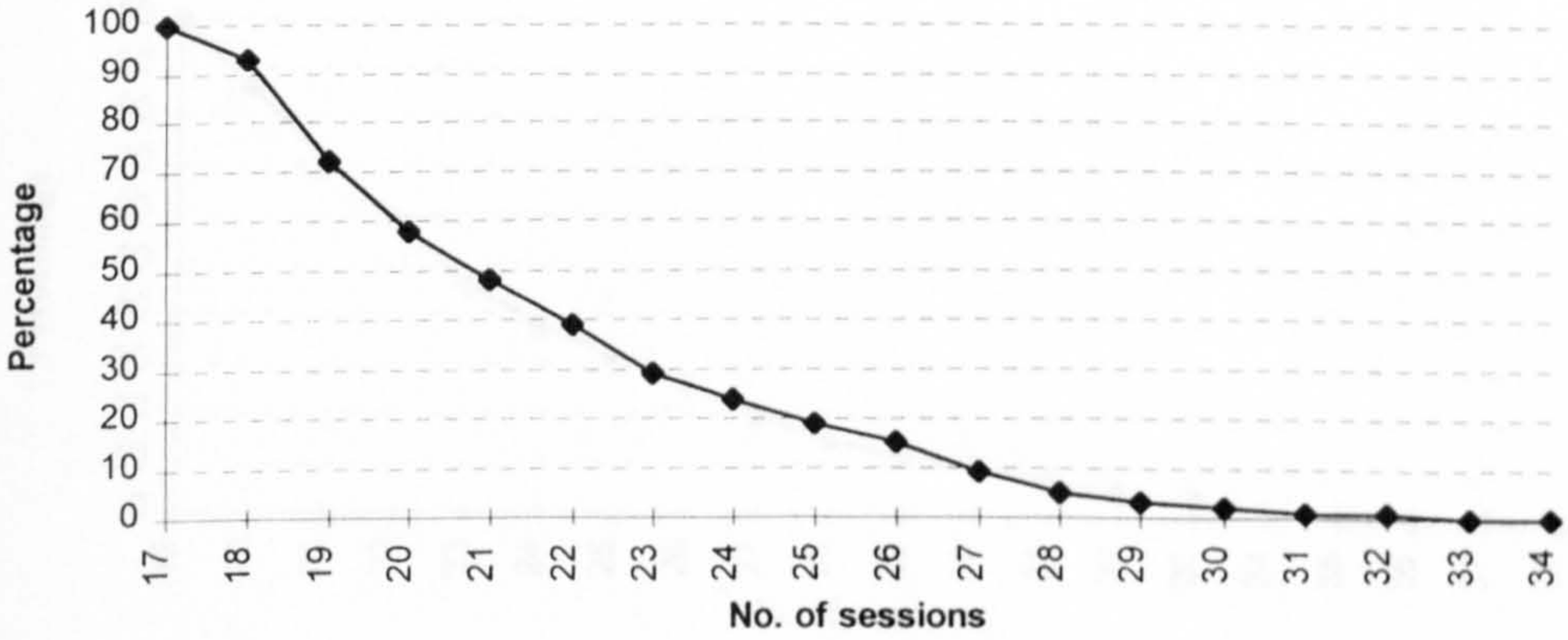


Figure 6.8 : The decrease in the total number of same-day exams in LSE

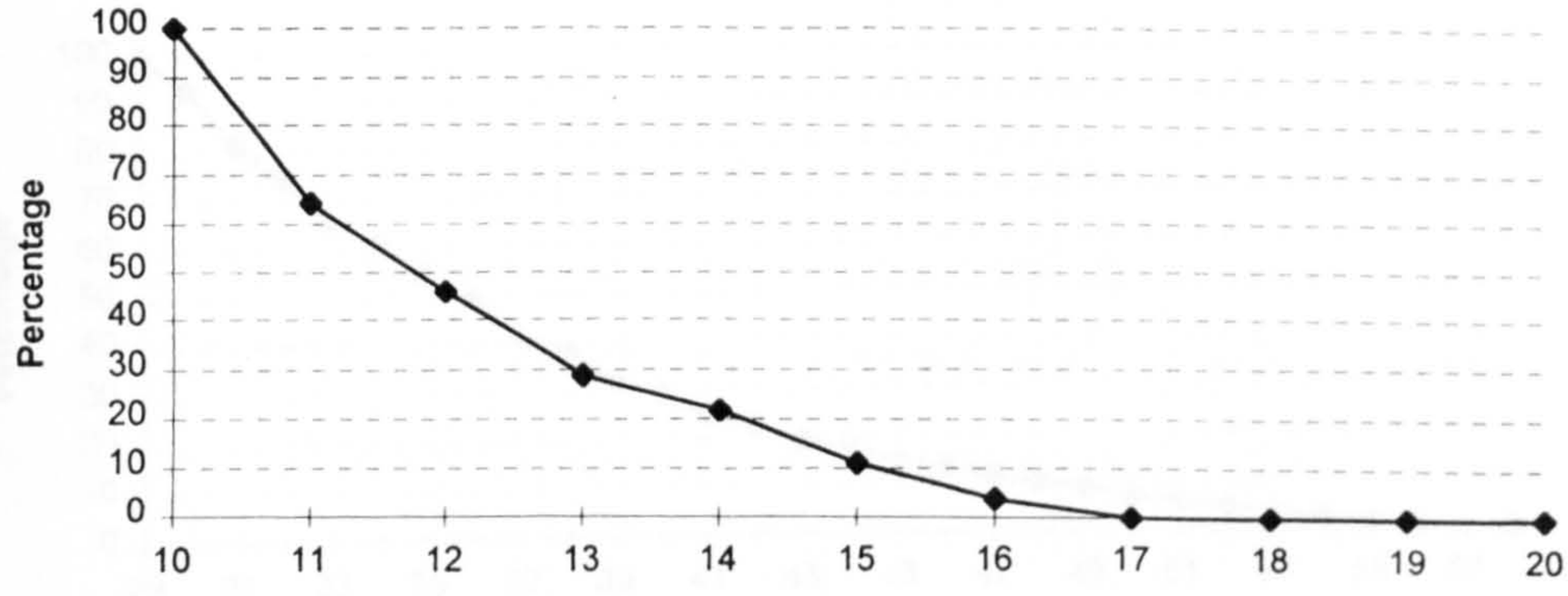


Figure 6.9 : The decrease in the total number of same-day exams in UTE

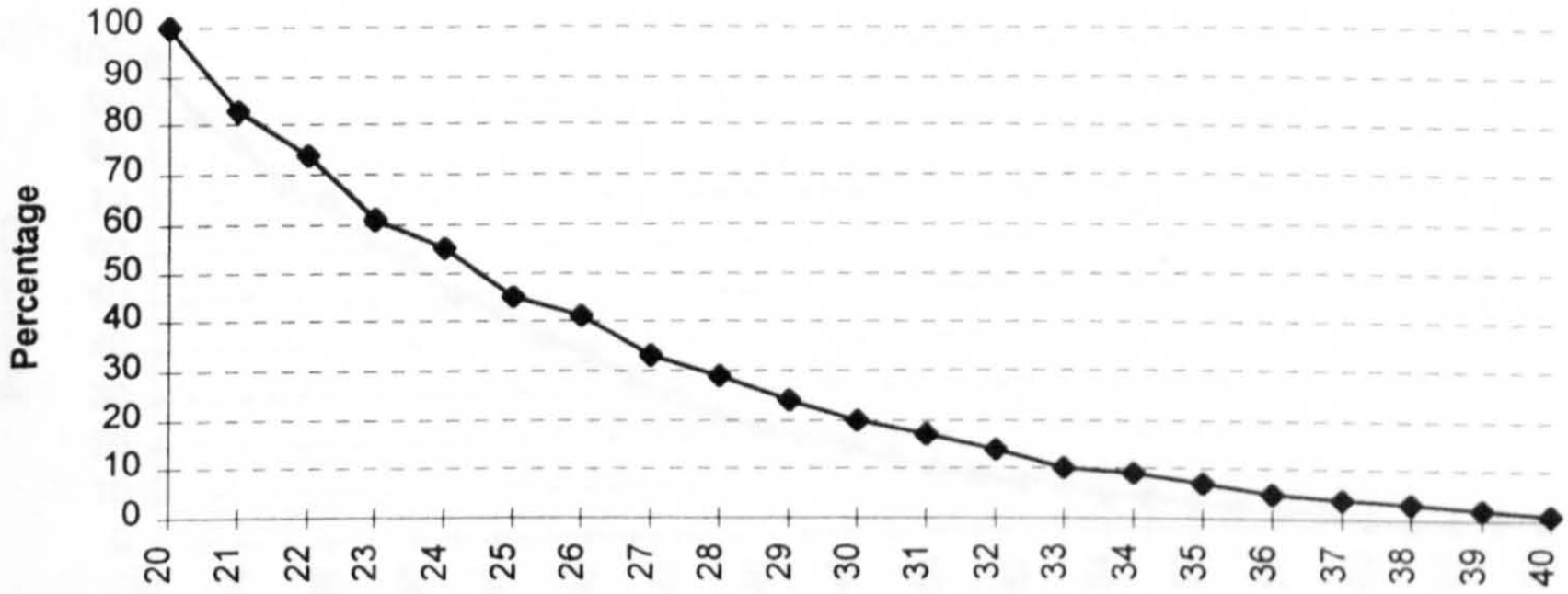


Figure 6.10 : The decrease in the total number of same-day exams in TRE

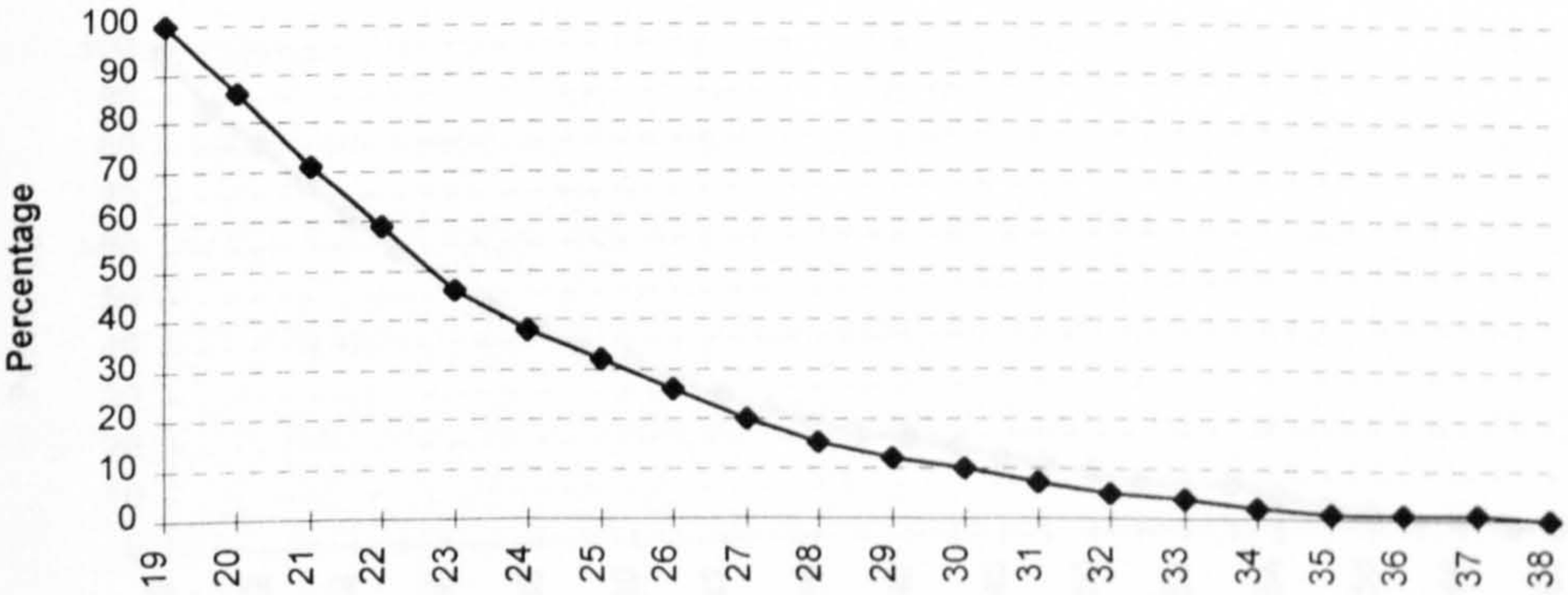


Figure 6.11 : The decrease in the total number of same-day exams in KFU

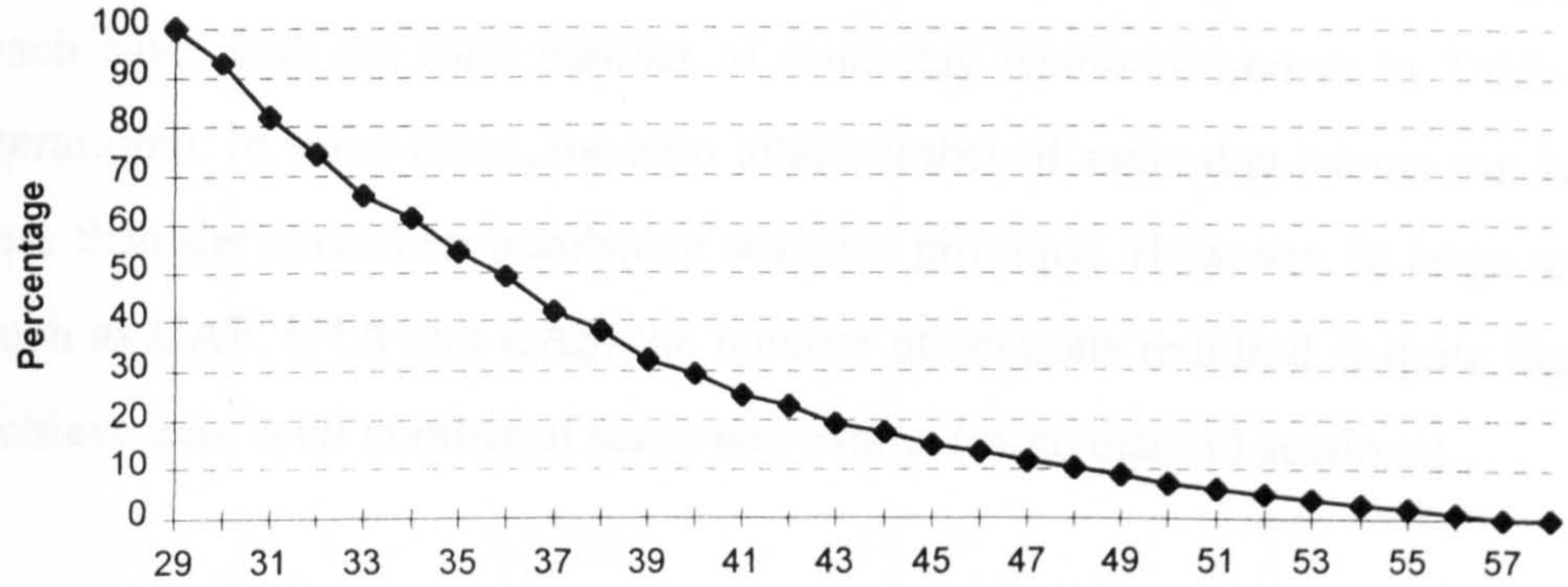


Figure 6.12 : The decrease in the total number of same-day exams in CAF

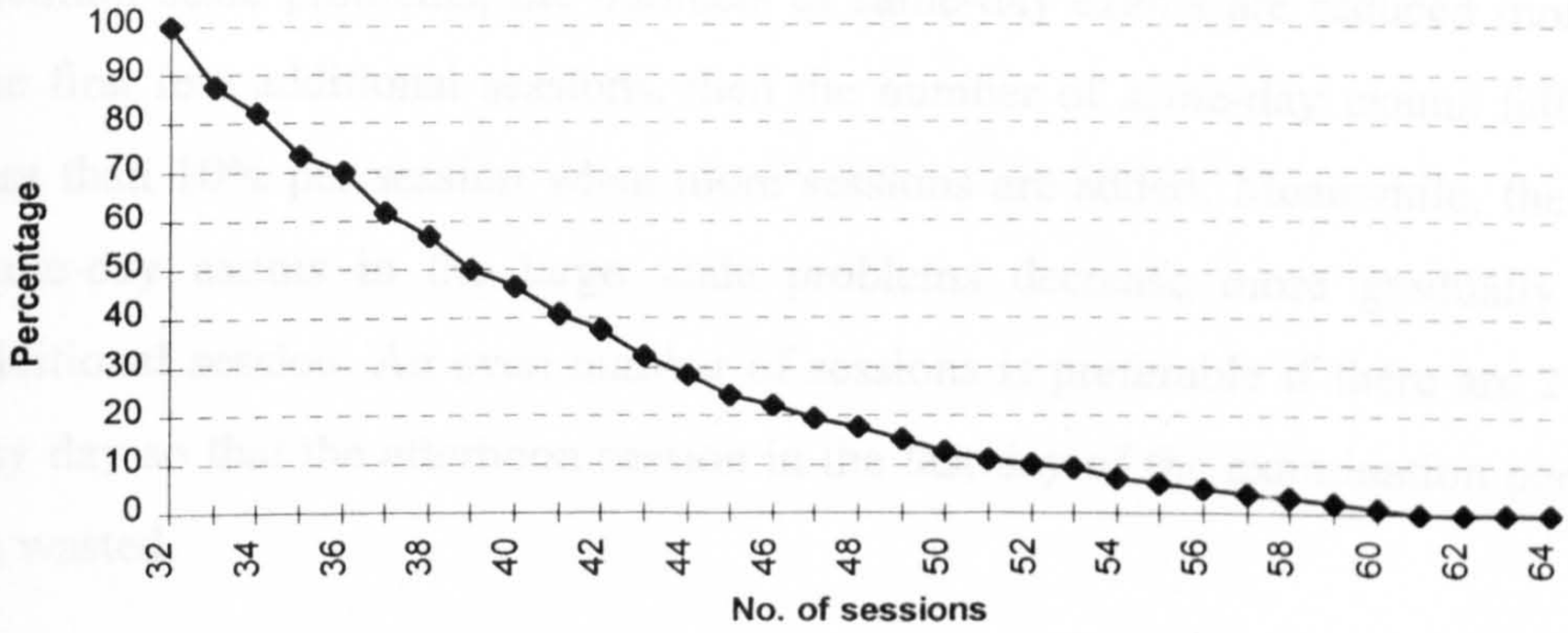


Figure 6.13 : The decrease in the total number of same-day exams in UTA

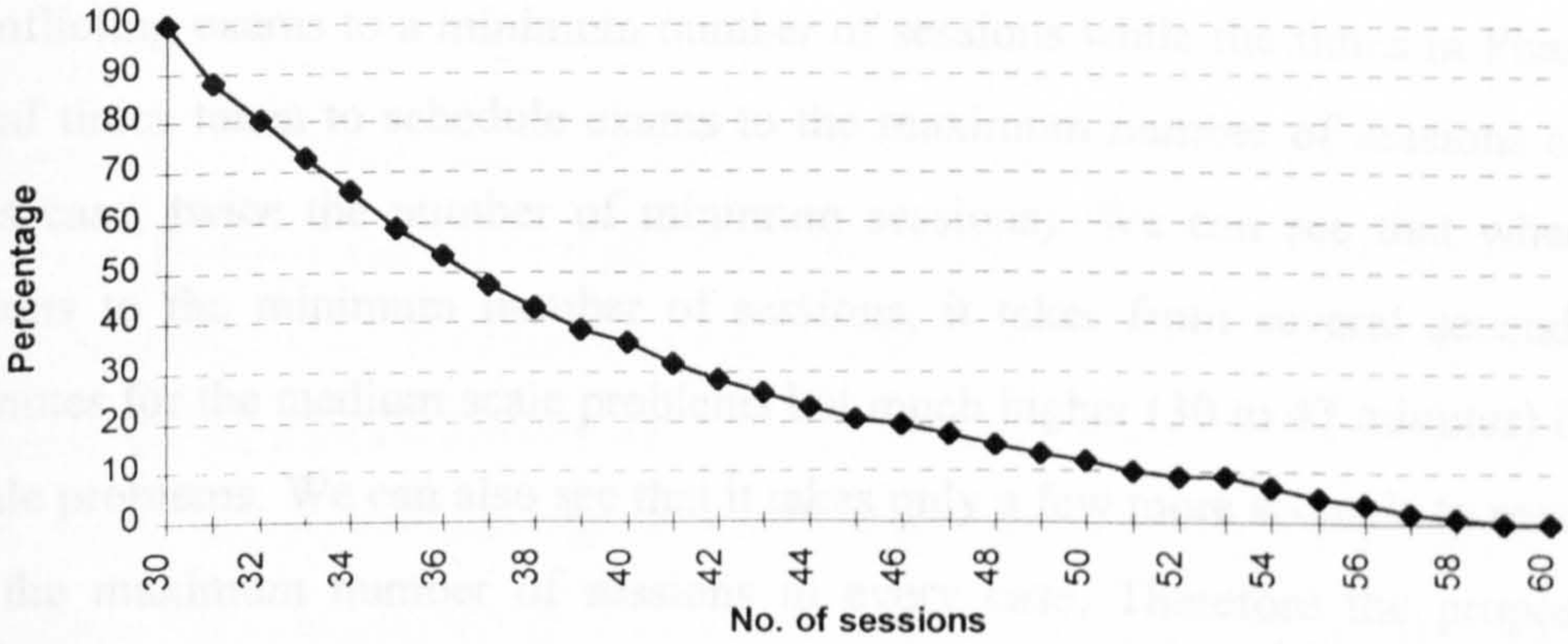


Figure 6.14 : The decrease in the total number of same-day exams in CAS

The graphs show that the number of sessions starts at the minimum value obtained in each case, with the total number of same-day exams as shown in Table 6.1 (using 3 iterations). In some cases, the zero total number of same-day exams can be achieved in less than the maximum number of sessions provided. However, in large scale problems such as CAF, UTA and CAS, the number of sessions required is quite high in order to achieve zero total number of same-day exams (more than 50 sessions).

