Automating Class Schedule Generation in the Context of a University Timetabling Information System

Kuldeep Singh Sandhu (Student Number 319995)
B. Bus, Grad Dip of Acctg, MBA

School of Management,
Nathan Campus, Griffith University.

Submitted as fulfilment of the requirements for the degree of Doctor of Philosophy.
ABSTRACT

This thesis examines the university timetable generation problem. It begins with a discussion on the conflicting terminology used and differentiates between the terms scheduling and timetabling. This discussion integrates with an overview of the problem itself both from practical and academic perspectives.

This is followed by a summary of the apparently effective timetable solution generation algorithms. The literature is then examined in detail from that perspective. This literature review is then summarised in a form to highlight the use of these different solution algorithms. A classification schema is developed and the summary of the literature presented within the framework of this schema. Trends in the research literature result from this presentation and an extrapolation to future research trends are suggested.

An information system based upon the need to support timetable production and maintenance is presented. Given the very practical outcomes expected of timetable research, the information system was designed to enable the whole range of administrative functions performed by teachers to be either directly supported or readily modified to prove such support. The implementation of this particular system is given and resulting timetables are presented and discussed. The system generated manual and automated timetables and these were produced by trailing a number of objective functions. It was noted that the determination of the optimal objective function is dominated by specific individual institutional criteria. It is suggested that this would make a more than significant project for future information systems research.

From the literature it is noted that the timetable generation problem, as reported time and time again in the literature, has been solved. Such claims lead to a benchmark which is proposed to enable an initial comparison of the effectiveness of proposed solutions by different researchers.

The thesis then presents a summary of the work that was carried out and offers direction for future research. It is noted that despite the fifty years of research conducted into this area there exists a significant number of research avenues still to be pursued.
# TABLE OF CONTENTS

ABSTRACT ............................................................................................................................................... 1

ACKNOWLEDGMENTS ........................................................................................................................... 7

STATEMENT OF ORIGINALITY ........................................................................................................... 8

CHAPTER 1 INTRODUCTION .................................................................................................................. 9
  SCHEDULING AND TIMETABLING ......................................................................................................... 9
  RESEARCH OBJECTIVES ..................................................................................................................... 15
  THESIS OVERVIEW .............................................................................................................................. 16

CHAPTER 2 LITERATURE ANALYSIS FROM A TIMETABLE GENERATION ALGORITHM PERSPECTIVE ................................................................................................................................. 18
  SOLUTION GENERATION ALGORITHMS ............................................................................................. 18
    Linear Programming/Integer Programming ....................................................................................... 19
    Evolutionary and Genetic Algorithms ................................................................................................. 21
    Simulated Annealing ........................................................................................................................... 23
    Tabu Search ......................................................................................................................................... 25
    Constraint Based Reasoning ............................................................................................................... 27
  Summary ............................................................................................................................................... 29
  THE LITERATURE ................................................................................................................................. 30
    Overview ........................................................................................................................................... 30
    A Detailed Survey of The Literature .................................................................................................. 31
    Discussion and Future Trends ............................................................................................................. 71
  ANALYSING THE LITERATURE ............................................................................................................ 71
    A Proposed Classification Schema .................................................................................................... 73
  Summary ............................................................................................................................................... 74

CHAPTER 3 A TIMETABLE INFORMATION SYSTEM ................................................................................. 76
  BACKGROUND ...................................................................................................................................... 78
  TIMETABLE SOLUTION GENERATOR .................................................................................................. 81
    The Algorithm for Automated Solution Generation ........................................................................... 82
    Implementation ................................................................................................................................... 85
  TIMETABLE MANAGEMENT INFORMATION SYSTEM ....................................................................... 85
    Problem Definition ............................................................................................................................. 87
    System Analysis and Design .............................................................................................................. 87
    Implementation .................................................................................................................................. 130
  RESULTS AND DISCUSSION ................................................................................................................ 161
  CONCLUSION AND FUTURE ENHANCEMENTS ............................................................................... 164

CHAPTER 4 EVALUATING THE EFFECTIVENESS OF TIMETABLE SOLUTION GENERATORS ................................................................................................................................. 165
  BACKGROUND ...................................................................................................................................... 165
    Hardware and/or Software Benchmarks ............................................................................................ 166
    Database Performance Benchmarks .................................................................................................. 167
    Research Benchmarks ....................................................................................................................... 168
  DATA REQUIREMENTS ........................................................................................................................ 168
  PROPOSED BENCHMARK ..................................................................................................................... 170
    Test 1 ................................................................................................................................................ 171
    Test 2 ................................................................................................................................................ 171
  RESULTS AND DISCUSSION ................................................................................................................ 172
    Test 1 ................................................................................................................................................ 172
    Test 2 ................................................................................................................................................ 173
    System Usability ................................................................................................................................ 174
  FUTURE RESEARCH ............................................................................................................................ 175
TABLE OF FIGURES

Figure 1: Timetable Publications as a Function of Time .......................................................... 72
Figure 2: Publications By Solution Generation Technique ...................................................... 73
Figure 3: Publications By Improvement Technique ................................................................. 74
Figure 4: Holistic Architecture of University Timetabling Information System ...................... 79
Figure 5: Detailed Architecture of University Timetabling Information System .................... 80
Figure 6: E-R Diagram – Subject and Related Entities ............................................................ 96
Figure 7: E-R Diagram – Student and Related Entities ........................................................... 97
Figure 8: ER-Diagram - Rooms and Related Entities ............................................................. 98
Figure 9: ER Diagram - Subject Class Timetables and Related Entities .................................. 99
Figure 10: ER Diagram - Department and Related Entities .................................................. 100
Figure 11: ER Diagram - Courses and Related Entities ......................................................... 101
Figure 12: ER Diagram - Staff and Related Entities ............................................................... 101
Figure 13: ER Diagram - Teaching Contact Types and Related Entities ................................ 102
Figure 14: ER Diagram - Faculties and Related entities .......................................................... 102
Figure 15: ER Diagram - Campuses and the Related Entities ................................................. 103
Figure 16: ER Diagram - Room Availabilities and Related Entities ....................................... 103
Figure 17: ER Diagram - Teaching Position and Related Entities ......................................... 103
Figure 18: Subject table showing related entities ...................................................................... 106
Figure 19: Student table and its related entities ....................................................................... 107
Figure 20: Room table and its related entities ........................................................................ 108
Figure 21: Subject class timetable and its related entities ..................................................... 109
Figure 22: Department table and its related entities ............................................................... 110
Figure 23: Course table and its related entities ...................................................................... 111
Figure 24: Staff table and its related entities .......................................................................... 112
Figure 25: Teaching contact types table and its related entities ............................................ 113
Figure 26: Campus table and its related entities .................................................................... 114
Figure 27: Room availability table and its related entities .................................................... 115
Figure 28: Screen - Just Timetables (Main Menu) ................................................................. 130
Figure 29: Screen - Room Maintenance - Facility Details Tab ............................................. 131
Figure 31: Screen - Maintenance of Buildings Table ...................................................133
Figure 32: Screen - Maintenance of Campuses – Campus Details and Buildings Table ...............................................................................................................................134
Figure 33: Screen - Maintenance of Campuses – Campus and Subject Available Table ...............................................................................................................................135
Figure 34: Screen - Maintenance of Departments – Department Details Table ........136
Figure 35: Screen - Maintenance of Departments – Staff Members and Profiles Table ...............................................................................................................................137
Figure 36: Screen - Maintenance of Departments – Subjects in the Department Table138
Figure 37: Screen - Maintenance of Staff Details – Main Details Table .................139
Figure 38: Screen - Maintenance of Staff Details – Staff CV Table .......................140
Figure 39: Screen - Maintenance of Staff Details - Teaching Responsibilities Table ...141
Figure 40: Screen - Maintenance of Staff Details - Subject Convened Table ..........142
Figure 41: Screen - Maintenance of Student Details – Personal Details Table ..........143
Figure 42: Screen - Maintenance of Student Details – Course Enrolment Table ......144
Figure 43: Screen - Maintenance of Student Details – Subject Enrolments and Results
Figure 44: Screen - Maintenance of Courses : Introduction Sub-Table ....................146
Figure 45: Screen - Maintenance of Courses – Degree Rules Sub-Table.................147
Figure 46: Screen - Maintenance of Courses – Further Information Sub-Table ........148
Figure 47: Screen - Maintenance of Courses – Subject Table ..................................149
Figure 48: Screen - Maintenance of Courses – Enrolments Table .........................150
Figure 49: Screen - Maintenance of Subjects – Classes And Timetabling Sub-Table .151
Figure 50: Screen - Maintenance of Subjects – Teaching Team Sub-Table................152
Figure 51: Screen - Maintenance of Subjects – Brief Description Sub-Table............153
Figure 52: Screen - Maintenance of Subjects – Objectives Sub-Table .....................154
Figure 53: Screen - Maintenance of Subjects – Content Sub-Table .......................155
Figure 54: Screen - Maintenance of Subjects - Assessment Sub-Table ....................156
Figure 55: Screen Subject Enrolments for each teaching type ..............................157
Figure 56: Generated Manual Timetable for one subject .......................................158
Figure 57: Generated Manual Timetable for a combination of subjects .................159
Figure 58: Generated Automated Timetable for one subject ...............................160
Figure 59: Generated Automated Timetable after Objective Function Adjustment ....161
INDEX OF TABLES

Table 1: Data Structure – Campus Table ................................................................. 89
Table 2: Data Structure – Rooms Table ................................................................. 90
Table 3: Data Structure – Subjects Table .............................................................. 91
Table 4: Data Structure – Students Table ............................................................. 92
Table 5: Data Structure – Department, Course and Staff .................................... 93
Table 6: Data Structure - Subject Class Timetables, Teaching Contact Types and Faculties ................................................................. 94
Table 7: Titles, Room Availability and Teaching Position .................................. 95
Table 8: List of Key Identifier Columns ................................................................. 105
Table 9: Table indicating the data-types and other information ......................... 116
Table 10: Buildings data dictionary ...................................................................... 117
Table 11: Campuses Data Dictionary .................................................................. 117
Table 12: Course Student Enrolments Data Dictionary ....................................... 117
Table 13: Course Teaching Department Data Dictionary .................................... 118
Table 14: Course Data dictionary ......................................................................... 118
Table 15: Department Data Dictionary ................................................................. 119
Table 16: Faculties Data Dictionary ..................................................................... 119
Table 17: Room Availabilities Data Dictionary .................................................... 119
Table 18: Room Teaching Usage ........................................................................ 119
Table 19: Room Timetable Data Dictionary ......................................................... 120
Table 20: Room Data Dictionary ......................................................................... 120
Table 21: Staff Data Dictionary ........................................................................... 121
Table 22: Student Subject Class Preference Data Dictionary ............................ 122
Table 23: Student Subject Enrolments Data Dictionary ....................................... 122
Table 24: Student Data Dictionary ....................................................................... 123
Table 25: Subject Assessment Items Data Dictionary ......................................... 123
Table 26: Subject Assessment Student Marks Data Dictionary ......................... 124
Table 27: Subject Class Clash Data Dictionary .................................................. 124
Table 28: Subject Class Possible Room Time Lists Data Dictionary ................... 124
Table 29: Subject Class Student Enrolments Data Dictionary ........................... 124
Table 30: Subject Class Timetables Data Dictionary ........................................... 125
Table 31: Subject Student Grade Data Dictionary .............................................. 125
Table 32: Subject Teaching Team Data Dictionary ............................................. 125
Table 33: Subject Data Dictionary ....................................................................... 126
Table 34: Teaching Contact Type Data Dictionary .............................................. 126
Table 35: Teaching Finish Time Data Dictionary .............................................. 127
Table 36: Teaching Position Data Dictionary .................................................. 127
Table 37: Teaching Start Time Data Dictionary ............................................... 127
Table 38: Trial Data Dictionary ....................................................................... 127
Table 39: Building ID data element ................................................................. 127
Table 40: Campus ID data element ................................................................. 128
Table 41: Course ID data element ................................................................. 128
Table 42: Department ID data element ........................................................... 128
Table 43: Faculty ID data element ................................................................. 128
Table 44: Room ID data element .................................................................. 128
Table 45: Room Time ID data element .......................................................... 129
Table 46: Staff ID data element ..................................................................... 129
Table 47: Student ID data element ................................................................. 129
Table 48: Subject class ID data element ......................................................... 129
Table 49: Subject ID data element ................................................................. 129
Table 50: Teaching Contact ID data element ................................................ 130
Table 51: Teaching position ID data element ................................................ 130
Table 52: Title ID data element ..................................................................... 130
Acknowledgments

The work that I have carried out could not possibly have come to successful fruition without the significant time and effort expended by my supervisor, Dr Geoff Harris, over the last three years. His unwavering devotion and attention to every detail has resulted in the masterpiece presented. No problem was too difficult or menial to discuss with me. No time of the day, and quite often night, was too inconvenient for him. A researcher who stands heads and shoulders above the rest, I have learnt and grown significantly in academic stature as a direct consequence of his supervision.

I would like to thank my wife for her patience and many lonely nights over the last three years. There are many others to whom thanks and acknowledgment are due. Of special note are the many administrative staff who have attempted to help me source much of the data required to complete chapters three and four of this thesis. The efforts of some in securing the data involving questionable ethics has not gone unnoticed. Many thanks are also due to Stephen and Pat for reviewing and correcting grammatical oversights that somehow crept into the manuscript undoubtedly due to the busy schedule and work load I had taken on.

Thanks again to all of you.
Statement of Originality

This work has not previously been submitted for a degree or diploma in any university. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made in the thesis itself.

_____________________________
Kuldeep Sandhu
CHAPTER 1
INTRODUCTION

This thesis examines the general and well established research into the timetable scheduling problem. The very real need to optimise timetable construction from a practical perspective has driven and provided funding for an extensive research activity resulting in a very well developed and ongoing research literature. The work contained in this thesis not only summarises the voluminous literature but also contributes through analysis as well as providing a benchmark for the objective comparison of future research in this area.

This chapter begins with a discussion of the concepts, terminology and general research direction in the broad area of scheduling. In particular, the usage and classification of the general timetable problem is highlighted. General trends in this timetable literature are presented and discussed.

An overview of the thesis is then detailed. A clear succinct coverage of the major timetable solution generation algorithms through an extensive literature review and proposed classification schema is presented. In this context, a generalised timetable and student information system is presented to aid practical outcomes. As a special case the significance of the benchmark is then discussed.

Scheduling and Timetabling

The general area of scheduling has been the subject of intense research for a number of decades. Scheduling and timetabling are typically viewed as two separate activities, with the term scheduling used as a generic term to cover specific types of problems in this area. Consequently, timetable construction can be considered as a special case of generic scheduling activity.

In the most general terms, scheduling can be described as the constrained allocation, of resources to objects, being placed in space-time in such a way as to minimise the total cost of a set of the resources used (Wren, 1996). Examples of this problem set can be seen in transport scheduling and delivery vehicle routing where the business driven objective is to minimise the “real dollar” total cost function. Timetable construction is the allocation, subject to constraints, of given resources to objects being placed in space-time in such a way as to satisfy or nearly satisfy a desirable set of possible
objectives (Wren, 1996). Class timetables and exam timetables are examples of these problems where all hard constraints must be satisfied to generate a valid solution.

Thus the term scheduling covers all aspects of the activity of allocating resources and, at the same time, satisfying some predetermined objective. However, due to the enormity of the problem, it becomes necessary to classify the scheduling problem into specialised activities such as timetabling. Thus, in practical terms the timetabling problem can be described as scheduling a sequence of lectures between teachers and students in a prefixed time period (typically a week), satisfying a set of varying constraints (Schaerf, 1999).

Many approaches and models have been proposed for dealing with the variety of timetable problems (Bardadyn, 1996). Problems range from the construction of semester or annual timetables in schools, colleges and universities to exam timetabling at the end of these periods. Early timetable activities were carried out manually and a typical timetable once constructed remained static with only a few changes necessary, in order to fine tune it every semester or year. However, the nature of education has changed substantially over the years and thus the requirements of timetables have become much more complicated than they used to be. Consequently the need for automated timetable generation is increasing and thus the development of a timetable generation system that generates valid solutions is essential. As a result, during the last 30 years, many papers related to automated timetabling have been published in conferences, proceedings and journals. In addition, several applications have been developed and implemented with various successes (Schaerf, 1999).

The early techniques used in solving timetabling problems were based on a simulation of the human approach in resolving the problem. These included techniques based on successive augmentation that were called direct heuristics. These techniques were based on the idea of creating a partial timetable by scheduling the most constrained lecture first and then extending this partial solution lecture by lecture until all lectures were scheduled (Schaerf, 1999). The next step was for researchers to apply general techniques like integer and linear programming, graph colouring and network flow to solve the timetable problem. Hence the first two papers published on timetable construction using these general techniques are generally attributed to Kuhn (1955) and Haynes (1959). Kuhn’s paper adopts a mathematical approach to the fundamental timetable problem in contrast to Haynes’ paper, which concentrates, on the more
practical problem aspects of scheduling events for a conference. Interest in timetable solution generators increased dramatically in the 1960’s mainly due to the more common availability of computers to perform the “number crunching” required by the algorithms developed. It was in these papers that the first use of look-ahead techniques was discussed. For instance, Appleby, Blake and Newman (1960) and Lewis (1961) used a primitive look-ahead technique involving heuristic counting arguments.

The first non-heuristic approach was developed by Gotlieb (1963) and discussed in the now famous process of reducing the availability array and presented at the Munich IFIP congress. This was arguably the first paper on the set partitioning approach and was further enhanced by Berghuis, Van der Heiden and Becker (1964) where the concept of virtual classes or teachers to obtain the classical bipartite problem was introduced. Csima (1965) embedded this basic problem into doubly stochastic problems that used pseudo classes. In addition to this, Csima and Gotlieb (1964) also treated the timetable problem as a three dimensional assignment problem by considering the close connection between the timetable problem and the problem of vertex colouring. This was the first paper to consider the now well-known special relationship between the various scheduling problems. With the help of the Gotlieb approach (1963) both Becker (1964) and Baraclough (1965) simulated their respective implementations with “hand” calculations. Typically these papers were based on a heuristic approach. Due to this work many other papers followed which discussed the problem but had very little new work in them.

Around the late 60’s some attempts at limiting the general problem by considering case examples were beginning to be published. For instance, Lawrie (1969) developed a model for the school timetable problem by using an integer linear programming approach. During the 1970’s several authors adopted the usage of the heuristic approach in tackling the timetable problem. For example Junginger (1972) provided a reduction of the timetable problem by applying it to a three dimensional transport problem. This theoretical oriented attempt to long range planning, in terms of using a Boolean matrix iteration, expanded on the concept of the relation descriptions and double integrals used previously.

Schmidt and Strohlein (1973) published the first extensive annotated bibliography of the timetable research literature, including more than 200 entries. They predicted the generation of timetables by computer would be heavily influenced by devices at hand,
with timetable programming moving from remote handling in huge computing centres to micro computer centres owned by schools and directly handled by teachers on their desktops. The major general techniques that seemed to have been prevalent in the 1970’s and 1980’s have their roots in artificial intelligence and are based on algorithms supported by simulated annealing, tabu search and genetic algorithm methods. Papers in the literature typically described a substantial software implementation and this is supported by the presentation of results of the application of the method in one or more cases (Schaerf, 1999). Furthermore, there were a number of important surveys of timetabling literature that were published in the 1980’s.

De Werra (1985) listed the various problems dealing with timetabling in a formal way and provided different formulations in an attempt to solve them. He also described the approaches considered the most important at that time, stressing the graph-theoretic ones. Carter (1986) analysed a survey, which discussed actual applications of timetables at several universities. He also provided details of a tutorial guide for practitioners on electing and/or designing an algorithm for their own institutions.

Junginger (1986) described research work done in Germany on the school timetable problem and the underlying approaches that were based on direct heuristics. In particular, he described the various software products implemented, and their actual utilisation of them at various institutions. Corne et al. (1994) provided a survey of Genetic Algorithm application to timetables, discussed future perspectives of such approaches and compared results obtained with respect to other approaches. Other surveys are provided in Dempster, Lethbridge and Ulph (1975); Hilton (1980); Klein (1983); Vincke (1984); and Ferland, Roy and Gia-Loc. (1986). Although there were papers published in the 1990’s solving timetable problems using the above artificial intelligence based techniques, there was a new approach emerging, also rooted in Artificial Intelligence, that has gained prominence called Constraint Satisfaction Programming (CSP).

Abramson (1991) used Simulated Annealing as an optimisation technique. The possibility of adding cost components was discussed in an attempt to include the more complex scheduling constraints that arise in schools. Also described is how the weighting of cost components allowed one component to be made more important than others. He implemented this in a parallel computer system and proved that the speed of the algorithm improved along with results. Cooper and Kingston (1993) described a
computer program that solved a problem within a large and highly constrained high school without any simplifications. A timetable specification language was provided that helped to avoid many constraints in a uniform way. Costa (1994) discussed many different types of constraints that must be taken into account. Tsang, Mills, Williams, Ford, & Borrett (1999) discusses the importance of constraint satisfaction techniques in solving the timetable problem and provides a quick introduction to this field. Schaerf (1999) provided a survey of the different techniques used in timetable generation. Constraint satisfaction techniques were stressed as an important addition to the tools that are used in solving the timetabling problem.

From the literature, there emerged a trend that in the last decades the topics of timetabling were mainly confined to the Operational Research community. The techniques used were mathematical in nature, the two most common being linear programming (LP) and integer programming (IP). In the current decade the contribution of artificial intelligence has provided the timetabling community with promising modern heuristics, like Genetic Algorithms, Simulated Annealing and Tabu Search (Schaerf, 1999).

Genetic Algorithms (GA) are a class of stochastic search algorithms that mimic natural selection by using a computational version of survival of the fittest (Sanchez, Shibata, and Zadeh, 1997). It is a problem solving method that uses genetics as its model for problem solving, applying reproduction, general crossover, and mutation to these pseudo-organisms so they can pass beneficial and survival-enhancing traits to new generations (Chambers, 1995). In a similar manner, Simulated Annealing (SA) has been described as “…a randomised optimisation method, which accepts deteriorations of the objective function with a probability depending on a control parameter, based on a physical analogy” (Rodrigues and Anjo, 1993). SA has been demonstrated to be a good technique for solving hard combinatorial optimisation problems. Due to its very nature of sampling across the entire search space SA can be very resource extensive and as such attempts at speeding up annealing algorithms have been based on shared memory multiprocessor systems, and parallelization for certain problems on distributed memory multiprocessor systems (Diekmann, Luling and Simon, 1993). In contrast Tabu Search (TS) is a meta-heuristic technique that guides a local heuristic search procedure to explore the solution space beyond local optimality (Glover and Laguna, 1997). “A meta-heuristic refers to a master strategy that guides and modifies other heuristics to produce the solutions beyond those that are normally generated in a quest
for local optimality” (Glover and Laguna, 1997). Due to it’s “shrinking” of the search space a TS normally generates a solution much faster than a SA method. Another contribution by Artificial Intelligence in the 1990's has seen the emergence of a new approach to solving timetabling problems called constraint satisfaction techniques.

Constraint based reasoning has proven to be a productive research method for researchers in the areas of artificial intelligence, operational research and logic programming (Eugene and Mackworth, 1994). Gradually a theoretical framework called Constraint Satisfaction Problems (CSP) has evolved (Lhomme, 1993). The process of Constraint Satisfaction involves finding values for variables within the framework of the problem, subject to constraints on acceptable combinations of values. As in some cases it is impossible to solve the problem completely, a subset of the problem is solved that is called partial constraint satisfaction (Tsang et. al., 1999). CSPs are solved by different versions of a backtracking search that attempts to improve the search efficiency and generally deals with the binary CSPs (Janssen, Jegou, Nouguier and Vilarem, 1989). The CSP approach allows for greater flexibility in the formulation of the problem, which is broadly recognised as a crucial issue in generating successful timetables. In most cases it is difficult to define the constraints of the problem as they are unclear and require many refinements and interactions with the user. This level of flexibility allows CSP systems to address at a practical level the problems that are encountered at the implementation stage. Furthermore due to the very nature of CSP definitions, the problem of uneven time-slots can be easily accommodated.

Almost all of the papers in the literature describe a substantial software implementation. In addition, this is supported by the presentation of results of the application of the method in one or more test cases. The results obtained are measured against manual results but unfortunately, the absence of a common definition of the various problems and of widely accepted benchmarks prevents the comparison of the algorithms among each other. The computational complexity of the proposed systems are determined only through computing time. However comparisons are difficult as hardware varies from case to case. Furthermore there seems to be a substantial gap between the theoretical discussion and implementation of the software to test cases in contrast to obtaining effective and realistic timetables that can be used in everyday operations. This gap can be attributed to the various problems that are not foreseen but encountered in the workplace. Some of the major problems that do occur are listed:
• Students tend to change their enrolment quite frequently, even up to week five of the semester, which leads to clashes that can require the regeneration of the timetable in the hope of finding a feasible solution.

• The availability of sessional lecturers and tutors, who generally have to juggle between two or more jobs, can provide an extra complexity to the timetable problem even after it has been generated.

• Most of the techniques utilised in timetable generation assume that lectures and tutorials are of equal length in order to facilitate the algorithm to successfully work. However in reality there are a number of subjects that do not conform to this time frame and as such normally need to be allocated to a time slot manually. This action reduces the efficiency of the automated timetable application and can lead to clashes in other parts of the generated solution.

• The majority of algorithms that have developed do not allow a temporal ordering of teaching slots within the same subject. This is normally expected by subject convenors (person responsible for managing a subject) and can lead to dissatisfaction with an automated timetable system.

Therefore in order to generate a timetable that is practical and effectual it needs to be flexible enough so that it can facilitate and overcome the problems listed above as they occur.

**Research Objectives**

Defining research objectives is a fundamental step that is necessary in providing an overall framework that derives the scope of the research. The two principle objectives of this research are:

1. Collate, annotate and critique the extensive research literature into the timetabling problem
2. To derive a suitable timetabling system from a succinct understanding of the various solution generation techniques that have persisted in the literature.

The first step, necessary to achieve these objectives, was the collection, analysis and critique of the vast literature that has developed over the past 5 decades into the
timetable problem. From this the solution techniques that have persisted were then compared to determine if there was any pattern in the direction to which the literature may, or may not, have been heading.