We can see from the graph that the bigger the number of sessions, the fewer the number of same-day exams. The examination officers will have to make their own judgement on choosing a suitable number of sessions based on the number of same-day exams. In the medium scale problems, the numbers of same-day exams are reduced more rapidly in the first few additional sessions, then the number of same-day exams falls steadily at less than 10% per session when more sessions are added. Meanwhile, the numbers of same-day exams in the large scale problems decrease more gradually with every additional session. An even number of sessions is preferable if there are 2 sessions on any day so that the afternoon session in the last day of the examination period will not be wasted.

Table 6.2 shows the times taken in seconds to produce a conflict-free timetable for each of the 11 test data sets. The times in Phase I are the times taken to assign all non-conflicting exams to a minimum number of sessions while the times in Phase II are the total times taken to schedule exams to the maximum number of sessions available (in this case, twice the number of minimum sessions). We can see that when assigning exams to the minimum number of sessions, it takes from several seconds to a few minutes for the medium scale problems but much higher (30 to 47 minutes) for the large scale problems. We can also see that it takes only a few more seconds to reassign exams to the maximum number of sessions in every case. Therefore the proposed general examination timetabling system is shown to be very fast and efficient.

Data	Time (sec) for Phase I	Time (sec) for Phase II
HEC	5.89	6.14
STA	16.44	16.67
YOR	25.01	25.52
EAR	34.49	35.02
LSE	250.96	253.41
UTE	38.09	39.16
TRE	137.71	139.67
KFU	964.01	972.47
CAF	1797.10	1804.16
UTA	2223.78	2236.88
CAS	2805.56	2826.75

Table 6.2 : Computation time using the proposed method

6.4 Assignment Of Exams To Rooms In Each Session

The next step involves incorporating two further objectives into the basic algorithmic rule. In our survey, Objective 7 (to avoid having exams with different durations in the same room) is ranked second while Objective 8 (not to split any exam in two or more rooms) is fourth in the list of eight objectives considered when constructing examination timetables. However, in order to make full use of the rooms availabilities and also not to increase the minimum number of sessions required to schedule all exams, we felt that these two objectives can be relaxed when constructing examination timetables. Therefore, large exams which exceed the capacity of the largest room would have to be split into 2 or more rooms, and exams with different durations would have to be scheduled together if necessary. In order to maximise use of the space, large rooms will be filled up with as many students as possible. The rule for assigning exams in each session to exam rooms is as follows:

Step1. Exam rooms are initially ordered in decreasing order of their sizes.

Step2. In each session, exams are sorted in decreasing order of the number of students.

Step3. If the total number of students taking the next exam does not exceed the capacity of the rooms with the most seats available, then assign the exam to it and go to Step 5, otherwise go to Step 4.

Step4. The students would have to be split into two or more rooms. Occupy the largest room in the list and put the rest of the students into the next room in the list which can accommodate the students.

Step5. If all exams have been assigned to rooms then STOP, otherwise go to Step 3 for the next exam in the list.

This is obviously a very simple and straightforward rule for assigning exams to exam rooms, but lack of information on room sizes in the test data makes it somewhat unrealistic to compare alternative, more complex rules.

6.5 Rescheduling Of Sessions To Days

Objective 6 should be incorporated into the timetabling system since from the survey, it was found to be ranked third. The purpose of this objective is to schedule exams with large numbers of students early in the examination period where it allows more time for the marking of the exam before the results come out at the end of the examination period.

The procedure for rescheduling sessions to days in order to have larger exams at the early part of the examination period starts with choosing the session containing the exam with the largest number of students. If this session has not been already assigned to the morning session of the first day available then schedule this exam, together with other non-conflicting exams in the same chosen session, to the first available morning session. In order to preserve the total number of same-day exams achieved in the previous section, the exams in the pair session (the session on the same day as the chosen session) will be rescheduled to the afternoon session of that day. The process continues to find the exam in sessions which have not yet been assigned which has the next largest number of students. The exams in the chosen session will be then rescheduled to the morning session of the new lowest numbered day available and the exams in its pair session to the afternoon session. The process is repeated until the last session of the last day has been assigned with exams.

Table 6.3 shows the size of the largest exam in each session for the eleven data sets. The number of sessions used in this case is the minimum number of sessions obtained in Section 5.3. The largest exam, together with other non-conflicting exams in the chosen session, are reassigned to the earliest available odd number session. Meanwhile the exams in the pair session are reassigned to the even number session (on the same day) to maintain the minimum number of same-day exams found in the previous section. That is why we see that the size of the largest exam in the even number session is smaller than in the odd number session. The solution quality in terms of having large exams scheduled early in the examination period might be improved by moving individual large exam from the even number session to the odd number session as long as the total number of same-day exams does not increase drastically. When moving an exam to its new session, we must make sure that it does not clash with any of the exams already assigned to it. Otherwise, we need to carry out a swapping process where the conflicting exams are rescheduled to other sessions. Again we need to consider the number of same-day exams when doing this. Therefore the examination officer must make a decision which objective is more important, the number of same-day exams or having large exams scheduled earlier.

Session	HEC	STA	YOR	EAR	LSE	UTE	TRE	KFU	CAF	UTA	CAS
1	634	237	175	232	382	482	407	1280	1566	1314	1385
2	289	122	34	29	113	351	139	348	158	642	148
3	579	209	136	220	359	367	390	1023	1557	1268	666
4	257	68	52	59	280	151	78	71	127	418	241
5	573	128	114	216	291	345	333	970	806	1242	565
6	127	88	59	99	150	187	206	185	413	571	196
7	469	120	114	178	255	307	325	970	618	1185	522
8	141	85	55	82	131	193	229	94	481	280	449
9	454	118	96	174	222	192	256	931	607	1120	466
10	114	35	49	76	186	158	307	444	390	728	376
11	367	20	88	159	189		189	494	567	1117	439
12	273	78	41	89	50		291	100	62	220	77
13	275	35	80	130	159		146	439	521	712	421
14	105		62	49	60		274	139	434	273	253
15	245		72	125	146		134	430	475	616	415
16	205		50	82	79		210	250	347	323	333
17	197		67	121	137		169	391	472	596	402
18			66	112			186	154	398	119	217
19			59	118			99	346	447	568	359
20			28	25			159	258	376	492	349
21				94					421	497	336
22				89					337	291	331
23				81					378	325	302
24				59					328	269	274
25									361	324	269
26									357	255	264
27									346	304	206
28									203	257	188
29									276	301	193
30									184	201	153
31										216	
32										188	

Table 6.3 : Size of the largest exam in each session

6.6 Summary

In this chapter, the construction of a general examination timetabling system for universities is described. The system is an extension to the system discussed in the previous chapter which assigns exams to a minimum number of sessions while avoiding conflicts for any student. It also uses a rule-based method to incorporate most of the important objectives and the constraints identified by the survey. The objectives which are taken into account are:

- to minimise the total number of same-day exams, where students need to take two exams on the same day.

- to schedule large exams early in the examination period.

and the constraints are :

- a specified time limit on the overall examination period in terms of the maximum number of sessions allowed.

- a maximum number of students that may be examined in any session, i.e. within the capacity of all examination rooms available.

A swapping rule where exams in sessions are interchanged with each other is introduced to reduce the total number of same-day exams within the minimum number of sessions. The use of the swapping rule when minimising the number of same-day exams means that the same cluster of exams would not be kept throughout the assignment process. This is because after each assignment of exam to its chosen session, the exams in the pair session will be swapped with exams in other sessions which have not been previously swapped. If we want to keep the same cluster of exams after assigning them to the minimum number of sessions, we need to use a standard algorithm such as a matching algorithm in order to minimise the number of same-day exams. However, we

decided to use the swapping rule when minimising the number of same-day exams due to its simplicity. We would also like to use an alternative approach to a standard algorithm in a form of a simple search method.

The evidence is that the swapping iteration should be done three times in order to produce the best results. The rules for reassigning exams in additional sessions to further decrease the total number of same-day exams are also discussed. The results show that as extra sessions are introduced, the number of same-day exams decreases to the point where the timetabler needs to use his/her judgement to decide whether the further reduction justifies the use of an extra session.

CHAPTER 7

SUMMARY AND FUTURE WORK

7.1 Summary

The objective of the research work presented in this thesis is to produce a general computerised examination timetabling system for universities in the U.K. and Northern Ireland. This has never been done before since the examination timetabling problems can vary greatly among universities. Nevertheless, of all possible situations, there should be some common requirements that will be the basis of the general examination timetabling system. The work was divided into two parts; firstly a survey of the existing examination timetabling systems in universities and secondly the construction of a general examination timetabler. The second part was our main focus of the research work where the method of assigning exams to sessions produced by Carter *et al.* [1996] was improved. The proposed system will be able to produce a timetable within a very reasonable computation time and also provides a choice of the number of sessions to be used, from the least number of sessions possible to the maximum number of sessions available. The examination officer will be able to choose the appropriate number of sessions in order to produce a good solution quality, in this case the number of students taking two exams on the same day.

7.1.1 The Survey

The survey was carried out to: (i) determine the extent to which the use of computerised examination timetabling procedures are used, (ii) identify the objectives and constraints which are commonly considered when constructing examination timetables and (iii) evaluate the effectiveness of the existing examination timetabling systems in achieving the objectives and satisfying the constraints. Questionnaires were sent to 93 selected institutions in the U.K and Northern Ireland. A total of 55 questionnaires were returned, which is about a 60% response rate. This rate is considered to be quite high for survey research of this nature. In addition to simple descriptive statistics and qualitative interpretations of the data, chi-square tests of independence and simple factorial ANOVAs were used to analyse the data obtained from the survey.

Generally, most of the survey respondents consist of semester based universities with a modular system for both undergraduate and taught postgraduate programmes. Semesterisation is favoured because it can offer more flexibility and freedom of choice for students. Students are able to take their chosen courses whenever they can thus allowing them to tailor their studies to personal circumstances and requirements. The system is also said to benefit students who prefer to work consistently throughout the year because examinations are usually held at the end of each semester. The average examination period in semester based universities is shorter than in term based universities since they usually have two exam diets in a year. This could create a problem for examination officers to schedule all exams in a short examination period.

The results showed that computerised timetabling systems were used by only about 30% of both term based and semester based universities. This implies that most of the university examination officers still construct the examination timetables manually. Those examination officers having to construct their timetables manually found that it was extremely difficult to avoid clashes for all students if the clash list is very long or complicated. This is largely due to the procedure of checking for clashes in each exam session which can be very tedious and laborious. They can also encounter major

problems if errors were detected after the timetable was completed and therefore the timetable has to be reconstructed. Other problems which also occurred in both universities with computerised and manual timetabling systems arise when there are not many large rooms to accommodate large numbers of students and where students have to be split between two or more rooms. Problems can also be caused by a lack of co-operation from the departments and students in getting accurate source data or information about the courses.

From the survey, the common objectives considered by university examination officers in decreasing order of importance are:

i) to produce a clash free timetable
ii) to avoid having exams with different durations in the same room
iii) to schedule large examinations early in the examination period
iv) to avoid splitting any exam between two or more rooms
v) to minimise the overall duration of the examination period.

The common constraints encountered are:

i) a specified time limit on the overall examination period
ii) a maximum number of students that may be examined in any session
iii) certain specified exams that must take place during the same session

The general conclusions gathered from the results of the tests regarding relationships between the university structure and the examination timetabling system, and the level of importance of objectives are:

- the degree of importance attaching to scheduling large exams early in the examination period, avoiding having exams with different durations in the same room and not splitting any exam between two or more rooms all relate to the structure of the university

- the degree of importance attaching to minimising the overall duration of the examination period and scheduling large exams early in the examination period both relate to the examination timetabling system used

- the degree of importance attaching to minimising the overall duration of the examination period and not splitting any exam between two or more rooms in each type of university structure depend on the examination timetabling system used

Regarding the relationship between the examination timetabling system and the university structure, and the level of effectiveness of their existing systems in achieving the objectives, the conclusions are:

- being able to schedule large exams early in the examination period relates to the structure of the universities

- being able to schedule large exams early in the examination period relates to the examination timetabling system used

- being able to minimise the overall duration of the examination period in each type of university structure depends on the examination timetabling system used

7.1.2 Development of A General Examination Timetabling System

The construction of the general examination timetabling system was divided into two stages. The first stage was to solve the examination timetabling problem with the objectives of producing a clash free timetable and minimising the overall duration of the examination period, but without any constraints. The second stage was to incorporate the common constraints found in the survey and also another objective, namely to minimise the number of students having to take more than one exam on the same day into the problem. The general approach is similar to that used by Carter *et al.* [1996].

Their method to solve examination timetabling problems with costs uses the graph colouring approach together with a clique initialisation strategy and a backtracking process. They also provided thirteen real life examination timetabling problems which can be obtained via the internet.

In the first stage, a new algorithmic rule was developed to assign exams to a number of sessions which is as close as possible to the lower bound of the number of sessions without creating conflicts for any student. The algorithmic rule adopts a clique initialisation strategy as a starting point as it proved to make a significant contribution to reducing the total number of sessions needed to schedule all non-conflicting exams. Instead of finding just one maximum clique, the procedure finds all possible maximum cliques. This is to determine whether one maximum clique used as a starting point produces consistently better results than the others. The results have shown that using the maximum clique with the lowest total degree yields the least number of sessions in most instances.

A procedure for finding critical exams and scheduling these exams at an early stage was introduced in order to avoid having to do any backtracking. The backtracking process could be time consuming if there are a lot of exams which clash with each other. An exam is said to be critical if it can only be assigned to a particular session. In other words, it clashes with all sessions but one. By scheduling these exams first, the chances of having to increase the minimum number of sessions needed will be reduced.

When there are no critical exams found in the search, the rest of the unscheduled exams are said to be non-critical. Several ways of scheduling the non-critical exams were examined. The results have shown that the best way to schedule the non-critical exams is to initially order the exams according to recursive largest degree, with the next exam chosen from the list being the one that clashes with most sessions. Among the sessions to which this exam can be assigned, choose the highest numbered session whose exams clash with most of the remaining unscheduled exams.

The results obtained by testing the new algorithmic rule on the eleven real data sets have shown that in most cases (mainly in the medium sized problems), the minimum number of sessions was either equal or very close to the lower bound to the number of sessions. The results have also been compared to the results produced by Carter *et al.* [1996] in order to analyse the performance of the new algorithmic rule. The number of sessions obtained by the new rule exceeds the number of sessions obtained by Carter in three cases; two cases with one extra session and another one case with two extra sessions. Therefore, the new algorithmic rule incorporating a graph colouring method and a clique initialisation strategy but without a backtracking process is capable of producing results which are almost as good as those of Carter.

In the second stage, two more objectives and two out of the three most important constraints obtained from the survey are incorporated into the problem. The two objectives taken into account are to minimise the total number of same-day exams, where students have to take two exams on the same day, and to schedule large exams early in the examination period. The constraints are a specified time limit on the overall examination period in terms of the maximum number of sessions allowed, and a maximum number of students that may be examined in any session, i.e. within the capacity of all examination rooms.

We have excluded two of the objectives found in the survey, namely to avoid having exams with different durations in the same room and not to split any exam between two or more rooms. The reason for not considering these objectives is that it is preferable to make full use of all the available examination rooms. However, we decided to include the objective to minimise the number of same-day exams even though it didn't come out strongly in the survey. This is because the objectives considered by examination officers are most likely to be centred on the convenience of the universities and lecturers instead of the students themselves. We have also left out the constraint where there are certain specified exams that must take place during the same session because no information relating to this was available for the data sets used to test the procedures.

The minimisation of the number of same-day exams was done in two parts. The first part was to carry out a swapping procedure where exams are interchanged between sessions within the minimum number of sessions initially obtained, and the second part involved seeking a further reduction in the number of same-day exams if there are additional sessions available. Exams already assigned to sessions in the first part may then be reassigned to the new sessions.

The swapping procedure is carried out every time an exam is scheduled to its chosen session if the total number of same-day exams can be reduced. Exams in the pair session, i.e. the session in the same day as the chosen session, will be swapped with exams in another session which gives the largest net decrease and whose exams have not been swapped before. Experimentation showed that the total number of same-day exams can usually be reduced by more than 50% and the best results were produced if the swapping procedure is repeated three times.

However, if the number of sessions allowable was found to be higher than the minimum number of sessions obtained in the first part, then exams could be reassigned to extra sessions. This will definitely reduce the total number of same-day exams even more. The minimum number of sessions could either be an odd number or an even number. If it was odd, then the extra session is added as the afternoon session on the last day of the examination period. Exams chosen to be reassigned to this new session would be those which have fewer students in conflict with exams in the morning session than with the exams in their current pair session, and also do not exceed the capacity of the session. If the minimum number of sessions obtained is even, then an extra session added would be a morning session on the next day. The total number of same-day exams can be reduced significantly if the exams in one of the two sessions with the largest number of same-day exams are reassigned to the new session.