Timetable research concentrates either from a technical perspective (as indicated from the large number of papers concentrating on the generation of a solution or optimal solution using a specific or combined technique eg. Abramson (1991), Sharma and Chandra (1999) and de Werra (1985)) or from an IS viewpoint (mainly from a DSS eg. Foulds and Johnson (2000)). Thus an integrated system that encapsulates the timetabling problem from both perspectives is proposed.

The second objective, as derived from the literature review, was to design and develop an integrative timetabling IS for the generation and management of university timetables. This included support for both automatic and manual development of semester timetables. The system developed would allow for plug and play of any solution generation technique into the IS. A constraint-based technique has been adopted but this could easily be substituted with a Genetic Algorithm (GA) or Simulated Annealing (SA) algorithms.

**Thesis Overview**

The next chapter presents an overview of the timetable generation literature. This overview is rationalised and classified by solution algorithm. It begins with a presentation of the major solution generation algorithms that have persisted in the literature. This required the treatment to be condensed to a succinct coverage of the salient features of each algorithm. The subtle nuances that differentiate the implementations of the algorithms from each other were covered in the extensive literature that follows. This broad literature survey is carried out in order to illustrate the evolution and trends of timetable problems from the 1950’s through to 2002. The objective of the survey is to ascertain the way the solution to timetable problems is heading and then to provide a system that can endeavour to assist in this process. In order to carry out this task it is necessary to summarise the results of the literature survey and they are presented in Chapter 2. Following this a rationalisation and classification of the literature is carried out in order to highlight the trends in
timetabling. This chapter then concludes with a concise analysis classification of the extent to which each of the solution algorithms has been used in the literature.

This is followed in chapter three by the discussion of the development and implementation of an information system generic to supporting all aspects of the timetable problem. The database system produced supports the generation of university timetables both from a manual and automated perspective. The other university administrative functions are supported from this fundamental perspective. The system developed satisfies the practical setting within which timetable generation and maintenance must integrate from the perspective of seamlessly facilitating the role of ongoing research into this fascinating problem. The chapter is concluded with a presentation of the results generated and a discussion on the effectiveness of the system.

Chapter four presents the first benchmark proposed as an agent to support the objective evaluation of competing claims made by researchers in this area. The details of the two tests that comprise this benchmark are discussed. Whilst the very nature of benchmarks dictate that their acceptance is conditional upon their uptake and use by researchers in an area this new work should at least raise the awareness of researchers working in timetabling to the very real need for this performance comparison and appraisal. A short discussion on the objective assessment of the system concludes this chapter.

The conclusion and future research chapter succinctly summarises the work presented in this thesis. Suggestions for the direction of future research efforts are presented.
CHAPTER 2
LITERATURE ANALYSIS FROM A TIMETABLE GENERATION ALGORITHM PERSPECTIVE

This chapter provides an analysis of the automated timetable literature broadly organized by algorithmic technique. It begins with a presentation of the major timetable solution generation algorithms that have persisted in the literature. A detailed examination of the academic literature is provided within the context of these fundamental solution generation algorithms. An analysis of the literature, grouped by the solution generation technique used, is then presented. The results show some interesting trends in the progressive cumulative research that has evolved over the last 5 decades.

The chapter concludes with a summary of the major findings from the analysis. It is demonstrated that the literature is currently converging on the use of constraint based solution algorithms and implementations. It is also noted that the next most commonly reported implementation involves the use of hybrid algorithms. Possible avenues to further this analysis are then presented. A small, though not insignificant, number of more recent publications are noted to not contribute directly to the solution generation debate. These papers report on development of Timetable Information Systems rather than extending generation algorithms and solution techniques.

Solution Generation Algorithms

There are currently many different solution generation algorithms in existence. Some are so well known that they are the subject of standard textbook material (see, eg., Scharef, 1996; Osman and Kelly, 1996). This section presents major algorithms that have appeared in the literature on generating timetabling solutions. For the sake of brevity only the fundamental algorithms themselves are presented, not the many hybrid approaches adopted (to examine some in detail see, for example, Gueret, Jussien, Boizumault and Prins, 1996; Mamede and Soares, 1996; Birbas, Daskalaki, and Housos, 1997).
The fundamental solution generation techniques developed in the literature are generally acknowledged to be (see Wren, 1996; Scharef, 1999, and references contained therein) the following:

1. Integer Programming/Linear Programming
2. Constraint Satisfaction Programming
3. Genetic and Evolutionary Algorithms
4. Simulated Annealing
5. Tabu Search

A brief description of these fundamental techniques is now presented. Each of these descriptions are structured to begin with a general formulation of the technique followed by a mathematical interpretation of each technique within the context of the generalised timetable problem.

**Linear Programming/Integer Programming**

The Linear and Integer Programming techniques, the first applied to timetabling, were developed from the broader area of mathematical programming. Mathematical programming is applicable to the class of problems characterised by a large number of variables that intersect within boundaries imposed by a set of restraining conditions (Thompson, 1967). The word "programming" means planning in this context and is related to the type of application (Feiring, 1986). This scheme of programming was developed during World War II in connection with finding optimal strategies for conducting the war effort and used afterwards in the fields of industry, commerce and government services (Bunday, 1984).

Linear Programming (LP) is that subset of mathematical programming concerned with the efficient allocation of limited resources to known activities with the objective of meeting a desired goal such as maximising profits or minimising costs (Feiring, 1986). Integer Programming (IP) deals with the solution of mathematical programming problems in which some or all of the variables can assume non-negative integer values only. Although LP methods are very valuable in formulating and solving problems related to the efficient use of limited resources they are not restricted to only these
problems (Bunday, 1984). Linear programming problems are generally acknowledged to be efficiently solved by just three methods, namely the graphical method, the simplex method, and the transportation method (see eg, Palmers and Innes, 1976; Makower and Williamson, 1985).

The construction of a linear programming model involves three successive problem-solving steps. The first step identifies the unknown or independent decision variables. Step two requires the identification of the constraints and the formulation of these constraints as linear equations. Finally, in step three, the objective function is identified and written as a linear function of the decision variables.

The formulation of LP/IP in the context of the general timetabling problem is represented as follows (see, for example, Schaefer, 1999, or de Werra, 1985).

There are $q$ subjects, represented as $S_1...S_q$ and each subject, $S_i$, consists of $n_i$ lectures. Furthermore, let there be $r$ curricula $K_1...K_r$ in which each curriculum is defined to a group of subjects that have common students. This means that subjects in $K_i$ must all be scheduled at different times. If the number of lecture time periods is $p$, and $M_k$ is the maximum number of lectures that can be scheduled at lecture time period $k$ (ie. the number of rooms available at time period $k$), then the formulation of this LP/IP timetable problem is as following:

Find $y_{ik}$ \hspace{1cm} (i = 1..q; k = 1..p )

Such that

\[ \sum_{k=1}^{p} y_{ik} = k_i \hspace{1cm} (i = 1..q ) \] (1)

\[ \sum_{i=1}^{q} y_{ik} \leq M_k \hspace{1cm} (k = 1..p ) \] (2)

\[ \sum_{i\in K_j} y_{ik} \leq 1 \hspace{1cm} (l = 1..r; k = 1..p ) \] (3)

\[ y_{ik} \in \{0,1\} \hspace{1cm} (i = 1..q; k = 1..p ) \] (4)

where we interpret $y_{ik} = 1$ if a lecture of subject $S_i$ is scheduled at lecture time period $k$, and $y_{ik} = 0$ otherwise.
Constraint set (1) imposes that each subject is composed of the correct number of lectures. Constraint set (2) enforces that at each time there are not more lectures than rooms. Constraint set (3) prevents conflicting lectures to be scheduled at the same period. Constraint set (4) contains the binary requirements that transform the problem from the LP domain into the LP/IP domain.

In the 1970’s IP was identified as being less efficient than Linear Programming (LP) for solving optimisation problems but was still considered to be a good problem solving technique (Zoints, 1974). However, in the 1980’s IP was recognised as not being uniformly efficient from the computational standpoint due to performance unreliability as the size and complexity of the problem increased (Taha, 1987).

**Evolutionary and Genetic Algorithms**

Evolutionary Algorithms (EAs) are a class of direct, probabilistic search and optimisation algorithms gleaned from the model of organic evolution. A Genetic Algorithm (GA) is a type of EA and is regarded as being the most widely known EA in recent times (Back, 1995). GAs are a class of stochastic search algorithms based on biological evolution whose search strategy mimics natural selection by using an automated version of the “survival of the fittest” analogy (for example see Sharma and Chandra, 1999; Sanchez, Shibata and Zadeh, 1997). It is a problem solving and optimisation method that uses genetics as its model problem and applies the rules of reproduction, general crossover and mutation to pseudo-organisations so those organisations can pass beneficial and survival-enhancing traits to the new (next) generation (Chambers, 1995).

A GA differs from other search techniques in the following ways:

- GAs optimise the trade-off between exploring new points in the search space and exploiting the information discovered thus far. This was proved using the K-armed bandit (an extension of the one-armed bandit) problem (Buckles and Petry, 1992).
- GAs have the property of implicit parallelism. Implicit parallelism means that the GA’s effect is equivalent to an extensive search of hyper planes of the given
space, without directly testing all hyper plane values (Goldberg, 1989). Each schema denotes a hyper plane.

- GAs are randomised algorithms, in that they use operators whose results are governed by probability. The results for such operations are based on the value of a random number (Buckles and Petry, 1992). This means GAs use probabilistic transition rules, not deterministic rules.
- GAs operate on several solutions simultaneously, gathering information from current search points to a direct subsequent search. Their ability to maintain multiple solutions concurrently makes them less susceptible to the convergence problem of local maxima and noise (Goldberg, 1989).
- GAs work with a coding of the parameter set, not the parameters themselves (Goldberg, 1989).
- GAs search from a population of points, not a single point (Goldberg, 1989).
- GAs use payoff (objective function) information, not derivatives or other auxiliary knowledge (Buckles and Petry, 1992).

An abstract view of the problem solution phase using a GA can be illustrated as follows (for example see Buckles and Petry, 1992):

1. generate an initial population, \(G(\text{at time}=0)\)
2. evaluate the fitness function \(f[G(0)]\)
3. \(t = 0\)
4. repeat
   1. \(t = t + 1\)
   2. Generate \(G(t)\) using \(G(t-1)\)
   3. Evaluate \(f[G(t)]\)
5. until either an acceptable solution is found or population convergence is achieved.

The representation of the timetable problem as a GA is very problem specific. Unlike LP/IP (where an almost “exact” mathematical formulation can be ascribed) the encoding scheme, which represents the template for the solution, must be determined. Here, we use a recent innovation (Sharma and Chandra, 1999) in which integer index strings are used (instead of the traditional bit strings) in the representation. This is claimed to improve the computational performance due to the vast reduction in the length of the encoded chromosome.

In this schema a sample examination timetable problem of, say, 17 subjects would be represented graphically as (Sharma and Chandra, 1999):
Examination Timetable (iterated to convergence)

<table>
<thead>
<tr>
<th>3</th>
<th>7</th>
<th>0</th>
<th>11</th>
<th>5</th>
<th>0</th>
<th>5</th>
<th>1</th>
<th>0</th>
<th>9</th>
<th>9</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject 0</td>
<td>Subject 1</td>
<td>Subject 2</td>
<td>. . .</td>
<td>Subject 16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Where \( a \) is the Room Index, \( b \) is the Day Index and \( c \) is the Session Index. The combination of room, day and session determine where the subject is scheduled to be examined.

It is claimed that GAs have a greater capacity than any other search method in finding the largest number of possible solutions and depict the best possible results (Maxfield, 1997). GAs demonstrate real-world significance when applied to timetabling problems where a complex set of scheduling constraints and a various collection of individuals exist. It is generally argued that a timetabling GA develops the best possible schedule (Plain, 1995). However, GAs still fall short in the choice of control parameters, the exact roles of crossover and mutation, and the characterisation of search landscapes for optimisation and convergence characteristics (Srinivas and Patnaik, 1994). Some applications where GAs have been used successfully are analogue circuits, fuzzy logic, and neural networks (Maxfield, 1997). These cover a wide range of practical applications such as job shop scheduling, training neural nets, image feature extraction, and image feature recognition (Buckles and Petry, 1992).

Simulated Annealing

Simulated Annealing (SA) is a randomised local search optimisation technique for finding solutions to optimisation problems. The name is derived from the analogue to the chemical physics simulation of the cooling of a collection of a Boltzmann distribution of atoms. SA is highly resource intensive and one of its setbacks is its requirement of utilising a large amount of computational time for obtaining a near-optimal solution. As such some attempts at speeding up annealing algorithms have been based on shared memory multiprocessor systems, and parallelization for certain problems on distributed memory multiprocessor systems (Abramson, 1991; Dikmann, Luling and Simon, 1993).

The process starts by creating a random initial solution. The main procedure consists of a loop that generates, at random and for each iteration, a neighbour (and thus
classification as a local search technique) of the current solution. The definition of neighbour in SA is dependent upon the specific structure of each problem. The basic steps involved in using an SA is as follows (Schaerf, 1999):

- Let $\Delta$ be the difference in the objective function between the new solution and the current one.
- If $\Delta < 0$ the new solution is accepted and becomes the current one.
- If $\Delta \geq 0$ the new solution is accepted with probability $e^{-\Delta T}$ where $T$ is a parameter called temperature.
- The temperature $T$ is initially set to an appropriate high value $T_0$. After a fixed number of iterations, the temperature is decreased by the cooling rate $a$, such that $T_n = a \times T_{n-1}$, where $0 < a < 1$.
- The procedure terminates when the temperature value is close and no other solution increases the value of the objective function, i.e. the system is frozen. The solution obtained when the system is frozen is obviously a local minimum.
- The control knobs of the procedure are the cooling rate $a$, the number of iterations at each temperature, and the starting temperature $T_0$.

The application of the timetable problem using SA is a relatively straightforward procedure. The atoms are replaced by elements and the system energy is replaced by the timetable costs. An initial allocation is made in which elements are placed in a randomly chosen period. The initial cost and the initial temperature are then computed. The cost is used to reflect the quality of the timetable and the temperature is used to control the probability of an increase in cost. At each iteration a period is chosen at random, called the from period, and an element randomly selected from that period. Another period is chosen at random, called the to period. The change in cost is then calculated from these two components (Abramson, 1991). An example of this process is illustrated using SA to decrease the cost of a school/university timetable solution is as follows: (Abramson and Dang, 1993).

```
Compute an Initial temperature

While (cost <> 0) and timetable not frozen repeat
  Repeat some constant number of times
  Choose a tuple Tn (class, room, period)
  Choose a new field value for Period called Period’
```
• Evaluate the cost of removing this tuple from Period
• Evaluate the cost of inserting this tuple into Period’
• Compute change in cost
• If (change in cost ≤ 0) or (change in cost is acceptable at this temperature)
  • then accept change and update cost
  • compute new temperature

SA, which was first introduced by Kirkpatrik, Gelatt and Vecchi (1983) in the paper “Optimisation by Simulated Annealing” for solving hard combinatorial optimisation problems, proved to be a good technique for many such applications (Dikmann, Luling, and Simon, 1993). Attention has been given to hard combinatorial problems like scheduling, the travelling salesman and the quadratic assignment problem (QAP) (Paulli, 1993). The advantage of SA for scheduling problems is its use as an optimisation procedure for solving the school and universities timetabling problems.

Tabu Search

Tabu Search (TS) is a meta-heuristic technique that guides a local heuristic search procedure to explore the solution space beyond local optimality (Glover and Laguna, 1997). Formally, a meta-heuristic is defined as a master strategy that guides and modifies other heuristics to produce solutions beyond those that are normally generated in a quest for local optimality in optimisation problems (Glover and Laguna, 1997). Meta-heuristic procedures have improved enormously due to advances in recent years, especially due to the developments of new implementations of TS that are more effective in solving difficult problems, than previously considered possible (Glover, 1997).

The initial design and development in TS started in the late 1960’s and early 1970’s. Glover produced the present form of TS in his paper “Future Paths for Integer Programming and Links to Artificial Intelligence” (Glover, 1986). Present research indicates that TS has become a well-established optimisation approach that is rapidly spreading in various new fields (Glover, 1993). TS helps to solve problems in a wide area of fields such as resource planning, telecommunications, VLSI design, financial analysis, scheduling, space planning, energy distribution, molecular engineering, logistics, pattern classification, flexible manufacturing, waste management, mineral exploration, biomedical analysis, environmental conversation and scores of other problems (Glover and Laguna, 1997).
The TS method was partly motivated by the observation that human behaviour appears to operate with a random element that leads to inconsistent behaviour given similar circumstances. The TS method operates in this way with the exception that new subjects are not chosen randomly. Instead the TS proceeds according to the supposition that there is no point in accepting a new (poor) solution unless it is to avoid a path already investigated. This ensures that new regions of a problem’s solution space will be investigated with the goal of avoiding local minima and which should ultimately lead to the desired solution.

The TS begins by seeking a local minimum. To avoid retracing the steps used, the method records recent moves in one or more Tabu lists. The original intent of the lists was not to prevent a previous move from being repeated, but rather to ensure it was not reversed. The Tabu lists are historical in nature and form the nucleus of the Tabu search memory. The role of the memory can change as the algorithm proceeds. At initialisation the goal is to make a broad examination of the solution space, known as ‘diversification’, but as candidate locations are identified the search is more focused to produce local optimal solutions in a process of ‘intensification’. In many cases the differences between the various implementations of the Tabu method have to do with the size, variability, and adaptability of the Tabu memory to a particular problem domain.

Hertz (1991) provides a good algorithm for the general tabu search technique as described below:

Initialisation

\[ s := \text{initial solution in } X \]
\[ nbiter := 0 \quad \{ \text{current iteration} \} \]
\[ bestiter := 0 \quad \{ \text{iteration when the best solution has been found} \} \]
\[ bestsol := s \quad \{ \text{best solution} \} \]
\[ T := 0 \]

initialise the aspiration function \( A \)

while \((f(s)>f^*)\) and \((nbiter-bestiter<nbmax)\) do

\[ nbiter := nbiter + 1 \]

generate a set \( V^* \) of solutions \( s \) in \( n(s) \) which are either not tabu or such that \( A(f(s))>= f(s) \)

choose a solution \( s^* \) minimising \( f \) over \( V^* \)

update the aspiration function \( A \) and the tabu list \( T \)

if \( f(s^*)<f(bestsol) \) then
\[
\text{bestsol} := s^* \\
\text{bestiter} := \text{nbiter} \\
s := s^*
\]

An example of using TS to solve the timetable problem can be formulated as follows: (see for example Glover, (1989); Glover and Laguna, (1997).

For each solution \(s\) this method requires the definition of a neighbourhood \(V(s)\), consisting of solution reachable in one step from \((s)\). The basic step is to move from the current solution \((s)\) to the best solution \(s^* V (s)\). A tabu list \((T)\) is used to avoid cycling as it stores descriptions of the last move or solution, while finding \((s^*)\). \(T\) is scanned to avoid so called tabu moves that could bring the search back to a previous iteration. The procedure stops after a maximum number of iterations or until the best solution is found.

In an optimisation problem the search space \((S)\) is minimised by the objective function \((f)\). A function \(N\) which depends on the structure of a specific problem is assigned to each feasible solution \((s)\) which belongs to \(S\) in its neighbourhood \([N (s) \subseteq S]\) each solution \(s^* \in N (S)\) is called a neighbour of \(S\).

TS is based on selected concepts that unite the fields of artificial intelligence and optimisation. In addition to Simulated Annealing (SA) and Genetic Algorithms (GA), TS was evaluated in the widely referenced report by the Committee on the next Decade of Operations Research (Condor, 1988) to be “extremely promising” for the future treatment of practical applications (Glover and Laguna, 1997).

**Constraint Based Reasoning**

Constraint based reasoning has proven to be a productive research method for researchers in the areas of artificial intelligence, operational research, and logic programming (Eugene and Mackworth, 1994). More recently a theoretical framework called Constraint Satisfaction Programming (CSP) has evolved and been defined (Tsang et. al., 1999). The process of *Constraint Satisfaction* involves finding values for problem variables subject to constraints on acceptable combinations of values. The satisfaction of all constraints can sometimes lead to conflicts. In these cases constraints have to be prioritised so that the more important constraints are satisfied first. Furthermore, the satisfaction of some constraints could also give rise to conflicts with the stated objective. Therefore, in some cases it is impossible to solve the problem.
completely and thus a subset of the problem is solved which is called *partial constraint satisfaction* (Freuder and Wallace, 1994).

Constraint Satisfaction Problems (CSPs) involve a set of problem variables, a domain of potential values for each variable and a set of constraints specifying which combinations of values are acceptable (Lhomme, 1993). A solution specifies an assignment of a value to each variable that does not violate any of the constraints (see Mackworth, 1994; Freuder and Wallace, 1994). A large number of problems in Artificial Intelligence (AI) and other areas of computer science can be viewed as special cases of CSP. Some examples are machine vision, belief maintenance, scheduling, temporal reasoning, graph problems, floor plan, the planning of genetic experiments, the satisfiability problem, circuit design, machine design and manufacturing, and diagnostic reasoning (Kumar, 1992).

A constraint-scheduling problem can be partially solved using search algorithms that search for best and feasible solutions with forward checking and constraint propagation. A solution to CSP involves assigning values from domains of all variables such that all constraints are satisfied. Constraint Logic Programming is the integration of two methods: logic programming and constraint solving. Logic programming has the capability of supporting declarative programming where formulation is in terms of true or false. Constraint solving reduces the search space by pruning all impossible values through constraint propagation. Constraint based reasoning is a reasoning process that uses an arc consistency technique to propagate constraints. The arc consistency algorithm is derived from the representation of constraints in a graph that can be associated with a constraint network where nodes correspond to variables in the constraint network and edges link nodes \( i \) and \( j \) if there is a relation \( R_{ij} \) between nodes. During assignment an algorithm (arc consistency algorithm) is used to check the consistency of labels for each couple of nodes that are related by a binary constraint and labels that cannot satisfy constraints are removed (Deris, Omatu, Ohta and Samat, 1997).

A solution to a timetabling problem using CSP is described as follows: (Deris et al., 1997).

The problem depicts each course as offering several subjects per semester and each subject having a specified number of lessons per week. Each lesson can thus be defined as the contact hours between lecturers and students at a specific time and place (room). Each lesson lasts for a specific period of time. Thus the timetabling problem can be
defined as an assignment of time \( t_j, 1 \leq j < m \) and rooms \( r_k, 1 \leq k \leq p \) to lessons \( s(i), 1 \leq i \leq n \) taught by a lecturer \( L(S(i)) \) such that all constraints \( C(S(i)) \) are satisfied. \( L(S(i)) \) and \( C(S(i)) \) are lecturers and constraints of a lesson respectively, while \( m, p \) and \( n \) are infinity variables. Thus the general CSP for a timetabling problem is as follows:

- A finite set of variables, \( X_1, ..., X_n \)
- For each variable \( X_i \), a set of domains \( D_1, ..., D_n \) containing possible values of \( X_i \)
- A finite set of constraints, \( C_1, ..., C_q \) representing relations between variables.

A solution to the CSP involves assigning values from domains of all variables such that all constraints are satisfied.

In terms of the CSP, a timetabling problem can be formulated by representing a timeslot and a room of a lesson as variables of the CSP, available timeslots and rooms as values of the CSP, whereas constraints are the various relationships between lessons. Therefore the CSP model for timetabling problems can be formulated by deciding the variables, values and constraints.

CSP is a decision making tool that satisfies all constraints and its major advantages are as follows:

1. Constraint propagation reduces the search space so it takes less time to search thus minimising backtracking;
2. Memory requirement is smaller since the search space has been reduced;
3. All available resources are represented in the form of constraints and hence user preferences and requirements can be easily satisfied.

**Summary**

The above completes the brief overview of the major fundamental techniques in the literature. As indicated there is a trend towards the use of constraint based reasoning. This becomes obvious as one surveys the literature presented in the next section. As will be shown in the final section this is the most utilised technique in hybrid approaches.
The Literature

Due to the immense interest in providing a complete solution to timetable generation over the years, an enormous body of literature has accumulated on this topic. The literature not only provides a chronological history of timetabling but also provides insight into the evolution of timetable generation techniques from the first manual heuristic procedures through to the present artificial intelligence based constraint satisfaction techniques. This section attempts to capture this historical perspective beginning with an overview followed by a detailed survey of the literature.

Overview

The earliest papers devoted to the timetabling problem were mathematical in nature and based on heuristics portraying manual generation systems (Kuhn, 1955; Haynes, 1959). Neither of these papers attempted to produce a realistic system, but rather they concentrated on creating a rigorous model based on the logic of the underlying problem.

The development and use of heuristics then emerged in the 1960’s. As the availability of computers increased during this period so did interest in utilising their power for automated timetabling. This reliance on the mathematical “number-crunching” ability of computers prompted the proliferation in the use of Linear and Integer Programming (LP/IP) techniques for timetable generation. However, it was not long before researchers realised computational power alone would not generate satisfactory automated solutions. Indeed, Gotlieb (1963) produced and published the first purely heuristic approach where he generated sub-problems by detecting some hard constraints and pseudo availabilities in order to reduce specific assignment arrays. This is generally accepted as the first paper on the now classic set partitioning approach and a number of other papers appeared soon after which improved upon this procedure.

Throughout the 1970’s is seen a trend in the literature towards firstly refining the LP/IP techniques and then, with the publication of the now classic text, “A Guide to the Theory of NP-Completeness”, in 1979, the realisation that perhaps a universal solution to timetable generation is not possible. Thus many of the papers in this decade, although highly mathematical in nature, were examining this new theory of intractability.
The 1980’s saw the emergence of the so-called Artificial Intelligence (AI) approach. According to Bardadym (1996), the emphasis of this approach was not on a unique mathematical solution, but on generating solutions that would be acceptable for use in a practical setting. Thus the need to identify hard and soft constraints arose. (Hard constraints being those that must not be violated, whereas a soft constraint should be imposed if possible.)

Finally, in the 1990’s the area of AI as a discipline matured and almost came of age (Schaerf, 1999). Early in the decade the rigour and detailed examination of the many AI techniques was provided. The understanding and applicability that came with this detailed theoretical examination and knowledge paved the way later in the decade, and for the start of the new millennium, for the more appropriate application of these techniques.

It seems apparent that no single technique will necessarily provide a general solution technique for the timetable problem. The general trend in the literature is towards a CSP approach.

A Detailed Survey of The Literature

The first extensive annotated bibliography of the timetable research literature was published by Smith and Schmidt (1973). Their article contained only brief commentary on the more relevant publications to that date but represented the first compilation of all articles on the subject to that point in time. Consequently, this section is structured into a discourse on the literature prior to the Smith and Schmidt bibliography (the literature pre-1978), followed by a more detailed discussion of the literature thereafter (the timetable literature of post 1978). The section is concluded with some general comments on the directions evident from contemporary research.

The Literature Pre-1978

The first two papers published on timetable construction, and the only papers published before 1960, are generally attributed to Kuhn (1955) and Haynes (1959). Kuhn’s paper adopted a mathematical approach to the fundamental timetable problem in contrast to Hayne's paper that concentrated on the more practical problem aspects of scheduling events for a conference. Given that the timetable problem has since been proven intractable (Garey,Johnson and Stockmeyer 1976) it is interesting to note that these
papers discussed the implementation of their respective algorithms claiming optimal or near-optimal solutions or solution generation.

During the 1960’s interest in timetable solution generators increased dramatically. This was linked to the more common availability of computers to perform the “number crunching” required by the algorithms developed. It was in these papers that the first use of look-ahead techniques is discussed. For instance, both Appleby, Blake and Newman (1960) and Lewis (1961) used a primitive look-ahead technique involving heuristic counting arguments.

The first non-heuristic approach was derived by Gotlieb (1963) and discussed in the now famous process (set-partitioning) of reducing the availability array and presented at the Munich IFIP congress. The main concept was to detect some tight assignments and pseudo-availabilities by generating certain sub-problems of minor complexity, which was claimed to be helpful in the reduction of the problem arrays. This was arguably the first paper on the set-partitioning approach and was further enhanced by Berghuis, Van der Heiden and Becker (1964) when the concept of virtual classes or teachers to obtain the classical bipartite problem was introduced. Csima (1965) embedded this basic problem into doubly stochastic problems which used pseudo classes.

In addition to this, Csima and Gotlieb (1964) also treated the timetable problem as a three dimensional assignment problem by considering the close connection between the timetable problem and the problem of vertex colouring. This was the first paper to consider the now well-known special relationship between the various scheduling problems. Extending the work of Csima and Gotlieb (1964) both Becker (1964) and Baraclough (1965) simulated their respective implementations with “hand” calculations. Typically these papers were based on a heuristic approach. Due to this work many other papers followed which discussed the problem but failed to cover any new ground.

An interestingly different paper, though, was published by Broder (1964) who maintained that minimisation of the objective function can be achieved by repetitively evaluating an appropriate set of nonlinear equations (derived from random or monte-carlo assignments). It was claimed that the solutions thus obtained may not be optimal but would be comparative to the local minimal results achieved by previous heuristic techniques.

Simultaneously, Csima and Gotlieb (1964) proposed an automated method of constructing timetables based on an iteration involving Boolean matrices. References were provided which relate the timetable problem to theorems on these Boolean
matrices (implemented as 0’s and 1’s), and also to theorems on bipartite graphs. In limited tests this method supposedly produced successful timetables on every attempt. However, they did discuss the problems that arose from the application of these methods to constructing timetables in real situations, thus at least acknowledging the intractability of NP-Complete problems.

Closely aligned to this is Lions (1966) description of the application of the original work of Kuhn's Hungarian method (1955) to the problem of matrix reduction as essential for the application of Gotlieb’s set partitioning method of timetable generation. This also produced extra efficiency in the basic algorithm itself. Lions (1969) extended this basic work with a generalisation method for the construction of Class/Teacher Timetables. In his paper he described an application of Kuhn's Hungarian Method to the problem of matrix reduction required in Gotlieb’s method for timetable generation. It was claimed that the resulting algorithm was useful for both hand and computer calculations. He also discussed proposed devices to improve the efficiency of the basic algorithm itself, which led to a faster solution generation technique.

One of the first discussion papers to explore the intractability of the timetable problem was presented by Macon and Walker (1966). They implemented a Monte Carlo algorithm for assigning students to classes that is a random choice technique and illustrates the problem of assigning students to a fixed schedule of courses. They noted that a final optimal result was limited by the use of the randomising algorithm.

Around the late 1960’s publications began to appear regarding certain attempts at limiting the general problem by considering case examples. For instance, Lawrie (1969) developed a model for just the school timetable problem by using an integer linear programming approach. This paper, one of the first implementing LP/IP in a “real” case study, describes an approach based around a large number of departments, grouping of pupils (generally year groups), and room layouts with integer programming formulations and computational methods to obtain solutions for the problem. Again, the paper claimed success in the generation of solutions.

In the 1970’s the usage of the heuristic approach in solving the timetable problem was adopted by several authors. Junginger (1972) provided a reduction of the timetable problem by applying it to a three dimensional transport problem. This theoretically oriented attempt to long range planning, in terms of using a Boolean matrix iteration, expanded on the concept of the relation descriptions and double integrals used by
Genrich (1966) and Friedman (1957) respectively. This approach was considered significant enough to be discussed further by Schmidt and Strohlein (1976).

Put forward by Hemmerling (1972) was a method developed for the generation of timetable solutions for South Australian schools at the University of Adelaide. The algorithm described performed an exhaustive search and determined whether or not a solution to a given school timetable problem did exist. Some results were included and various computing times quoted. The machine used for all tests was the CDC 6400 and extensive use was made of binary word patterns. It was claimed to be faster and to have been adopted in many secondary schools in the state at the time. However, given that it used an exhaustive tree-spanning algorithm, it obviously will be defeated by the combinatorial growth in the phase space.

Around this time, the growth in the size of the search space was considered to be a limiting factor for any algorithm. While investigating this, Schmidt and Strohlein (1976) introduced max-flow min-cut algorithms in hyper-graphs to further theoretical analysis. This facilitated the use of graph colouring in terms of formulating the timetable problem as certain techniques and algorithms used in this method could be re-interpreted as timetable algorithms. At the time this was considered significant although no actual formal mapping was produced.

This generated significant controversy in the literature and, for example, the state of complexity considerations put forward by Broder (1964) were disputed by Duncan and Griffith (1974) wherein they argued them to be quite imperfect. Further theoretical work was generally quite narrow and usually related to existence conditions. For instance a discussion based on preconditions corresponding to the unavailability constraints and pre-assigned meetings in the class teacher timetable problem was presented by Neufeld and Tartar (1974). Again their paper involved an examination of an earlier paper by Csima (1965).

Moving the theory at unifying these problems, Garey, Johnson, and Stockmeyer (1976) proved that the k-coloration is closely related to the timetable problem. However, it was Itai and Shamir (1975) who proved for the first time that the timetable and other problems where indeed related and belonged to the NP Complete problem set. The textbook interpretation and presentation of this theory however was presented by Garey, Johnson, and Stockmeyer (1976) in their now famous book ‘Some simplified NP Complete problems’.
The controversy in the literature, however, continued unabated with an entire sub-literature examining each problem instance in isolation. For example, an algorithm using “alternative sub-graphs” facilitated the exchange options for dead ends, which starts by satisfying the subjects most likely to lead to conflict, was discussed by Knauer (1974). He presented a program that analysed the results and used the exchange routine for an improvement within certain limits. The algorithm was then applied to school timetables and a considerable improvement in timetable quality was achieved. These computer generated timetables were used in some of the larger high schools of Wurzburg. Again, however, no reference to the intractability of the general problem was discussed.

Further in this sub-literature White (1975) considered the problem by dealing with a column generation method purported to find the right combination of timetable assignments. He also examined the solution of 0-1 sub-problems arising from duality considerations. Being highly mathematical in nature the paper made no attempt to solve the general problem nor offer how this work could be integrated with the mainstream literature.

A survey of automated construction of school timetables in the United Kingdom was published in 1975. From this, Miles and Roger (1975) discussed issues directly connected to the organisation and related activities common to computer timetable generation. Further Egner (1975) gave a general review of the evolution of the use of the computers in the construction of school timetables in Great Britain.

In the same year Tillet (1975) proposed an operations research model which discussed the feasibility of determining the optimality of an algorithm in generating a solution to the problem. The results indicated that there was improvement in only fifty percent of the cases when applying the model. It was shown that there is considerable merit in taking a quantitative optimisation approach to the problem. Such fine tuning to speed up solution generation was just starting to be considered by a number of researchers.

In the same vein, Harwood and Lawless (1975) discussed the achievement of organisational goals using mathematical models (mixed integer programming and goal programming). The goal programming related to the provision of explicit slack-variables (variables that are not mandatory but could assist in finding an optimal solution) and the mixed integer programming referred to variables, which can only take on integer values in the final solution. After the first feasible solution was reached, the
test run was terminated. It was concluded that the mixed-integer codes that were available at the time required uneconomically large amounts of computer time to obtain an optimal solution.

Similarly, a heuristic algorithm was developed by Aust (1976) for improving infeasible timetables that reduce the teaching resource, break and spread (contact hours that are broken up into different days of the week) infeasibilities into three stages. The first stage involved a series of transportation problems and the other two stages involved solving a series of small integer programming problems. An implementation of the algorithm was produced in FORTRAN. The system was then tested on four schools. Feasible solutions were obtained for two schools only and further refinements had to be carried out to achieve results in the remaining schools. For the purpose of the academic exercise, it was assumed that teacher availability was a constant for each subject. This is an unrealistic assumption.

Other related publications included educational timetable software outlined by Hulskamp (1976) in which he developed software to assist engineering students in exploring many alternative solutions to a design problem. The software was implemented at Caulfield Institute of Technology with positive results in demonstrating the complexity and phase space of the problem. This is arguably the first publication of a DSS (Decision Support System) for timetable solution generation.

Around the same time Breslaw (1976) developed a program which was both efficient and effective and which did not rely on integer programming. As a DSS, it allowed easy interaction between users and the program, in determining the optimal solution. Empirical tests provided successful construction (as opposed to generation) of timetables.

A model that was capable of providing an assignment that satisfied departmental course offerings and teaching load objectives was demonstrated by Garey, Johnson and Stockmeyer (1976). At the same time, the preferences of the faculty concerned were considered during the assignment process. The model proposed was implemented at the University of Nebraska with favourable results.

A solution facilitated by human interaction, in the general vein of the DSS, was being published next to highly mathematical treatments of the fundamental problem. In this case Ulph (1977) generalised a model to arrive at conditions that are sufficient for the existence of a solution to the simple timetable problem to allow for classes or teachers to be available more than once per period. Further extensions allowed these
availabilities to vary from period to period, which demonstrated that stronger conditions must be set when trying to achieve an optimal solution.

This move away from the highly mathematical approach resulted in a number of papers considering sub-problems or special cases. A number of papers explored the problem from this less mathematical perspective. For example, a common class of problems in the area of course timetabling is the scheduling of optional subjects so that each member of the class is able to attend as many of his or her preferred choices as possible without clashes. Barham and Westwood (1978) described a simple heuristic framework, using the above conditions, that produced a near optimal solution very quickly. Various decision rules which were tested on actual course data are identified by the authors. The results of their tests were described in great detail. It was suggested that the number of clashes for student options was dramatically reduced and better groupings were obtained.

Another example of solving a sub-problem within the general timetable activity is given by Punter (1978). His technique involved manually produced allocation plans that listed the lesson requirements for each class, and then transferred this data onto coding sheets before processing was carried out by a computer. An analysis by using a graph colouring approach enabled the identification of inconsistent sub-sets of lesson requirements. It was asserted that with appropriate amendments the scheme could be used to generate viable timetables. However, no follow-up paper was uncovered in the literature to support his assertion.

It was in this period of change in the direction of research efforts that Schmidt and Strohlein (1973) produced their well known annotated bibliography of the literature. It is noteworthy that Schmidt and Strohlein predicted the generation of timetables by computer to be heavily influenced by computing devices at hand. They further expanded their prediction in stating that timetable programs will move from remote handling in huge computing centres to micro computer centres owned by schools and handled by teachers on their own desktops directly. Twenty three years later we still do not see any evidence of their prediction and timetabling is still a manual activity carried out by administrators and not by the staff or students.
The Timetable Literature 1978 – 2001

The literature since the annotated bibliography mentioned above continued to expand with less emphasis being placed on number crunching abilities of computers. This section will show the development of new solution techniques where the use of the computer was merely as a fast calculator to assist in evaluating the technical aspects of these new algorithms. More emphasis has been placed on the meaning and interpretation of the various search techniques.

In looking at the objective function, a model which is mainly concerned with the generic roots in the so called “assignment problems” was presented by Ritzman, Bradford and Jacobs (1979) with the added feature of recognising multiple objectives. They suggested that with proper modifications, the ideas presented might well be extended to location-distribution and other types of assignment problems. These models have been applied to college timetables, with the final implementation being to study these schedules to ascertain their feasibility.

An algorithm based on a savings heuristics was developed at the University of Montana as a result of a study by Nguyen and McDonald (1980). This study was initiated to evaluate the possibility of computerising the class scheduling process. Obstacles that prevented a feasible implementation of the project are listed. As with the previous paper this work detailed an analysis of the problem as opposed to being goal orientated in producing a feasible solution.

Similarly, to help lower school timetable problems in the process of planning and implementation, a computer system was developed and published by Zarraga and Bates (1980). It was suggested by the results produced that this system could provide an invaluable tool to assist in the planning and implementation of all schools’ curriculum. Again the emphasis was on the analysis of a specific problem and not on the final timetables generated. This system added to the interactive DSS systems used by timetable planners.

In stark contrast Tripathy (1980) proposed a study on mathematical programming approaches to timetable problems which resulted in the development of an algorithm based on the Lagrangean relaxation method embedded in a branch-and-bound procedure. This algorithm was considered suitable for small problems. Its use on larger problems was also discussed. It was suggested that the solution method proposed could be used successfully in many similar timetabling situations and could even be extended
into various other scheduling problems such as examination timetabling. Consequently this paper produced a phase space spanning algorithm which did not lead to any resolution of the combinatorial problem inherent in solution generation.

Not to be outdone, Babaev and Druganov (1981) used a branch-and-bound method to obtain an optimal weekly timetable and complemented their findings with a survey conducted in Dutch schools on the timetable problem. They included a designed data structure used in a combinatorial/heuristic software system to handle most curriculum requirements and provide feasible solutions to the timetable problem. The importance of the main concept underlying this construction procedure was highlighted by de-Gans (1981).

In an attempt to develop a new solution technique, Romero (1982) published a process where solutions are generated in successive steps during interactions with the computer. This method had the advantage of coping with the aim or objective without the necessity of a normally defined objective function. It was suggested that this allowed the flexibility to provide the scope to permit the choice of an acceptable solution from the feasibility set. It was concluded that the interactive and participative procedure described could “probably” be extended to other scheduling and allocation problems due to its flexibility to permit dynamic changes in any solution.

Bulmer (1982) developed a program enabling users to deploy a pattern that matched pupils and optional subjects for that year. He also published the program code representing the implementation of the algorithm. This was the beginning of a trend where significant portions of an implementation that highlight a uniqueness factor would be published.

In conjunction with this trend, de Werra and Krarup (1982) addressed this issue at a more practical level by concentrating on real-world scheduling problems as well as using discrete models in decision-making by using the Chromatic Optimisation method. The paper illustrated in detail examples in applying this approach but drew no real conclusions. Given this new direction taken by the literature it can be seen that these two papers were indeed breaking new territory by using new improved techniques. Indeed, Osterman and de-Werra (1982) discussed the adoption of heuristic methods using various types of constraints which were used in public schools in Switzerland.

The following year, an improved method for solving typical timetable problems was described by Selim (1983). This research was a consequence of a work project undertaken for the American University in Cairo. As a consequence of the very
practical work that this was based upon, the author outlined the 26-step algorithm, which indicated computer storage requirements, showing how the algorithm copes with conflicts and explaining how to obtain the final output in convenient format. It produced feasible course timetables as well as lecturers timetables. Satisfactory results were obtained from a test at a university on a small computer. The claim that this method was applicable to any course scheduling problem in any university or institute was not only contentious but also involved an extrapolation from his small phase space to a very large phase space involving combinatorial explosion. Thus this claim does not address the underlying complexity of the problem itself. For instance complete enumeration is an algorithm that is guaranteed to locate the optimal solution regardless of problem size. But a complete enumeration algorithm is not feasible for anything but a very small problem size. The author did not demonstrate that his solution technique would remain feasible for large problem sizes.

Post 1982 could almost be termed the "era of the computer solution". The IBM PC/XT was appearing on the desktops of academics in computer science departments around the world. The ability to implement an algorithm and test it and have results to analyse in a matter of hours, as opposed to submitting a job to a computer centre which then had to schedule it with other requests, resulted in a change to the way timetabling research was being carried out. For example, Ford (1983) described the development of software and hardware specifications developed by teaching staff to meet their daily timetabling requirements. The performance of the resulting systems was outlined with the systems applying specifically to a continuous-intake, self-paced, skill-oriented program using Individualised Learning Program (LIP) methods. In this program, a computer successfully handled all testing and evaluations. No claim was made that the software would solve the general timetable problem.

In a very focused study, Dowsland and Lim (1983) described a 56k Z80-based microcomputer program for timetable creation that included general program structures, data inputs, class assignments, timetable displays, improvement programs, possible enhancements to computer models, quality of timetables produced and implications of data/program structures. In their feasibility analysis, the authors concluded that their timetable generation system was most suitable for a medium-sized school. No attempt was made to suggest the algorithm could be applied to a large scale or general timetable problem.
In ordering and then applying constraints regarding room utilisation and time restrictions for examinations, Laporte and Desroches (1984) demonstrated an algorithm that is efficient and reliable, accessed through a simple and comprehensive input routine. The system was implemented at Ecole des Hautes Etudes Commerciales de Montreal (HEC – the University of Montreal Business School). As one of the first hybrid approaches, this method utilised an integer programming model that had the advantage over other IP models as the decision variables represented the assignment of complete teaching schedules. Feasible solutions for the real and artificial problems upon which the implementation was tested were produced. No attempt was made to suggest the same algorithm would work for the general problem.

In a similar study, McClure and Wells (1984) proposed a model using decision variables to represent the assignment of complete teaching schedules rather than courses to faculty members. Their proposed model was compared with previous models for application in an academic department. Based on this comparison and the tests to which they subjected their implementation, successful results were obtained. It was pointed out that their improvement techniques could successfully speed up solution generation of existing models. However, they concluded that a significant amount of research still had to be carried out before the general problem could be considered solved.

In extending his earlier work, Tripathy (1984) proposed a branch-and-bound procedure solution that incorporated a special order set of variables, based on the Lagrangean relaxation method coupled with a sub-gradient optimisation. He presented the computational results of the algorithm for a large timetable problem involving 900 subjects in a yearlong graduate program. The author concluded there was strong potential of the Lagrangean relaxation method for tackling large-scale linear programming problems. This appears to represent the final stages of the application of purely management science approaches to the timetabling problem.

Similarly a mathematical programming approach was proposed by Ferland and Roy (1985) that solved two sub-problems in sequence. The sub-problems used have the same structure, consisting of a 0-1 assignment problem of conflicting activities to resources. A solution method is derived for them by using a relaxed version of an equivalent 0-1 quadratic assignment programming problem. This procedure is then repeated until all conflicts are resolved and an optimal solution is derived. This method was expanded upon in 1986 (see Ferland, Roy and Gia-Loc, 1986) by formulating the course timetable problem as an assignment problem in which conflicting activities are
assigned to resources. The authors then used a solution method with a heuristic exchange procedure that modified the assignments of one or two activities at a time. This solution method was facilitated by the authors developing an interactive computer system incorporating data input, data update, the use of the main procedure and the modification of a solution generated by the main procedure. This work was one of the last of the exhaustive line of research into variations of a strongly LP/IP modelling approach.

In a departure from the LP/IP driven primary algorithm, Glassey and Mizrach (1986) proposed a new method using a large 0-1 integer program which was approximately solved by a very fast heuristic approach as part of a decision support system for assigning classes to rooms at the University of California at Berkeley. It was claimed that the system was successfully implemented and in regular use since the fall of 1983. The authors concluded that the success of their method was due to the adoption of the decision support approach instead of the prescriptive model approach.

This trend towards the open incorporation of heuristic techniques was also used by Loo, Goh and Ong (1986) who started with a blank timetable in which classes are allocated according to various dependent criteria. These criteria can change according to the situation being modelled. The authors claimed that after a few trials a good solution could be obtained. The solutions produced by using this technique can be deployed by a number of academic departments. They also discussed the limitations and possible future enhancements of their approach. This approach was able to satisfy complex conditions such as pre-assignment of classes, provision of lunch and dinner breaks, reservation or blocking of certain periods and some other requirements. As with all such heuristic approaches, the mere incorporation of a heuristic narrows down the focus of application (of the search space) of the resulting implementation.

At the Memorial University of Newfoundland, Canada, Sabin and Winter (1986) trailed a registration system consisting of a collection of more than 10 different computer programs. Two of these programs dealt directly with the scheduling problem and were developed using greedy algorithms. It was noted that the first program prepared a student request file for the second program to process and be used by the other programs to manipulate files and produce reports. One of the drawbacks of this system was its inability to respond to particular time slots or sections that were requested by students. This inability to manage each and every constraint in the general problem appears to be consistent with the use of heuristics.
An analysis of a survey carried out discussing actual applications of timetables at several universities was provided by Carter (1986). Also provided were details of a tutorial guide for practitioners on electing and/or designing an algorithm for their own institutions. He described the examination timetabling problem as consisting of a set of examinations that must be assigned to a fixed number of periods to avoid clashes between students and examination. No implementation was attempted as the paper itself was more of a survey and analytical in content.

Returning to a more a mathematical approach, Schniederjans and Kim (1987) described an application of a 0-1 goal programming approach at the University of Nebraska. The results of this application show the model’s capability to provide an assignment that satisfies departmental course offerings and teaching load objectives by considering personal preferences. However, the need for significant computational power limited the application of this algorithm.

In an attempt to formalise the emerging heuristic based research approach, Tripathy (1987) recommended using a Decision Support System for class scheduling problems. The author supported this recommendation by discussing the quality of class schedules being more of a perception of the scheduler especially in cases of rescheduling. Conclusions drawn from this paper were that the use of a computer combined with a DSS framework relieves the scheduler of the monotony of preparing class schedules. It also had the added advantage of human intervention to avoid convergence on locally optimal results that plagued the more mathematical approaches.

In conjunction with this now stated trend towards heuristic modelling, an analysis of an initially constructed timetable in order to clarify the composition of the data required for programming was presented by Yuda (1987). A heuristic allocation processing algorithm was then suggested for use in programming the timetable deploying the above data. Its system design and implementation are described in the paper. Similar to Tripathy’s (1987) approach, this system allowed interactive replacement and as such the author characterises this timetable as incorporating sophisticated user judgment. However, the author’s anticipated widespread use of this timetable due to its high allocation ratio represented more of a statement regarding the success of heuristic based as opposed to mathematical approaches rather than anything inherently unique or novel with his implementation.

Also working with a heuristic method, Eglese and Rand (1987) incorporated an Annealing algorithm to help conference organisers develop timetables. This was the
first time that an algorithm used in the natural sciences had now been used as the fundamental solution generation algorithm to solve the timetable problem. The authors claimed that the developed program permitted the user to update the designed timetable and provided a sample demonstration which illustrated this.

Described by Sharratt (1987) was a design exercise where postgraduate students used his specially developed Command Language Grammar (CLG) to define interfaces and design a prototype for a transport timetable system. Outlining a detailed analysis of CLG, the author discussed the issues encountered in the CLG specifications that are used to demonstrate the problems associated with a top-down design process. Further verified were additional features that would provide more adequate support for such a process. However, there was no further work reported on this approach.

This work, though, seemed to coincide with an investigation into the use of constraint based reasoning engines, such as Prolog, to enforce the hard and soft rules that proliferate in timetabling exercises. In examining this, Murphy (1987) discussed the process of manipulating data within the constraints of a set of rules, implemented in the programming language Prolog. He outlined an attempt to combine the rule-based approach of Prolog with the interactive power of a spreadsheet to produce an intelligent design aid for school timetabling. Although no more than an elaborate, rule-based DSS, it demonstrated the then common use of supplementing a manual approach with automated tools.

The following year, Ganeshan and Bowyer (1988) also used Prolog to successfully generate timetables for schools in Singapore. They concentrated on the “real-life” constraints. This particular system simulated the behaviour of a human assistant by allowing the user to intervene, guide and override the generation process whenever necessary. It was concluded that the then recent advances in artificial intelligence and computer hardware and software made it feasible to use computers to successfully generate and maintain timetables. They proposed that further work was necessary to realise the potential of their system. Such work, though, has not come to light in the literature to date.

In keeping with the now recognised use of artificial intelligence modelling, Monfroglio (1988) described an approach in which he reduced the large set of constraints into their component rule base. This rule base was then reordered to minimise backtracking and to afford the maximum amount of the parallelism. This highly computational approach was illustrated by including implementations of it in both Prolog and Parlog systems.
along with an assessment of its performances. It was claimed that the same system, with little changes, could also solve analogous scheduling, resource allocation and planning problems. Again, for an algorithm that promised so much, it disappeared without trace.

Focusing upon the cost and time scale benefits of a particular development methodology, Forster (1988) used a knowledge based scheduling system for school timetable generation. Included in the system was an automatic and advisory mode that can provide a 15-fold improvement rate in the coverage of the phase space, which produced high quality timetables for large secondary schools.

A piece-wise incremental algorithm for constructing school timetables was discussed by Sharratt (1988). Several classes of algorithms (graph theoretic method, linear optimisation, etc) which used heuristic methods were identified and considered. Although these algorithm classes generally create satisfactory solutions they usually create other problems that are specific to a particular timetabling problem. The algorithm developed used a sequential application of these methods to avoid specific pitfalls. This largely theoretical work was not fully tested, although it represented in this AI period, a step back from the exploratory practical approaches being published in this period.