The tests showed that it is better to reassign the sessions with the lower total degree. At this stage, the session whose exams have been reassigned is now empty. Therefore, some other exams which give a net decrease in the total number of same-day exams can

be reassigned to the empty session. These exams must also not exceed the room capacity. For both phases of the work done, the processing times achieved by the new method of assigning exams to sessions are quite small. For medium scale problems, the results can be produced within a few minutes while the largest problem (CAS) took less than an hour. Therefore, the new method has been shown to be very fast and efficient.

7.2 Future Work

The method developed to solve the general examination timetabling problem in universities has adopted a clique initialisation strategy as discussed in Chapter 5. It is vital to find the maximum clique of the exams since these exams represent the 'difficult' exams which all clash with each other and also with other exams. Instead of finding just one maximum clique, an existing method is modified to produce several maximum cliques. The maximum clique with the lowest sum of degrees is chosen as a starting point. Exams occurring in other maximum cliques are then put back into the waiting list of unscheduled exams. It could be worthwhile exploring further the issue of utilising more of the maximum cliques at the start of the scheduling. This will involve the development of algorithms for selecting a subset of the maximum cliques which can be overlaid on each other such that the examinations occurring in common do not clash with each other. If a large number of cliques are found, it could be possible to allocate many times more exams at the initial stage than is currently the case with just one clique being allocated.

In the present system, the number of sessions in a day is restricted to only two, a morning session and an afternoon session. Further improvement to the system is to allow examination officers to have any number of sessions they would wish to allocate in a day. There could be three sessions a day, i.e. a morning session, an afternoon session and an evening session. The allocation of exams to sessions will be the same as before but the procedure of minimising the number of same-day exams will have to be

changed. This is because now we are having three sessions in a day so some of the students might have to take two or three exams. The main priority now is to eliminate or at least reduce the number of students taking three exams on the same day. Therefore the method of swapping exams will now take into account the number of students taking three exams on the same day as well as the number of students taking two exams. Fixed exams or time windows can be incorporated by adding a clique of dummy vertices (i.e. a set of vertices which are all pairwise adjacent), one for each session (Balakrishnan [1991]). Edges are added between time-windowed exam vertices and any unsuitable session vertices.

The general examination timetabling system excludes two common objectives discovered in the survey which are to avoid having exams with different durations in the same room and not to split any exam between 2 or more rooms. Nevertheless, in order not to split any exam between 2 or more rooms, the examination officers must make sure that the size of the largest room is larger than the size of the largest exam. Any exam to be assigned to a particular session can be assigned to a room at the initial stage where two conditions must be satisfied:

1. the size of the exam must not exceed the capacity of the room.
2. the duration of the exam must be similar to other exams already scheduled to the room.

Finally, the commonly occurring constraint where specified exams must take place during the same session can also be included in the general examination timetabling system. The exams which must take place during the same session can be treated as one exam so that all of them can be scheduled together. However, this can give rise to problems with room allocation because this new amalgamated exam could be allocated to different rooms corresponding to the actual exams that it comprises. The algorithm must therefore 'remember' that this combined group can be split in certain ways for room allocation purposes.

REFERENCES

ABRAMSON, D. (1991). Constructing School Timetables using Simulated Annealing: Sequential and Parallel Algorithms. *Management Science*, **37**(1), 98-113.

ARANI, T., AND LOTFI, V. (1991). A Three Phased Approach to Final Exam Scheduling. *IIE Transaction*, **21**, 86-96.

BALAKRISHNAN, N. (1991). Examination Scheduling : A Computerised Application. *Omega*, **19**, 37-41.

BOIZUMAULT, P., DELON, Y. AND PÉRIDY, L. (1995). Constraint Logic Programming for Examination Timetabling. *Journal of Logic Programming*, 1-17.

BRELAZ, D. (1979). New Methods to Colour the Vertices of a Graph. *Communications of the A.C.M.*, **22**, 251-256.

BRODER, S. (1964). Final Examination Scheduling. *Communications of the A.C.M.*, **7**, 494-498.

BRUSCO, M. J., JACOBS, L. W., BONGIORNO, R. J., LYONS, D. V. AND TANG, B. (1995). Improving Personnel Scheduling At Airline Stations. *Operations Research*, **43**(5), 741-751.

BURKE, E., ELLIMAN, D., FORD, P. AND WEARE, R. (1996). Examination Timetabling in British Universities : A Survey. In *The Practical and Theory of Automated Timetabling*, Burke, E.K. and Ross, P. (eds.), Springer-Verlag Lecture Notes in Computer Science.

CARRAGHAN, R. AND PARDALOS, P. M. (1990). An Exact Algorithm for the Maximum Clique Problem. *Operations Research Letters*, **9**, 375-382.

CARTER, M. W. (1986). A Survey of Practical Applications of Examination Timetabling Algorithms. *Operations Research*, **34**, 193-202.

CARTER, M. .W. AND GENDREAU, M. (1992). A Practical Algorithm for Finding the Largest Clique in a Graph. *Publication CRT-820*, Centre de recherche sur les transports, Montreal.

CARTER, M. W., LAPORTE, G., AND CHINNECK, J. W. (1994). A General Examination Scheduling System. *Interfaces*, **24**(3), 109-120.

CARTER, M. W., LAPORTE, G. (1996). Recent Developments in Practical Examination Timetabling. In *The Practical and Theory of Automated Timetabling*, Burke, E.K. and Ross, P. (eds.), Springer-Verlag Lecture Notes in Computer Science.

CARTER, M. W., LAPORTE, G., AND YAN LEE, S. (1996). Examination Timetabling : Algorithmic Strategies and Applications. *Journal of Operational Research Society*, **47**, 373-383.

COLE, A. J. (1964). The Preparation of Examination Timetables Using a Small-Store Computer. *The Computer Journal*, **7**, 117-121.

CORNE, D., FANG, H. L. AND MELLISH, C. (1993). Solving the Modular Exam Scheduling Problem with Genetic Algorithm. In *Proceedings of the Sixth International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems.* Chung, P.W.H., Lovegrove, G. and Ali, M. (eds.), 370-373.

DESROCHES, S., LAPORTE, G., AND ROUSSEAU, J. M. (1978). HOREX : A Computer Program for the Construction of Examination Schedules. *INFOR*, **16**, 294-298.

DUNSTAN, F. D. J. (1976). Sequential Colourings of Graphs. In *Proceedings of the 5th British Combined Conference, Aberdeen 1975, Cong. Num. XV, Utilitas Math.*, 151-158.

ERGÜL, A. (1996). GA-Based Examination Scheduling Experience at Middle East Technical University. In *The Practical and Theory of Automated Timetabling*, Burke, E.K. and Ross, P. (eds.), Springer-Verlag Lecture Notes in Computer Science.

EGLESE, R. W. (1990). Simulated Annealing: A Tool for Operational Research. *European Journal of Operational Research*, **46**, 271-281.

EGLESE, R.W. AND RAND, G K. (1987). Conference Seminar Timetabling. *Journal of Operational Research Society*, 38(7), 591-598.

GLOVER, F. (1989). Tabu Search – Part I. *ORSA Journal on Computing*, **1**(3), 190-206.

HANKE, J. E. AND ARTHUR, G. R. (1991). *Understanding Business Statistics*, 2nd Edition, America : Richard D. Irwin Inc.

HERTZ, A. (1991). Tabu Search for Large Scale Timetabling Problems. *European Journal of Operational Research*, **54**, 39-47.

HOWITT, D. AND CRAMER, D. (1997). *A Guide to Computing Statistics with SPSS for Windows*, Prentice Hall.

JOHNSON, D. (1990). Timetabling University Examinations. *Journal of Operational Research Society*, **41**(1), 39-47.

KARP, R. M. (1972). Reducibility among Combinatorial Problems. In *Complexity of Computer Computations*, Plenum Press, New York, 85-104.

KINNEAR, P. R. AND COLIN, D.G. (1994). *SPSS for Windows Made Simple*, Erlbaum (UK) Taylor & Francis.

LAPORTE, G., AND DESROCHES, S. (1984). Examination Timetabling by Computer. *Computers & Operations Research*, **11**, 351-360.

LOTFI, V., AND CERVENY, R. (1991). A Final-exam-scheduling Package. *Journal of Operational Research Society*, **42**(3), 205-216.

MANVEL, B. (1981). Colouring Large Graphs. *Proceedings of the 1981 Southeast Conference on Graph Theory, Combinatorics and Computer Science.*

MATULA, D. W., MARBLE, G. AND ISAACSON, I. D. (1972). Graph Colouring Algorithms. In *Graph Theory and Computing*, R. C. Read (ed.). New York : Academic Press.

MEHTA, N. K. (1981). The Application of A graph Colouring Method to An Examination Scheduling Problem. *Interfaces*, **11**(5), 57-65.

NUIJTEN, W. P. M., KUNNEN, G. M., AARTS, E. H. L. AND DIGNUM, F. P. M. (1995). Examination Timetabling Problem. In *Proceedings of the ECAI'94 Workshop on Constraint Satisfaction Issues Raised by Practical Applications*, 11-19.

NORUSIS, M. J. (1993). *SPSS for Window : Base System User Guide Release 6.0*, Chicago III, SPSS Inc.

OPPENHEIM, A. N. (1992). *Questionnaire Design, Interviewing and Attitude Measurements*, New Edition, New York : Printer Publisher Ltd.

POTTS, C. N. AND VAN WASSENHOVE, L. N. (1991). Single Machine Tardiness Sequencing Heuristics. *IIE Transactions*, **23**, 346-354.

REEVES, C. R. AND BEASLEY, J. E. (1993). *Modern Heuristic Techniques for Combinatorial Problems*, Colin R. Reeves(Ed), Blackwell Scientific Publications.

SEKARAN, U. *Research Methods for Business: A skill building Approach* (2nd Edition). John Wiley & Sons.

THOMPSON, J. AND DOWSLAND, K (1995). TISSUE Wipes Away Exam Time tears - A Computerised System Helps Swansea University Improve Examination Timetabling. *OR Insight*, 8(4), 28-32.

THOMPSON, J. AND DOWSLAND, K (1996). General Cooling Schedules for a Simulated Annealing Based Timetabling System. In *The Practical and Theory of Automated Timetabling*, Burke, E.K. and Ross, P. (eds.), Springer-Verlag Lecture Notes in Computer Science.

VAN LAARHOVEN, P. J. M., AARTS, E. H. L. AND LENSTRA, J. K. (1992). Job shop Scheduling by Simulated Annealing. *Operations Research*, **40**, 113-125.

WELSH, D. J. A. AND POWELL, M. B. (1967). An Upper Bound for the Chromatic Number of a Graph and Its Application to Timetabling Problems. *The Computer Journal*, **10**, 85-86.

WHITE, G. M., AND CHAN, P. W. (1979). Towards the Construction of Optimal Examination Schedules. *INFOR*, **17**, 219-229.

WILSON, J. (1981). The Scheduling of Magistrates to Courts. *Journal of Operational Research Society*, **32**, 121-124.

WOOD, D.C. (1968). A System for Computing University Examination Timetables. *The Computer Journal*, **11**, 41-47.

WRIGHT, M. (1994). Timetabling County Cricket Fixtures Using Form of Tabu Search. *Journal of Operational Research Society*, **45**(7), 758-770.

APPENDIX 1

QUESTIONNAIRE

Q/naire No. .....

UNIVERSITY EXAMINATION TIMETABLING SURVEY

Institution : .....

Examination Officer : .....

Please enter the total number of students in each of the following categories during the **1994/1995** academic year.

	Undergraduate	Postgraduate (Taught)	Postgraduate (Research)
Full-time			
Part-time			

UNIVERSITY SYSTEM

For questions 1 to 5, please tick one of the boxes in each case.

1. For the academic year 1995/1996, which of the following best describes the academic structure of the university regarding the way in which the **majority** of student teaching is organised?

   ☐ Term based
   ☐ Semester based

2. Are there any definite plans to change the present academic structure of the university as described in Question 1?

   ☐ Yes
   ☐ No

   # If *Yes*, please state when .....

3. Are the university's academic programmes organised *predominantly* on a Modular basis?

   Undergraduate ☐ Yes Go to Question 5.
   ☐ No Go to Question 4.

Postgraduate (Taught) ☐ Yes Go to Question 5.
☐ No Go to Question 4.

4. If programmes are not currently organised on a Modular basis, are there any definite plans to introduce a Modular system?

Undergraduate ☐ Yes
☐ No

If *Yes*, please state when .....

Postgraduate(Course) ☐ Yes
☐ No

If *Yes*, please state when .....

5. How often will university wide examinations be held during academic year 1995/1996?

☐ Once a year
☐ Twice a year
☐ Other

If *Other*, please give brief details .....

EXAMINATION DETAILS

For questions 6 to 10, please refer to the most recent university wide examinations held.

6. How many different examination papers were taken? .....

7. What was the total number of student-examinations? .....

8. Using all of the examination rooms normally available, what is the maximum number of students that could be examined at the same time? .....

9. What was the overall duration of the examination period? (in days) .....

10. Please indicate below how many different examination sessions are held in each day of the entire examination period.

(N.B. Different examination sessions represent the non-overlapping time slots available in each day. For example, 9 a.m. - 11 a.m. and 9 a.m. - 12.00 p.m.time slots are considered as one examination session, and 9 a.m. - 12.00 p.m. and 12.00 p.m. - 2.00 p.m. time slots as two different sessions.)

Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday

EXAMINATION TIMETABLING

11. Do you presently use a computerised examination scheduler to allocate examinations to particular sessions?

☐ Yes.
☐ No Go to Question 13.

12. Is the examination scheduler that you presently use

☐ Commercially available as a general examination scheduling package.

If so, what is the name of the package .....

☐ Designed and used specifically by your institution.

☐ Other

If *Other*, please specify .....

13. Briefly describe the specific procedure used in your institution for constructing the examination timetable.

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

14. Briefly describe any problems regularly encountered when constructing the examination timetable for your institution.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

15. Below are a number of objectives that might be considered when constructing an examination timetable. Please indicate the relative importance of each of these objectives for your institution by *circling* the appropriate number in each case:

OBJECTIVE	Not important	Slightly important	Fairly important	Desirable	Essential
Eliminate exam conflicts for all students.	1	2	3	4	5
Ensure that no student takes more than one exam on the same day.	1	2	3	4	5
Minimise the number of exams taken by any student in 2 consecutive days.	1	2	3	4	5
Minimise the overall duration of the examination period.	1	2	3	4	5
Schedule examinations to take account of year of course.	1	2	3	4	5
Schedule the larger examinations early in the examination period.	1	2	3	4	5
Avoid having exams with different durations in the same room.	1	2	3	4	5
Not to split any examination between two or more rooms.	1	2	3	4	5

Please specify below any additional objectives considered by your institution to be at least fairly important when timetabling examinations.

ADDITIONAL OBJECTIVES	Fairly important	Desirable	Essential
A	3	4	5
B	3	4	5

16. Below are a number of constraints that might arise when timetabling examinations. Please *tick* the appropriate boxes for those that apply in your institution.

- ☐ Specified time limit on the overall examination period.
- ☐ Maximum number of students that may be examined in any session.
- ☐ Maximum number of exams that may take place in any session.
- ☐ Certain specified exams require special facilities or equipment.
- ☐ Certain specified exams must take place during the same session.
- ☐ Certain specified exams cannot take place during particular sessions.
- ☐ Certain specified exams must take place during a defined period.

Please list any other constraints that are taken into account when timetabling examinations in your institution.

I .....
.....
.....

II .....
.....
.....

17. This question concerns the effectiveness of your examination timetabling system in achieving those objectives which are relevant to your institution?

For each of the objectives in question 15 which you have rated as at least fairly important, please indicate the extent to which it is achieved by *circling* the appropriate number in each case:

OBJECTIVE	Not at all	A little	Reasonably	Largely	Totally
Eliminate exam conflicts for all students.	1	2	3	4	5
Ensure that no student takes more than one exam on the same day.	1	2	3	4	5
Minimise the number of exams taken by any student in 2 consecutive days.	1	2	3	4	5
Minimise the overall duration of the examination period.	1	2	3	4	5
Schedule examinations to take account of year of course.	1	2	3	4	5
Schedule the larger examinations early in the examination period.	1	2	3	4	5
Avoid having exams with different durations in the same room.	1	2	3	4	5
Not to split any examination between two or more rooms.	1	2	3	4	5
Additional objective A	1	2	3	4	5
Additional objective B	1	2	3	4	5

18. This question concerns the effectiveness of your examination timetabling system in satisfying the constraints that arise in your institution?