Arani and Lotif (1989) proposed a multi-phase examination scheduling process, at Buffalo (SUNYAB), in which the scheduling phase was independently solved. In phase one (assignment) the objective was to minimise the number of students taking more than one exam in the same exam block. This situation was solved by using a variation of the quadratic assignment problem. Phase two (scheduling) was formulated as a set covering problem with an extra constraint. Phase three of the scheduling process involved the assignment of exam blocks to exam periods in each day to obtain an optimal ordering of exam days. This was solved heuristically by using the travelling salesman problem as a part of the solution procedure. In conclusion, the authors claimed that the performance of the algorithms devised for the multi-phase scheduling process were tested both in terms of quality of the solutions obtained and the computer time utilised in generating these solutions. An extension from exam timetabling to the more general problem was not offered.

In an attempt to produce alternative solution techniques, a new network flow model for constructing timetables at an adult training school was created by Chahal and de-Werra (1989). This type of timetable problem differed in several ways from the classical
class-teacher problem. An interactive system based on this model was described and the algorithm presented was used to produce feasible timetables in schools much more efficiently than previous methods. This paper is very well referenced in the literature. Its uniqueness is in the development of a model specifically for use as the guiding solution engine for a DSS approach.

Focusing on a dual approach, Aubin and Ferland (1989) categorised the timetable problem into two components - namely a timetable problem and a grouping problem. The authors discussed an approach that was able to handle both these problems and also able to deal with the problems associated with the strong relationship existing between these two categories. Although very strongly related to the set-partitioning approach, the significance of this work is in the reordering and separation of constraints according to the sub problem and its interaction with the main problem. Several heuristic procedures were used to implement this approach. The article also reported on the numerical results obtained from the process which indicated a 75% reduction in the number of residual student conflicts when compared to the process actually used in the school.

In a heuristic-based approach, Carter (1989) developed a Lagrangean Relaxation method for solving the multiple period assignment problem (MPAP). A cost function model showing the general preference structure of the faculty and the administration was provided. This algorithm was claimed to have been implemented on 1400 courses at the University of Waterloo and had been successfully used for four terms, improving upon the old manual system which had been in use. The actual details of the final implementation were not discussed, thus making it difficult to infer the amount of human input required for solution generation.

Using a combination of a decision support and expert systems Falcao, Pereira, Ribeiro and Barahona (1990) investigated the concepts of timetabling at the Universidade Nova de Lisboa. Their system used several criteria to qualify a proposed timetable, whose relative importance may be adjusted by the user. The user’s dialog was facilitated by a sophisticated graphical interface which supported the creation of comprehensive reports. The system also provided complete and comprehensive report writing capabilities. When published the system’s results were being compared and contrasted with those obtained by the manual process still in use at the faculty.

Gudes, Kuflik and Meisels (1990) presented a general paradigm also based on an expert-system approach that searches for a feasible solution. The generated solution
purportedly satisfied the problem’s real-life constraints for solving resource allocation, timetable and scheduling problems within the new paradigm. It was demonstrated that the new paradigm included generic concepts for resources, activities, constraints and allocations. This paradigm was tested on three real-world problems: crews assignment to air force missions, class scheduling for a university department, and timetabling of final examinations for the faculty of natural sciences. The final system, however, does not appear to have stood the test of time and no further mention or reference was located in the literature.

Working with a heuristic approach Johnson (1990) suggested a method that is more practical and still produces acceptable solutions. The implementation was reported to produce more optimal timetables compared to the manual procedure it was replacing. One of the practical benefits of the system was a considerable saving in the clerical effort required. Another benefit was that in one set of test results, when compared with the previous manual approach to the same data, the computer generated timetable was demonstrably better in terms of the number of examination clashes eliminated.

Three models (Graph Colouring, Set Partitioning and Simulated Annealing) were investigated by Dowsland (1990) in an attempt to find a satisfactory solution for a multi-objective timetabling problem. The problem was expressed in terms of the optimisation of a single objective function by translating some of the objectives into constraints. She discussed the advantages and disadvantages of each of these models. The significant conclusion was the proposition that given the success of an exact solution approach it is worthwhile to investigate standard models that may fit a problem even if some modification may be necessary to fit the model exactly into the given problem specific domain.

In his seminal paper (as judged by both references to in the literature and the academic community) Abramson (1991) applied Simulated Annealing within a Monte Carlo scheme as an optimisation technique for use in the timetable problem. He discussed the possibility of adding cost components in order to include the more complex scheduling constraints that arise in schools. The weighting of cost components allowed a component to be made more important than others. Implementing this in a parallel program, it was demonstrated that the speed of the algorithm improved along with the results. However, although the speed of the algorithm can be improved by implementation as a parallel program the results obtained improved until overtaken by a
complexity issue related to the large number of competing processors. This hardware issue was left unresolved and an issue for future research.

Trying to anticipate the eventual contradictions contained inside a partial action and state scheme Gudes, Kuflik and Meisels (1990) presented a rule based system conceived with the purpose of dealing with plan generating systems, automatic programming, timetabling or scheduling. It was claimed that this approach separately handles the semantic aspects and the temporal constraints of the system. Based on the findings of their research, although overlapping with the concepts of set partitioning, the authors had begun developing an expert system shell to assist in the timetabling process. Again this work appears to have faded away.

In keeping with the development within the context of an expert system shell, Solotorevsky, Gudes and Meisels (1991) presented a paradigm RAPS (Resource Allocation Problem Specification) for specifying resource allocation and timetable problems. They claimed that their language enabled the specification of a problem in terms of resources, activities, allocation rules and constraints. The powerful syntax of RAPS allowed it to perform allocations that are independent of it’s inference engine. This language, built within the expert system paradigm, enabled the solution of resource allocation problems by using experts’ knowledge and heuristics.

In one of the first papers in what was to become a mainstream technique in timetable solution generation, Hertz (1991) investigated a number of techniques that incorporated the use of a Tabu Search. The stated aimed was to minimise conflicts due to overlapping courses that involved common students, teachers, or requiring the same classroom. In addition to these classical constraints the other constraints affecting this problem that were investigated included the grouping of students (courses taken by a large number of students have to be repeated several times during the week), compactness and precedence requirements, and geographical constraints. The model was implemented for assignment problems in general with reportedly good results.

Using a branch-and-bound technique, Cangalovic and Schreuder (1992) presented an exact algorithm based on enumeration for determining the interval chromatic number of a weighted graph. It was claimed that the developed algorithm and its modifications were used for solving timetable problems with lectures of different lengths. Continuing their work the following year, another paper was published which described a special group of class-teacher timetable problems. These problems were modelled as discrete lexicographic optimisation problems that considered a partial ordering between topics of
curriculum and special requirements. This method combined a general level heuristic approach for resource constrained scheduling with graph colouring projects. Based on the results obtained further applications of their method on real-life problems should prove successful, or so it is asserted.

Interestingly enough, Kang, Von-Schoenberg and White (1991) described a complete university timetable program that schedules student-professor-course combinations into suitable classrooms at specified times. It was suggested that a number of primary constraints are satisfied by the timetable and a number of secondary constraints can be enforced if necessary. The program, written in Wprolog, consumes about 25 minutes of CPU time on an Amdahl 2880. Although the goals of the program were accomplished no promises of further application were made.

In an approach based on logic programming using first order predicate calculus Kang and White (1992) attempted to provide a formal basis for the use of expert systems. An algorithm was developed and implemented in WProlog, and also thoroughly tested on an Amdahl 5880 under VM HPL. Being research at such a low level no practical problems were investigated or reported.

Lin (1992) wrote a set of APL code for the optimisation in multiple choice programming (MCP) using the special-ordered-set of a branch-and-bound procedure. This incorporated the partitioning strategy of a weighted-mean method. It was argued that the MCP problem, originating from W. C. Healey (1964), belongs to a special class of combinatorial problems. This class of problems can be used for generalised assignment problems, knapsack problems, sales resource allocation, multi-item scheduling, timetabling, etc. Several heuristics were reported in the literature for general MCP along with the aforementioned solution procedures. Due to the mathematical nature of the paper, no conclusion or summary was provided.

Using the completeness nature of the fundamental problem Kiaer and Yellen (1992) described a fast heuristic algorithm that utilised a graph colouring approach to search for solutions. Presented were the results obtained which denoted the performance of this algorithm on several test problems. Although overlapping time slots, varying classroom sizes and other standard timetable complexities were not included in the test data, it was claimed that these considerations would only require minor “bookkeeping” changes in the algorithm.

Working in much the same area, Ferland and Lavoie (1992) proposed the approach of a heuristic iterative procedure where the assignment of one item is modified at each
iteration and this is used to develop an algorithm to implement a timetable. To induce this a theoretical framework using a geometric interpretation of an exchange along with two other procedures to prevent jamming situations outside local optimum values, and a Lagrangean relaxation method, were provided. Also provided was a detailed analysis of the numerical results obtained but no valid conclusions were offered. As with the briefly discussed research this paper has added volume to the literature.

The rationale for a MCADSS (multi criteria allocation decision support system) was described by Weitz and Jelassi (1992) along with design criterion, system methodology and application results. The goal of this system was to maximise the diversity of the members within groups while minimising the differences between these groups. Also discussed was the potential of future research in this area and the possibility of adapting this program for use in other academic or training situations. The system was in regular use at the European Institute of Business Administration at the time the paper was published. No other mention of this paper could be located in the literature.

In a more fundamental investigation Tripathy (1992) noted that a special characteristic of the timetable problem is the highly dense conflict matrix. Consequently it was recommended that computer-aided decision making be used to arrive at a desirable timetable. The related problems were discussed in terms of lack of an objective function and quality. The main conclusion of this paper was that it is desirable to actively involve the decision maker in the timetable generation process. This was one of the first papers in the now well established area of intelligent scheduling systems.

The use of KADS (a methodological approach to a small knowledge-based systems project) was described by Kingston (1992). This system assisted undergraduate students in choosing courses that complied with university regulations and timetable restrictions. He also described the three major phases of a KBS project, giving a brief outline of the KADS approach for each of these stages, and then discussed the techniques that were actually used for Course Selector. Conclusions drawn from this paper recommended techniques for future small KBS projects. Nothing further emerged from this work.

Graves, Linus and Shankaran (1993) proposed a method that deployed a market approach for allocating course selections to students. This represented a change from the “tackling the problem” approach to looking at those involved as participants in the entire process. The students were allowed to express preferences for combinations of courses rather than for a single course. This allowed the students to express their
preferences for courses over multiple school terms rather than just one term and thus students need not wait in registration queues. Using this system the initial goals of avoiding physical queues were achieved. This paper represented a significant departure from the mainstream literature but was well received in the sense that it paved the way for work looking at the management of constraints and database based approaches.

Discussed by Johnson (1993) was the use of a database management system for the “bookkeeping” aspects of timetable development. It was claimed that the system was implemented at Loughborough University Business School and proved to be effective. It was demonstrated that there is seldom any real likelihood of being able to completely and effectively automate or computerise the timetabling process thus adding weight to the work done by Graves et al (1993).

Adopting a different approach, Cooper and Kingston (1993) described a timetable specification language which allowed the program to handle the many idiosyncratic constraints in a uniform manner. It was claimed that the new algorithms introduced could tackle the problem more intelligently than any of the traditional search methods. The conclusions drawn indicated that an assignment algorithm which is unaware of a tightly constrained sub-problem laying in its path will fail to solve that sub-problem. On the other hand an assignment algorithm which is designed to solve that sub-problem in particular will actually benefit from the tight constraints imposed upon it.

The development of a microcomputer-based timetable generator using expert system technology was described by Martinsons and Kwan (1993). This system was developed for a Hong Kong tertiary institution with a very dynamic academic environment. The knowledge, strategies and heuristics of a small, centralised group of schedulers were modelled. This was represented in a readily available expert system shell which can run on a standard IBM-type microcomputer. Also discussed in the article was the broad feasibility of expert-level timetable generators and, more generally, the application of this knowledge-based system approach. The results achieved by this system were argued to be cost effective. The state of the system as it currently stands is unknown as no further reference to it can be found.

Using an expert system that was developed with Domlog, Azevedo and Barahona (1994) presented the solution of the timetable problem at the Computer Science Department of the FCT/UNL. Domlog is a Constraint Logic Programming (CLP) (FD) system, that extends CHIP with features such as user-defined heuristics, and a more flexible look-ahead constraint solving method. It was claimed that this CLP approach
could result in the generalised Logic Programming unification to constraint solving over a computational domain. The results indicated that although these problems were specified quite declaratively, the system required a relatively short implementation time and would still be able to solve the problem efficiently. However the application to the general timetable problem was neither discussed nor referenced.

Discussed by Gunadhi, Anand and Yeong (1994) was a Decision Support System (DSS) approach to the timetable problem that was made up of knowledge, data and model bases, as well as various user interfaces. The architecture facilitated a more flexible and maintainable system with the inclusion of human expertise in the model. It was demonstrated that an object-oriented development approach benefited the system with a more efficient design and code implementation. Also noted was the use of objects, methods, messages and functions providing greater flexibility for solving the problem. The preliminary results obtained from test runs carried out with 3500 students were very encouraging but no further work could be located.

The same year Ferland and Fleurent (1994) also developed a Decision Support System for use in a limited environment. By incorporating efficient optimisation procedures and interactive tools to help the user adjust the schedule to individual specifications, an initial solution was generated which the user could then modify. This system was based on heuristic algorithmic methods that proved to be successful at the computer science department in the University of Montreal, Canada, and at the college of sciences of Sherbrooke University, Canada. However, no further work can be located on this project.

In an article littered with incredible claims Yoshikawa, Kaneko, Nomura and Watanabe (1994) described a case study on a general purpose Constraint Relaxation Problem solver, COASTOOL. It was concluded that, by using COASTOOL, a problem can be solved merely by declaring “what is the problem”, without programming “how to solve it”. The problem was solved by a novel method that generated a high-quality initial assignment using arc-consistency, and refined using a Hill-Climbing approach. This system was successfully implemented in a high school situation, proving to be efficient. However, efficiency comparisons were not presented in any objective way.

Around this time the first papers using Genetic Algorithms began to appear. Thus Corne, Ross and Fang (1994) demonstrated how Evolutionary Algorithms (EAs) can be employed to effectively address arbitrary instances of the General Examination/Lecture Timetable Problem (GELTP). Also discussed were several research and
implementation issues concerning EAs in timetabling. A class of specialised mutation operators for use in conjunction with the commonly employed penalty function based EA approach to timetabling was included in the discussion. This approach showed significant improvement in performance over a range of real and realistic problems. The use of delta evaluation, an obvious and recommended technique, which speeds up the implementation of the approach was also reviewed. The paper recommended that on implementation their operators lead to a more pertinent measure of speed than the commonly used “number of evaluations”.

Alternatively, Zajicek and Brownsey (1994) described the interface characteristics of an interactive constraint based resource allocation system (ICBRAS). Discussed in the article were specification methods existing at the time, including a brief analysis of their suitability. Their findings were justified by the implementation of the system into Oxford Brookes University. An analysis of a special test case was also discussed.

A heuristic approach was demonstrated by Minciardi, Paolucci and Puliafito (1994) based on the decomposition of the original problem instance into a sequence of successive instances of a simpler sub-problem, each corresponding to a certain decisional horizon. Furthermore, heuristics were used to generate feasible schedules for the sub-problem instances considered. The overall schedule turned out to be semi-active, and determined a set of decisions for the assignment and sequencing of tasks over the resources. Such decisions were then retained and interpreted as constraints on the final step, i.e. timetable optimisation. This promising initial work represented a new look at the old set partitioning techniques.

By combining both traditional methods and advanced modern methods Burke, Elliman and Weare (1994) discussed an automatic timetable generation method that used applications such as Graph Colouring and Genetic Algorithms. It was claimed that their method was being implemented at some United Kingdom Universities. Although cumbersome in style, the use of hybrid approaches on this scale was successfully achieved.

In the same year Burke, Elliman and Weare (1994) also presented a description of the methods and techniques involving Graph Colouring and room allocation algorithms. The combination of these two methods provided a basis for a flexible and widely applicable timetable system. The problems of intractability were overcome by producing a spreadsheet-style system that the user could guide in an informed and useful way. This gave control of the search and offered the possibility of backtracking
when no reasonable solution could be found, while still allowing the heuristic algorithms to do the hard work. It was concluded that although this approach did not guarantee an optimal solution, it could guarantee a solution that was feasible.

Continuing their work the following year, Burke, Elliman and Weare (1995) discussed a series of recombination operators which acted upon a direct representation of the timetable for the timetable problem. The authors compared different approaches to solving the timetable problem using evolutionary computing methods. They presented a recombination of operators and described various alternatives for incorporating heuristic knowledge into the search. This continuation of research is unusual in this area, demonstrating the effectiveness of the use of hybrid approaches.

By applying Object Oriented Constraint Logic Programming, Deris, Omatu and Ohta (1995) described the solution for the timetable problems of privately owned colleges. The results showed that the highly constrained problems could be solved using minimal computing resources compared to substantial man-months. It was proposed that this method (Object Oriented Constraint Logic Programming (OCLP)) offered many advantages, as the model presented could be applied to similar types of institutions with a minimum number of changes. Such claims are not unusual in the expert systems based papers.

Frangouli, Harmandas and Stamatopoulos (1995), used a method incorporating Constraint Logic Programming to construct optimum timetables for the University of Athens Department of Informatics courses. The software platform used for the implementation was an instance of the Constraint Logic Programming class of languages, the ECL/sup i/PS/sup e/ system, and proved that the system was an appropriate vehicle for managing the complexity of the timetable problem. A specific system was used for the timetabling procedure of a department within a university with supposedly satisfactory results.

A Genetic Algorithm technique was proposed by Paechter, Cumming and Luchian (1995) for timetable generation. The uniqueness of this algorithm involved the use of suggestion lists for the placement of events into time slots. The experimental results were provided in order to compare the performance of the operators with each other and with a system not using the suggestion lists. This paper demonstrated how preference constraints can be encoded into an evolutionary based approach.

Using a different approach, Nice (1995) described the chromosome representation employed for the examination timetable problem. A genetic algorithm was developed
to schedule the lectures and practicals. Multiple time slot constraints with respect to students which are to be minimised, were provided along with a discussion of their performances. As with all such approaches a discussion of the quality of the solutions generated ensued.

Making comparisons between Genetic Algorithms (GA), Simple Heuristics (SH), and Simulated Annealing (SA) on a collection of real timetable problems, Ross and Corne (1995) concluded that SH and SA are generally the best strategy as far as solution quality is concerned. However for a fairly small range of particular problems the GA either equals or betters the performance of SA and SH, only with the added value of a large number of usefully distinct, equally good solutions. This work has value in being one of the very few to perform a direct comparison between different approaches on the same specific problem.

A two stage examination scheduling system using a microcomputer for the University of Calgary, Canada, was developed by Colijn and Anton (1995). In the first stage mathematical methods were used to create a timetable that is conflict free. These methods used a minimum number of periods, and minimised the number of students with multiple exams in one day. The next step was an interactive stage that allowed the examination officer to accommodate late requests and special circumstances. It was claimed to have a successful implementation of this system. However this must be viewed in the context of the relatively straightforward examination sub-problem not as a solution for the complete timetable generation process.

Exploring a technique for developing a conference (or class) schedule, Sampson and Weiss (1995) maximised the servicing of participant requests for sessions which have been prioritized. The data regarding participant interests was collected beforehand and a schedule generated by considering (1) the feasibility to assign session offerings to time periods and rooms; and (2) to assign participants to sections of multiple offering sessions. A heuristic procedure that simultaneously solved the above two parts was described and tested. In conclusion, it was noted that the procedure considered not only the participants’ requests for sessions, but also the relative importance of the various requests.

A combinatorial approach was presented by Roux, Bouquard and Partant (1995) using different automatic generation methods (heuristics and methods of optimal solution searching). The timetabling problem consisted of placing courses, which share resources (such as teachers or classrooms), in a calendar. They have successfully
developed a program using CHIP (Constraints Handling In Prolog) which was completed by two students of the E3I (Ecole d'Ingenieurs en Informatique pour l'Industrie-Tours, France). As has been noted is the norm with such papers it was claimed that the implementation of this program and its results were satisfactory.

Babes and Quilliot (1995) developed an interactive approach, which was expressed to resolve and formulate logical, resource type and temporal constraints. To achieve the above some parts of a Prolog program were used and a determination of flows, along with the constraint propagation and least engagement techniques.

Also using Prolog, Erben (1995) aimed not only at finding a feasible solution to the problem but also one with good quality i.e. to satisfy as many secondary constraints as possible by using a Genetic Algorithm. This algorithm was claimed to have been implemented in a real-life environment and produced promising results. It was concluded that a number of refinements still needed to be implemented, including a component which allowed the user to adapt various parameters at run time.

In the same vein, Gueret, Jussien, Boizumault and Prins (1996) claimed that CSP formalism improved the flexibility of the Constraint Logic Programming (CLP) paradigm. It was claimed that this system provided the ability to rapidly design efficient search procedures that provide feasible solution. These implementations were developed using CHIP.

Also using Constraint Logic Programming, Boizumault, Delon and Peridy (1996) developed an application where the CLP was used over finite domains. A brief overview of OR approaches for solving examination timetable problems was provided. Also discussed was the examination timetable problem for universities showing how Constraint Logic Programming over finite domains can be used to solve the problem efficiently. It was concluded that the experiences of the authors illustrated the important potentialities of Constraint Logic Programming for the prototyping and implementation of real life applications.

The humane and profane facets of using the Constraint Logic Programming approach were discussed by Cheng, Le-Kang, Norrus-Leung and White (1996). The constraints were hierarchical i.e. the primary constraints were rigidly enforced and the secondary constraints were relaxed according to their priority if a solution could not be found. Presented was a solution based on a Prolog description of the constraints and goals. Tests with synthetic data and real data from a university showed that good timetables could be cast using this method in a reasonable amount of time.
Discussed by Burke, Elliman, Ford and Weare (1996) were the results of a questionnaire on examination timetables sent to the registrars of ninety-five British universities. The survey asked questions on three specific categories about the nature of their examination timetable problems, how the problem was solved at their institution and whether a manual or automated system was used. The registrars were also asked to list the qualities required in a good timetable and recommend which automated systems should be used. Their analysis of the survey was presented along with some comments from the surveys which indicated the type of criteria a general automated timetabling system must meet.

Based on these responses Burke, Newall and Weare (1996) claimed that Hybrid Evolutionary Algorithms yielded better results than Evolutionary Algorithms even though evolutionary techniques with general purpose have optimisation capabilities. A Mimetic algorithm was used which incorporated local search methods into the Hybrid approach. It was noted in conclusion that this Hybrid Evolutionary method gave better results.

Commenting on their current research Paechter, Cumming, Norman and Luchian (1996) described work that was in progress which aimed to increase the performance of a Mimetic timetabling system. Experimental results showed favourable performance improvements with directed and targeted mutation and acceptable first results with the structured population.

Researching in the same area, Rankin (1996) described the experience of using an automatic timetable system, based on a Mimetic algorithm, for courses over 500 students. This automatic algorithm was supported by a records system, which permitted manual data collection and editing, viewing and printing of timetables. The method developed to use the system was outlined in detail and the benefits and difficulties arising from its practical application were also discussed. It was indicated that although there are difficulties with its practical application, the system could produce a working timetable which was used regularly.

In an attempt to categorise different techniques, Carter and Laporte (1996) classified different algorithms and discussed the reported results from a survey. The survey did show that there had been little interest in Simulated Annealing approaches but Tabu Search methods had proven to be more popular. However, no attempt was made to perform experimental comparisons on different methods (cluster, sequential, generalised search and constraint-based approaches).
A dual approach using a combination of Evolutionary Algorithms (EA) and greedy algorithms was proposed by Corne and Ross (1996), in which they retained the speed of the simpler method while adopting the greedy method to bootstrap the process. The initial population was obtained by a peckish timetable method that is similar to a greedy algorithm. It was shown that the peckish initialisation aided a simple Hill Climbing approach. It was demonstrated that there is a growing observation that Hill-Climbing methods often outperform an EA on timetable problems, but that this effect was reversed on particularly over-constrained or difficult problems.

Carrying on this work Ross, Corne and Terashima (1996) investigated some of the constraint satisfaction issues and uncovered some new phase transition regions on timetable problems. These occurred in the context of varying degrees of problem homogeneity and graph connectivity. It was concluded that a simple stochastic Hill Climber was the best method compared to a stochastic Hill Climber in certain phase transitions.

A heavily constrained university timetable problem using a Genetic Algorithm based approach for solution generation was described by Erben and Keppler (1996). To facilitate this a problem-specific chromosome representation and knowledge-augmented genetic operators was developed. It was asserted that these operators “intelligently” avoided building illegal timetables. The prototype timetable system that was presented was implemented in C and Prolog, and included an interactive graphical user interface. The test data included in this paper indicated promising results.

Working in the same area Rich (1996) developed a software solution, based on a Genetic Algorithm (GA) for optimisation which was designed for creating a university class timetable. The GA, assisted by a dynamic penalty function and greedy algorithms using domain knowledge, resolved the constraints imposed in a complex environment. It was concluded that the developed techniques create an intelligent Genetic Algorithm for solving discontinuous, complex and highly stochastic optimisation problems.

Using a dual approach Ergul (1996) suggested a method which included two adaptive mutation operators that yielded a more robust genetic search, a proximity matrix for efficient computation of the fitness function, a scaled conflict matrix and temporal suspension of highly conflicting exams which resulted in schedules with better patterns. It was claimed that this system was implemented with favourable results in a Middle Eastern technical university involving 682 exams in one case and 1449 in another.
Introduced by Gunadhi, Anand and Yeong (1996) was an automated timetabler that combined data, model and knowledge bases using object-oriented methodologies. Their results were claimed to be promising as the system had automatically scheduled over 2000 students and instructors into more than 300 classes. This approach was concluded to be preferable to traditional methods due to the lack of flexibility, computational intractability and poor results obtained from these techniques.

Discussing a typical scheduling problem Henz and Wurtz (1996) attempted to compute a feasible timetable for an academic college. The problem contained a variety of complex constraints and necessitated special purpose search strategies. The techniques from operations research and traditional Constraint Logic Programming were not able to express constraints and search strategies on a high enough level of abstraction. Comparing the different techniques, it was concluded that the higher order concurrent constraint language, Oz, is more effective due to its high level of expressivity.

Carrying on this work Wurtz and Muller (1996) described Finite Domain Programming (FDP) which is a technique that employs disjunctive constraints to solve combinatorial problems such as timetabling. Constructive disjunction was a rather new approach to model those constraints, whereby common information is lifted from the alternatives, aiming for stronger pruning of the search space. The concurrent constraint language OZ was made use of and showed where constructive action provided for stronger pruning and where it failed to do so. For several problems, including a real-world college timetabling application, the benefits and limitations of constructive disjunction were exemplified. It was shown that constructive disjunction does not work for scheduling and timetabling problems where domain specific knowledge can be used in a constraint setting. However it was very useful for complex problems where a special purpose strategy was unknown and flexibility required.

A practical demonstration of this approach was illustrated by Lajos (1996), in preparation for the changeover to a new modular degree structure, at the University of Leeds for the year 1993-94 academic session. Constructing a large-scale modular timetable using Constraint Logic Programming techniques was described.

Using a different approach, the application of multiple context reasoning and truth maintenance to the automated generation of timetables was described by Ram and Scogings (1996). The system used was made up of a rule-based problem-solver that made inferences and an assumption-based truth maintenance system that maintained a record of the justification. Various implementation prototypes were constructed and
tested on a subset of a university-timetabling problem and the positive results obtained were discussed in the conclusion.

Using a sectionalised approach Robert and Hertz (1996) divided the main problem of timetabling into several sub-problems (assignment types) and tackled them individually, subject to certain constraints. In doing this the total number of overlapping situations was reduced in the subsequent phases by means of another series of assignment type problems. The implementation of this approach provided satisfactory results.

Described by Schaerf (1996) was a General Examination/Lecturer Timetable Problem (GELTP), which covered a very broad range of real problems faced continually in educational institutions. The use of how Evolutionary Algorithms should be employed to address arbitrary constraint instances of GELTP effectively was described. Benchmark and implementation issues were also covered.

Not directly related with timetabling but still being relevant, Wren (1996) explored the relationship between the problem types. The history of the use of computers for solving timetabling, scheduling, and rostering problems was discussed. The early age of mathematical formulations was identified as being unable to solve the problems due to limited computer power, however very specialised heuristics were used from an early age. Mathematical approaches were nourished again during the 1980s and some subsequent developments were found to be very powerful in practical situations. The developments as applied principally in the areas of vehicle routing and scheduling, driver scheduling, job shop scheduling and personnel rostering were discussed. It was asserted that a comparison can be made with class and examination timetables. The similarities of the different types of application to the same problem were examined.

An automated solution was developed by Carter, Laporte and Yan-Lee (1996) using a computerised examination timetable system called EXAMINE. A comparison was made among several algorithmic strategies for this problem. It was shown that the combined use of a backtracking strategy combined with a saturation degree sorting rule almost always delivers shorter schedules in less computing time.

A categorisation approach was described by Monfroglio (1996) where the classification of constraints and constraint ordering was utilised to obtain the minimisation of backtracking and the maximisation of parallelism. Also illustrated was an approach based on the hybridisation of constrained heuristic search methods with novel Genetic Algorithm techniques. This compared favourably with known programs to solve decision problems under logic constraints. Also reported was the cost of the new
The University Timetable Problem
K. Sandhu

algorithm and the quality of the solutions obtained in significant experiments. Presented were the implementation of the presented algorithms during seven years of research and the potential for its use in artificial intelligence.

On the other hand Wright (1996) discussed the various complexities of the school’s lesson structure along with constraints and objectives. The resultant timetable was successfully implemented into a large and complex secondary school in Lancashire, England, in September, 1994. He used a four-phase solution method with heuristic search that had little or no manual intervention. It was claimed that in comparison with previous timetable systems this method was of a higher quality and standard, and a marked improvement.

An automated methodology adopted by Mamede and Soares (1996) described the implementation of a timetable generator that used a constrained heuristic search method, combining constraint programming with search techniques. Designed to cater for a university with seven thousand students, in particular the Instituto Superior Tecnico, Lisboa, a strong effort was made to achieve a generic timetabling system. A short summary described the technique used and suggested that the representation of preferences (soft constraints) can be related to any particular problem.

An interesting proposition by Mauser, Magazine and Moore (1996) described an application of annealed neural networks to solve a timetable problem with contiguity and availability constraints. The problem formulation was organised by the implementation of annealed neural networks and results were computed as a series of randomly generated problems. The main point of note was that the solution quality varied depending upon the size and complexity of the problem.

Concerned with the use of Simulated Annealing in the solution of the multi-objective examination timetable problem, Thompson and Dowsland (1996) proposed a solution method that optimised groups of objectives in different phases. Depending on the solution quality with respect to earlier phases in the algorithm, some decisions may be altered which do not deteriorate the final result. These limitations may disconnect the solution phase by causing optimal or near optimal solutions to be missed. It was suggested that a Simulated Annealing method can be used to overcome these variants. Thompson and Dowsland (1996) then discussed the utilisation of TISSUE, a Timetable package using a robust cooling schedule based on the Simulated Annealing approach. It was claimed that the developed package performs well on real life test data without
sacrificing solution quality. The package was successfully implemented at the University of Wales, Swansea.

In the same year, de-Werra (1996) formulated the problem in terms of Graph Colouring (or hypergraph colouring) for a collection of simple class-teacher timetable problems and a review of complexity issues for these formulations is given. The extensions and variations of basic chromatic scheduling models, motivated by applications in automated production systems and course timetabling were examined. Although the contexts were very different, it was proved that the basic models were very similar. Various types of requirements that were common to production scheduling and timetabling were also presented and illustrated. It was concluded that more research was required in this area.

A new technique for improving the number of usefully distinct solutions was presented by Turner, Corne, Ritchie and Ross (1996) where a Genetic Algorithm (GA) was applied to multi-modal problems. A comparison of the technique with two well-known niching methods (crowding and sharing) was given. It was concluded that the tribes technique can greatly improve the efficiency with which a GA can obtain multiple distinct solutions to a problem.

A different approach by Boufflet and Negre (1996) described the problem of examination timetables at the University of Technology of Compiègne, France, where administrative and physical constraints using three tools were considered. The first tool was an exact method based on a tree search, the second based on a Tabu technique, and the third was an interactive computer aided design system that can be used if all other techniques fail. It was suggested that the most effective method was the tree search method.

Working along the same vein, Alvarez-Valdes, Martin and Tamarit (1996) developed a three-phased algorithm for Spanish school timetabling. In phase I, an initial solution using a parallel heuristic algorithm with priority rules was deployed. In phase II, a Tabu Search procedure started from the solution of phase I and obtained a feasible solution to the problem. The solution obtained was then improved in phase III. The system made use of negative cost cycles and shortest paths in a solution graph to produce more timetables. Compact timetables were generated by the algorithm and used in Spanish schools. It was claimed that the procedures could be adapted to other education systems in other countries.
A study carried out by Mausser and Magazine (1996) made a comparison between an annealed neural network with that of graph colouring and pricing heuristics in solving a timetable problem. The above methods were applied to randomly generated problems of varying size and difficulty based upon the solution quality and execution time. Based on the study it was claimed that even though annealed neural networks have the potential to find solutions, the heuristic procedure was far more effective for practical purposes.

In the following year Baumgart, Kunz, Meyer and Cirkel (1997) presented a timetable application for the University of Siegen, Germany, based on an object-oriented taxonomy which split constraints into hard and soft categories. In their approach an Object oriented method was used where the solver provided the satisfaction of as many soft constraints as possible. It was proved that the hard constraints must be satisfied to achieve a consistent solution and soft constraints were ordered by priority. The prototype achieved the same results as the conventional planners in significantly less time.

In the same year Birbas, Daskalaki and Housos (1997) presented an efficient solution to the timetabling problem for the secondary educational system in Greece. Using an integer programming approach and utilising commercial tools that were available for a class of scheduling problems, the process of locating the optimal solution was facilitated. Demonstrating the effectiveness of the model, it was illustrated that their method produced optimal results for the specific constraints of the problem.

A novel experiment by McIlhagga (1997) discussed a comparative study that was carried out where a Distributed Genetic Algorithm (DGA), a random search and a heuristic method of scheduling using 100 very large scale problems of the order of 500 tasks was conducted. The study showed that only DGA was capable of providing solutions that had as high as a 60% reduction rate. The GA-based scheduler allowed the user to define and solve any scheduling problem using a scheduling description language (SDL). A unique chromosome-coding scheme that allowed simple representation, straightforward chromosome recombination and fast schedule building was described. It was concluded that the scheduler was able to solve problems such as job-shop scheduling (JSS), timetabling, and resource sequencing with favourable results.

In another study carried out by de Werra (1997) various formulations of timetable problems in terms of colouring problems in graphs were mapped to a collection of
simple class-teacher timetable problems. A review of the complexity issues for these formulations was made. Also provided was a brief sketch of a Tabu Search procedure that handled many specific requirements and provided an efficient heuristic technique.

Carrying on this work de-Werra and Mahadev (1997) considered the problem of extending a pre-colouring to an edge $k$-colouring of the whole graph (where $k$ is fixed) and described some polynomial solvable cases of the problem which is generally NP-Complete. They were represented by pre-colourings in the graphs associated with the models. A model with costs was provided and showed that the problem can be solved in polynomial time. It was concluded that in open shop and class-teacher timetable problems pre-assignment requirements are often present.

From an object-oriented perspective Deris, Omatu, Ohta and Abd-Samai (1997) discussed mainly the maintainability and extendibility of object-oriented technologies, enhanced model formulation and maintenance by inheritance and polymorphism properties of classes and objects. It was suggested that the object-oriented approach also enhanced constraint propagation and search by object interactions and communications. By applying the object-oriented approach, it was concluded that the declaratively and efficiency of the solution generator can be enhanced.

Another paper was published during the same year by Deris, Omatu, Ohta and Samat (1997) in which a case study was presented applying constraint-based reasoning to university timetable planning. This was formulated as a constraint satisfaction model and could be solved using constrained-directed search algorithms with built-in forward checking and constraint propagation. Using the constraint based technique with recent real data, a timetable which meets user requirements and the objectives of the timetable was claimed to have been produced.

In the same year Drexl and Salewski (1997) showed that several types of constraints may be modelled using the unifying framework of partially renewable resources. Presented are two-phase parallel greedy randomised and genetic methods. An instance generator for the generation of a representative set of instances was also given. Computational results showed that this method solved the instances investigated close to optimality.

A research project was presented by Kragelund (1997) where Genetic Algorithms (GAs) are used as the basis for solving a timetable problem. The strength of co-evolutionary constraint satisfaction technique with that of two other GA approaches was discussed. Distributed GAs and a simple special-purpose Hill Climber were introduced.
to improve the performance of the three algorithms. It was concluded that the performance of the distributed hybrid GAs on a very difficult real problem were mostly successful.

A survey of Hong Kong secondary schools on the scope of timetabling was conducted by Kwok, Kong and Kam (1997). The results indicated that there existed a need for further research into appropriate software to automate the timetabling process. It was concluded that human interaction was required to complete any automated generation of a timetable, however a complete solution was highly recommended to reduce the workload of timetablers.

In a study carried out by Meisels, Ell-Sana and Gudes (1997) adequate proof of the existence of necessary and sufficient conditions requiring the provision of a solution to decomposed School Timetable (STT) problem grids by using a decomposition procedure was provided. Due to the size of STTs, it was concluded that good constraint satisfaction problem search techniques (i.e. back-jumping and forward checking) do not terminate in reasonable times for STT Grids that have larger than 300 variables.

Using a dual approach Soares and Mamede (1997) introduced a new paradigm for automated timetabling based on models and techniques for scheduling. The developed paradigm is constraint-based and object-oriented. These two methods allowed for an easy representation of the timetable problem and the handling of new timetable constraints, either “hard” (should be satisfied) or “soft” (should preferably be satisfied). Scheduling concepts such as activity and resource were translated to the timetabling domain, and a general Bardadym's scheduling method, named micro-opportunistic approach, was applied in this novel domain. It was suggested that computer assistance was of great importance in solving timetabling problems and was described in different aspects such as intuitive graphical interface to an automated timetabler.

In the same vein Stallaert (1997) implemented a course timetable system for the Anderson School of Management at UCLA. She divided the overall problem into two sub-problems: 1) to schedule the core courses using an integer-programming algorithm; and 2) to generate a timetable by using a heuristic algorithm to solve a variant of a quadratic assignment problem. She used DSS for generating solutions that she claimed would work faster.

A novel approach by Ucoluk (1997) was presented where a new method for representing r-permutations of n-elements as GA chromosomes was introduced. Claiming that his proposed representation was not handicapped under crossover and
mutation, this method could also be used in various scheduling and timetable GA applications problems. It was observed that this new method performed extremely well in most situations.

In a study carried out in the same year Weitz and Lakshminarayanan (1997) concluded that an adaptation of a pair-wise exchange procedure drawn from the final exam scheduling literature outperforms a more sophisticated graph theoretic approach. It was claimed that their method outperforms the more sophisticated approach that was previously shown to be a top performer for VLSI design.

In the following year Monfroglio (1998) termed the finite constraint satisfaction problems also called shared resource allocation (SRA) that was tractable, i.e. it is in the P class of complexity. It was identified that the discrete optimisation, operation research, database management, pattern recognition, and multi-tasking problems can be restructured as Finite Constraint Satisfaction Problems (FCSP). Moreover, an important FCSP restriction named shared resource allocation (SAR) and several neural network approaches were also described to solve conjunctive normal form-satisfiability problems. In conclusion different neural network approaches to solve conjunctive normal form problems were proposed.

A comprehensive survey carried out by Schaefer (1999) presented an annotated bibliography of the evolution of the timetable problem from its traditional roots in the operational research field through to techniques belonging to Artificial Intelligence. Different variants of the timetabling problem were proposed and discussed based on the type of institution involved and the type of constraints used. The various formulations of the problem, techniques and algorithms used to generate solutions were also described.

Describing a solution generator for timetabling in high schools, Schaerf (1999), discussed the alternative direction taken by her algorithm. Traditionally the problem was attempted to be solved using heuristic methods however the author utilised and implemented a solution algorithm based on local search techniques. It was claimed that the algorithm was successfully implemented in large high schools with various kinds of side constraints. In all cases the timetable produced appeared to be better than manually created schedules.

In the same year Drexl and Sprecher (1999) used a branch-and-bound algorithm in relation to semi-active schedules. It was demonstrated that these techniques were effective in providing solutions to these types of problems. Although no conclusion
was given it was suggested that the algorithm would generate an optimal solution which need not be semi-active.

A comparison of six different simulated annealing cooling schedules, for solving the school timetabling problem, by Abramson, Amoorthy and Dang (1999) showed that incorporating a way of calculating the temperature that is used as the reheating point produced better results. The basic geometric cooling schedule, a scheme using two cooling rates and four schemes allowing reheating as well as cooling were examined. The third and fourth reheating schedules incorporated computing the temperature used as the reheating point. Extensive testing showed that the fourth reheating schedule produced higher quality and quicker results.

A different approach asserts that many scheduling and timetabling problems can be formulated as Constraint Satisfaction Problems (CSP's) (Brailsford, Potts and Smith 1999). They defined and examined basic techniques for solving CSP's, then compared them with other approaches such as integer programming, branch and bound and simulated annealing. As opposed to more developed OR methods the authors viewed CP as being a relatively new area, however they still see that there are problems for which it is competitive. With further improvements to CP methodology being anticipated, it is considered that awareness of CP as a technique for tackling combinatorial optimization problems is important for researchers in this area.

It is suggested that the use of genetic algorithms for solving timetable planning problems is problematic due to the ambiguity in deciding the fitness function (Deris, Omatu, Ohta & Saada, 1999). Their view is that most solutions to constraint-satisfaction problems using genetic algorithms are problem-dependent and therefore difficult to apply to real-world problems. Deris et al. (1999) suggested using a hybrid algorithm consisting of a genetic algorithm and constraint-based reasoning. They tested the proposed algorithm using real data for university timetable planning. Unlike other techniques, this approach did not require modifications to the components of the genetic algorithm procedures and is therefore generic and problem independent. Consequently, the algorithm can be adapted to different kinds of problems with minimum changes and applied to most constraint-satisfaction problems.

Following on from their previous work, Deris, Omatu and Ohta (2000) viewed the timetabling problem as being combinatorial and dynamic, and thus proposed that any approach used must be efficient, flexible, portable and adaptable. The solution procedure used was based on a constraint based reasoning technique and implemented
in an object-oriented approach. It was argued that since the object-oriented approach was implemented the system proposed could be easily modified and adapted to support changes. It was suggested that the most feasible and best timetable for the college was found using the proposed algorithm.

Another perspective on examination timetabling is gained through using multi-criteria decision problems techniques. A variant of this approach was developed (Burke, Bykov and Petrovic 2000) and tested using real data from a number of universities. A number of criteria were examined such as room capacities, proximity of the exams for the students and the order and locations of events. A two phase approach was then used. The first phase looked at finding a solution examining each criterion separately. In the second phase trade-offs between criteria values were made, trying to find a compromise solution addressing all criteria. This approach attempted to focus in on the ideal point with respect to a certain defined distance measure, however still recognizing that a perfect solution was not possible.

Tabu search solutions are then looked at, bringing in several features from graph coloring research (Di Gasperol and Schaerf, 2000). They discuss a family of solution algorithms for a set of variants of the examination timetabling problem, which are then tested on both public benchmarks and random instances, and compared with historical results. The results show that the presented algorithms performed as well as constructive methods and mimetic algorithms however they were often outperformed by a decomposition based approach, providing the possibility of faster more efficient solution generation.

Adopting a different approach Foulds and Johnson (2000), had implemented a user friendly, menu-driven decision support system (DSS) that can be utilised for the construction of university timetables. The system was tested successfully in a university environment and provided a useful edition to the university timetabler’s tool kit. This was an interesting application of the DSS approach in an important area.

A multi-objective genetic algorithm was proposed for a class/teacher timetabling problem, incorporating two distinct objectives, the minimization of the violations of both types of constraints, hard and soft, while respecting the two competing aspects - teachers and classes (Carrasco and Pato, 2000). The characteristics of a genetic multi-objective meta-heuristic and a non-dominated sorting genetic algorithm were discussed, using a standard fitness-sharing scheme improved with an elitist secondary population and favourable results were obtained through an application of the algorithm to a real example.
Using a case based reasoning approach, previous solutions can be incorporated to help find new solutions for new problems when solving academic timetabling problems (Burke, MacCarthy, Petrovic and Qu (2000). They present Attribute graphs as a suitable method with the case base being organised as a decision tree. Their experiment verified the effectiveness of the proposed method.

A two-phase genetic algorithm (TGA) was used to examine university timetabling (Ueda, Ouchi, Takahashi, and Miyahara, 2000). Two kinds of populations, class scheduling and room allocation were looked at, developed independently, and a cost value for each person assigned. Instances where the lowest costs occur were selected from each population, and these subjects were combined to calculate the fitness values. To assess its performance, TGA was applied to several problems and outcomes compared with those obtained by the simple GA (SGA). In the case that was used it was found that TGA obtained a better solution than the simple GA when the room utilization ratio was high. Therefore in some instances TGA can find a solution that satisfies all constraints where SGA cannot find a feasible solution.

Boronico (2000) presented a model (applied at Monomuth's University School of Business) that was used in scheduling undergraduate courses. It was found that important priorities should be examined include minimizing expected student course conflicts, constraining the use of adjunct faculty, and adhering to ex ante declared recommendations in the assignment of faculty to course sections. The model presented relied on a hierarchical mathematical integer programming model in conjunction with discrete event simulation.

It was found that using combinations of memory types, when looking at examination scheduling using a tabu search algorithm could improve the quality of the resulting solutions (White and Xie 2000). Practical application of a system (OTTABU using recency-based short-term memory and move (or frequency)-based longer-term memory) found that using longer-term memory produced better schedules than those produced without such memory. Long term memory list length was also found to be important. It was found better to have a list that was too long, which only marginally effected performance, than a list that was too short which had a greater negative impact. To estimate the appropriate length of the longer-term tabu list a quantitative analysis method was applied and to improve effectiveness a controlled tabu relaxation technique was also used.

Timetabling problems was examined by Nonobe and Ibaraki (2001) using a tabu search algorithm of weighted constraint satisfaction problem (WCSP). This example
incorporated an evaluation function, defined in terms of the modified weights of constraints, for guiding the search, and it incorporates an automatic control mechanism of the weights in the evaluation function. The control mechanism of weights made the tabu search more powerful.

Typical examination timetabling problems, unlike random graphs, will often have many alternative maximum cliques, and even larger dense subsets of nodes that are almost cliques. To extend the scope of the clique initialisation so that a larger subset of examinations can be included, Carter and Johnson (2001) suggested a number of different methods that considered sub-maximum cliques and/or quasi-cliques. These methods avoided arbitrarily identifying a single maximum clique, providing for several possible alternatives that could lead to an optimal solution.

A specific timetabling and examination scheduling problem is examined that includes specific difficulties including the restricted availability of classrooms and the increased flexibility of students' choices of courses (Dimopoulou & Miliotis, 2001). The IP model attempted to improve schedule quality by constructing groups of courses that are assigned to groups of time periods to handle the relative position of the courses assigned to the available time periods. The solution also attempts to exploit the user's experience and knowledge of the problem. An initial solution is constructed then a heuristic algorithm is generated and refined until a good feasible solution is reached. The system is designed to be flexible, allowing the easy construction and testing of alternative schedules which are pre-conditioned according to requirements specified by the user.

Answers to a combinatorial optimization problem (maximising contact among student cohorts over time) are proposed by Baker, Magazine and Polak (2002). Integer programming and Constraint Programming models are proposed with the Constraint Programming model reducing the solution equivalence class sizes, making the search for an optimum solution easier.

Alvarez-Valdes, Crespo & Tamarit (2002) used a set of heuristic algorithms in a program for solving course timetabling related problems. A Tabu Search procedure had several strategies developed for it and tested, leading to a potent and quick algorithm which produced satisfactory results.
Discussion and Future Trends

This concludes the survey of the literature to date. It has been shown that there have been significant trends and movements away from the highly mathematical analyses of the 60’s, towards the development and application of new AI techniques such as integrated hybrid approaches. The application of these hybrid approaches lead to either, solving significant but focussed sub-problems (eg. the examination scheduling problem) or in providing automated tools (which are assisted by manual intervention) which assist in finding acceptable quality solutions (the maturing of the intelligent scheduling approach).

From the analysis presented above it would appear that the current decade will see a move away from the theoretical basis of the CSP approach to the practical applications that emerge from integrating this latest area into the maturing hybrid intelligent scheduling methodology. The following section summarises the trends in the literature by examining solution technique and graphing this against time. The work presented in the chapters that follow this is my contribution towards this new area of emerging research.

Analysing The Literature

The preceding section presented a detailed examination of the timetable literature. This literature comprised 260 refereed journal papers or refereed conference publications. The graphs and statistics presented below are derived from this body of reported research activity.

Over time the size of this literature has increased as a consequence of a number of factors. The graph in Figure 1 depicts the increase in research into the timetable problem in five year time intervals. This graph, when analysed statistically, yields a positive growth rate that is significantly different from zero at a 99% confidence interval. From the analysis presented above it is reasonable to expect that this growth rate should continue with the new areas of research methodology that are emerging.

Concerning Figure 1, it can be noted that the period between 1991-1995 shows a noticeable increase in timetabling literature. This is easily attributed to the emergence of both the new AI techniques developed around 1990 and the phenomenal increase in computing power available on the desktop to researchers. The period 1996 – 2001 also
represents a period in which some not insignificant theoretical developments were being made. It is reasonable to therefore expect another five year “boom” period in publications for the period 2001 - 2005 as the results of applying these new techniques to practical problems emerge.

![Figure 1: Timetable Publications as a Function of Time.](image)

But how can such a prediction be argued? The points to argue in favour of this assertion are because the timetable problem remains of current and topical interest for the following factors:

1. Being a member of the intractable problem set, the NP Complete Problem set, it is reasonably well accepted that it defies general solution and thus remains of continuing research interest.

2. It is a well known practical problem that the business world must deal with on a day-to-day basis costing commerce significantly money, time and resources to generate and maintain and consequently realises a significant source research funding.

3. As was highlighted in the previous section most research articles claim to have successfully solved the problem and yet despite this optimism, there is no widespread use of any solution implementation resulting in a form of academic pride in the need to find such a solution algorithm.

In this section is presented a classification of the timetable literature by solution generation technique. This classification is based upon a detailed analysis presented in the previous section. The general discussion is then followed by an overall summary, in the form of figures and graphs, of the current state of the literature. It is noted that some
of the papers in the literature, whilst employing algorithms, make a contribution by way of design and development of an Information System rather than the implementation of the algorithm itself.

A Proposed Classification Schema

In this section the literature is classified according to the initial solution generator technique used and the improvement algorithm employed to arrive at a final solution. The papers used for this analysis were limited to those referenced in the thesis. The miscellaneous category includes fifteen papers that did not present a solution generator (reported research that concentrated on a Timetable IS). Figure 2 shows the literature classified according to the solution technique used in obtaining the initial solution. The graph illustrates that the more dominant solution technique is a constraint based artificial intelligence approach.

![Bar Chart]

**Figure 2 : Publications By Solution Generation Technique**

It is noted that the use of the highly mathematical operations research approach (integer and linear programming) has tended to wane in recent years. Interestingly enough there has been some usage of stochastic generation (included as miscellaneous) of initial timetables in the literature, which has persisted over time. Typically the use of a stochastic initial solution set relates with a strong dependence on arriving at a final solution by iterative improvement.

Figure 3 presents a classification of the literature by the improvement algorithm employed in each paper. The significant factor by far is the rather large reported use of Genetic, Evolutionary Algorithms and Constraint Based approaches.
Indeed from this figure the most outstanding point to note is how the literature surrounding the use of improvement techniques has grown exponentially relative to discussions based on initial solution generation. The most popular of these improvement techniques is the use of constraint approaches followed by Evolutionary and Genetic Algorithms. The literature, examined in this thesis, shows a marked increase in the usage of Constraint Based approaches as an improvement algorithm. It has been suggested by Bardadym (1996), that this represents the new wave of timetable solution generators.