For each of the constraints which you have specified in question 16, please indicate the extent to which it is satisfied by *circling* the appropriate number in each case:

CONSTRAINT	Not at all	A little	Reasonably	Largely	Totally
Specified time limit on the overall examination period.	1	2	3	4	5
Maximum number of students that may be examined in any session.	1	2	3	4	5
Maximum number of exams that may take place in any session.	1	2	3	4	5
Certain specified exams require special facilities or equipment.	1	2	3	4	5
Certain specified exams must take place during the same session.	1	2	3	4	5
Certain specified exams cannot take place during particular sessions.	1	2	3	4	5
Certain specified exams must take place during a defined period.	1	2	3	4	5
Other constraint I	1	2	3	4	5
Other constraint II	1	2	3	4	5

APPENDIX 2

LIST OF INSTITUTIONS

	Id no.	
*	38	University of Hull
*	39	University of Abertay
*	40	Loughborough University
*	41	University of Reading
*	42	University of Surrey
*	43	Anglia Polytechnic University
*	2	Aston University
*	3	University of Bath
*	4	Queen's University of Belfast
*	5	Bournemouth University
*	6	University of Brighton
*	8	University of Hertfordshire
*	9	University of Bristol
*	10	Brunel University
*	11	University of Central Lancashire
*	12	University of Central England in Birmingham
*	13	City University
*	14	Coventry University
*	15	De Montfort University
*	16	University of Derby
*	17	University of Dundee
*	18	Napier University
*	19	University of Durham
*	20	University of East Anglia
*	21	University of Edinburgh
*	22	University of Essex
*	23	University of Exeter
*	24	University of Glamorgan
*	25	University of Glasgow
*	26	University of Greenwich
*	27	Heriot-Watt University
*	28	University of Hertfordshire
*	29	University of Huddersfield
*	30	University of Humberside
*	31	University of Keele
*	32	University of Kent at Canterbury
*	33	Kingston University
*	34	University of Lancaster
*	35	University of Leeds
*	36	Leeds Metropolitan University
*	37	University of Leicester
*	38	University of Liverpool
*	39	Liverpool John Moores University
*	40	University of London, Birkbeck College
*	41	University of London, Goldsmiths College

* Survey respondent

	Id no.	Name of Institutions
*	P1	University of Hull
*	P2	University of Aberdeen
*	P3	Loughborough University
*	P4	University of Reading
*	P5	University of Sussex
	1	Anglia Polytechnic University
*	2	Aston University
*	3	University of Bath
*	4	Queen's University of Belfast
*	5	University of Birmingham
*	6	Bournemouth University
*	7	University of Bradford
*	8	University of Brighton
*	9	University of Bristol
*	10	Brunel University
	11	University of Central England in Birmingham
*	12	University of Central Lancashire
*	13	City University
*	14	Coventry University
	15	De Monfort University
*	16	University of Derby
*	17	University of Dundee
	18	Dundee Institute of Technology
	19	University of Durham
*	20	University of East Anglia
	21	University of Edinburgh
*	22	University of Essex
	23	University of Exeter
*	24	University of Glamorgan
	25	University of Glasgow
	26	University of Greenwich
*	27	Heriot-Watt University
*	28	University of Hertfordshire
	29	University of Huddersfield
	30	University of Humberside
*	31	University of Keele
*	32	University of Kent at Canterbury
	33	Kingston University
	34	University of Lancaster
*	35	University of Leeds
*	36	Leeds Metropolitan University
	37	University of Leicester
*	38	University of Liverpool
*	39	Liverpool John Moores University
	40	University of London, Birkbeck College
*	41	University of London, Goldsmiths' College

* Survey respondent

	42	University of London, Imperial College of Sci, Tech & Medicine
	43	University of London, King's College London
*	44	University of London, LSE & Political Science
	45	University of London, Queen Mary & Westfield College
	46	University of London, Royal Holloway
*	47	University of London, University College London
*	48	University of London, Wye College
*	49	London Guidhall University
*	50	UMIST
	51	Manchester Metropolitan University
*	52	Victoria University of Manchester
	53	Middlesex University
*	54	Napier University
*	55	University of Newcastle upon Tyne
*	56	University of North London
*	57	University of Northumbria at Newcastle
*	58	University of Nottingham
	59	Nottingham Trent University
*	60	Oxford Brookes University
	61	University of Paisley
	62	University of Plymouth
*	63	University of Portsmouth
	64	Robert Gordon University
*	65	University of St. Andrews
*	66	University of Salford
	67	University of Sheffield
	68	Sheffield Hallam University
	69	University of Southampton
	70	South Bank University
	71	Staffordshire University, Stoke-on-Trent : College Road
	72	University of Stirling
*	73	University of Strathclyde
*	74	University of Sunderland
*	75	University of Surrey
*	76	University of Teeside
*	77	Thames Valley University, Ealing Campus : St. Mary's Road
	78	University of Ulster
*	79	University College of Wales, Aberystwyth
*	80	University College of North Wales, Bangor
	81	University of Wales College of Cardiff
	82	University College of Swansea
*	83	Saint David's University College, Lampeter
*	84	University of Warwick
	85	University of Westminster
	86	University of West England, Bristol
	87	University of Wolverhampton
	88	University of York

* survey respondent

APPENDIX 3

TABLES OF ANOVA FOR THE RELATIONSHIPS BETWEEN EXAMINATION TIMETABLING SYSTEM AND UNIVERSITY STRUCTURE WHEN CONSIDERING OBJECTIVES

* * * A N A L Y S I S O F V A R I A N C E * * *

OBJ1 objective1
by STRUCTUR university structure
COMPUT comput. system

UNIQUE sums of squares
All effects entered simultaneously

Source of Variation	Sum of Squares	DF	Mean Square	F	Sig of F
Main Effects	.426	2	.213	1.067	.352
STRUCTUR	.048	1	.048	.241	.625
COMPUT	.245	1	.245	1.228	.273
2-Way Interactions	.205	1	.205	1.027	.316
STRUCTUR COMPUT	.205	1	.205	1.027	.316
Explained	.658	3	.219	1.098	.358
Residual	10.179	51	.200		
Total	10.836	54	.201		

55 cases were processed.
0 cases (.0 pct) were missing.

* * * A N A L Y S I S O F V A R I A N C E * * *

OBJ2 objective2
by STRUCTUR university structure
COMPUT comput. system

UNIQUE sums of squares
All effects entered simultaneously

Source of Variation	Sum of Squares	DF	Mean Square	F	Sig of F
Main Effects	7.700	2	3.850	2.377	.103
STRUCTUR	6.131	1	6.131	3.785	.057
COMPUT	.130	1	.130	.080	.778
2-Way Interactions	1.067	1	1.067	.659	.421
STRUCTUR COMPUT	1.067	1	1.067	.659	.421
Explained	12.555	3	4.185	2.584	.064
Residual	79.369	49	1.620		
Total	91.925	52	1.768		

55 cases were processed.
2 cases (3.6 pct) were missing.

* * * A N A L Y S I S O F V A R I A N C E * * *

OBJ3 objective3
by STRUCTUR university structure
COMPUT comput. system

UNIQUE sums of squares
All effects entered simultaneously

Source of Variation	Sum of Squares	DF	Mean Square	F	Sig of F
Main Effects	6.146	2	3.073	2.754	.073
STRUCTUR	2.490	1	2.490	2.232	.141
COMPUT	5.526	1	5.526	4.952	.031
2-Way Interactions	2.656	1	2.656	2.380	.129
STRUCTUR COMPUT	2.656	1	2.656	2.380	.129
Explained	6.289	3	2.096	1.879	.145
Residual	56.911	51	1.116		
Total	63.200	54	1.170		

55 cases were processed.
0 cases (.0 pct) were missing.

* * * A N A L Y S I S O F V A R I A N C E * * *

OBJ4 objective4
by STRUCTUR university structure
COMPUT comput. system

UNIQUE sums of squares
All effects entered simultaneously

Source of Variation	Sum of Squares	DF	Mean Square	F	Sig of F
Main Effects	7.263	2	3.632	2.821	.069
STRUCTUR	.680	1	.680	.528	.471
COMPUT	7.241	1	7.241	5.625	.022
2-Way Interactions	17.082	1	17.082	13.271	.001
STRUCTUR COMPUT	17.082	1	17.082	13.271	.001
Explained	20.791	3	6.930	5.384	.003
Residual	65.645	51	1.287		
Total	86.436	54	1.601		

55 cases were processed.
0 cases (.0 pct) were missing.

* * * A N A L Y S I S O F V A R I A N C E * * *

OBJ5 objective5
by STRUCTUR university structure
COMPUT comput. system

UNIQUE sums of squares
All effects entered simultaneously

Source of Variation	Sum of Squares	DF	Mean Square	F	Sig of F
Main Effects	24.835	2	12.418	7.000	.002
STRUCTUR	1.025	1	1.025	.578	.451
COMPUT	18.331	1	18.331	10.334	.002
2-Way Interactions	9.061	1	9.061	5.108	.028
STRUCTUR COMPUT	9.061	1	9.061	5.108	.028
Explained	31.799	3	10.600	5.976	.001
Residual	86.918	49	1.774		
Total	118.717	52	2.283		

55 cases were processed.
2 cases (3.6 pct) were missing.

* * * A N A L Y S I S O F V A R I A N C E * * *

OBJ6 objective6
by STRUCTUR university structure
COMPUT comput. system

UNIQUE sums of squares
All effects entered simultaneously

Source of Variation	Sum of Squares	DF	Mean Square	F	Sig of F
Main Effects	10.274	2	5.137	3.021	.058
STRUCTUR	8.713	1	8.713	5.124	.028
COMPUT	4.810	1	4.810	2.829	.099
2-Way Interactions	1.074	1	1.074	.631	.431
STRUCTUR COMPUT	1.074	1	1.074	.631	.431
Explained	22.244	3	7.415	4.361	.008
Residual	85.015	50	1.700		
Total	107.259	53	2.024		

55 cases were processed.
1 cases (1.8 pct) were missing.

* * * A N A L Y S I S O F V A R I A N C E * * *

OBJ7 objective7
by STRUCTUR university structure
COMPUT comput. system

UNIQUE sums of squares
All effects entered simultaneously

Source of Variation	Sum of Squares	DF	Mean Square	F	Sig of F
Main Effects	22.085	2	11.042	7.905	.001
STRUCTUR	22.081	1	22.081	15.808	.000
COMPUT	2.825	1	2.825	2.022	.161
2-Way Interactions	.340	1	.340	.243	.624
STRUCTUR COMPUT	.340	1	.340	.243	.624
Explained	26.529	3	8.843	6.331	.001
Residual	69.841	50	1.397		
Total	96.370	53	1.818		

55 cases were processed.
1 cases (1.8 pct) were missing.

* * * A N A L Y S I S O F V A R I A N C E * * *

OBJ8 objective8
by STRUCTUR university structure
COMPUT comput. system

UNIQUE sums of squares
All effects entered simultaneously

Source of Variation	Sum of Squares	DF	Mean Square	F	Sig of F
Main Effects	14.478	2	7.239	5.751	.006
STRUCTUR	14.473	1	14.473	11.498	.001
COMPUT	1.681	1	1.681	1.335	.253
2-Way Interactions	16.089	1	16.089	12.782	.001
STRUCTUR COMPUT	16.089	1	16.089	12.782	.001
Explained	21.150	3	7.050	5.601	.002
Residual	64.196	51	1.259		
Total	85.345	54	1.580		

55 cases were processed.
0 cases (.0 pct) were missing.

APPENDIX 4

TABLES OF CHI-SQUARE FOR THE ASSOCIATIONS BETWEEN UNIVERSITY STRUCTURE AND EXAMINATION TIMETABLING SYSTEM WHEN CONSIDERING CONSTRAINTS

STRUCTUR university structure by COMPUT comput. system
Controlling for..
CONST1 constraint1 Value = 1 considered

STRUCTUR	COMPUT yes (1)	COMPUT no (2)	Row Total
term based	4	15	19
	21.1	78.9	38.0
	23.5	45.5	
	8.0	30.0	
semester based	13	18	31
	41.9	58.1	62.0
	76.5	54.5	
	26.0	36.0	
Column Total	17	33	50
	34.0	66.0	100.0

Chi-Square	Value	DF	Significance
Pearson	2.28930	1	.13027
Continuity Correction	1.45326	1	.22801
Likelihood Ratio	2.38159	1	.12277
Mantel-Haenszel test for linear association	2.24351	1	.13418

Minimum Expected Frequency - 6.460

Number of Missing Observations: 0

STRUCTUR university structure by COMPUT comput. system
Controlling for..
CONST2 constraint2 Value = 1 considered

STRUCTUR	COMPUT yes (1)	COMPUT no (2)	Row Total
1 term based	3 15.8 25.0 6.1	16 84.2 43.2 32.7	19 38.8
2 semester based	9 30.0 75.0 18.4	21 70.0 56.8 42.9	30 61.2
Column Total	12 24.5	37 75.5	49 100.0

Chi-Square	Value	DF	Significance
Pearson	1.27031	1	.25971
Continuity Correction	.61807	1	.43177
Likelihood Ratio	1.32668	1	.24940
Mantel-Haenszel test for linear association	1.24438	1	.26463
Fisher's Exact Test:			
One-Tail			.21813
Two-Tail			.32294

Minimum Expected Frequency - 4.653
Cells with Expected Frequency < 5 - 1 OF 4 (25.0%)

Number of Missing Observations: 0

STRUCTUR university structure by COMPUT comput. system
Controlling for..
CONST3 constraint3 Value = 1 considered

STRUCTUR	Count / Row Pct / Col Pct / Tot Pct	yes 1	no 2	Row Total
1		1	4	5
term based		20.0	80.0	38.5
		33.3	40.0	
		7.7	30.8	
2		2	6	8
semester based		25.0	75.0	61.5
		66.7	60.0	
		15.4	46.2	
Column Total		3 23.1	10 76.9	13 100.0

Chi-Square	Value	DF	Significance
Pearson	.04333	1	.83510
Continuity Correction	.00000	1	1.00000
Likelihood Ratio	.04392	1	.83400
Mantel-Haenszel test for linear association	.04000	1	.84148
Fisher's Exact Test:			
One-Tail			.68531
Two-Tail			1.00000

Minimum Expected Frequency - 1.154
Cells with Expected Frequency < 5 - 3 OF 4 (75.0%)

Number of Missing Observations: 0

STRUCTUR university structure by COMPUT comput. system
Controlling for..
CONST4 constraint4 Value = 1 considered

STRUCTUR	Count / Row Pct / Col Pct / Tot Pct	yes 1	no 2	Row Total
1		4	11	15
term based		26.7	73.3	38.5
		33.3	40.7	
		10.3	28.2	
2		8	16	24
semester based		33.3	66.7	61.5
		66.7	59.3	
		20.5	41.0	
Column Total		12 30.8	27 69.2	39 100.0

Chi-Square	Value	DF	Significance
Pearson	.19259	1	.66077
Continuity Correction	.00677	1	.93442
Likelihood Ratio	.19472	1	.65901
Mantel-Haenszel test for linear association	.18765	1	.66488
Fisher's Exact Test:			
One-Tail			.47175
Two-Tail			.73424

Minimum Expected Frequency - 4.615
Cells with Expected Frequency < 5 - 1 OF 4 (25.0%)

Number of Missing Observations: 0

STRUCTUR university structure by COMPUT comput. system
Controlling for..
CONST5 constraint5 Value = 1 considered

STRUCTUR	COMPUT		
Count / Row Pct / Col Pct / Tot Pct	yes 1	no 2	Row Total
1	4	13	17
term based	23.5	76.5	35.4
	28.6	38.2	
	8.3	27.1	
2	10	21	31
semester based	32.3	67.7	64.6
	71.4	61.8	
	20.8	43.8	
Column Total	14 / 29.2	34 / 70.8	48 / 100.0

Chi-Square	Value	DF	Significance
Pearson	.40489	1	.52457
Continuity Correction	.09261	1	.76088
Likelihood Ratio	.41340	1	.52025
Mantel-Haenszel test for linear association	.39646	1	.52892
Fisher's Exact Test:			
One-Tail			.38584
Two-Tail			.74135

Minimum Expected Frequency - 4.958
Cells with Expected Frequency < 5 - 1 OF 4 (25.0%)

Number of Missing Observations: 0

STRUCTUR university structure by COMPUT comput. system
Controlling for..
CONST6 constraint6 Value = 1 considered

STRUCTUR	Count / Row Pct / Col Pct / Tot Pct	yes 1	no 2	Row Total
term based	1	3	7	10
		30.0	70.0	34.5
		27.3	38.9	
		10.3	24.1	
semester based	2	8	11	19
		42.1	57.9	65.5
		72.7	61.1	
		27.6	37.9	
Column Total		11 37.9	18 62.1	29 100.0

Chi-Square	Value	DF	Significance
Pearson	.40779	1	.52309
Continuity Correction	.05570	1	.81343
Likelihood Ratio	.41487	1	.51951
Mantel-Haenszel test for linear association	.39373	1	.53035
Fisher's Exact Test:			
One-Tail			.41119
Two-Tail			.69415

Minimum Expected Frequency - 3.793
Cells with Expected Frequency < 5 - 1 OF 4 (25.0%)

Number of Missing Observations: 0

STRUCTUR university structure by COMPUT comput. system
Controlling for..
CONST7 constraint7 Value = 1 considered

STRUCTUR	COMPUT yes (1)	COMPUT no (2)	Row Total
1 term based	4 28.6 28.6 11.1	10 71.4 45.5 27.8	14 38.9
2 semester based	10 45.5 71.4 27.8	12 54.5 54.5 33.3	22 61.1
Column Total	14 38.9	22 61.1	36 100.0

Cell Contents: Count / Row Pct / Col Pct / Tot Pct

Chi-Square	Value	DF	Significance
Pearson	1.02614	1	.31107
Continuity Correction	.43869	1	.50775
Likelihood Ratio	1.04594	1	.30645
Mantel-Haenszel test for linear association	.99764	1	.31788

Minimum Expected Frequency - 5.444

Number of Missing Observations: 0

APPENDIX 5

TABLES OF ANOVA FOR THE RELATIONSHIPS BETWEEN EXAMINATION TIMETABLING SYSTEM AND UNIVERSITY STRUCTURE IN THE LEVEL OF EFFECTIVENESS IN ACHIEVING THE OBJECTIVES

* * * A N A L Y S I S O F V A R I A N C E * * *

EFFOBJ1 effectiveness of obj1
by STRUCTUR university structure
COMPUT comput. system

UNIQUE sums of squares
All effects entered simultaneously

Source of Variation	Sum of Squares	DF	Mean Square	F	Sig of F
Main Effects	.508	2	.254	.686	.508
STRUCTUR	.012	1	.012	.034	.855
COMPUT	.379	1	.379	1.025	.316
2-Way Interactions	.056	1	.056	.153	.698
STRUCTUR COMPUT	.056	1	.056	.153	.698
Explained	.710	3	.237	.640	.593
Residual	18.494	50	.370		
Total	19.204	53	.362		

55 cases were processed.
1 cases (1.8 pct) were missing.

* * * A N A L Y S I S O F V A R I A N C E * * *

EFFOBJ2 effectiveness of obj2
by STRUCTUR university structure
COMPUT comput. system

UNIQUE sums of squares
All effects entered simultaneously

Source of Variation	Sum of Squares	DF	Mean Square	F	Sig of F
Main Effects	.752	2	.376	.219	.804
STRUCTUR	.602	1	.602	.351	.557
COMPUT	.435	1	.435	.253	.618
2-Way Interactions	.235	1	.235	.137	.714
STRUCTUR COMPUT	.235	1	.235	.137	.714
Explained	2.097	3	.699	.407	.749
Residual	63.513	37	1.717		
Total	65.610	40	1.640		

55 cases were processed.
14 cases (25.5 pct) were missing.

* * * A N A L Y S I S O F V A R I A N C E * * *

EFFOBJ3 effectiveness of obj3
by STRUCTUR university structure
COMPUT comput. system

UNIQUE sums of squares
All effects entered simultaneously

Source of Variation	Sum of Squares	DF	Mean Square	F	Sig of F
Main Effects	1.036	2	.518	.368	.694
STRUCTUR	.000	1	.000	.000	.993
COMPUT	.975	1	.975	.693	.411
2-Way Interactions	.501	1	.501	.356	.554
STRUCTUR COMPUT	.501	1	.501	.356	.554
Explained	1.448	3	.483	.343	.794
Residual	52.064	37	1.407		
Total	53.512	40	1.338		

55 cases were processed.
14 cases (25.5 pct) were missing.

* * * A N A L Y S I S O F V A R I A N C E * * *

EFFOBJ4 effectiveness of obj4
by STRUCTUR university structure
COMPUT comput. system

UNIQUE sums of squares
All effects entered simultaneously

Source of Variation	Sum of Squares	DF	Mean Square	F	Sig of F
Main Effects	.750	2	.375	.227	.798
STRUCTUR	.729	1	.729	.442	.510
COMPUT	.346	1	.346	.210	.649
2-Way Interactions	4.903	1	4.903	2.971	.092
STRUCTUR COMPUT	4.903	1	4.903	2.971	.092
Explained	5.989	3	1.996	1.210	.318
Residual	69.315	42	1.650		
Total	75.304	45	1.673		

55 cases were processed.
9 cases (16.4 pct) were missing.

* * * A N A L Y S I S O F V A R I A N C E * * *

EFFOBJ5 effectiveness of obj5
by STRUCTUR university structure
COMPUT comput. system

UNIQUE sums of squares
All effects entered simultaneously

Source of Variation	Sum of Squares	DF	Mean Square	F	Sig of F
Main Effects	9.922	2	4.961	4.470	.019
STRUCTUR	4.863	1	4.863	4.381	.044
COMPUT	.117	1	.117	.105	.748
2-Way Interactions	.515	1	.515	.464	.500
STRUCTUR COMPUT	.515	1	.515	.464	.500
Explained	11.048	3	3.683	3.318	.032
Residual	36.628	33	1.110		
Total	47.676	36	1.324		

55 cases were processed.
18 cases (32.7 pct) were missing.

* * * A N A L Y S I S O F V A R I A N C E * * *

EFFOBJ6 effectiveness of obj6
by STRUCTUR university structure
COMPUT comput. system

UNIQUE sums of squares
All effects entered simultaneously

Source of Variation	Sum of Squares	DF	Mean Square	F	Sig of F
Main Effects	9.433	2	4.716	4.045	.025
STRUCTUR	3.768	1	3.768	3.232	.080
COMPUT	7.820	1	7.820	6.707	.013
2-Way Interactions	.433	1	.433	.372	.546
STRUCTUR COMPUT	.433	1	.433	.372	.546
Explained	15.184	3	5.061	4.341	.010
Residual	45.468	39	1.166		
Total	60.651	42	1.444		

55 cases were processed.
12 cases (21.8 pct) were missing.

* * * A N A L Y S I S O F V A R I A N C E * * *

EFFOBJ7 effectiveness of obj7
by STRUCTUR university structure
COMPUT comput. system

UNIQUE sums of squares
All effects entered simultaneously

Source of Variation	Sum of Squares	DF	Mean Square	F	Sig of F
Main Effects	1.586	2	.793	.705	.500
STRUCTUR	.426	1	.426	.378	.542
COMPUT	.095	1	.095	.085	.772
2-Way Interactions	1.424	1	1.424	1.265	.267
STRUCTUR COMPUT	1.424	1	1.424	1.265	.267
Explained	8.154	3	2.718	2.416	.081
Residual	45.005	40	1.125		
Total	53.159	43	1.236		

55 cases were processed.
11 cases (20.0 pct) were missing.

* * * A N A L Y S I S O F V A R I A N C E * * *

EFFOBJ8 effectiveness of obj8
by STRUCTUR university structure
COMPUT comput. system

UNIQUE sums of squares
All effects entered simultaneously

Source of Variation	Sum of Squares	DF	Mean Square	F	Sig of F
Main Effects	1.184	2	.592	.477	.625
STRUCTUR	.861	1	.861	.693	.410
COMPUT	.031	1	.031	.025	.875
Explained	1.184	2	.592	.477	.625
Residual	48.435	39	1.242		
Total	49.619	41	1.210		

55 cases were processed.
13 cases (23.6 pct) were missing.

Due to empty cells or a singular matrix,
higher order interactions have been suppressed.

APPENDIX 6

TABLES OF ANOVA FOR THE RELATIONSHIPS BETWEEN EXAMINATION TIMETABLING SYSTEM AND UNIVERSITY STRUCTURE IN THE LEVEL OF EFFECTIVENESS IN SATISFYING THE CONSTRAINTS

* * * A N A L Y S I S O F V A R I A N C E * * *

EFFC1 effectiveness of c1
by STRUCTUR university structure
COMPUT comput. system

UNIQUE sums of squares
All effects entered simultaneously

Source of Variation	Sum of Squares	DF	Mean Square	F	Sig of F
Main Effects	1.270	2	.635	1.395	.259
STRUCTUR	.034	1	.034	.075	.786
COMPUT	1.243	1	1.243	2.730	.106
2-Way Interactions	.120	1	.120	.264	.610
STRUCTUR COMPUT	.120	1	.120	.264	.610
Explained	1.270	3	.423	.930	.435
Residual	19.581	43	.455		
Total	20.851	46	.453		

55 cases were processed.
8 cases (14.5 pct) were missing.

* * * A N A L Y S I S O F V A R I A N C E * * *

EFFC2 effectiveness of c2
by STRUCTUR university structure
COMPUT comput. system

UNIQUE sums of squares
All effects entered simultaneously

Source of Variation	Sum of Squares	DF	Mean Square	F	Sig of F
Main Effects	1.343	2	.672	1.439	.249
STRUCTUR	.935	1	.935	2.003	.164
COMPUT	.796	1	.796	1.706	.199
2-Way Interactions	.022	1	.022	.047	.829
STRUCTUR COMPUT	.022	1	.022	.047	.829
Explained	1.878	3	.626	1.342	.274
Residual	19.600	42	.467		
Total	21.478	45	.477		

55 cases were processed.
9 cases (16.4 pct) were missing.

* * * A N A L Y S I S O F V A R I A N C E * * *

EFFC3 effectiveness of c3
by STRUCTUR university structure
COMPUT comput. system

UNIQUE sums of squares
All effects entered simultaneously

Source of Variation	Sum of Squares	DF	Mean Square	F	Sig of F
Main Effects	1.786	2	.893	1.273	.318
STRUCTUR	1.182	1	1.182	1.686	.221
COMPUT	1.182	1	1.182	1.686	.221
2-Way Interactions	.189	1	.189	.270	.614
STRUCTUR COMPUT	.189	1	.189	.270	.614
Explained	2.019	3	.673	.960	.446
Residual	7.714	11	.701		
Total	9.733	14	.695		

55 cases were processed.
40 cases (72.7 pct) were missing.

* * * A N A L Y S I S O F V A R I A N C E * * *

EFFC4 effectiveness of c4
by STRUCTUR university structure
COMPUT comput. system

UNIQUE sums of squares
All effects entered simultaneously

Source of Variation	Sum of Squares	DF	Mean Square	F	Sig of F
Main Effects	2.542	2	1.271	1.752	.189
STRUCTUR	2.539	1	2.539	3.500	.070
COMPUT	.063	1	.063	.087	.770
2-Way Interactions	1.189	1	1.189	1.639	.209
STRUCTUR COMPUT	1.189	1	1.189	1.639	.209
Explained	2.765	3	.922	1.271	.300
Residual	23.938	33	.725		
Total	26.703	36	.742		

55 cases were processed.
18 cases (32.7 pct) were missing.

* * * A N A L Y S I S O F V A R I A N C E * * *

EFFC5 effectiveness of c5
by STRUCTUR university structure
COMPUT comput. system

UNIQUE sums of squares
All effects entered simultaneously

Source of Variation	Sum of Squares	DF	Mean Square	F	Sig of F
Main Effects	1.313	2	.656	1.189	.315
STRUCTUR	.180	1	.180	.326	.571
COMPUT	.928	1	.928	1.681	.202
2-Way Interactions	.605	1	.605	1.096	.301
STRUCTUR COMPUT	.605	1	.605	1.096	.301
Explained	2.136	3	.712	1.290	.290
Residual	23.190	42	.552		
Total	25.326	45	.563		

55 cases were processed.
9 cases (16.4 pct) were missing.

* * * A N A L Y S I S O F V A R I A N C E * * *

EFFC6 effectiveness of c6
by STRUCTUR university structure
COMPUT comput. system

UNIQUE sums of squares
All effects entered simultaneously

Source of Variation	Sum of Squares	DF	Mean Square	F	Sig of F
Main Effects	1.153	2	.576	.566	.575
STRUCTUR	.545	1	.545	.536	.471
COMPUT	.351	1	.351	.345	.562
2-Way Interactions	1.718	1	1.718	1.688	.206
STRUCTUR COMPUT	1.718	1	1.718	1.688	.206
Explained	3.597	3	1.199	1.178	.338
Residual	25.438	25	1.018		
Total	29.034	28	1.037		

55 cases were processed.
26 cases (47.3 pct) were missing.

* * * A N A L Y S I S O F V A R I A N C E * * *

EFFC7 effectiveness of c7
by STRUCTUR university structure
COMPUT comput. system

UNIQUE sums of squares
All effects entered simultaneously

Source of Variation	Sum of Squares	DF	Mean Square	F	Sig of F
Main Effects	.521	2	.260	.411	.667
STRUCTUR	.113	1	.113	.177	.677
COMPUT	.313	1	.313	.493	.488
2-Way Interactions	.501	1	.501	.791	.381
STRUCTUR COMPUT	.501	1	.501	.791	.381
Explained	1.208	3	.403	.635	.599
Residual	19.028	30	.634		
Total	20.235	33	.613		

55 cases were processed.
21 cases (38.2 pct) were missing.

APPENDIX 7

PROGRAM LISTING

```
{ This is the main program for assigning
  exams to a minimum number of sessions
  without creating any conflicts. The input
  files are the course file, the clash file and
  the room capacity file. The program will
  produce an output file listing all the exams
  assigned to each of the two sessions in
  each day of the examination period}

Program Assigning(input,output);

const
   max_exam = 700;
   max_room = 10;
   max_session = 70;

type
   nametype = packed array[1..20] of char;
   crstype=
         record
            no              : integer;
            size            : integer
         end;
   scheduletype=
         record
            exam_no         : integer;
            size            : integer;
            room_no         : array[1..max_room] of integer;
            examroom        : array[1..max_room] of integer;
         end;
sessiontype=
         record
            capacity        : integer;
            exam            : array[1..200] of scheduletype;
            room_size       : array[1..max_room] of integer;
         end;
 examtype=
         record
            no              : integer;
            size            : integer;
            degree          : integer
         end;
sortype=
         record
            no              : integer;
            edge            : integer
         end;
    temptype=
         record
            no              : integer;
            size            : integer
         end;

var
   i,j,k,l,s,t,u,x,y,z,sessno,day,exam      : integer;
   biggest_exam,sess_cap,clq_num            : integer;
   room,totstusess,capacity                 : integer;
   num,count1,count2,mclq,exam_i            : integer;
```

```
initsess,max_count,biggestclash      : integer;
max,chosensess,sessct,extrasess      : integer;
maxdegbsm, biggestmaxnum             : integer;
biggestsubmaxclq, totexcla           : integer;
count,countsess,countdeg             : integer;
maxclique,iteration,addsess          : integer;
totsess,mincla,temp,totsameday       : integer;
maxtotdeg,totdeg,choose              : integer;
tot1,tot2,examtot1,examtot2,pair     : integer;
chosenexam,maxtotexcla,taboono       : integer;
exampos,examsess,diff,mindiff        : integer;
choosesameday,maxsameday             : integer;
crsrec                               : crstype;
crslist                              : array[1..max_exam] of crstype;
schedulerec                          : scheduletype;
schedulelist                         : array[1..200] of scheduletype;
sessrec                              : sessiontype;
sesslist                             : array[1..max_session] of sessiontype;
examrec                              : examtype;
examlist                             : array[1..max_exam] of examtype;
sortrec                              : sortype;
sortlist                             : array[1..max_exam] of sortype;
temprec                              : temptype;
templist                             : array[1..max_exam] of temptype;
arr_1                                : array[1..max_exam,1..max_exam] of integer;
arr_2,arr_3,deg                      : array[1..max_exam] of integer;
cli,sessnum_1,sessnum_2              : array[1..100] of integer;
maxnum,ts,examno                     : array[1..100] of integer;
submaxclq,exno,sumdegbsm             : array[1..100] of integer;
multisess,sess_bsm,same_day          : array[1..max_session] of integer;
mark1,mark2,chsess,taboosess         : array[1..max_session] of integer;
newcli_no,newclique,examinsess       : array[1..max_session,1..100] of integer;
roomnum                              : array[1..max_session,1..max_room] of integer;
sessday                              : array[1..50,1..2] of integer;
name1,name2,name3,name4              : nametype;
inf1,inf2,inf3,outf                  : text;
stop,allassigned,improvement         : boolean;
donotclash                           : boolean;


procedure findmaxclique(var totexam : integer); { procedure to find the maximum cliques }

var
   node,D,Dtemp,exam_j               : integer;
   min,minnode,maxD                  : integer;
   cla                               : array[1..600,1..max_exam] of integer;
   tdeg,actnode,start,last           : array[1..max_exam] of integer;
   a,b,no                            : integer;

begin
   for a := 1 to exam do
    begin
     if examlist[a].no <> 0 then
      begin
           exam_j := examlist[a].no;
           tdeg[exam_j] := arr_2[exam_j]
      end
    end;
```

```
no := 0;
node := totexam;
while node <> 0 do
 begin
  min := maxint;
  for a := 1 to exam do                    { exams are sorted according to the
   begin                                     recursive smallest degree }
        if examlist[a].no <> 0 then
         begin
          exam_j := examlist[a].no;
          if tdeg[exam_j] < min then
           begin
            min := tdeg[exam_j];
            minnode := exam_j
           end
         end
 end;
no := no + 1;
actnode[no] := minnode;
node := node - 1;
for a := 1 to exam do
 begin
      if examlist[a].no <> 0 then
       begin
        exam_j := examlist[a].no;
        if exam_j = actnode[no] then
         tdeg[exam_j] := maxint
        else if tdeg[exam_j] <> maxint then
         begin
          if arr_1[exam_j,actnode[no]] > 0 then
           begin
            if tdeg[exam_j] > 0 then
             tdeg[exam_j] := tdeg[exam_j] - 1
            else tdeg[exam_j] := 0
           end
         end
       end
   end
 end;
maxD := 0;
D := 1;
maxclique := 0;
for a := 1 to totexam do
 cla[D,a] := actnode[a];
start[D] := 0;
last[D] := totexam;
while D <> 0 do
 begin
  start[D] := start[D] + 1;
  if (D+last[D]-start[D]) > maxclique then        { finds the set of largest maximum cliques }
   begin
        Dtemp := D;
        D := D + 1;
        start[D] := 0;
        last[D] := 0;
        for a := (start[Dtemp]+1) to last[Dtemp] do
         begin
          if arr_1[cla[Dtemp,start[Dtemp]],cla[Dtemp,a]] > 0 then
```

```
        begin
         last[D] := last[D] + 1;
         cla[D,last[D]] := cla[Dtemp,a]
        end
       end;
    if last[D] = 0 then
     begin
      D := D - 1;
      if D > maxclique then
       begin
        maxclique := D;
        for a := 1 to maxclique do
          cli[a] := cla[a,start[a]]
       end
     end
    else
     D := D - 1
   end;
  writeln('maxclique = ',maxclique:2);
  initsess := maxclique;
  for a := 1 to maxclique do
   write(cli[a]:5);
  writeln
end;


procedure assign(var a,b,c : integer); { procedure to assign the exam to its chosen session }
var
   examrec      : examtype;
   sessrec      : sessiontype;

begin
   sesslist[a].exam[c].exam_no := examlist[b].no;
   sesslist[a].exam[c].size := examlist[b].size;
   sesslist[a].capacity := sesslist[a].capacity - examlist[b].size;
   examlist[b].no := 0
end;

procedure findnewclique(var examct,sessnum : integer);

var
   a,b,c,number,node,D,Dtemp  : integer;
   max,maxnode,exam_j         : integer;
   cla                        : array[1..600,1..max_exam] of integer;
   tdeg,actnode               : array[1..max_exam] of integer;
   start,last                 : array[1..100] of integer;

begin
   for a := 1 to examct do
    begin
     exam_j := examlist[newcli_no[sessnum,a]].no;
     tdeg[exam_j] := arr_2[exam_j]
    end;
   number := 0;
   node := examct;
   while node <> 0 do
```

```
begin
 max := -maxint;
 for a := 1 to examct do
  begin
       exam_j := examlist[newcli_no[sessnum,a]].no;
       if tdeg[exam_j] > max then
        begin
         max := tdeg[exam_j];
         maxnode := exam_j
        end
  end;
 number := number + 1;
 actnode[number] := maxnode;
 node := node - 1;
 for b := 1 to examct do
  begin
       exam_j := examlist[newcli_no[sessnum,b]].no;
       if exam_j = actnode[number] then
        tdeg[exam_j] := -exam
       else
        begin
         if tdeg[exam_j] <> -exam then
          begin
           if arr_1[exam_j,actnode[number]] <> 0 then
            begin
             if tdeg[exam_j] > 0 then
              tdeg[exam_j] := tdeg[exam_j] - 1
             else tdeg[exam_j] := 0
            end
          end
        end
  end;
 mclq := 0;
 D := 1;
 for a := 1 to examct do
  cla[D,a] := actnode[a];
 start[D] := 0;
 last[D] := examct;
 while D <> 0 do
  begin
   start[D] := start[D] + 1;
   if (D+last[D]-start[D]) > mclq then
    begin
         Dtemp := D;
         D := D + 1;
    start[D] := 0;
    last[D] := 0;
    for c := (start[Dtemp]+1) to last[Dtemp] do
     begin
      if arr_1[cla[Dtemp,start[Dtemp]],cla[Dtemp,c]] <> 0 then
       begin
        last[D] := last[D] + 1;
        cla[D,last[D]] := cla[Dtemp,c]
       end
     end;
    if last[D] = 0 then
     begin
```

```
            D := D - 1;
            if D > mclq then
             begin
              mclq := D;
              for c := 1 to mclq do
               newclique[sessnum,c] := cla[c,start[c]]
             end
           end
      end
     else
      D := D - 1
    end
end;
```

```
procedure swapsession(var sessiona,sessionb : integer); { procedure to swap exams in
                                                          sessions }

var
   a,sess1,sess2          : integer;
   arr                    : array[1..max_exam] of integer;
   tempsessrec            : sessiontype;
   tempsesslist           : array[1..max_session] of sessiontype;

begin
   sess1 := sessiona;
   arr[sess1] := arr_3[sessiona];
   for a := 1 to arr[sess1] do
    begin
     tempsesslist[sess1].exam[a].exam_no := sesslist[sessiona].exam[a].exam_no;
     tempsesslist[sess1].exam[a].size := sesslist[sessiona].exam[a].size
    end;
   sess2 := sessionb;
   arr[sess2] := arr_3[sessionb];
   for a := 1 to arr[sess2] do
    begin
     tempsesslist[sess2].exam[a].exam_no := sesslist[sessionb].exam[a].exam_no;
     tempsesslist[sess2].exam[a].size := sesslist[sessionb].exam[a].size
    end;
   arr_3[sessionb] := arr[sess1];
   for a := 1 to arr_3[sessionb] do
    begin
     sesslist[sessionb].exam[a].exam_no := tempsesslist[sess1].exam[a].exam_no;
     sesslist[sessionb].exam[a].size := tempsesslist[sess1].exam[a].size
    end;
   arr_3[sessiona] := arr[sess2];
   for a := 1 to arr_3[sessiona] do
    begin
     sesslist[sessiona].exam[a].exam_no := tempsesslist[sess2].exam[a].exam_no;
     sesslist[sessiona].exam[a].size := tempsesslist[sess2].exam[a].size
    end
end;
```

```
procedure findsession(var sesschosen,iteno : integer); { procedure to find the session in which
                                                         the exams will be swapped
                                                          in another session
with exams
which produce the
```

```
                                                          maximum net decrease }
var
   cpair,pair,totcla,excla,no1,no2,no      : integer;
   temp1,temp2,tempsum,temptotcla          : integer;
   picksess,num1,num2,a,b,c,d,e            : integer;
   taboo                          : array[1..max_session] of integer;

begin
   no := 1;
   taboo[no] := sesschosen;
   if sesschosen mod 2 = 1 then          { identifying the pair session }
    begin
     if sesschosen <> k then
      pair := sesschosen+1
     else
      pair := 0
    end
   else if sesschosen mod 2 = 0 then
    pair := sesschosen-1;
   if pair <> 0 then
    begin
     no := no + 1;
     taboo[no] := pair;
     e := 1;
     while e <= iteno do
      begin
           totcla := 0;
           for a := 1 to arr_3[pair] do
            for b := 1 to arr_3[sesschosen] do
             totcla := totcla +
arr_1[sesslist[pair].exam[a].exam_no,sesslist[sesschosen].exam[b].exam_no];
           picksess := pair;
           temp1 := 0;
           temp2 := 0;
           tempsum := 0;
           if sesschosen mod 2 = 1 then
            num1 := (sesschosen div 2) + 1
           else if sesschosen mod 2 = 0 then
            num1 := sesschosen div 2;
           temptotcla := totcla - same_day[num1];
           c := 1;
           while c <= k do
            begin
             d := 1;
             while d <= no do
              begin
               if c <> taboo[d] then
                d := d + 1
               else
                begin
                 d := max_exam;
                 c := c + 1
                end
              end;
             if d = no+1 then
              begin
               excla := 0;
               for a := 1 to arr_3[c] do
```

```
                begin
                 for b := 1 to arr_3[sesschosen] do
                  excla := excla +
arr_1[sesslist[c].exam[a].exam_no,sesslist[sesschosen].exam[b].exam_no]
                end;
               temp1 := excla - same_day[num1];
               if c mod 2 = 1 then
                cpair := c + 1
               else if c mod 2 = 0 then
                cpair := c - 1;
               if cpair <= k then
                begin
                 excla := 0;
                 for a := 1 to arr_3[pair] do
                  begin
                       for b := 1 to arr_3[cpair] do
                        excla := excla +
arr_1[sesslist[pair].exam[a].exam_no,sesslist[cpair].exam[b].exam_no]
                 end;
                if c mod 2 = 1 then
                 num2 := (c div 2) + 1
                else if c mod 2 = 0 then
                 num2 := c div 2;
                temp2 := excla - same_day[num2]
               end
              else temp2 := 0;
               tempsum := temp1 + temp2;
               if tempsum < temptotcla then { choose the session which produces
                begin                                    the maximum net decrease in the
                 temptotcla := tempsum;                  total same-day exams }
                 no1 := temp1;
                 no2 := temp2;
                 picksess := c
                end
              end;
             c := c + 1
            end;
           if picksess <> pair then
            begin
             swapsession(pair,picksess);
             if pair mod 2 = 1 then
              same_day[(pair div 2)+1] := same_day[(pair div 2)+1] + no1
             else if pair mod 2 = 0 then
              same_day[pair div 2] := same_day[pair div 2] + no1;
             if picksess mod 2 = 1 then
              begin
               if picksess <> k then
                same_day[(picksess div 2)+1] := same_day[(picksess div 2)+1] + no2
              end
             else if picksess mod 2 = 0 then
              same_day[picksess div 2] := same_day[picksess div 2] + no2;
             no := no + 1;
             taboo[no] := picksess
            end
           else
            begin
             e := max_exam;
             if pair mod 2 = 1 then
```

```
            same_day[(pair div 2)+1] := totcla
           else if pair mod 2 = 0 then
            same_day[pair div 2] := totcla
          end;
         e := e + 1
     end
   end
end;


procedure remainingexams(var uns_exam,sess : integer); { procedure to assign the remaining
of                                                       the exams which have not yet been
                                                              scheduled because they
clash only                                                          with each other
and not with the                                                 others }


var
   exam_j,a,b,c,d,e          : integer;

begin
   findmaxclique(uns_exam);
   b := 1;
   while b <= maxclique do
    begin
     c := 1;
     while c <= exam do
begin
     if examlist[c].no = cli[b] then
      begin
       exam_j := c;
       d := 1;
       while d <= sess do
        begin
         if examlist[exam_j].size <= sesslist[d].capacity then
          begin
           arr_3[d] := arr_3[d] + 1;
           assign(d,exam_j,arr_3[d]);
           totsameday := 0;
           findsession(d,iteration);
           for e := 1 to sess do
            totsameday := totsameday + same_day[e]
           d := max_session
          end
         else
          d := d + 1
        end;
       if d = sess+1 then
        begin
         sess := sess + 1;
         if sess <= totsess then
          begin
           arr_3[sess] := 1;
           assign(sess,exam_j,arr_3[sess]);
           totsameday := 0;
           findsession(sess,iteration);
           for e := 1 to sess do
```

```
    totsameday := totsameday + same_day[e]
   end
  else
   begin
    stop := true;
    allassigned := false;
    writeln('The number of sessions required exceeds the maximum');
   writeln('number of sessions available');
    writeln('You need more sessions');
    writeln
   end
  end;
                end;
                c := max_exam;
                b := b + 1
               end
              else
               c := c + 1
        end
      end
end;

begin
   write('course file : ');
   readln(name1);
   reset(inf1,name1);
   write('clash file : ');
   readln(name2);
   reset(inf2,name2);
write('room capacity file : ');
readln(inf3,name3);
reset(name3);
write('output file : ');
readln(name4);
rewrite(outf,name4);
count := 0;
while not eof(inf1) do
 begin
  readln(inf1,x,y);
  count := count + 1;
  crsrec.no := x;
  crsrec.size := y;
  crslist[count] := crsrec;
  if y > biggest_exam then
   biggest_exam := y
 end;
exam := x;
for i := 1 to exam do
 for j := 1 to exam do
  arr_1[i,j] := 0;f
Repeat
 readln(inf2,x,y,z);
 arr_1[x,y] := arr_1[x,y] + z;
 arr_1[y,x] := arr_1[x,y]
Until eof(inf2);
for i := 1 to exam do
 begin
  count := 0;
  for j := 1 to exam do
```

```
  begin
       if j <> i then
        begin
         if arr_1[i,j] <> 0 then
          count := count + 1
        end
   end;
  arr_2[i] := count
 end;
for i := 1 to exam do
 begin
  examrec.no := i;
  examrec.size := crslist[i].size;
  examrec.degree := arr_2[i];
  examlist[i] := examrec
 end;
findmaxclique(exam);
capacity := 0;
Repeat
 readln(inf3,x,y);
 sessrec.room_size[x] := y;
 capacity := capacity + sessrec.room_size[x]
Until eof(inf3);
room := x;
if biggest_exam <= capacity then        { the biggest exam must not exceed the room
 begin                                    capacity }
 write('Total no. of sessions available = ');
 readln(totsess);
 if totsess >= initsess then            { the maximum number of sessions allocated must
be                                        larger than the size of the maximum clique }
     begin
      iteration := 3;                   { the number of times the exams in sessions are
       for i := 1 to totsess do           swapped is set to 3 }
        begin
         sessrec.capacity := capacity;
         sesslist[i] := sessrec
        end;
       if initsess mod 2 = 1 then
        begin
         for i := 1 to initsess-2 do
          begin
           if i mod 2 = 1 then
            same_day[(i div 2)+1] := arr_1[cli[i],cli[i+1]]
          end;
         same_day[(initsess div 2)+1] := 0
        end
else if initsess mod 2 = 0 then
 begin
  for i := 1 to initsess-1 do
   begin
    if i mod 2 = 1 then
     same_day[(i div 2)+1] := arr_1[cli[i],cli[i+1]]
   end
 end;
k := 0;                                 { assigning the clique exams to their respective
while k < initsess do                     sessions }
 begin
  k := k + 1;
```