**Summary**

Overall the literature suggests the use of heuristics to constrain the search for an initial timetable solution/s followed by the use of a Genetic Algorithm as an improvement technique to arrive at a final near optimal solution. Obviously, as argued from the existing literature, an exact recipe for the optimal timetable to any general timetable problem does not currently exist.

In conclusion it can be seen that a very large and significant literature has evolved into the timetable problem. The nature of the fundamental problem set to which timetable solution generation belongs has resulted in the development of new techniques. This has been a symbiotic relationship in that some techniques have been applied to timetable generation after being developed for the more general problem (eg, graph colouring and CSP) whilst other techniques developed for timetable problems have been applied back to the NPC problem set (eg, Simulated Annealing and Tabu Search).
It is concluded from the analysis of the literature that the emerging use of hybrid solution generators, normally produced from relatively stochastic initial positions, would be iterated upon by an appropriate iterative improvement scheme to arrive at a high quality, though not necessarily optimal, solution. There is also a strong emerging trend in the reporting of University timetabling information systems. The systems reported make little or no attempt at generating timetable solutions. Rather, they are aimed at providing an IS “solution” to the administrative problem of managing the life cycle of a Timetable.

Hence, as detailed in the literature, there is a clear distinction between the technical aspects of timetable generation and management aspects of university timetable administration. In an attempt to bridge this gap it is proposed that a system that encompasses both perspectives of timetable generation and administration be developed. There has been no successful implementation of any system that integrates these activities.

In the following chapters is presented such a system. Its novelty is in its ability to be integrated within the framework of any existing university information systems. That is, the system’s database scheme was normalised to 3\textsuperscript{rd} Normal Form for complete coverage of a ‘typical’ university’s data requirements (the tables are not decomposed any further than 3NF due to the linking nuances of the Object Orientated application tool used).
CHAPTER 3
A TIMETABLE INFORMATION SYSTEM

This chapter proposes and develops an information system to support the production and management of timetables in a university setting. The work presented here attempts to integrate the work of implementation of a solution generation algorithm with the very real problem of administration and management of the resulting timetable solution. As noted in the previous chapter, there have been many reported attempts at developing timetable information systems. All of these systems, however, have begun with an analysis of the information needs of the university administration of timetables.

The chapter begins with a brief discussion of the background literature related to the development of a Timetabling IS. The algorithm used in this work to generate solutions is then presented. Especial attention is paid to the data that are required for the generation procedure.

Following this, the development of the Timetable Information System is discussed. It is noted that the IS presented here begins with an analysis of the data entities required to support the generation and maintenance of a university timetable. This is then expanded to accommodate the other administrative activities associated with the management of student enrolment and assessment data. The SDLC (Systems Development Life Cycle) was used in the construction of this system.

The analysis and design phase is developed from the timetable generation perspective. This is noted to be contra to the usual life cycle of such a system which is normally derived from a platform in which a supporting information system has already been developed and deployed (Senn, 1989). It is suggested that this contra-normal view of the system should more fully support the integration of the timetable support role during deployment.

In order to ascertain the requirements of the proposed IS system, as determined from the second objective, the SDLC method was deployed. An initial study was conducted, at the Faculty of Business at Griffith University, with administrators involved in the timetabling function to gauge the need and use of such a system. The results indicated that a system that provided all student administrative functions and at the same time integrated the timetabling process would be very useful. As a result from the information gained from the Literature review and the results of the initial staff study a
A proposal for a prototype of the system was agreed upon. The next step in the SDLC required a design of the system. An analysis of the requirement was carried out from a generation perspective. In doing this analysis entities required and their relationships to each other were identified and pictorial representation was made using Entity Relationship Diagram (ERD). From this ERD’s the database requirements were designed to determine the attributes of each data table in the database. Having established the database requirements for timetable generation the other administrative aspects of the IS system were integrated into the database schema. A detailed illustration of the database is discussed in this chapter. Once the database requirements were satisfactorily ascertained the next step in the SDLC, that of development, was carried out. The development phase requirements consisted of developing a timetabling Information System that assisted in generating manual timetables as well as having an automated generator that generates timetables. The automated generator uses a dynamic slot table algorithm (DSTA), which is described in detail later in this chapter. In conjunction with timetable generation the system also required the administration and management of all university functions pertaining to students to be included. A prototype of the system was developed and is illustrated and discussed. The system was tested for functionality and useability and the implementation phase consisted of timetable generation, both manual and automated, and the administration of student information.

The results generated from the system are then presented and discussed. Of special interest is the need to integrate the use of any timetable solutions produced in the usual maintenance operations of an information system. In particular, the ongoing need to be able to refine any “final” timetable is catered for in the system specifications.

The ability to generate automated solutions from a variety of initial starting points is accommodated. For instance, start from last years timetable. Or one may wish to “fix” the position of some of the larger lectures, and so on. The system considers this in its design. In particular, it is noted that this feature allows for the intervention in the automated solution generation process by a timetable “expert” or evaluation team.

In the next section is presented an overview of the analysis and design of the system. The detail provided covers almost all the work performed. Nonetheless to keep the size of the chapter manageable some of the more pedestrian work, not directly affecting the research work, has been omitted.
Following this is a brief discussion of the implementation including some screen dumps of the final system. The chapter concludes with a brief discussion of the effectiveness of the final system.

**Background**

The literature includes many examples of timetable based information systems (see, eg., Schmidt et al 1973; or Schaef, 1999). The common environment that these systems are developed in, is one in which there is already some form of a student information system (SIS). Thus the reported system is an attachment, and support only, to this overriding SIS. Obviously, this is due to the nature of a university: initially there are few students in an institution and timetabling is able to be well managed manually. As enrolments increase, the first need is to automate the administrative functions related to processing student enrolments and progression through their students. Finally, a situation is reached where the student population to faculty resources reaches a critical ratio, and timetabling is no longer a trivial task. Consequently (see Hulskamp, 1976; Ulph 1977; Foulds and Johnson, 2000) attempts at developing even a timetable support system, are driven by the data and capabilities of the SIS. Such system developments are usually ad hoc and involve prototyping and evolution life cycles (Johnson, 2000).

The system developed here does not follow these cycles. Rather this is the first system reported to date which has been designed and developed from a timetable solution generation perspective as opposed to a response to an ad hoc generated need. Given the above, the system was developed using the standard SDLC as given by the 5 linear cycles that are espoused in textbooks (see, for eg., Senn 1989; Wu and Wu 1994; Whitten, Bentley and Dittman, 2001). Consequently the database schema has been developed to support both student administration and facilitate both manual and automated timetable automation needs.
Figure 4: Holistic Architecture of University Timetabling Information System

Figure 4 above illustrates a holistic view of the proposed information system architecture. It shows the inputs, processes and outputs of the system. Figure 5 below is a more detailed illustration of this system. The student information system database is the foundation of the system, the data within is sourced from students, courses & subjects, campuses, buildings & rooms, and staff. This data is essential for providing information on timetable administration as well as used in timetable generation. The other principle input to the database is the existing timetable data. This information is essential in providing seeding points so that the generation process can be adapted incrementally. As can be seen from the figure below the Data Maintenance module maintains the data and this updated data is utilised by the Administration and Timetable Generation modules to process the required output.
Once the student information system database is enriched with classified data, it allows efficient administration of the university timetabling function as well as generates timetable views. To further enhance the administration function, these timetable views allow the information to be queried from various perspectives namely: rooms, student, staff and courses. These views generate manual and automated timetables and provide not only screen viewing but also the ability to print hardcopies. Furthermore an ad hoc report generator, embedded in the application, allows for the creation of management report from both the Maintenance and Administration modules.

The above mentioned information system architecture was designed during the requirement specification process. It entailed the detail view of the system, which comprises of ERD’s, data structures and Data Dictionary. These components of the requirements specification document are illustrated and discussed later in this chapter. Furthermore, the proposed output, derived from the requirement specification was classified into manual as well as automated timetables and management reports. Hence the requirements of the system entail a relational database that allows the logical storage of data with minimal redundancies. This will allow the university timetabling application to maintenance data and provide information to management for university administration. The system should also encapsulate a timetable generator that allows for
both manual and automated timetables. The outputs of the system are the various management and timetabling reports.

The manual timetable generator creates timetables that assist administrators in a decision support system capacity by the provision of informed information. It allows the user to create timetables from the data in the student database system and facilitates manual allocation. The core strategy adopted was developing an application from an IS perspective to encapsulate key administrative functions and enhancement of the overall timetable generation process. With manual timetable generation that supports the core timetabling process, the system with its automated timetable generator, acts as a support facility to the core process. Furthermore the application encapsulates not only the manual and automated timetable generators but also provide an integrated solution to the other administrative functions of the university. Thus the system generates manual timetables and management reports from the data stored in the database and generates automated timetables using this data and a Dynamic Slot Table Algorithm (DSTA). However, the focus of the university timetabling information system is on decision based timetable generation as opposed to algorithm-based generation.

**Timetable Solution Generator.**

The choice of using a DSTA for automated timetabling relates to information obtained from the literature review. From this analysis it was determined that constraint-based reasoning techniques have become prevalent in algorithm usage in timetable generation. Hence the use of a constraint based reasoning method was decided upon. A DSTA was deployed as one of it greatest advantage is its ability to very quickly determine whether a particular problem is over constrained and thus will provide no solutions. The algorithm utilised had been first applied to generating crossword puzzle solutions. As these crossword problems and timetable problems belong to the NP-Complete set of problems, by definition a solution to one could be mapped to a solution to the other (see Itai et al, 1975; Garey, et al 1976). As a result it was decided to use this crossword puzzle algorithm and apply it to generating timetable solutions. It could be argued that as crossword puzzles are mainly concerned with hard constraints it would be inappropriate to apply this algorithm to time tabling, which consists mainly of soft constraints. Although there is a general perception that Timetabling deploys soft
constraints, quite often in practice many of these soft constraints are converted to hard constraints. Moreover this DSTA has also been deployed successfully to a variant of crosswords called Go-Words or Crozzle that deals mainly with soft constraints (Spring, 1993). Furthermore the DSTA is relatively new and has been successfully applied to generating crossword puzzle solutions (Harris, Spring and Forster, 1993). Additionally as the DSTA had not been applied to the timetable problem previously it was decided to utilise this as a novel idea in the area of generating timetable solutions.

This section describes the algorithm developed to generate solutions. The algorithm employed belongs to the CSP class of solution generators. Its implementation is derived from the slot table formulation. In brief, the algorithm is a constraint based approach to timetabling derived from the successful slot table formulation pioneered by Smith and Steen (1981). Here we consider a timetable as a teaching slot table with a list of potential room-time slot allocations available to each entry in the teaching slot table. That is, the variables of the CSP algorithm are the class teaching requirements and the domain of each of these variables is a list of the possible room-timetable slots that each class could be assigned. The Boolean constraints that need to be satisfied are as follows:

- Room capacity
- Room usage
- Equipment availability
- Room is currently empty
- Participant availability

It is the dynamic nature of the last 2 constraints that makes solution generation a very hard task.

**The Algorithm for Automated Solution Generation**

A deterministic constraint based technique has been deployed to develop the algorithm used to generate solutions for this particular university timetable problem. The algorithm employs a recursion technique that facilitates a faster and more efficient traversal of the search space.

The algorithm itself is a an innovative application of the dynamic slot table algorithm (Harris, Spring and Forster, 1993) that was used to solve the crossword puzzle problem. The algorithm traverses the search space via recursion with backtracking upon constraint failure.
The present algorithm uses recursion to traverse the search space. Backtracking occurs upon either failure to fill a slot (assign a value to a variable from its domain) or a successfully completed timetable is generated. Recursion is used to move through a slot table defined as a list of all the classes that must be taught. At each entry in this slot table in turn, the algorithm attempts to assign a room-time tuple from the domain of that slot variable. If an assignment is possible recursion proceeds otherwise backtracking is invoked. Upon backtracking to a higher level slot, the class is assigned the next tuple in its domain and recursion invoked again. The algorithm terminates when the entire search space has been traversed. The algorithm proceeds towards finality in a number of steps that are described below.

Step 1 - Initialisation
Initially, each Domain is created from the room list by matching subject classes and their constraints. Hence each element of the Domain represents a valid instantiation to its variable in the absence of any other variable instantiation.

Each element of a domain consists of a room timetable pair, for example:

\[\text{NT1, 8:00am – 9:00am Monday}\]
\[\text{NT1, 9:00am - 10:00am Monday}\]

which represents a subject class. Obviously an element can occur in one or many domains.

Step 2 – Phase space traversal
The DSTA proceeds by instantiating all Variables with all possible combinations of their dynamically maintained Domains. This procedure is implemented by the use of standard recursion and backtracking procedures.

Recursion is employed on moving from one variable to the next. The choice of which variable to move to be selected is simply an option of the next slot with the smallest domain. In the case of a tie a random selection process is simply applied.

Backtracking is triggered when the domain of the current variables is exhausted so that all elements have, in turn, been instantiated.

Upon recursion into a variable, all elements of its domain are instantiated. After instantiation of an element, into a variable, a number of checks are performed to complete the process. These are as follows:
I. Have all variables been instantiated? If so then a valid solution has been generated. The “score” for this solution is evaluated and recorded if this is the best solution generated so far.

II. All domains are updated

III. Employ recursion again as described previously.

IV. Move to the next element and repeat the above steps until the end of the domain is reached.

V. At the end of the domain list the backtracking procedure as described previously is initiated.

Step 3 – Finalisation and termination of the DSTA

This final step requires the storage of the best solution that has been generated into the database.

One of the objectives in running the DSTA was to produce timetables that holistically had the minimum number of clashes between subjects. Minimising clashes of subjects was included as a weighting in the objective function in order to generate timetables that did not assign students to two or more subjects at the same time and day. However a standard clash matrix was not deployed to do this, as clashes would then have to be set as hard constraints. The need to leave student clashes in the objective function and not code them as hard constraints is due to the very nature of university timetables. Enrolment details of a student are typically not finalised prior to the timetable generation procedure. Consequently a system was derived whereby solutions could be weighted, without actual enrolment data, so that the probability of a clash could be reduced. In order to do this, subject enrolments (reasonably firm) are taken and students are randomly assigned to tutorials. A high penalty is invoked, in the objective function, if student clashes occur in lectures but a lower penalty incurs for clashes in just tutorials or tutorials and lectures combined. However as pointed out by Tripathy (1987) the use of a clash matrix can improve the quality and construction of the timetable. Hence it is intended to include a standard clash matrix in future improvements to this work.

The current algorithm used does produce timetables that can be utilised. However when the requirement is to traverse a large search space it can be proven to be inefficient. Thus it is proposed that in future work this algorithm could be improved by including a Branch and Bound technique in the process of traversal. The utilisation of this technique ensures that incomplete but obviously low value timetables are not
completely generated before being discarded. Observations from previous work carried out in this area have shown a tendency for the DST to achieve this (Harris et al 1993). The usage of this technique on the crossword variant Go-Word problem resulted in an increase in the speed of solution generation and a overall efficacy in the generation process (Tybon & Harris, 1999).

Implementation

The system is being prototyped in Delphi 6.01 on a Pentium 4 workstation running Windows 2000. The database was implemented in the Flash Filer DBMS and a large number of data views encoded in Delphi to allow for standard data maintenance functions to be undertaken.

The database was populated with data taken from the Nathan Campus of Griffith University for semester 2 in 2002. The system has already demonstrated its potential by enabling timetable data and information to be easily entered. A limited run of the automated solution generator was successfully integrated with the room view of the timetable management system described later.

The essential details of the implementation are illustrated in the following pseudo-code:

```
Insert_word_into_slot(slot_table_row_nr)

    Save global vars onto the stack for backtracking
    For each element in the domain
        Instantiate element into variable of row nr
        Update slot table, domain elements and global variables
        Perform a 1-level lookahead
        If not fails lookahead then
            If not at bottom of slot table then
                Insert_word_into_slot(slot_table_row_nr + 1)
            Else save timetable solution

            Clear element from variable of row nr
            Reset slot table domain and global variables

    The implementation continues until normal termination, which occurs when the algorithm tries to backtrack above the first row in the slot table.
```

Timetable Management Information System

The procedures for analysis, design and implementation of a computerised information system are well known. The systems development life cycle is considered to be the set
of activities that analysts, designers, and users carry out to develop and implement an information system. In most situations all activities are closely related and often inseparable. Even the order of the steps in these activities may be difficult to determine. Different parts of a project can be in different phases at any one point in time (Senn, 1989).

The systems development life cycle (SDLC) method consists of various phases. The first phase is a preliminary investigation which involves tasks like defining the problem, estimating time and cost requirements, and a feasibility study of the project. The major output of this process is the feasibility report (Wu and Wu, 1994).

Secondary tasks initiate after the acceptance of the feasibility report and this stage involves determination and analysis of the system requirements. Modelling the proposed new system is done during this stage. The output of this process is a proposal for the new system.

The design stage follows this analysis procedure. This is an important stage as it involves tasks like determining and acquiring the hardware and software required, designing the data files, designing the programs to be used, preparing training guidelines, and preparing preliminary testing procedures. The output of the design process is a design specification.

Taking the design specification as a base, the programmers develop or code the individual programs which are to be used within the system. A meticulous testing of all programs used in the system is then performed. “Bugs” located within the system are remedied. (Senn, 1989).

Once the programs have been debugged, the system is implemented at the client’s end. The post implementation phase involves the review of the development process immediately upon completion and also every three to six months. At the end of every evaluation a system evaluation report is generated.

Generally, SDLC methodology helps designers and developers to avoid any flaws in the final system. The techniques used during the analysis and design phases facilitate smooth production of working programs that perform the appropriate functions for the information system. However there are three major disadvantages with SDLC methodology. Firstly, SDLC requires more time to develop and integrate an information system. Secondly, the costs of developing a system using the SDLC
approach are very high. Finally, the use of the SDLC approach demands heavy documentation of the various processes used. (Whitten, Bentley and Dittman, 2001).

Applying this methodology to the timetable and student information system is justified on the grounds of the well-documented features of the system that should be modelled using the SDLC approach. It is important to have a clearly defined problem as this will facilitate and save development time in relation to gathering accurate information from the client (Wu and Wu, 1994).

The features of the timetable and student information system includes systematic inputs and outputs in a well-defined, well-structured system that produces all necessary reports, and the implementation of the system can be done on completion and not in a phase-in method. These features collectively enable application of the SDLC methodology (Senn, 1989).

**Problem Definition**

An information system is required to support the activities of student enrolment, administration and timetable generation, and maintenance in an integrated fashion. The system should support all aspects of student administration, including enrolments, grading, and maintenance of personal details. The database for this part of the system must integrate in a seamless fashion with the timetable functions. The timetable module must support all activities, including timetable generation, maintenance and administration. In particular, the timetable generation component must support all forms of timetable generation from manual assignment through to complete automated generation.

**System Analysis and Design**

Another option in choosing the process for the development of the timetable generation application is to consider some of the modern evolutionary methodologies. One of the more favoured approaches is Rapid Application Development (RAD).

The RAD methodology was readily adopted by software developers who are currently using this approach to build better, higher-quality applications, in faster time spans (Burger, K. 1995). RAD is an iterative approach to software development that delivers
working pieces of an application incrementally, while continuously incorporating feedback from the end-users (Hanna, M. 1995).

The RAD method consists of five distinct phases: modelling, prototyping, optimisation, integration and deployment. During the modelling phase, an enterprise model is created and developers can automate the modelling process using upper-CASE tools to generate entity-relationship diagrams, or logical data models, in a format that can be read by prototype builders, Structured Query Language, or a flat file. At this stage the use of a prototype builder can significantly reduce the coding effort as an application prototype can be built automatically which will include default user interfaces, reports and logic modules, and a menu system that controls all of the tables, attributes, and forms.

The major advantage of using this methodology is that users can begin experimenting with an application prototype immediately and feedback from the users can provide valuable insights into the proposed system. Upon completion of the prototype, configuration management tools are then used to help complete the application so that it can be readied for production. Once the system has been fully configured, the application is then optimised for a specific physical environment, making it ready for integration with other applications and deployment (Jacques, T. 1992).

In another article Jacques, T. (1992) suggested that the 1990’s would be a decade of integrated tool sets and rapid application development (RAD) methods. The RAD approach combines the effective use of tools within an integrated development environment and one of its major advantages is that the applications developed by using this methodology can be reused as templates for other developments. With the current interest and advancements in utilising visual application development tools, it is imperative that methods used for system development integrate many modelling perspectives within and across all life-cycle phases and also include the usage of applications over multiple domains (Norman et. al., 1992). The RAD methodology covers all aspects of this requirements and is an ideal platform for use in developing applications using the latest programming tools available.

Delphi, being a popular tool utilising a Graphic User Interface (GUI), allows prototyping as one of its main advantages in the development process. As a result RAD is ideally suited as the methodology to be used in conjunction with Delphi.

Delphi was deployed as the prototype builder and used to interact with a selected number of end users. An initial model was used to allow end users to experiment with.
The final outcome of this was a user friendly design that could be developed into a full system.

**Entity-Relationship Modelling**

During the design of the database for the timetabling and student information system there were various entities identified. These entities were further classified into major and related entities. The major entities were identified as the primary source of data, which in turn post relation setting with related entities generated a complete and robust database design for the system.

This taxonomy of entities comprised of *Students, Subjects, Rooms, Subject Class Timetables, Departments, Courses, Staff, Teaching Contact Type, Faculties, Campus, Room Availability and Teaching Position*. The tables below give a brief description of each of the above mentioned major entities and the entities related to them, which form the database structure of the Timetable and Student Information System.

<table>
<thead>
<tr>
<th>Major Entities</th>
<th>Description</th>
<th>Related Entities</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Campus</td>
<td>It records the various campuses of the university.</td>
<td>Buildings</td>
<td>It records all the building within a campus.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rooms</td>
<td>It stores information of all the rooms available within each campus and respective buildings within them.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Subjects</td>
<td>It stores data for all the subjects offered under a course which is offered by a department within a campus.</td>
</tr>
</tbody>
</table>

Table 1: Data Structure – Campus Table
<table>
<thead>
<tr>
<th>Major Entities</th>
<th>Description</th>
<th>Related Entities</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rooms</td>
<td>It stores information of all the rooms available within each campus and respective buildings within them.</td>
<td>Room Timetables</td>
<td>This entity stores data with respects to the weekly room schedules.</td>
</tr>
<tr>
<td>Subject Class Timetables</td>
<td>This entity stores data in context to the subject offered and the room where it will be conducted and the type of class in terms of the teaching contact and other related information.</td>
<td>Staff Information regarding the department with whom the staff is affiliated and the room allocated to them is stored by this entity. It also stores the title of the staff member.</td>
<td></td>
</tr>
<tr>
<td>Room Teaching Usage</td>
<td>This entity stores the purpose of the room.</td>
<td>Room Availability</td>
<td>This entity stores information regarding the various rooms and their availability status.</td>
</tr>
</tbody>
</table>

**Table 2 : Data Structure – Rooms Table**
<table>
<thead>
<tr>
<th>Major Entities</th>
<th>Description</th>
<th>Related Entities</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subjects</td>
<td>It stores data for all the subjects offered under a course which is offered by a department within a campus.</td>
<td>Subject Assessment Items</td>
<td>This entity stores details of the various subjects assessment within a subject offered.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Subject Assessment Student Marks</td>
<td>It stores data with regards to marks obtained by student in the various assessment for a subject.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Student Subject Grade</td>
<td>It store the grade scored by a student in each respective subject.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Student Subject Enrolment</td>
<td>It records the details of the various subjects the student is enrolled in and the other details such as year, semester etc.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Subject Teaching Teams</td>
<td>It stores information of the staff and the position of the staff member conducting the subject.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Student Subject Class Preference</td>
<td>It stores information with regards to any preferences of the students for a subject and details of the preference.</td>
</tr>
</tbody>
</table>

Table 3 : Data Structure – Subjects Table
<table>
<thead>
<tr>
<th>Major Entities</th>
<th>Description</th>
<th>Related Entities</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Students</td>
<td>This entity stores the personal data of the student. In addition, it stores information of the title of the student and the course the student is enrolled with.</td>
<td>Subject Student Grade</td>
<td>It stores the grade scored by a student in respective subject.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Subject Class Student Enrolment</td>
<td>It keeps track on the student enrolled for a subject, which in turn is related to the Subject Class Timetable entity.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Subject Assessment Student Marks</td>
<td>It stores data with regards to marks obtained by the student in the various assessment criteria for a subject.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Student Subject Enrolments</td>
<td>It records the details of the various subjects the student is enrolled and the other details like year, semester etc.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Student Subject Class Preference</td>
<td>It stores information with regards to any preferences of the students for a subject and the details of the preference.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Course Student Enrolments</td>
<td>It stores data with respects to the student and courses they are enrolled with.</td>
</tr>
</tbody>
</table>

Table 4: Data Structure – Students Table
<table>
<thead>
<tr>
<th>Major Entities</th>
<th>Description</th>
<th>Related Entities</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Department</strong></td>
<td>It stores information with regards to the faculty in which the department is and the details of the administrative staff of the department.</td>
<td>Course Teaching Department</td>
<td>This entity keeps a log of the departments offering various courses.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Staff</td>
<td>Information regarding the department with whom the staff is affiliated and the room allocated to them is stored by this entity. It also stores the title of the staff member.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Subjects</td>
<td>It stores data for all the subjects offered under a course which is offered by a department within a campus.</td>
</tr>
<tr>
<td><strong>Course</strong></td>
<td>This entity stores information of the faculty who offers the course and the details of the course and related information.</td>
<td>Course Student Enrolment</td>
<td>It stores data with respects to the student and courses enrolled with.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Course Teaching Department</td>
<td>This entity keeps a log of the departments offering various courses.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Students</td>
<td>This entity stores the personal data of the student. In addition, it stores information of the title of the student and the course the student is enrolled within.</td>
</tr>
<tr>
<td><strong>Staff</strong></td>
<td>Information regarding the department with whom the staff is affiliated and the room allocated to them is stored by this entity. It also stores the title of the staff member.</td>
<td>Student Teaching Teams</td>
<td>It stores information of the staff member and the position of the staff member conducting the subject.</td>
</tr>
</tbody>
</table>

Table 5: Data Structure – Department, Course and Staff
<table>
<thead>
<tr>
<th>Major Entities</th>
<th>Description</th>
<th>Related Entities</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject Class</td>
<td>This entity stores data in context to the subject offered and the room where</td>
<td>Subject Class Enrolments</td>
<td>It keeps track on the student enrolled for a subject, which in turn is related to the Subject Class Timetable entity.</td>
</tr>
<tr>
<td>Timetables</td>
<td>it will conducted. It also stores the type of class in terms of the teaching</td>
<td>Subject Class Clashes</td>
<td>This entity keeps a record on the clashes of the subject timetables for future analysis.</td>
</tr>
<tr>
<td></td>
<td>contact and other related information.</td>
<td>Subject Class</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible Rooms Time lists</td>
<td>This entity keeps a record on the various subject classes available and the related information of the times it is available.</td>
</tr>
<tr>
<td>Teaching Contact</td>
<td>This entity stores basic information of the teaching contact type.</td>
<td>Room Teaching Usage</td>
<td>This entity stores the purpose of the usage of the room.</td>
</tr>
<tr>
<td>Types</td>
<td></td>
<td>Subject Class Timetables</td>
<td>This entity stores data in context to the subject offered and the room where it will conducted and the type of class in terms of the teaching</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>contact and other related information.</td>
</tr>
<tr>
<td>Faculties</td>
<td>This entity lists the faculty information within the university.</td>
<td>Courses</td>
<td>This entity stores information of the faculty who offer the course and the details of the course and related information.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Departments</td>
<td>It stores information with regards to the faculty in which the department is and the details of the administrative staff of the department.</td>
</tr>
</tbody>
</table>

Table 6: Data Structure - Subject Class Timetables, Teaching Contact Types and Faculties
<table>
<thead>
<tr>
<th>Major Entities</th>
<th>Description</th>
<th>Related Entities</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Titles</td>
<td>This entity has the titles related to the names of the students or staff members.</td>
<td>Student</td>
<td>This entity stores personal data of the student. In addition, it stores information of the title of the student and the course the student is enrolled within.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Staff</td>
<td>Information regarding the department with whom the staff is affiliated and the room allocated to them is stored by this entity. It also stores the title of the staff member.</td>
</tr>
<tr>
<td>Room Availability</td>
<td>This entity stores information with regards to the various rooms and their availability status.</td>
<td>Subject Class Possible Room Time Lists</td>
<td>This entity keeps a track on the various subject classes available and the related information of the times it is available.</td>
</tr>
<tr>
<td>Teaching Position</td>
<td>This entity stores data with respects to the various position of the teaching staff.</td>
<td>Student Teaching Teams</td>
<td>It stores information of the staff member and the position of the staff member conducting the subject.</td>
</tr>
</tbody>
</table>

Table 7: Titles, Room Availability and Teaching Position

In context to the timetabling problem the major issue has been with respects to identifying various clashes within the generated timetable. This issue within the system is supervised with a unique Artificial Intelligence based algorithm which takes into consideration all the historical data in regards to timetable clashes and generates results considering the percentage of possible clashes. In a similar pattern, the timetable clashing for subjects, rooms and staff is managed.
Subject and Related Entities

In Figure 6 above Subjects is the major entity, which is linked to various entities like the Student Subject Enrolments, Subject Teaching Teams, Student Subject Class Preference, Subject Student Grades, Subject Assessment Student Marks and Subject Assessment Items. Closer evaluation of these entities, their attributes and relationships are provided in the Database Schema Section.