\ (

```
i := 1;
while i <= exam do
 begin
  if examlist[i].no = cli[k] then
   begin
    exam_i := i;
    arr_3[k] := 1;
    assign(k,exam_i,arr_3[k]);
    i := max_exam
   end
  else
   i := i + 1
  end
 end;
allassigned := true;
stop := false;
while stop <> true do                { the process of finding the critical and noncritical
exams
 begin                                 and assigning them to their chosen sessions }
  count1 := 0;
  count2 := 0;
  for x := 1 to k do
   begin
    sess_cap := sesslist[x].capacity;
    num := 0;
    i := 1;
    while i <= exam do
     begin
      if examlist[i].no <> 0 then
begin
    y := 1;
    while y <= arr_3[x] do
     begin
      if arr_1[examlist[i].no,sesslist[x].exam[y].exam_no] <> 0 then
       begin
        y := max_exam;
        i := i + 1
       end;
      y := y + 1
     end;
    if y = arr_3[x]+1 then
     begin
      if examlist[i].size <= sess_cap then
       begin
        count := 0;
        for z := 1 to k do
begin
 if z <> x then
  begin
      l := 1;
      while l <= arr_3[z] do
       begin
        if arr_1[examlist[i].no,sesslist[z].exam[l].exam_no] <> 0 then
         begin
          count := count + 1;
          l := max_exam
         end;
        l := l + 1
```

```
            end
        end
       end;
      if count = k-1 then
       begin
        sess_cap := sess_cap - examlist[i].size;
        num := num + 1;
        newcli_no[x,num] := i
       end;
      i := i + 1
     end
    else
     i := i + 1
   end;
  if num = 1 then
   begin
    count1 := count1 + 1;
    sessnum_1[count1] := x
   end
  else if num > 1 then
   begin
    count2 := count2 + 1;
    sessnum_2[count2] := x;
    findnewclique(num,sessnum_2[count2]);
    submaxclq[count2] := mclq;
    maxnum[count2] := num
   end
  end;
 if count2 <> 0 then                    { more than one critical exams found in any session }
  begin
   biggestsubmaxclq := 0;
   countsess := 0;
   for l := 1 to count2 do
    begin
     if submaxclq[l] > biggestsubmaxclq then
      begin
          biggestsubmaxclq := submaxclq[l];
          countsess := 1;
          sess_bsm[countsess] := sessnum_2[l]
      end
     else if submaxclq[l] = biggestsubmaxclq then
      begin
          countsess := countsess + 1;
          sess_bsm[countsess] := sessnum_2[l]
      end
    end;
  if biggestsubmaxclq = 1 then              { the biggest maximum subclique is 1 in any
   begin                                      session }
    for l := 1 to count2 do
     begin
          mark1[l] := sesslist[sess_bsm[l]].exam[1].exam_no;
          for s := 1 to maxnum[l] do
           examinsess[l,s] := newcli_no[sess_bsm[l],s]
     end;
```

```
if count1 <> 0 then
 begin
      for l := 1 to count1 do
       begin
        mark2[l] := sesslist[sessnum_1[l]].exam[1].exam_no;
        examno[l] := newcli_no[sessnum_1[l],1]
       end
 end;
for l := 1 to count2 do
 begin
      x := 1;
      while x <= k do
       begin
        if sesslist[x].exam[1].exam_no = mark1[l] then
         begin
          sess_bsm[l] := x;
          x := max_session
         end
        else
         x := x + 1
       end;
      for s := 1 to maxnum[l] do                    { assign all of the maximum
subclique
       begin                                          to their respective sessions }
        arr_3[sess_bsm[l]] := arr_3[sess_bsm[l]] + 1;
        assign(sess_bsm[l],examinsess[l,s],arr_3[sess_bsm[l]])
       end;
      totsameday := 0;
      findsession(sess_bsm[l],iteration);          {
      for x := 1 to (k div 2) do
       totsameday := totsameday + same_day[x]
 end;
if count1 <> 0 then
 begin
      for l := 1 to count1 do
       begin
        x := 1;
        while x <= k do
         begin
          if sesslist[x].exam[1].exam_no = mark2[l] then
           begin
            sessnum_1[l] := x;
            x := max_session
           end
          else
           x := x + 1
         end;
        arr_3[sessnum_1[l]] := arr_3[sessnum_1[l]] + 1;
        assign(sessnum_1[l],examno[l],arr_3[sessnum_1[l]]);
        totsameday := 0;
        findsession(sessnum_1[l],iteration);
        for x := 1 to (k div 2) do
         totsameday := totsameday + same_day[x]
       end
  end
 end
else if biggestsubmaxclq > 1 then      { the biggest maximum subcliques is larger
 begin                                   than 1 in any session }
```

```
if countsess = 1 then                    { the biggest maximum subclique is unique,
 begin                                     assign the exam with the largest degree to
     for l := 1 to biggestsubmaxclq do     the existing session and the rest to new
      begin                                sessions }
       for s := 1 to exam do
        begin
         if examlist[s].no = newclique[sess_bsm[countsess],l] then
          exno[l] := s
        end
      end;
     arr_3[sess_bsm[countsess]] := arr_3[sess_bsm[countsess]] + 1;
     assign(sess_bsm[countsess],exno[1],arr_3[sess_bsm[countsess]]);
     totsameday := 0;
     findsession(sess_bsm[countsess],iteration);
     for x := 1 to (k div 2) do
      totsameday := totsameday + same_day[x];
     for l := 2 to biggestsubmaxclq do
      begin
       k := k + 1;
       if k >= totsess then
        begin
         arr_3[k] := 1;
         assign(k,exno[l],arr_3[k]);
         totsameday := 0;
         findsession(k,iteration);
         for x := 1 to (k div 2) do
          totsameday := totsameday + same_day[x]
        end
       else
                    begin
                     stop := true;
                     allassigned := false;
                     writeln('The number of sessions required exceeds the maximum');
                     writeln('number of sessions available');
                      writeln('You need more sessions');
                      writeln
                     end
                   end
              end
             else if countsess > 1 then          { the biggest maximum subclique is not
unique,
             begin                                 choose the session with the largest sum of
                 for l := 1 to countsess do        degrees. Assign the exam with the largest
                  begin                            degree to the existing session and the rest
                   countdeg := 0;                  to new sessions }
     for s := 1 to biggestsubmaxclq do
      begin
       for i := 1 to exam do
        begin
         if examlist[i].no <> 0 then
          begin
              if arr_1[examlist[i].no,newclique[sess_bsm[l],s]] <> 0 then
               countdeg := countdeg + 1
          end
        end
      end;
     sumdegbsm[l] := countdeg
    end;
```

```
maxdegbsm := 0;
for l := 1 to countsess do
 begin
  if sumdegbsm[l] > maxdegbsm then
   begin
    maxdegbsm := sumdegbsm[l];
    sessno := sess_bsm[l]
   end
 end;
for l := 1 to biggestsubmaxclq do
 begin
  for s := 1 to exam do
   begin
    if examlist[s].no = newclique[sessno,l] then
     exno[l] := s
   end
 end;
arr_3[sessno] := arr_3[sessno] + 1;
assign(sessno,exno[1],arr_3[sessno]);
totsameday := 0;
findsession(sessno,iteration);
for x := 1 to (k div 2) do
 totsameday := totsameday + same_day[x];
for l := 2 to biggestsubmaxclq do
 begin
  k := k + 1;
  if k >= totsess then
   begin
    arr_3[k] := 1;
    assign(k,exno[l],arr_3[k]);
    totsameday := 0;
    findsession(k,iteration);
    for x := 1 to (k div 2) do
     totsameday := totsameday + same_day[x]
   end
  else
   begin
    stop := true;
    allassigned := false;
   writeln('The number of sessions required exceeds the maximum');
  writeln('number of sessions available');
   writeln('You need more sessions');
   writeln
   end
 end
end
  end
 end
else if count2 = 0 then                    { one critical exam found in any session.
 begin                                       Assign each of them to their respective
  if count1 <> 0 then                        sessions }
   begin
    if count1 = 1 then
     begin
        arr_3[sessnum_1[count1]] := arr_3[sessnum_1[count1]] + 1;
        assign(sessnum_1[count1],newcli_no[sessnum_1[count1],1],arr_3[sessnum_
1[count1]]);
        totsameday := 0;
```

```
      findsession(sessnum_1[count1],iteration);
      for x := 1 to (k div 2) do
       totsameday := totsameday + same_day[x]
 end
else if count1 > 1 then
 begin
      for l := 1 to count1 do
       begin
        mark2[l] := sesslist[sessnum_1[l]].exam[1].exam_no;
        examno[l] := newcli_no[sessnum_1[l],1]
       end;
      for l := 1 to count1 do
 begin
 x := 1;
 while x <= k do
  begin
   if sesslist[x].exam[1].exam_no = mark2[l] then
    begin
     sessnum_1[l] := x;
     x := max_session
    end
   else
    x := x + 1
  end;
 arr_3[sessnum_1[l]] := arr_3[sessnum_1[l]] + 1;
 assign(sessnum_1[l],examno[l],arr_3[sessnum_1[l]]);
 totsameday := 0;
 findsession(sessnum_1[l],iteration);
          for x := 1 to (k div 2) do
           totsameday := totsameday + same_day[x]
         end
   end
 end
else if count1 = 0 then                    { no critical exams found }
 begin
  for l := 1 to exam do
   begin
        if examlist[i].no <> 0 then
         begin
          deg[i] := arr_2[i];
          for x := 1 to k do
           begin
            for y := 1 to arr_3[x] do
             begin
              if arr_1[examlist[i].no,sesslist[x].exam[y].exam_no] <> 0 then
               deg[i] := deg[i] - 1
             end
           end
         end
        else
         deg[i] := -exam
 end;
for i := 1 to exam do                      { exams are sorted according to decreasing
 begin                                       order of recursive largest degree }
      if examlist[i].no <> 0 then
       sortrec.no := i
      else
       sortrec.no := 0;
```

```
        sortrec.edge := deg[i];
        sortlist[i] := sortrec
    end;
for j := exam downto 2 do
 for i := 1 to j-1 do
        if sortlist[i].edge < sortlist[i+1].edge then
         begin
          sortrec := sortlist[i];
          sortlist[i] := sortlist[i+1];
          sortlist[i+1] := sortrec
         end;
exam_i := 0;
max_count := 0;
for t := 1 to exam do                    { choose the exam which clashes with
 begin                                     most of the available sessions }
      if sortlist[t].no <> 0 then
       begin
        count := 0;
        for l := 1 to k do
         begin
          y := 1;
          while y <= arr_3[l] do
           begin
            if arr_1[sortlist[t].no,sesslist[l].exam[y].exam_no] <> 0 then
             begin
                  count := count + 1;
                  y := maxint
             end
            else
              y := y + 1
           end;
          if count > max_count then
           begin
            max_count := count;
            exam_i := t
           end
         end
 end;
if exam_i > 0 then
 begin
      sessct := 0;
      s := 1;
      while s <= k do
       begin
        u := 1;
        while u <= arr_3[s] do
         begin
          if arr_1[examlist[sortlist[exam_i].no].no,sesslist[s].exam[u].exam_no] <> 0
then
           u := maxint
          else
           u := u + 1
         end;
        if u = arr_3[s]+1 then
         begin
          if examlist[sortlist[exam_i].no].size <= sesslist[s].capacity then
           begin
```

```
                    sessct := sessct + 1;
                    multisess[sessct] := s
                   end
                  end;
                 s := s + 1
                end;
               if sessct <> 0 then                { the exams can be assigned to more than
                begin                               one sessions. Choose the session which
                 biggestclash := -maxint;           clashes with most of the unscheduled
exams }
                 num := 0;
                 for x := 1 to sessct do
                  begin
                   count := 0;
                   for i := 1 to exam do
                    begin
                     if examlist[i].no <> 0 then
                      begin
                           y := 1;
                           while y <= arr_3[multisess[x]] do
                            begin
                             if arr_1[sesslist[multisess[x]].exam[y].exam_no,examlist[i].no] <> 0
then
                              begin
                               count := count + 1;
                               y := max_exam
                              end
                             else
                              y := y + 1
                            end
                      end
                    end;
                   if count > biggestclash then
                    begin
                     num := 1;
                     chsess[num] := multisess[x];
                     biggestclash := count
                    end
                   else if count = biggestclash then
                    begin
                     num := num + 1;
                     chsess[num] := multisess[x]
                    end
                  end;
                 maxtotdeg := -maxint;
                 if num > 1 then
                  begin
                   for t := 1 to num do
                    begin
                     totdeg := 0;
                     for y := 1 to arr_3[chsess[t]] do
                      totdeg := totdeg + arr_2[sesslist[chsess[t]].exam[y].exam_no];
                     if totdeg > maxtotdeg then
                      begin
                           maxtotdeg := totdeg;
                           chosensess := chsess[t]
                      end
                    end
                  end
```

```
 end
 else chosensess := chsess[1];
 arr_3[chosensess] := arr_3[chosensess] + 1;
 assign(chosensess,sortlist[exam_i].no,arr_3[chosensess]);
 totsameday := 0;
 findsession(chosensess,iteration);
 for x := 1 to (k div 2) do
  totsameday := totsameday + same_day[x]
end
else                          { the exam clashes with all sessions. Assign it
 begin                          to a new session }
 k := k + 1;
 if k <= totsess then
  begin
   arr_3[k] := 1;
   assign(k,sortlist[exam_i].no,arr_3[k]);
   totsameday := 0;
   findsession(k,iteration);
   for x := 1 to (k div 2) do
    totsameday := totsameday + same_day[x]
  end
 else
  begin
   stop := true;
   allassigned := false;
  writeln('The number of sessions required exceeds the maximum');
 writeln('number of sessions available');
   writeln('You need more sessions');
   writeln
          end
         end
  end
 else
  begin
      z := 0;                 { find whether there are exams which have not yet
      for t := 1 to exam do     been scheduled }
       begin
        if examlist[t].no <> 0 then
         z := z + 1
       end;
      if z > 0 then
       remainingexams(z,k)
      else
       stop := true
  end
end
  end
 end;
if allassigned <> false then          { determine the total number of same-day
 begin                                  exams }
  totsameday := 0;
  if k mod 2 = 1 then
   begin
    for l := 1 to (k div 2)+1 do
     totsameday := totsameday + same_day[l]
   end
  else if k mod 2 = 0 then
   begin
```

```
            for l := 1 to (k div 2) do
             totsameday := totsameday + same_day[l];
            end;
           extrasess := totsess - k;
           if extrasess <> 0 then                    { the minimum number of sessions obtained
            begin                                      is less than the maximum number of
sessions
            writeln('There are ',extrasess:2,' extra sessions available');      allowable }
            write('How many sessions to add? : ');
            readln(addsess);
           taboono := 0;
           sessno := 1;
           if k mod 2 = 1 then                       { the minimum number of sessions is odd }
            begin
             k := k + 1;
             arr_3[k] := 0;
             improvement := true;
             while improvement <> false do
              begin
                  chosenexam := 0;
                  mindiff := maxint;
                  for l := 1 to (k-2) do
                   begin
                    sess_cap := capacity;
                    for x := 1 to arr_3[l] do
                     begin
                      if sesslist[l].exam[x].size <= sess_cap then
                       begin
                        donotclash := false;
                        if arr_3[k] <> 0 then
                         begin
                             z := 1;
                             while z <= arr_3[k] do
                              begin
                               if arr_1[sesslist[l].exam[x].exam_no,sesslist[k].exam[z].exam_no] <>
0 then
                                z := max_exam
                               else
                                z := z + 1
                              end;
                             if z = arr_3[k]+1 then
                              donotclash := true
                         end
                        else
                         donotclash := true;
                        if donotclash = true then
                         begin
                             tot1 := 0;
                             for y := 1 to arr_3[k-1] do
                              tot1 := tot1 + arr_1[sesslist[l].exam[x].exam_no,sesslist[k-
1].exam[y].exam_no];
                             tot2 := 0;
                             if l mod 2 = 1 then
                              begin
                               for y := 1 to arr_3[l+1] do
                                tot2 :=
arr_1[sesslist[l].exam[x].exam_no,sesslist[l+1].exam[y].exam_no]
                              end
```

```
                    else if l mod 2 = 0 then
                     begin
                      for y := 1 to arr_3[l-1] do
                       tot2 := tot2 + arr_1[sesslist[l].exam[x].exam_no,sesslist[l-
1].exam[y].exam_no]
                     end;
                    diff := tot1 - tot2;
                    if (diff < 0) and (diff < mindiff) then        { choose the exam to be
                     begin                                          reassigned to the new
session
                      mindiff := diff;                             which produces the
maximum
                      chosenexam := sesslist[l].exam[x].exam_no;   net
decrease in
                      examsess := l;                               same-day
                      exampos := x;                                exams }
                      examtot1 := tot1;
                      examtot2 := tot2
                     end
                   end
                  end
                 end;
                if chosenexam <> 0 then
                 begin
                  arr_3[k] := arr_3[k] + 1;
                  sesslist[k].exam[arr_3[k]].exam_no := chosenexam;
                  sesslist[k].exam[arr_3[k]].size := crslist[chosenexam].size;
                  same_day[(k div 2)] := same_day[(k div 2)] + examtot1;
                  if examsess mod 2 = 1 then
                   same_day[(examsess div 2)+1] := same_day[(examsess div 2)+1] -
examtot2
                  else if examsess mod 2 = 0 then
                   same_day[(examsess div 2)] := same_day[(examsess div 2)] - examtot2;
                  if exampos <> arr_3[examsess] then
                   begin
                    for y := exampos to arr_3[examsess]-1 do
                     begin
                      sesslist[examsess].exam[y].exam_no :=
sesslist[examsess].exam[y+1].exam_no;
                      sesslist[examsess].exam[y].size := sesslist[examsess].exam[y+1].size
                     end;
                    arr_3[examsess] := arr_3[examsess] - 1
                   end
                  else
                    arr_3[examsess] := arr_3[examsess] - 1;
                   totsameday := totsameday + mindiff
                  end
                 else
                  improvement := false
               end
              end
             else if k mod 2 = 0 then                              { the minimum number of sessions is
              begin                                                 even }
               maxsameday := 0;
               for l := 1 to (k div 2) do                          { choose the session with lower total
                begin                                               degree in the day with the largest
                    if same_day[l] > maxsameday then                same-day exams }
```

```
        begin
         maxsameday := same_day[l];
         choosesameday := l
        end
 end;
totsameday := totsameday - maxsameday;
same_day[choosesameday] := 0;
tot1 := 0;
for y := 1 to arr_3[choosesameday*2-1] do
 tot1 := tot1 + arr_2[sesslist[choosesameday*2-1].exam[y].exam_no];
tot2 := 0;
for y := 1 to arr_3[choosesameday*2] do
 tot2 := tot2 + arr_2[sesslist[choosesameday*2].exam[y].exam_no];
if tot1 < tot2 then
 chosensess := choosesameday*2-1
else
 chosensess := choosesameday*2;
taboono := taboono + 1;
taboosess[taboono] := chosensess;
k := k + 1;
arr_3[k] := arr_3[chosensess];
arr_3[chosensess] := 0;
for y := 1 to arr_3[k] do
 begin
      sesslist[k].exam[y].exam_no := sesslist[chosensess].exam[y].exam_no;
      sesslist[k].exam[y].size := sesslist[chosensess].exam[y].size
 end;
same_day[(k div 2)+1] := 0;
if chosensess mod 2 = 1 then
 pair := chosensess+1
else if chosensess mod 2 = 0 then
 pair := chosensess-1;
taboono := taboono + 1;
taboosess[taboono] := pair;
improvement := true;
while improvement <> false do
 begin
      chosenexam := 0;
      maxtotexcla := 0;
      for l := 1 to (k-1) do
       begin
        sess_cap := capacity;
        u := 1;
        while u <= taboono do
         begin
         if l = taboosess[u] then
          u := max_exam
         else
          u := u + 1
        end;
       if u = taboono+1 then
        begin
         for x := 1 to arr_3[l] do
          begin
           if sesslist[l].exam[x].size <= sess_cap then
            begin
                donotclash := false;
                 if arr_3[k] <> 0 then
```

```
                                begin
                                 z := 1;
                                 while z <= arr_3[k] do
                                  begin
                                   if arr_1[sesslist[l].exam[x].exam_no,sesslist[k].exam[z].exam_no]
<> 0 then
                                    z := max_exam
                                   else
                                    z := z + 1
                                  end;
                                 if z = arr_3[k]+1 then
                                  donotclash := true
                                 end
                                else
                                 donotclash := true;
                                if donotclash = true then
                                 begin
                                  tot1 := 0;
                                  for y := 1 to arr_3[k-1] do
                                   tot1 := tot1 + arr_1[sesslist[l].exam[x].exam_no,sesslist[k-
1].exam[y].exam_no];
                                  tot2 := 0;
                                  if l mod 2 = 1 then
                                   begin
                                    for y := 1 to arr_3[l+1] do
                                     tot2 := tot2 +
arr_1[sesslist[l].exam[x].exam_no,sesslist[l+1].exam[y].exam_no]
                                   end
                                  else if l mod 2 = 0 then
                                   begin
                                    for y := 1 to arr_3[l-1] do
                                     tot2 := tot2 + arr_1[sesslist[l].exam[x].exam_no,sesslist[l-
1].exam[y].exam_no]
                                   end;
                                  diff := tot1 - tot2;
                                  if (diff < 0) and (diff < mindiff) then      { choose the exam to be
                                   begin                                       reassigned to the empty
                                    mindiff := diff;                           session which produces
the
                                    chosenexam := sesslist[l].exam[x].exam_no;          maximum
net
                                    examsess := l;                                      decrease in
the
                                    exampos := x;                                       same-day
                                    examtot1 := tot1;                                   exams }
                                    examtot2 := tot2
                                     end
                                   end
                                  end
                                 end;
                                if chosenexam <> 0 then
                                 begin
                                  arr_3[chosensess] := arr_3[chosensess] + 1;
                                  sesslist[chosensess].exam[arr_3[chosensess]].exam_no := chosenexam;
                                  sesslist[chosensess].exam[arr_3[chosensess]].size :=
crslist[chosenexam].size;
                                  if examsess mod 2 = 1 then
```

```
                    same_day[(examsess div 2)+1] := same_day[(examsess div 2)+1] -
maxtotexcla
                   else if examsess mod 2 = 0 then
                    same_day[(examsess div 2)] := same_day[(examsess div 2)] -
maxtotexcla;
                   if exampos <> arr_3[examsess] then
                    begin
                     for t := exampos to arr_3[examsess]-1 do
                      begin
                          sesslist[examsess].exam[t].exam_no :=
sesslist[examsess].exam[t+1].exam_no;
                          sesslist[examsess].exam[t].size := sesslist[examsess].exam[t+1].size
                      end;
                     arr_3[examsess] := arr_3[examsess] - 1
                    end
                   else
                    arr_3[examsess] := arr_3[examsess] - 1;
                   totsameday := totsameday - maxtotexcla;
                  end
                 else
                  improvement := false
                end
   end;
  sessno := sessno + 1
 end;
writeln('session = ',k:2,' , Total number of same-day exams = ',totsameday:2);
for l := 1 to k do
 begin
for y := arr_3[l] downto 2 do
 for x := 1 to y-1 do
      if sesslist[l].exam[x].size < sesslist[l].exam[x+1].size then
       begin
        schedulerec := sesslist[l].exam[x];
        sesslist[l].exam[x] := sesslist[l].exam[x+1];
        sesslist[l].exam[x+1] := schedulerec
       end;
for x := 1 to arr_3[l] do                    { assign exams in each session to rooms }
 begin
      num := 0;
      z := 1;
      while z <= room do
       begin
        if sesslist[l].exam[x].size <= sesslist[l].room_size[z] then
         begin
          num := num + 1;
          sesslist[l].exam[x].room_no[num] := z;
          sesslist[l].exam[x].examroom[num] := sesslist[l].exam[x].size;
          sesslist[l].room_size[z] := sesslist[l].room_size[z] - sesslist[l].exam[x].size;
          z := max_room
         end
        else
         z := z + 1
       end;
if z = room+1 then
 begin
 t := 1;
 while t <= room do
  begin
```

```
                    if sesslist[l].room_size[t] > 0 then
                     begin
                      if sesslist[l].exam[x].size > sesslist[l].room_size[t] then
                       begin
                            num := num + 1;
                            sesslist[l].exam[x].room_no[num] := t;
                            sesslist[l].exam[x].examroom[num] := sesslist[l].room_size[t];
                            sesslist[l].exam[x].size := sesslist[l].exam[x].size -
sesslist[l].room_size[t];
                            sesslist[l].room_size[t] := 0;
                            t := t + 1
                       end
                      else
                       begin
                            num := num + 1;
                            sesslist[l].exam[x].room_no[num] := t;
                            sesslist[l].exam[x].examroom[num] := sesslist[l].exam[x].size;
                            sesslist[l].room_size[t] := sesslist[l].room_size[t] -
sesslist[l].exam[x].size;
                            t := max_room
                       end
                     end
                    else
                     t := t + 1
                   end
                  end;
                 roomnum[l,x] := num
               end
              end;
             for i := 1 to exam do                          { reschedule sessions to days in
order
              begin                                          to have large exams at an early
stage
               temprec.no := crslist[i].no;                  of the examination period }
               temprec.size := crslist[i].size;
               templist[i] := temprec
              end;
             for j := exam downto 2 do
              for i := 1 to j-1 do
               if templist[i].size < templist[i+1].size then
                begin
                     temprec := templist[i];
                     templist[i] := templist[i+1];
                     templist[i+1] := temprec
                end;
             if k mod 2 = 1 then
              day := (k div 2) + 1
             else if k mod 2 = 0 then
              day := k div 2;
             for l := 1 to day do
              begin
               i := 1;
               while i <= exam do
                begin
                     if templist[i].no <> 0 then
                      begin
                       x := 1;
                       while x <= k do
```

```
begin
 y := 1;
 while y <= arr_3[x] do
  begin
   if sesslist[x].exam[y].exam_no = templist[i].no then
    begin
         sessday[l,1] := x;
         if x mod 2 = 1 then
          sessday[l,2] := x + 1
         else if x mod 2 = 0 then
          sessday[l,2] := x - 1;
         for s := 1 to 2 do
          begin
           for t := 1 to arr_3[sessday[l,s]] do
            begin
             for z := 1 to exam do
              begin
               if sesslist[sessday[l,s]].exam[t].exam_no = templist[z].no then
                templist[z].no := 0
              end
            end
          end;
         i := max_exam;
         y := max_exam;
         x := max_session
    end
   else
    y := y + 1
  end;
 x := x + 1
          end
         end
        else
         i := i + 1
   end
  end;
 for l := 1 to day do                                    { the output which lists the exams
  begin                                                    together with their rooms and the
   writeln(outf,'EXAMS IN DAY ',l:2);                      number of students in each room }
   writeln(outf,'***************');
   writeln(outf);
   writeln(outf,'Number of same-day exams for day ',l:2,' = ',same_day[l]:2);
   writeln(outf);
   for x := 1 to 2 do
    begin
         totstusess := 0;
   if x = 1 then
    begin
     write(outf,'MORNING SESSION : ');
     writeln(outf,'EXAM':15,'ROOM':15,'STUDENT NO':20);
     writeln(outf,'========':35,'========':15,'==============':20)
    end
   else if x = 2 then
    begin
     write(outf,'AFTERNOON SESSION : ');
     writeln(outf,'EXAM':13,'ROOM':15,'STUDENT NO':20);
     writeln(outf,'========':35,'========':15,'==============':20)
    end;
```

```
for y := 1 to arr_3[sessday[l,x]] do
 begin
  for z := 1 to roomnum[sessday[l,x],y] do
   begin
    write(outf,sesslist[sessday[l,x]].exam[y].exam_no:32);
    write(outf,sesslist[sessday[l,x]].exam[y].room_no[z]:15);
    writeln(outf,sesslist[sessday[l,x]].exam[y].examroom[z]:17);
    totstusess := totstusess + sesslist[sessday[l,x]].exam[y].examroom[z]
   end
 end;
writeln(outf);
writeln(outf,'Total no. of students = ',totstusess:2);
writeln(outf)
              end
             end;
             writeln(outf);
             writeln(outf,'Total number of same-day exams = ',totsameday:2)
            end
          end
   end
  else
   writeln('session should be more than or equal to ',initsess:1)
 end
else
 begin
  writeln('The biggest exam exceeds the capacity of any session');
  writeln('You need at least ',biggest_exam:1,' seats')
 end;
    close(inf1);
    close(inf2);
    close(inf3);
    close(outf)
end.
```

APPENDIX 8

PAPER PRESENTED AT THE OR YOUNG RESEARCHERS FORUM AT SOUTHAMPTON UNIVERSITY ON 17 - 18 APRIL 1997.

AN ALTERNATIVE METHOD FOR SOLVING EXAMINATION TIMETABLING PROBLEMS

Wan Rosmanira Ismail and David G. Johnson

Loughborough University Business School

Ashby Road, Loughborough

Leics., LE11 3TU

Abstract

An algorithm for finding the minimum number of sessions required to assign all examinations without creating any conflicts, i.e. no students has to take more than one examination at any one time is introduced. The algorithm does not however consider any side constraints. Computational results for eleven real life problems are provided and the method is then compared with one existing method. The algorithm is coded in Standard Pascal and run on a Hewlett-Packard HP-UX 9000/755 computer.

Introduction

The simplest form of an examination timetabling problem is to produce a clash-free or conflict-free timetable without considering any side constraints. This particular problem which can be defined as a graph colouring problem is known to be NP-Complete where the computing time required grows exponentially with the number of examinations. Therefore, solving a large scale timetabling problem using a graph colouring method

becomes unworkable. To tackle the problem, graph colouring heuristics were developed and applied to practical timetabling problems (Desroches, Laporte and Rousseau[5], Mehta[8], Laporte and Desroches[7], Carter, Laporte and Chinneck[2]). The examinations are initially sorted in order of decreasing difficulty according to some criterion and they will then be sequentially assigned to sessions without creating conflicts.

The drawback of using this approach is that the number of sessions produced can sometimes be excessive. Recently, a clique initialisation strategy has been incorporated in the examination timetabling system to improve the method (Carter, Laporte and Lee[3]). A clique is a set of mutually conflicting examinations and represents the most difficult group of examinations to be scheduled. The size of the largest clique can be used as a lower bound to the number of sessions required to schedule all examinations, i.e. the minimum number of sessions is at least as large as the size of the largest clique. Another significant approach used to alleviate this problem is to implement backtracking procedures (Carter, Laporte and Lee[3], Laporte and Desroches[7], Hertz[6]). Whenever an examination to be scheduled is in conflict with every available session, some assignments already made will be undone in order to schedule that examination. Any examination removed will then be rescheduled.

New Algorithmic Rule

A new method is developed to schedule examinations to a minimum number of sessions without causing any clash for the students. The algorithm does include the clique initialisation strategy but without the use of backtracking procedures. The first step in the algorithm is to determine the largest clique in the graph. Each of the exams which form the clique is then assigned to only one of the **S** sessions where **S** is the size of the largest clique.

The next step involves in finding critical exams amongst the unscheduled exams. A critical exam is the one which can only be assigned to only one of the **S** sessions, i.e. the exam

does not clash with one member of the clique only. If such exams can be found then find the maximum sub-clique (MAXSC) in each of the sessions. $MAXSC_i$ is the maximum clique found among the critical exams associated with session *i*. If each of the MAXSCs is equal to one, i.e. all of the critical exams do not clash with each other, then assign them to the appropriate sessions. If there is only one MAXSC = K (where K is the biggest value and >1) then create K-1 new sessions. Assign the exam with the largest degree to the existing session and the rest to the new sessions. If there are more than one MAXSC = K , then calculate the sum of degrees of exams in each session. Choose the session with the largest sum of degrees and the process is repeated as previously. This process can be explained more clearly with an example.