The relationships that the Subject entity shares with almost all the related entities are a one-to-many. This is because in every case subject related information can have more than one record in the related entities.
Students and Related Entities

The Student entity in Figure 7 is an important entity for the timetabling and student information system as it is related to various entities which track and record student information. The related entities enable this information to be used for timetabling purposes. The various related entities involved in this case are Course Student Enrolments, Student Subject Class Preference, Student Subject Enrolments, Subject Assessment Student Marks, Subject Class Student Enrolments and Subject Student Grades.

The Student entity has one-to-many relationships with all the related entities. These relationships imply that there are more than one combinational record which can exist with the same student details resulting into the attribute Student ID becoming a foreign key in all the related entities.
Rooms and Related Entities

The Rooms entity is important in context to allocation of resources for subjects and classes. The allocation of rooms form is an important part of the system as proper care needs to taken to avoid room clashing. Various details have to be maintained to avoid or minimise such issues.

As the representation of the system in Figure 8 above signifies the room entity being the primary entity linked with a one-to-many relationship with Staff as they are allocated rooms; Room Availabilities to keep a track on which rooms are available for various activities; Room Teaching Usage to monitor which rooms are used for which activities; Room Timetabling to schedule the allocation of rooms for various activities; Subject Class Timetable to recognise the room utilised for what type of teaching contact.
Subject Class Timetables and Related Entities

This particular set of entities and relations depicted in Figure 9 above is very critical for the tracking of information for the purpose of managing issues related to room clashing. The major entity being the Subject Class Timetables, which keeps a record of the subject, the type of teaching contact and the room allocated for teaching. With this base information following related entities like Subject Class Possible Room Time Lists the system can generate reports, which for example, list the various rooms available. In addition, entities like the Subject Class Clashes assist the algorithm with relevant data for processing and generating results of the percentage of prospective clashes within the timetable. Other entities like the Subject Class Student Enrolments enables the tracking of students enrolled within various subjects and classes.
Department and Related Entities

Departments within the university system come at the second level in terms of course organisational structure. Various departments come under one faculty and various courses can be offered by one department.

Taking a closer look at above mentioned context, Departments is the key entity storing important details and is related to other entities like Staff indicating that every staff member is a part of one department, but that there can be more than one staff member within the department. Thus, a one-to-many relationship is generated. In addition, the relationship works in a similar manner for Subjects offered by the department and also the various courses offered by the department i.e. Course Teaching Department.
Courses and Related Entities

Entity *Courses* stores information of the various courses offered by the various department (*Course Teaching Departments*) within the university. *Students* enrol in these various courses. The *Course Student Enrolments* entity records the details of the various students enrolled in the various courses offered.

At any given time no more than one *Course Teaching Department* can offer one course and a *Course Teaching Department* can offer no more than one course. In addition, many *Students* can enrol in one course. This indicates that all the related entities have a one-to-many relationship with the major *Course* entity. **FIGURE 11 TO BE MENTIONED**

Staff and Related Entities

*Staff* is the major entity which stores all the information about the various staff members of the university. It has some related entities, like the *Student Teaching Team* entity, which holds data about the teaching teams. The *Title* entity has the list of titles that can
be offered to a student or a staff member. All of these entities have a one-to-many relationship.

![Diagram](image1.png)

**Figure 13: ER Diagram - Teaching Contact Types and Related Entities**

**Teaching Contact Types and Related Entities**

The *Teaching Contact Types* entity comprises of the various types of teaching methods used within the course. This entity has a one-to-many relationship with related entities such as the *Room Teaching Usage* entity which gives the details of the room usage based on Teaching Contact type and the *Subject Class Timetable*, which handles information about subjects, appropriate teaching contact types and other relevant details.

![Diagram](image2.png)

**Figure 14: ER Diagram - Faculties and Related entities**

**Faculties and Related entities**

The *Faculties* entity provides all the details of the faculty and its key members. In addition, the faculty offers various *Courses* and the faculties comprise of various *Departments*, which in turn have various staff members.
Campuses and the Related Entities

Campuses is the first entity in context to the hierarchy. The university has Campuses, each of which has many Buildings and every building has various Rooms. The relationship between Campuses – Buildings and Buildings – Rooms is both of type one-to-many. In addition, Campuses offer various subjects where it also makes a one-to-many relationship.

Room Availabilities and Related Entities

The Room Availabilities entity provides information about lists of rooms available for teaching or other purposes. The Subject Class Possible Room Time Lists provides the list of all the possible rooms for teaching purposes by receiving data from the major entity. They share a one-to-many relationship with each other.
Teaching Position and Related Entities.

The major entity *Teaching Position* has a list of all teaching positions. This teaching positions data is utilised for storing information related to *Student Teaching Teams*. They also share a one-to-many relationship.

The various entity relationship diagrams are generated using Microsoft Visio Modeller.

Database Schema

During system design, the logical data model will be transformed into a physical data model from a chosen database management system called a Database Schema. This model will reflect the technical capabilities and limitations of that database technology, as well as the performance tuning requirements suggested by the database administrator. Database Schema is the physical model or blueprint for a database. It represents the technical implementation of the logical data model (Whitten, Bentley and Dittman, 2001).

Database Schema specifies details based on the capabilities, terminology and constraints of the chosen database management system. The database schema is a little different compared to the ER Diagram. As the ER Diagrams has the basic entity and does not provide any further details with respect to the various attributes and the key fields within the table or entity.

The table below mentions all of the major tables and their related tables with the common identifier column, which is a primary key in the major table and a foreign key in the related table. Using the normalisation method derives these tables and the respective relationships are achieved at the third-normal form level.
<table>
<thead>
<tr>
<th>Major Entities</th>
<th>Relationship Column</th>
<th>Related Entities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Campus</td>
<td>Campus ID</td>
<td>Buildings, Rooms, Subjects</td>
</tr>
<tr>
<td>Rooms</td>
<td>Room ID</td>
<td>Room Timetables, Subject Class Timetables, Staff, Room Teaching Usage, Room Availability</td>
</tr>
<tr>
<td>Subjects</td>
<td>Subject ID</td>
<td>Subject Assessment Items, Subject Assessment Student Marks, Student Subject Grade, Student Subject Enrolment, Subject Teaching Teams, Student Subject Class Preference</td>
</tr>
<tr>
<td>Students</td>
<td>Student ID</td>
<td>Subject Student Grade, Subject Class Student Enrolment, Subject Assessment Student Marks, Student Subject Enrolments, Student Subject Class Preference, Course Student Enrolments</td>
</tr>
<tr>
<td>Department</td>
<td>Department ID</td>
<td>Course Teaching Department, Staff, Subjects</td>
</tr>
<tr>
<td>Course</td>
<td>Course ID</td>
<td>Course Student Enrolment, Course Teaching Department, Students</td>
</tr>
<tr>
<td>Staff</td>
<td>Staff ID</td>
<td>Student Teaching Teams</td>
</tr>
<tr>
<td>Subject Class Timetables</td>
<td>Subject Class ID</td>
<td>Subject Class Enrolments, Subject Class Clashes, Subject Class Possible Room Time Lists</td>
</tr>
<tr>
<td>Teaching Contact Types</td>
<td>Teaching Contact ID</td>
<td>Room Teaching Usage, Subject Class Timetables</td>
</tr>
<tr>
<td>Faculties</td>
<td>Faculty ID</td>
<td>Courses, Departments</td>
</tr>
<tr>
<td>Room Availability</td>
<td>Room Availability ID</td>
<td>Subject Class Possible Room Time Lists</td>
</tr>
<tr>
<td>Teaching Position</td>
<td>Teaching Position ID</td>
<td>Student Teaching Teams</td>
</tr>
<tr>
<td>Title</td>
<td>Title ID</td>
<td>Student, Staff</td>
</tr>
</tbody>
</table>

Table 8: List of Key Identifier Columns
Figure 18: Subject table showing related entities

The key entity within Figure 18 is Subjects, which is linked to various entities like the Student Subject Enrolments, Subject Teaching Teams, Student Subject Class Preference, Subject Student Grades, Subject Assessment Student Marks and Subject Assessment Items. In the subject table the PK notation in the diagram indicates primary key. **Subject ID** is a primary key for the Subject table, which is referenced into all the related entities by place **Subject ID** in related tables or entities with a FK notation indicating foreign key.
The Student table has the Student ID as a primary key, which shares a one to many relationship various related entities like the Course Student Enrolments, Student Subject Class Preference, Student Subject Enrolments, Subject Assessment Student Marks, Subject Class Student Enrolments and Subject Student Grades. The Student ID field is also a part of each of the listed tables as it becomes a foreign key for those related tables.
In Figure 20 Rooms is the key entity, which is related to various other entities like Staff, Subject Class Timetables, Room Timetables, Room Teaching Usage, Room Availabilities. In this diagram the key field which links the major entity to the related entities is the **Room ID**. **Room ID** is a primary key in the Rooms table and all the other related tables are a foreign key.
Figure 21: Subject class timetable and its related entities

The schema represents Subject Class Timetable tables as a major entity and Subject Class Possible Room Time Lists, Subject Class Student Enrolments and Subject Class Clashes are all the related entities or tables. In each of the related Subject Class ID is a foreign key as it creates a one to many relationship with the Subject Class Timetable, being the master table with Subject Class ID as a primary key. **FIGURE 21 TO BE MENTIONED**
Figure 22: Department table and its related entities

Department table in Figure 22 is the primary or the master table, which is in turn linked with all the other tables like the Subject, Course Teaching Department and Staff. There is a clear one-to-many relationship, thus field **Department ID** is a foreign key in all related tables and a primary key in the department table.
Figure 23: Course table and its related entities

Course table in Figure 23 is the key entity. It is linked to various entities like the Course Teaching Department, Student and Course Student Enrolments. In each of the cases Course ID is the foreign key in the related table and a primary key in the course table creating a one-to-many relationship.
Figure 24: Staff table and its related entities

There are two major entities involved in Figure 24. The Staff entity which has the Staff ID as a primary key is related to Student Teaching Teams in a one-to-many relationship pattern. In addition, the title table links itself to the Student Table and the Staff Table, again with a common identifying field called Title ID, which is a primary key in the Title Table and foreign key in respective related tables.
Figure 25: Teaching contact types table and its related entities

In Figure 25 there are two sets of table relations revealed. The first set of tables and relations involves the Teaching Contact Type, which is the master table linked to related tables like the SubjectClass, TimeTables and Room Teaching Usage. The link is generated by a common identifier field call the Teaching Contact ID. The second set of tables and relations involves Faculties and its relationship with Courses and Departments. As Departments are under a Faculty and Courses are offered by a Faculty they become the related entities or table connected to the Department table. This relationship is of a one-to-many type and has a common identifier field called Department ID.
Figure 26: Campus table and its related entities

According to Figure 26 the Campuses table is the key entity. It is related to Subjects and Buildings entities with a common field called Campus ID. This relation is a one-to-many relationship. In turn the Buildings table forms a base information for the Rooms table. As Rooms are in Buildings, rooms becomes a related entity to Buildings and shares a common data field of Building ID to set a one-to-many relationship.
Figure 27: Room availability table and its related entities

In Figure 27 the Room Availability table is related to Subject Class Possible Room Time Lists table where they share a one to many relationship with a Room Time ID being a common field as primary key in the master table and foreign key in the related table.

In a similar manner, the Teaching Position entity is related to Student Teaching Team table. They also share a one-to-many relationship and have a common field Teaching Position ID.

Data Dictionary

A data dictionary is a catalogue – a repository of the elements in a system. In a data dictionary is a listing of the various elements composing of the data flow within the system. The major elements of a data dictionary are data flows, data stores and processes.

Usage of data dictionaries in real time system development is very important. It is important to use data-dictionaries to manage the details of a large system. In addition, it also plays a significant role in communicating a common meaning of all system elements. It enables the developer to document and record the features of the system and also to facilitate analysis of the details in order to evaluate characteristics and
determine where system changes should be made. It is also important to have a data module, which would locate errors and omissions in the system.

As mentioned in the previous paragraph elements like the data store are mentioned. These data stores comprise of various data fields. These fields have relevant data types which form an important part of the database structure. There are various data types which are used within this system development eg. The various data types used for the Timetable and Student Information System are from this complete list of data-types provided by Borland®Delphi. The table below represents all the data-types and size and range of each data-type.

<table>
<thead>
<tr>
<th>Data-Type</th>
<th>Size</th>
<th>Possible Range of Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>ShortInt</td>
<td>1</td>
<td>-128 to 127</td>
</tr>
<tr>
<td>Byte</td>
<td>1</td>
<td>0 to 255</td>
</tr>
<tr>
<td>Char</td>
<td>1</td>
<td>0 to 255 (same as Byte)</td>
</tr>
<tr>
<td>WideChar</td>
<td>2</td>
<td>0 to 65535 (same as Word)</td>
</tr>
<tr>
<td>SmallInt</td>
<td>2</td>
<td>-32768 to 32767</td>
</tr>
<tr>
<td>Word</td>
<td>2</td>
<td>0 to 65535</td>
</tr>
<tr>
<td>LongInt</td>
<td>4</td>
<td>-2147483648 to 2147483647</td>
</tr>
<tr>
<td>Int64</td>
<td>8</td>
<td>-9223372036854775808 to 9223372036854775807</td>
</tr>
<tr>
<td>Integer</td>
<td>4</td>
<td>Same as LongInt</td>
</tr>
<tr>
<td>Cardinal</td>
<td>4</td>
<td>0 to 2147483647</td>
</tr>
<tr>
<td>Single</td>
<td>4</td>
<td>1.5 x 10^-45 to 3.4 x 10^38</td>
</tr>
<tr>
<td>Double</td>
<td>8</td>
<td>5.0 x 10^-324 to 1.7 x 10^308</td>
</tr>
<tr>
<td>Real</td>
<td>8</td>
<td>5.0 x 10^-324 to 1.7 x 10^308</td>
</tr>
<tr>
<td>Extended</td>
<td>10</td>
<td>3.4 x 10^-4932 to 1.1 x 10^4932</td>
</tr>
<tr>
<td>Comp</td>
<td>8</td>
<td>-92233720854775808 to 92233720854775807</td>
</tr>
<tr>
<td>Currency</td>
<td>8</td>
<td>-92233720854775808 to 92233720854775807</td>
</tr>
<tr>
<td>Boolean</td>
<td>1</td>
<td>True Or False</td>
</tr>
<tr>
<td>Variant</td>
<td>16</td>
<td>Varies</td>
</tr>
</tbody>
</table>

Table 9: Table indicating the data-types and other information.

The database created has a set size and the amount of disk space used depends more on the number of records. Delphi enables a multiple implementation pattern. Post development the database can be implemented in a single-user or a multi-user environment. There are no changes to be done when using Delphi as the front end for development. In the case of the database, Flash Filer 7 (being a relational database management system) manages multiple access issues such as concurrency and deadlocks in a very organised and procedural manner.
Data Structures within the Data Dictionary

Shown below are the data structures that represent the database that has been utilised for the timetable application. These data structures are comprehensive and complete within this relational database.

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Buildings. db</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Buildings in University Campuses</td>
</tr>
<tr>
<td>Contents</td>
<td>Building ID</td>
</tr>
<tr>
<td></td>
<td>Building</td>
</tr>
<tr>
<td></td>
<td>Campus ID</td>
</tr>
</tbody>
</table>

Table 10: Buildings data dictionary

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Campuses. db</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>All Campuses of the University</td>
</tr>
<tr>
<td>Contents</td>
<td>Campus ID</td>
</tr>
<tr>
<td></td>
<td>Campus</td>
</tr>
</tbody>
</table>

Table 11: Campuses Data Dictionary

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Course Student Enrolments. db</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Enrolment Details of the Student; Also Indicating the Status of the Enrolment</td>
</tr>
<tr>
<td>Contents</td>
<td>Course ID</td>
</tr>
<tr>
<td></td>
<td>Student ID</td>
</tr>
<tr>
<td></td>
<td>Enrolled</td>
</tr>
<tr>
<td></td>
<td>Graduated</td>
</tr>
<tr>
<td></td>
<td>Specified Credit Points</td>
</tr>
<tr>
<td></td>
<td>Unspecified Credit Points</td>
</tr>
<tr>
<td></td>
<td>Specified Notes</td>
</tr>
<tr>
<td></td>
<td>Unspecified Notes</td>
</tr>
<tr>
<td></td>
<td>File Notes</td>
</tr>
</tbody>
</table>

Table 12: Course Student Enrolments Data Dictionary

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Course Teaching Departments. db</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Indicates Which Course if Conducted by What Department</td>
</tr>
<tr>
<td>Contents</td>
<td>Department ID</td>
</tr>
<tr>
<td>Data Structure</td>
<td>Courses. db</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Indicates the Course Details and to Which Faculty it Forms a Part of.</td>
</tr>
<tr>
<td><strong>Contents</strong></td>
<td>Course ID</td>
</tr>
<tr>
<td></td>
<td>Course</td>
</tr>
<tr>
<td></td>
<td>Short Name</td>
</tr>
<tr>
<td></td>
<td>Faculty ID</td>
</tr>
<tr>
<td></td>
<td>Director ID</td>
</tr>
<tr>
<td></td>
<td>Total Credit Points</td>
</tr>
<tr>
<td></td>
<td>Introduction</td>
</tr>
<tr>
<td></td>
<td>Degree Rules</td>
</tr>
<tr>
<td></td>
<td>Further Information</td>
</tr>
</tbody>
</table>

**Table 13: Course Teaching Department Data Dictionary**

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Departments. db</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Indicates the Department Details and the Faculty it is Affiliated to and What Campus it is in.</td>
</tr>
<tr>
<td><strong>Contents</strong></td>
<td>Department ID</td>
</tr>
<tr>
<td></td>
<td>Department</td>
</tr>
<tr>
<td></td>
<td>Department Code</td>
</tr>
<tr>
<td></td>
<td>Faculty ID</td>
</tr>
<tr>
<td></td>
<td>Campus ID</td>
</tr>
<tr>
<td></td>
<td>Head Of Department ID</td>
</tr>
</tbody>
</table>

**Table 14: Course Data dictionary**
<table>
<thead>
<tr>
<th>Acting HOD ID</th>
<th>Deputy HOD ID</th>
<th>Secretary ID</th>
<th>Admin Officer ID</th>
</tr>
</thead>
</table>

**Table 15: Department Data Dictionary**

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Faculties. db</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Indicates Various Faculties With the University</td>
</tr>
<tr>
<td>Contents</td>
<td>Faculty ID</td>
</tr>
<tr>
<td></td>
<td>Faculty</td>
</tr>
<tr>
<td></td>
<td>Faculty Code</td>
</tr>
</tbody>
</table>

**Table 16: Faculties Data Dictionary**

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Room Availabilities. db</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Indicates the Time and the Room Available.</td>
</tr>
<tr>
<td>Contents</td>
<td>Room Time ID</td>
</tr>
<tr>
<td></td>
<td>Room ID</td>
</tr>
<tr>
<td></td>
<td>Day</td>
</tr>
<tr>
<td></td>
<td>Start ID</td>
</tr>
<tr>
<td></td>
<td>Finish ID</td>
</tr>
</tbody>
</table>

**Table 17: Room Availabilities Data Dictionary**

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Room Teaching Usage. db</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Indicates the Room Utilised by Which Teaching Contact</td>
</tr>
<tr>
<td>Contents</td>
<td>Room ID</td>
</tr>
<tr>
<td></td>
<td>Teaching Contact ID</td>
</tr>
</tbody>
</table>

**Table 18: Room Teaching Usage**

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Room TimeTables. db</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Indicates the Timetable Based on Rooms and Weekdays</td>
</tr>
<tr>
<td>Contents</td>
<td>Room ID</td>
</tr>
<tr>
<td></td>
<td>Start</td>
</tr>
<tr>
<td></td>
<td>Sunday</td>
</tr>
<tr>
<td></td>
<td>Monday</td>
</tr>
<tr>
<td>Day</td>
<td>Subject Class ID</td>
</tr>
<tr>
<td>----------</td>
<td>------------------</td>
</tr>
<tr>
<td>Tuesday</td>
<td></td>
</tr>
<tr>
<td>Wednesday</td>
<td></td>
</tr>
<tr>
<td>Thursday</td>
<td></td>
</tr>
<tr>
<td>Friday</td>
<td></td>
</tr>
<tr>
<td>Saturday</td>
<td></td>
</tr>
<tr>
<td>Sun</td>
<td></td>
</tr>
<tr>
<td>Mon</td>
<td></td>
</tr>
<tr>
<td>Tue</td>
<td></td>
</tr>
<tr>
<td>Wed</td>
<td></td>
</tr>
<tr>
<td>Thu</td>
<td></td>
</tr>
<tr>
<td>Fri</td>
<td></td>
</tr>
<tr>
<td>Sat</td>
<td></td>
</tr>
</tbody>
</table>

**Table 19: Room Timetable Data Dictionary**

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Rooms. db</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Indicates the Various Rooms Within the Building and Other Variables</td>
</tr>
<tr>
<td><strong>Contents</strong></td>
<td>Room ID</td>
</tr>
<tr>
<td></td>
<td>Room</td>
</tr>
<tr>
<td></td>
<td>Room Number</td>
</tr>
<tr>
<td></td>
<td>Building ID</td>
</tr>
<tr>
<td></td>
<td>Capacity</td>
</tr>
</tbody>
</table>

**Table 20: Room Data Dictionary**

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Staffs. db</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Indicates the Staff and Their Details.</td>
</tr>
<tr>
<td><strong>Contents</strong></td>
<td>Staff ID</td>
</tr>
<tr>
<td></td>
<td>Staff Number</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>Surname</td>
<td></td>
</tr>
<tr>
<td>First Name</td>
<td></td>
</tr>
<tr>
<td>Preferred Name</td>
<td></td>
</tr>
<tr>
<td>Title ID</td>
<td></td>
</tr>
<tr>
<td>Appointed</td>
<td></td>
</tr>
<tr>
<td>Terminated</td>
<td></td>
</tr>
<tr>
<td>Department ID</td>
<td></td>
</tr>
<tr>
<td>Room ID</td>
<td></td>
</tr>
<tr>
<td>Office Phone</td>
<td></td>
</tr>
<tr>
<td>Email</td>
<td></td>
</tr>
<tr>
<td>Home Phone</td>
<td></td>
</tr>
<tr>
<td>Mug Shot</td>
<td></td>
</tr>
<tr>
<td>CV</td>
<td></td>
</tr>
</tbody>
</table>

**Table 21: Staff Data Dictionary**
<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Student Subject Class Preferences. db</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Indicates Students and the Subject Preferences they have Offered</td>
</tr>
<tr>
<td>Contents</td>
<td>Student ID</td>
</tr>
<tr>
<td></td>
<td>Subject Class ID</td>
</tr>
<tr>
<td></td>
<td>Preference</td>
</tr>
<tr>
<td></td>
<td>Notes</td>
</tr>
</tbody>
</table>