Example 1

Session	Scheduled exams	Critical exams
S1	X1	C1
S2	X2	-
S3	X3	C2,C3,C4
S4	X4	C5
S5	X5	C6,C7
S6	X6	-
S7	X7	C8,C9

There is more than one critical exam in S3, S5 and S7, therefore find MAXSC. Suppose that MAXSC3 represents the maximum sub-clique in S3, MAXSC5 in S5 and MAXSC7 in S7. There can be three possible different outcomes:-

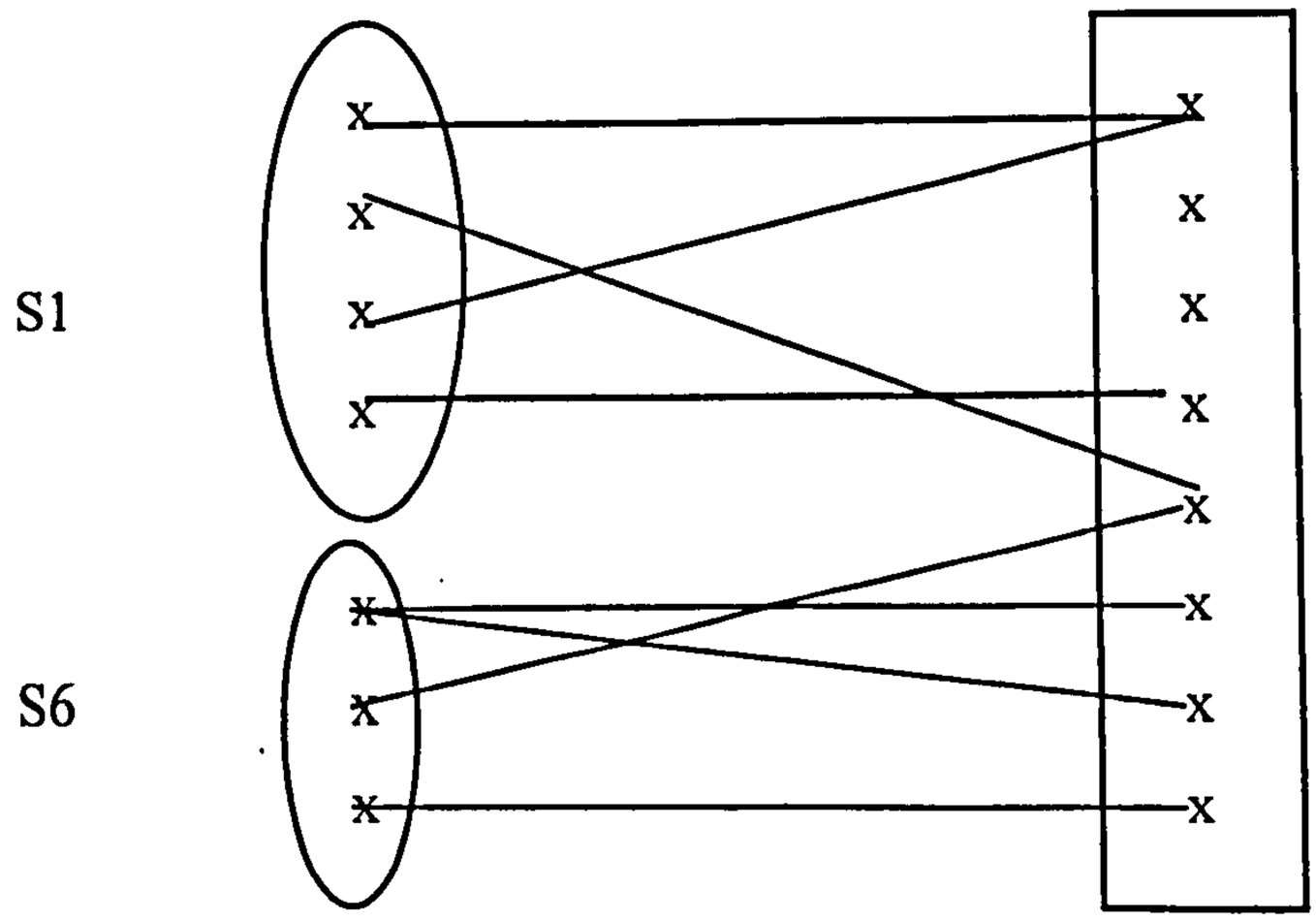
1) Let's say MAXSC3 = MAXSC5 = MAXSC7 = 1. This means that critical exams C2, C3, and C4 do not clash with each other so they can be assigned to S3. Similarly, C6 and C7 can be assigned to S5 and C8 and C9 to S7. Since no new session is created then all other critical exams can be assigned to their sessions (C1 to S1 and C5 to S4).

2) Let's say that MAXSC3 = 3, MAXSC5 = 2 and MAXSC7 = 1, then choose MAXSC3 since K = 3 is the largest. Create 2 new sessions S8 and S9. Sort C2, C3 and C4 in decreasing order of degree and assign them to the sessions S3, S8 and S9 successively. The remaining critical exams are put back in the unscheduled list.
3) Let's say MAXSC3 = MAXSC5 = 2 and MAXSC7 = 1. Calculate the sum of degrees of critical exams in S3 and S5 and choose the one with the largest sum of degrees. The process is then repeated as in (2).

If no critical exam can be found then the unscheduled exams are sorted according to decreasing order of recursive largest degree. The exams are initially sorted in decreasing order of degree and after the exam with the largest degree has been scheduled, the exam is removed from the list and the degrees of the rest of exams are recalculated and the list is then resorted. From the list, choose the exam which clashes with most of the available sessions. If the exam clashes with all sessions, then the exam will be assigned to a new session. If it can be put in two or more sessions, then the clash-free session which clashes with most of the unscheduled exams is chosen. Again, an example is provided to illustrate the process.

Example 2

Consider the sub-graph below:-



Suppose that an exam (exam_i) does not clash with any of the exams in S1 and S6. From the graph, S1 clashes with 3 of unscheduled exams and S6 clashes with 4 so assign exam_i to S6.

The steps of the algorithm are as follows :-

Step1 : Find the maximum clique of the examinations. Assign each of them to one of **S** sessions where **S** is the size of the maximum clique.

Step2 : Scan the list of unscheduled exams for critical exams. If such exams can be found then go to Step 3, otherwise go to Step 7.

Step3 : If there is only one critical exam which can be assigned in any one particular session, then assign each of them to the appropriate sessions and go to Step 9. If there exists more than one critical exams in one particular session, go to Step 4.

Step4 : Find the maximum sub-clique (MAXSC). If MAXSC = 1, then assign all critical exams to the appropriate sessions and go to Step 9. Otherwise, go to Step 5.

Step5 : If MAXSC = K (where K > 1) is unique then create K-1 new sessions. Assign the exam with the largest degree to the existing session and assign the rest to the new sessions and go to Step 9. If MAXSC is not unique, go to Step 6.

Step6 : Calculate the sum of degrees of the critical exams and choose the ones with the largest. Go to Step 5.

Step7 : Order the unscheduled exams according to the recursive largest degree and choose the one from the list which clashes with most of the available sessions. If the exam clashes with all of the available sessions, then assign it to a new session and go to Step 9. Otherwise go to Step 8.

Step8 : There exist at least two clash-free sessions and choose the one which clashes with most of the rest of unscheduled exams. Assign the exam to that session and go to Step 9.

Step9 : If all of the exams are scheduled then STOP. Otherwise go to Step 2.

The flowchart of this algorithm can be seen in the Appendix.

Test Problems

The data used in the study have been supplied by Carter, Laporte and Lee[3] and they can be obtained via the electronic mail. The main characteristics of the data can be seen in Table 1. The data can be divided into 2 groups; medium scale and large scale problems. The last three data, CAR-F-92, UTA-S-93 and CAR-F-91 fall in the latter group. In one of the columns, the density of the conflict matrix represents the proportion of non-zero and non-diagonal entries of the conflict matrix C_{ij} where C_{ij} is the number of students taking both exam i and j. High density in the conflict matrix shows a large number of exams which conflict with each other.

Table 1 : Main characteristics of the problems

Code	Institution	Number of examinations	Number of students	Number of student examinations	Density d of the conflict matrix
HEC-S-92	École des Hautes Études Commerciales, Montreal	81	2823	10632	0.42
STA-F-83	St. Andrew's Junior High School, Toronto	139	611	5751	0.14
YOR-F-83	York Mills Collegiate Institute, Toronto	181	941	6034	0.29
EAR-F-83	Earl Haig Collegiate Institute, Toronto	190	1125	8109	0.27
LSE-F-91	London School of Economics	381	2726	10918	0.06
UTE-S-92	Faculty of Engineering, University of Toronto	184	2749	11793	0.08
TRE-S-92	Trent University, Peterborough, Ontario	261	4360	14901	0.18
KFU-S-93	King Fahd University of Petroleum and Minerals, Dhahran	461	5349	25113	0.06
CAR-F-92	Carleton University, Ottawa	543	18419	55522	0.14
UTA-S-93	Faculty of Arts and Science, University of Toronto	622	21266	58979	0.13
CAR-S-91	Carleton University, Ottawa	682	16925	56877	0.13

The Results

Carter, Laporte and Lee[3] have used a heuristic by Carter and Gendreau[4] to determine the maximum clique. Since at the early stage of the research, the paper has been very difficult to obtain, we have decided to use an alternative algorithm by Carraghan and Pardalos[1] instead. The results obtained using the method by Carraghan and Pardalos are shown in Table 2.

Table 2 : Size of maximum clique

Code	Carraghan and Pardalos
HEC-S-92	17
STA-F-83	13
YOR-F-83	18
EAR-F-83	21
LSE-F-91	17
UTE-S-92	10
TRE-S-92	20
KFU-S-93	19
CAR-F-92	24
UTA-S-93	26
CAR-S-91	23

Carter, Laporte and Lee[3] compared 4 different sorting criteria; Largest Degree (LD), Saturation Degree (SD), Largest Weighted Degree (LWD) and Largest Enrolment (LE) as well as a Random Order (RO) as a benchmark. The results in Table 3 are based on how well each strategy performs in terms of the minimum number of sessions obtained, the number of backtracks required and the computing time in seconds. In almost all cases except in YOR-F-83, the best results are produced by the SD strategy.

In most cases of the medium scale problems, the minimum number of sessions obtained are equal to the size of the maximum clique, i.e. the lowest number of sessions possible. Unfortunately, the minimum number of sessions obtained in the large scale problems are considerably higher than the lowest bound to the number of sessions. Comparing the new algorithmic rule with the SD strategy, it managed to produce the same minimum number of sessions in 8 cases. In 3 cases where the results are different (EAR-F-83, CAR-F-92 and CAR-S-91), the number of sessions obtained is increased by at most 2 sessions. The

processing times achieved by the new algorithmic rule are quite small for both medium and large scale problems.

Table 3 : The minimum number of sessions obtained

		Carter *et al.*					New Algorithmic
		LD	SD	LWD	LE	RO	Rule
HEC-S-92	Sessions	18	17	17	17	17	17
	Backtracks	5	0	102	155	39	-
	Seconds	3.2	0.5	66.8	105.0	30.8	0.73
STA-F-83	Sessions	13	13	13	13	13	13
	Backtracks	0	0	0	0	2	-
	Seconds	2.7	2.7	2.7	2.8	90.3	4.02
YOR-F-83	Sessions	20	20	20	19	20	20
	Backtracks	255	1	378	310	208	-
	Seconds	673.1	6.6	966.5	705.8	740.2	7.70
EAR-F-83	Sessions	22	22	22	22	22	23
	Backtracks	100	1	35	56	114	-
	Seconds	391.6	8.7	111.5	184.5	630.4	11.24
LSE-F-91	Sessions	17	17	17	17	17	17
	Backtracks	95	10	25	164	79	-
	Seconds	551.9	78.0	158.8	943.3	848.1	102.34
UTE-S-92	Sessions	10	10	10	10	10	10
	Backtracks	0	0	2	0	4	-
	Seconds	1.6	1.6	2.7	1.6	4.3	9.51
TRE-S-92	Sessions	22	20	20	22	22	20
	Backtracks	51	2	63	2	132	-
	Seconds	468.3	32.8	402.6	29.8	1342.5	27.89
KFU-S-93	Sessions	19	19	19	20	19	19
	Backtracks	27	101	124	6	247	-
	Seconds	522.1	1159.6	1156.9	73.6	3361.6	233.86
CAR-F-92	Sessions	31	28	30	31	32	29
	Backtracks	104	510	432	6	205	-
	Seconds	392.5	227.2	1775.6	19.1	996.6	325.12
UTA-S-93	Sessions	33	32	33	33	34	32
	Backtracks	5	2	92	11	89	-
	Seconds	549.8	272.3	6878.4	755.6	6539.2	512.63
CAR-S-91	Sessions	32	28	30	32	35	30
	Backtracks	1	311	210	107	59	-
	Seconds	8.9	1080.6	807.9	412.1	265.2	631.12

Conclusions

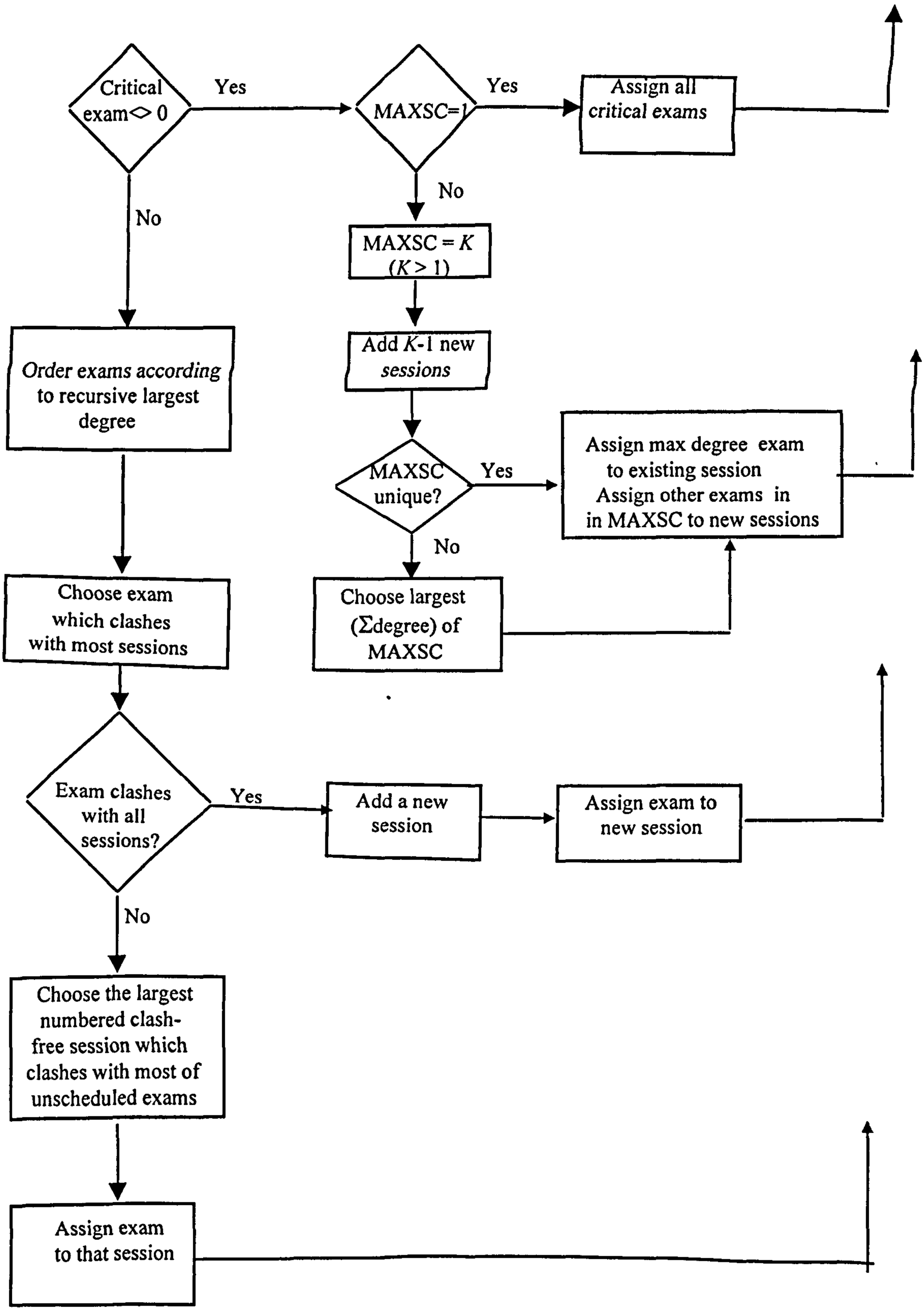
The objective is to produce a conflict-free examination timetable where no side constraints are considered at all. An algorithm to find the minimum number of sessions required to schedule all examinations is described in detail. The algorithm incorporates a clique

initialisation strategy but does not include backtracking procedures. The new method is then compared with the method by Carter, Laporte and Lee[3] in terms of the minimum number of sessions and the processing times. In most cases, the new method managed to achieve the same minimum number of sessions. Therefore, the new method is quite capable of assigning examinations to a minimum number of sessions possible without creating conflicts within a reasonable computing time.

References

1. Carraghan, R. and Pardalos, P. M. (1990). An Exact Algorithm for the Maximum Clique Problem. *Operations Research Letters*, **9**, 375-382.
2. Carter, M. W., Laporte, G., and Chinneck, J. W. (1994). A General Examination Scheduling System. *Interfaces*, **24**(3), 109-120.
3. Carter, M. W., Laporte, G., and Yan Lee, S. (1996). Examination Timetabling : Algorithmic Strategies and Applications. *Journal of Operational Research Society*, **47**, 373-383.
4. Carter, M. W., and Gendreau, M. (1992). A Practical Algorithm for finding the largest clique in a graph. Publication CRT-820, Centre de recherche sur les transports, Montreal.
5. Desroches, S., Laporte, G., and Rousseau, J. M. (1978). HOREX : A Computer Program for the Construction of Examination Schedules. *INFOR*, **16**, 294-298.
6. Hertz, A. (1991). Tabu Search for Large Scale Timetabling Problems. *European Journal of Operational Research*, **54**, 39-47.
7. Laporte, G., and Desroches, S. (1984). Examination Timetabling by Computer. *Computers & Operations Research*, **11**, 351-360.
8. Mehta, N. K. (1981). The Application of A graph Colouring Method to An Examination Scheduling Problem. *Interfaces*, **11**(5), 57-65.

APPENDIX


Critical exam<> 0
Yes
MAXSC=1
Yes
Assign all critical exams
No
MAXSC = K (K > 1)
No
Order exams according to recursive largest degree
Add K-1 new sessions
MAXSC unique?
Yes
Assign max degree exam to existing session Assign other exams in in MAXSC to new sessions
No
Choose largest (Σdegree) of MAXSC
Choose exam which clashes with most sessions
Exam clashes with all sessions?
Yes
Add a new session
Assign exam to new session
No
Choose the largest numbered clash-free session which clashes with most of unscheduled exams
Assign exam to that session