Table 22: Student Subject Class Preference Data Dictionary

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Student Subject Enrolments. db</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Indicates Students and the Subject They have Enrolled in</td>
</tr>
<tr>
<td>Contents</td>
<td>Student ID</td>
</tr>
<tr>
<td></td>
<td>Subject ID</td>
</tr>
<tr>
<td></td>
<td>Year</td>
</tr>
<tr>
<td></td>
<td>Semester</td>
</tr>
<tr>
<td></td>
<td>Grade</td>
</tr>
<tr>
<td></td>
<td>CP</td>
</tr>
</tbody>
</table>

Table 23: Student Subject Enrolments Data Dictionary
<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Students. Db</td>
<td>Indicates the Student Information and Personal Details</td>
</tr>
<tr>
<td></td>
<td>Student ID</td>
</tr>
<tr>
<td></td>
<td>Student Number</td>
</tr>
<tr>
<td></td>
<td>Surname</td>
</tr>
<tr>
<td></td>
<td>First Name</td>
</tr>
<tr>
<td></td>
<td>Preferred Name</td>
</tr>
<tr>
<td></td>
<td>Title ID</td>
</tr>
<tr>
<td></td>
<td>Enrolled</td>
</tr>
<tr>
<td></td>
<td>Completed</td>
</tr>
<tr>
<td></td>
<td>Course ID</td>
</tr>
<tr>
<td></td>
<td>Email</td>
</tr>
<tr>
<td></td>
<td>Home Phone</td>
</tr>
<tr>
<td></td>
<td>Mug Shot</td>
</tr>
<tr>
<td></td>
<td>CV</td>
</tr>
</tbody>
</table>

**Table 24: Student Data Dictionary**

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject Assessment Items. db</td>
<td>Indicates the Various Subjects and the Assessment Items in Each of Them.</td>
</tr>
<tr>
<td></td>
<td>Subject ID</td>
</tr>
<tr>
<td></td>
<td>Assessment Item ID</td>
</tr>
<tr>
<td></td>
<td>Assessment Item</td>
</tr>
<tr>
<td></td>
<td>Marks</td>
</tr>
<tr>
<td></td>
<td>Weighting</td>
</tr>
<tr>
<td></td>
<td>Notes</td>
</tr>
</tbody>
</table>

**Table 25: Subject Assessment Items Data Dictionary**
### Data Structure
Subject Assessment Student Marks. db

<table>
<thead>
<tr>
<th>Description</th>
<th>Indicates Marks Score in Assessment for Subject Enrolled by Students.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contents</td>
<td>Subject ID</td>
</tr>
<tr>
<td></td>
<td>Assessment Item ID</td>
</tr>
<tr>
<td></td>
<td>Student ID</td>
</tr>
<tr>
<td></td>
<td>Mark</td>
</tr>
<tr>
<td></td>
<td>Notes</td>
</tr>
</tbody>
</table>

**Table 26: Subject Assessment Student Marks Data Dictionary**

### Data Structure
Subject Class Clash. db

<table>
<thead>
<tr>
<th>Description</th>
<th>Indicates the Subjects That Clash in Terms of Timetable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contents</td>
<td>Subject Class ID</td>
</tr>
<tr>
<td></td>
<td>Subject Class ID Clash</td>
</tr>
</tbody>
</table>

**Table 27: Subject Class Clash Data Dictionary**

### Data Structure
Subject Class Possible Room Time Lists. db

<table>
<thead>
<tr>
<th>Description</th>
<th>Indicates the Possible Room Time for a Subject Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contents</td>
<td>Subject Class ID</td>
</tr>
<tr>
<td></td>
<td>Room Time ID</td>
</tr>
</tbody>
</table>

**Table 28: Subject Class Possible Room Time Lists Data Dictionary**

### Data Structure
Subject Class Student Enrolments. db

<table>
<thead>
<tr>
<th>Description</th>
<th>Indicates the Students Enrolled in a Subject Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contents</td>
<td>Subject Class ID</td>
</tr>
<tr>
<td></td>
<td>Student ID</td>
</tr>
</tbody>
</table>

**Table 29: Subject Class Student Enrolments Data Dictionary**
<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Subject Class Timetables. db</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Indicates the Subject Class Details</td>
</tr>
<tr>
<td><strong>Contents</strong></td>
<td>Subject Class ID</td>
</tr>
<tr>
<td></td>
<td>Subject ID</td>
</tr>
<tr>
<td></td>
<td>Teacher ID</td>
</tr>
<tr>
<td></td>
<td>Subject Class Title</td>
</tr>
<tr>
<td></td>
<td>Duration</td>
</tr>
<tr>
<td></td>
<td>Capacity</td>
</tr>
<tr>
<td></td>
<td>Size</td>
</tr>
<tr>
<td></td>
<td>Room ID</td>
</tr>
<tr>
<td></td>
<td>Day</td>
</tr>
<tr>
<td></td>
<td>Start ID</td>
</tr>
<tr>
<td></td>
<td>Finish ID</td>
</tr>
<tr>
<td></td>
<td>Notes</td>
</tr>
</tbody>
</table>

Table 30: Subject Class Timetables Data Dictionary

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Subject Student Grades. db</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Indicates the Grades Obtained by Student for a Subject</td>
</tr>
<tr>
<td><strong>Contents</strong></td>
<td>Subject ID</td>
</tr>
<tr>
<td></td>
<td>Student ID</td>
</tr>
<tr>
<td></td>
<td>Mark</td>
</tr>
<tr>
<td></td>
<td>Grade</td>
</tr>
<tr>
<td></td>
<td>Notes</td>
</tr>
</tbody>
</table>

Table 31: Subject Student Grade Data Dictionary

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Subject Teaching Team. db</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Indicates the Details of the Teaching Team Based on Subjects</td>
</tr>
<tr>
<td><strong>Contents</strong></td>
<td>Subject ID</td>
</tr>
<tr>
<td></td>
<td>Staff ID</td>
</tr>
<tr>
<td></td>
<td>Teaching Position ID</td>
</tr>
<tr>
<td></td>
<td>Hours per Week</td>
</tr>
<tr>
<td><strong>Volume</strong></td>
<td></td>
</tr>
</tbody>
</table>

Table 32: Subject Teaching Team Data Dictionary
### Table 33: Subject Data Dictionary

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Subject Details</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Indicates the Subject Details</td>
</tr>
<tr>
<td><strong>Contents</strong></td>
<td>Subject ID</td>
</tr>
<tr>
<td></td>
<td>Subject Code</td>
</tr>
<tr>
<td></td>
<td>Subject</td>
</tr>
<tr>
<td></td>
<td>Credit Point</td>
</tr>
<tr>
<td></td>
<td>Department ID</td>
</tr>
<tr>
<td></td>
<td>Campus ID</td>
</tr>
<tr>
<td></td>
<td>Year</td>
</tr>
<tr>
<td></td>
<td>Semester</td>
</tr>
<tr>
<td></td>
<td>Convenor ID</td>
</tr>
<tr>
<td></td>
<td>Planner ID</td>
</tr>
<tr>
<td></td>
<td>Timetable Notes</td>
</tr>
<tr>
<td></td>
<td>Brief Description</td>
</tr>
<tr>
<td></td>
<td>Objectives</td>
</tr>
<tr>
<td></td>
<td>Content</td>
</tr>
<tr>
<td></td>
<td>Organization</td>
</tr>
<tr>
<td></td>
<td>Assessment Rationale</td>
</tr>
<tr>
<td></td>
<td>Text Books</td>
</tr>
<tr>
<td></td>
<td>DSO</td>
</tr>
<tr>
<td></td>
<td>Lecture Guide</td>
</tr>
<tr>
<td></td>
<td>Tutorial Guide</td>
</tr>
<tr>
<td></td>
<td>Lab Guide</td>
</tr>
<tr>
<td></td>
<td>General Handouts Notes</td>
</tr>
<tr>
<td></td>
<td>Internal Notes</td>
</tr>
</tbody>
</table>

### Table 34: Teaching Contact Type Data Dictionary

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Teaching Contact Types. db</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Indicates the Various Teaching Contact</td>
</tr>
<tr>
<td><strong>Contents</strong></td>
<td>Teaching Contact ID</td>
</tr>
<tr>
<td></td>
<td>Teaching Contact</td>
</tr>
</tbody>
</table>

Table 33: Subject Data Dictionary

Table 34: Teaching Contact Type Data Dictionary
<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Teaching Finish Time. db</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Indicates the Various Finish Times</td>
</tr>
<tr>
<td>Contents</td>
<td>Finish ID</td>
</tr>
<tr>
<td></td>
<td>Finish</td>
</tr>
</tbody>
</table>

Table 35: Teaching Finish Time Data Dictionary

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Teaching Positions. db</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Indicates the Various Teaching Positions.</td>
</tr>
<tr>
<td>Contents</td>
<td>Teaching Position ID</td>
</tr>
<tr>
<td></td>
<td>Teaching Position</td>
</tr>
</tbody>
</table>

Table 36: Teaching Position Data Dictionary

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Teaching Start Time. db</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Indicates the Various Start Times</td>
</tr>
<tr>
<td>Contents</td>
<td>Start ID</td>
</tr>
<tr>
<td></td>
<td>Start</td>
</tr>
</tbody>
</table>

Table 37: Teaching Start Time Data Dictionary

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Titles. db</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Indicates the various Titles to staff and students.</td>
</tr>
<tr>
<td>Contents</td>
<td>Title ID</td>
</tr>
<tr>
<td></td>
<td>Title</td>
</tr>
</tbody>
</table>

Table 38: Trial Data Dictionary

Elements Include all the Primary Keys of all the Data Structure mentioned above. Also some of these Primary keys are referenced as Foreign Key in another table.

<table>
<thead>
<tr>
<th>Data Element</th>
<th>Building ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Identification and Authorisation for Each Building to the University</td>
</tr>
<tr>
<td>Type</td>
<td>Primary Key</td>
</tr>
<tr>
<td>Data Type</td>
<td>Integer</td>
</tr>
</tbody>
</table>

Table 39: Building ID data element
<table>
<thead>
<tr>
<th>Data Element</th>
<th>Campus ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Identification and Authorisation for Each Campus on the University</td>
</tr>
<tr>
<td>Type</td>
<td>Primary Key</td>
</tr>
<tr>
<td>Data Type</td>
<td>Integer</td>
</tr>
</tbody>
</table>

Table 40: Campus ID data element

<table>
<thead>
<tr>
<th>Data Element</th>
<th>Course ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Identification and Authorisation for Each Course that a Student Enrols Within</td>
</tr>
<tr>
<td>Type</td>
<td>Primary Key</td>
</tr>
<tr>
<td>Data Type</td>
<td>Integer</td>
</tr>
</tbody>
</table>

Table 41: Course ID data element

<table>
<thead>
<tr>
<th>Data Element</th>
<th>Department ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Identification and Authorisation for Each Department Within the University</td>
</tr>
<tr>
<td>Type</td>
<td>Primary Key</td>
</tr>
<tr>
<td>Data Type</td>
<td>Integer</td>
</tr>
</tbody>
</table>

Table 42: Department ID data element

<table>
<thead>
<tr>
<th>Data Element</th>
<th>Faculty ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Identification and Authorisation for Each Faculty of the University</td>
</tr>
<tr>
<td>Type</td>
<td>Primary Key</td>
</tr>
<tr>
<td>Data Type</td>
<td>Integer</td>
</tr>
</tbody>
</table>

Table 43: Faculty ID data element

<table>
<thead>
<tr>
<th>Data Element</th>
<th>Room ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Identification and Authorisation for Each Room in the University</td>
</tr>
<tr>
<td>Type</td>
<td>Primary Key</td>
</tr>
<tr>
<td>Data Type</td>
<td>Integer</td>
</tr>
</tbody>
</table>

Table 44: Room ID data element
<table>
<thead>
<tr>
<th>Data Element</th>
<th>Room Time ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Identification and Authorisation for Which Time What Room is Available</td>
</tr>
<tr>
<td>Type</td>
<td>Primary Key</td>
</tr>
<tr>
<td>Data Type</td>
<td>Integer</td>
</tr>
</tbody>
</table>

Table 45: Room Time ID data element

<table>
<thead>
<tr>
<th>Data Element</th>
<th>Staff ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Identification and Authorisation for Each Staff Member in the University</td>
</tr>
<tr>
<td>Type</td>
<td>Primary Key</td>
</tr>
<tr>
<td>Data Type</td>
<td>Integer</td>
</tr>
</tbody>
</table>

Table 46: Staff ID data element

<table>
<thead>
<tr>
<th>Data Element</th>
<th>Student ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Identification and Authorisation for Each Student of the University</td>
</tr>
<tr>
<td>Type</td>
<td>Primary Key</td>
</tr>
<tr>
<td>Data Type</td>
<td>Integer</td>
</tr>
</tbody>
</table>

Table 47: Student ID data element

<table>
<thead>
<tr>
<th>Data Element</th>
<th>Subject Class ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Identification and Authorisation for Each Subject and its Details Conducted Within the University</td>
</tr>
<tr>
<td>Type</td>
<td>Primary Key</td>
</tr>
<tr>
<td>Data Type</td>
<td>Integer</td>
</tr>
</tbody>
</table>

Table 48: Subject class ID data element

<table>
<thead>
<tr>
<th>Data Element</th>
<th>Subject ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Identification and Authorisation for Each Subject Conducted Within the university</td>
</tr>
<tr>
<td>Type</td>
<td>Primary Key</td>
</tr>
<tr>
<td>Data Type</td>
<td>Integer</td>
</tr>
</tbody>
</table>

Table 49: Subject ID data element
Data Element | Teaching Contact ID
---|---
Description | Identification and Authorisation of a Teaching Contact for a Subject Within the University
Type | Primary Key
Data Type | Integer

**Table 50: Teaching Contact ID data element**

Data Element | Teaching Position ID
---|---
Description | Identification and Authorisation of a Position for each Teaching Contact University.
Type | Primary Key
Data Type | Integer

**Table 51: Teaching position ID data element**

Data Element | Title ID
---|---
Description | Identification and Authorisation for each Staff as well as Student in the University
Type | Primary Key
Data Type | Integer

**Table 52: Title ID data element**

**Implementation**

The system as described above was implemented in Delphi 5.01 on a Pentium III workstation running NT5/SP2. 256mb RAM ensured that the system could be coded, tested and run under the IDE. This significantly shortened the time required to take the design of the previous section and create the running system described in this section.

Whilst the design of the system took 18 months to finalise the entire implementation using Delphi required only three months. The main operational forms within the system are shown next.

![Figure 28: Screen - Just Timetables (Main Menu)](image-url)
This is the screen which launches on start of the application. It links various forms through a main menu. In IDE terminology it is the MDI parent window, which indicates that this window would always be the base window within which all the other windows would open or close.

![Room Maintenance - Facility Details Tab](image)

Figure 29: Screen - Room Maintenance - Facility Details Tab

Selecting Rooms from the Facilities option on the main menu tool bar will activate the Room Maintenance form as displayed above. This form enables the user to access information and also store data about various rooms and any related information. The left side of the form contains a list of all the rooms and enables the user to quickly scroll through the existing rooms. The right side of the screen has three tab pages detailing all relevant information in regards to the room which is currently highlighted in the list to the left. The attributes of each room is listed on the Facility details tab page. This information can be altered as required, and rooms added or deleted as is necessary.
The University Table Problem

K. Sandhu

Figure 30: Screen - Room Maintenance – Room Timetable Tab

The Room Timetable tab page generates the timetable and displays the room utilisation during the week. As shown in Figure 30, the user can select a room from the list on the left hand side and instantly view a schedule of the current room utilisation. The timetable is generated on a day-to-day basis, giving an organised tabular view of room utilisation. There is also a facility for displaying the timetable for a number of selected rooms at the same time. This will allow the user to schedule specific queries by being able to list the useability of all types of rooms the query relates to.
The University Table Problem

K. Sandhu

Figure 31: Screen - Maintenance of Buildings Table

The Maintenance of Buildings form is accessed by selecting it from Facilities on the drop down menu on the main form. This form enables the user to create an entry of the various buildings within the university and the corresponding rooms inside each building. On the left side of the form the details of the building within the university are displayed. The set of buttons on the right hand side enables the user to add, delete or navigate through the building information. In addition users can make a selection from the drop down list as to which campus the building belongs to.
The Maintenance of Campuses form is accessed from the facilities toolbar. The Campus Details and Buildings tab enables the user to add and view buildings listed on the relevant Campus. In addition, the user can use the navigation tool bar to manipulate data and navigate around with the existing data.
Within the Maintenance of Campuses form there is a Subjects Available on This Campus tab which enables the user to select all the subjects that are offered by a particular campus. The user can set other parameters such as the year and semester when the subject would be offered, thus establishing a link between campuses and subjects offered.
The Maintenance of Departments form is accessed from a selection from the Resources drop down menu option on the main form. As shown in Figure 34, on the left hand side of the form there is a list of all the departments within the university. Information relating to the highlighted Department is shown on the three tab pages to the right. The Department Details tab lists all departmental information relating to Faculty, Campus and Head of School, etc. The navigator toolbar facilitates data manipulation.

Figure 34: Screen - Maintenance of Departments – Department Details Table
The Staff Members and Profile tab located on the Maintenance of Departments form enables the user to list all employees within the department and manage their profile. The staff details can be selected by using a drop down menu to select the name of the staff member.
The Subject in the Department tab within the Maintenance of Department form links all subjects offered by the department selected from the list on the left hand side of Figure 36 shown above. After highlighting the required department the subjects offered are displayed in the grid on the right hand side of the screen.
The Maintenance of Staff Details form is loaded by selecting the Staff option under the Resources toolbar. This form enables the user to enter data relating to staff members of the university. In addition to the personal details of the staff members there are other tabs, which contain other related information. Within the Main Details tab, the Department field enables the user to make a selection of which department the staff member is affiliated with and also contains a drop down list for which room is allocated to which staff member.
The Staff CV tab on the Maintenance of Staff Details form has a large text area on the right hand side that the user can copy and paste the CV of the staff member in a text format. The user can select the staff member from the left and read their CV in the right side of the form.
The Teaching Responsibilities tab of the Maintenance of Staff Details indicates and lists all the teaching responsibilities attached to the selected staff member. The workload details for a staff member can be referred in this tabulated area on this form. The various columns (Hours per week, Subject Code, Subject name, and Notes) enables the user to have a quick view of a staff member’s workload.
The final tab (Subject Covered) on the Maintenance of Staff Details screen lists all subjects that the staff member convenes for the department. It indicates the Subject Code, Subject Name, year offered and semester offered details on the right hand side. Listed on the left hand side are the names of all staff members.
Figure 41: Screen - Maintenance of Student Details – Personal Details Table

The Maintenance of Student Details form is activated from the main menu toolbar by selecting the Student option. The first tab is the Personal Details tab which records personal information about the student including their enrolment date and other related fields. The right hand corner of the screen has a grey area which can contain a picture of the student for future reference.
The next tab on the Maintenance of Student Details screen is the Course Enrolment tab. This tab contains information relating to the course the student is enrolled in. The course can be selected from the drop down list provided and other details like the specified CP and enrolment dates are updated at the same instance.
Figure 43: Screen - Maintenance of Student Details – Subject Enrolments and Results Table

Subject Enrolments and Results is the final tab of the Maintenance of Student Details screen. This tab page contains the subjects the student is enrolled in and grades received. The subject codes are accessed from a drop down list on the right hand side of the form. The user can select the student from the list on the left hand side of the page to view the relevant subject information on the right hand side.
The Maintenance of Courses screen is activated by selecting the Programmes option from the main menu toolbar on the main form. On selection Figure 44 appears on the screen, allowing the user to update current course information or add new courses offered. The first tab, Main Details, contains three sub-tabs. The Introduction sub-tab provides details about the course and would also enable data entry and manipulation of the above details. As with all sub-tab pages on the Maintenance of Courses screen, updated information can be saved or cancelled by using the standard buttons in the toolbar at the top of the screen.
Figure 45: Screen - Maintenance of Courses – Degree Rules Sub-Table

Figure 45 shows the Degree Rules sub-tab of the tab mentioned in the description. This sub-tab provides a memo to cut and copy or recreate the set of Rules and Regulation relating to the selected Course. As with all sub-tab pages on the Maintenance of Courses screen, updated information can be saved or cancelled by using the standard buttons in the toolbar at the top of the screen.
The Further Information sub-tab provides additional details about the selected course and enables data entry and manipulation of the above details. This sub-tab can be used for various purposes at top-level management. As with all sub-tab pages on the Maintenance of Courses screen, updated information can be saved or cancelled by using the standard buttons in the toolbar at the top of the screen.
The next tab on the Maintenance of Courses form is the Subjects tab. After completing the Main Details form the user needs to register all the subject offered within a course. Thus this form enables the user to enter data for the selected course. Information is updated by using the navigator buttons located directly above the grid on the right hand side of the screen.
The Enrolments tab indicates all students registered for the selected course. This can be manipulated using the navigation tools located directly above the Student list in the centre of the screen. Other related information is also displayed to the right hand side of the form.

Figure 48: Screen - Maintenance of Courses – Enrolments Table
The Maintenance of Subjects form is executed by selecting Subjects from the Programmes drop down menu option of the main menu located on the main form. Of the various tabs involved on this form, the first tab, by default, is the Main Details tab. The second tab page, Enrolments, indicates the details of how many people are enrolled for the selected course. The third tab page, Course, Faculty and Other Links, indicates the various courses and other related tabs. There are a number of sub-tabs on the main details tab page. The first sub-tab, Class and Timetabling, enables the user to select the subject, assign the subject class time and then create all static information related to it. This form is an integral part for construction of a subject.
In Figure 50, the next sub-tab on the Maintenance of Subjects screen is the Teaching Team sub-tab. Here the user can select a staff member by using a drop down list which only offers those teachers who are involved with the program. Relevant information about the staff member’s teaching position, hrs/week, notes etc is also displayed.
The sub-tab displayed in Figure 51, Brief Description, is located on the Maintenance of Subjects form. It enables the user to store a brief outline of the subject highlighted in the list on the left hand side. It also has the capability of making internal notes for reference purposes.
The next sub-tab on the Main Details tab page of the Maintenance of Subjects screen is the Objectives sub-tab. This sub-tab enables the user to store and recollect all information in regards to the objective of the subject. The user can cut and copy the text into the memo text area.
Figure 53: Screen - Maintenance of Subjects – Content Sub-Table

Figure 53 shows the Content sub-tab located on the Maintenance of Subjects form. This sub-tab enables the user to store all information regarding the content of the subject. The user can also provide other details, such as recommended text books, within this form.
The final sub-tab on the Maintenance of Subjects screen is the Assessment sub-tab. This sub-tab is used to store all information regarding the assessment of the subject. Also provided on this form is a text box to record information such as recommended text books etc.

![Figure 54: Screen - Maintenance of Subjects - Assessment Sub-Table](image)

<table>
<thead>
<tr>
<th>ASSESSMENT:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
</tr>
<tr>
<td>Mid-Semester Exam</td>
</tr>
<tr>
<td>1 Project</td>
</tr>
</tbody>
</table>

Text Books:
- **No Prescribed text**
- **Recommended Text Books**
  - *Depths in 21 Days* by SAMS Publishing House
The enrolment tab gives access to all necessary details required for timetable generation. In the figure above details of 1001MGT are displayed and on completion of these details the manual timetable can be generated as shown in the next figure.
The figure above depicts the generated timetable for 1001MGT. It displays the lecture times, tutorial times, room details and lecturer name for this subject. The duration of each teaching segment can be worked out from the timetable itself that lists each segment in hourly intervals.
Figure 57: Generated Manual Timetable for a combination of subjects

The figure above shows the ability of the system to generate a timetable for a combination of a number of subjects. In this example the timetable for all first year subjects related to the Bachelor of Commerce are displayed. Again the tutorials and laboratories are grouped as close to the lecture as possible and on the same day.
The University Timetable Problem
K. Sandhu

Figure 58: Generated Automated Timetable for one subject

The figure above depicts the automated generated timetable for the subject 1001MGT. It can be seen that the lecture and tutorials are evenly spread over the week and not grouped together. The figure below is another example of the same subject generated where 100 points are added to the objective function for each tutorial that is before or follows the lecture. In this instance we have a better grouping of the tutorials.
A discussion on the ability of the system to generate manual and automated timetables, with hard copy presentation, is detailed in the next section.

**Results and Discussion**

The system as developed and described in detail above could not be put into a real world setting for very practical reasons. Nonetheless the level of efficacy of the implementation needed to be established. The system itself enabled a smooth operation and indeed the objective of being able to support timetable construction on a smooth scale from complete manual setting through to automated generation was achieved.

In order to generate the timetables reliance was placed on the information relating to Semester 2 2002 timetables. This data was used incrementally in seeding the Subject Enrolment screen in Figure 55. From this the manual timetables for individual subjects were generated and hard copies of a number of subjects are shown in Appendix 1. This
The automated timetable generation is carried out using a DSTA. Appendix 7 shows a hard copy of the timetable for 1001MGT. It can be seen that this is a lower quality timetable than the manually generated copy. The tutorials do not fall on the same day and do not precede or follow the lecture. Appendix 8 illustrates the timetable for the same subject (1001MGT) where 100 points have been added to the objective function for each tutorial follows a lecture. It can be seen that this has improved the quality of the timetable and tutorials are grouped in a more suitable fashion. Thus it can be observed that the objective function can indeed be used to improve the quality of the solution.

The construction of the objective function is a multi-faceted complex issue. At one extreme complete enumeration while generating solutions is required to satisfy all constraints. This typically is operationally unfeasible. At the other extreme sacrificing hard constraints to ensure rapid convergence towards solutions may lead to an overstrained definition resulting in no solutions. Consequently a balance needs to be found between generation time and the need to minimise potential conflicts. The determination of an optimal mix would lead to a more efficient generation method. However the setting of this optimal mix is a complex process and in this work many versions of the objective function were trialed. Positive points were awarded for
conditions that were deemed to be desirable and negative points for unwanted conditions. This was an arbitrary formulation and requires significant research in the future. It is proposed that this future research should focus on developing a dynamic objective function that can be manipulated by end users to satisfy institutional specific criteria. For example in an institution where rooms is not a scarce resource the objective function could be weighted heavily to having tutorials and lectures on the same day while on the other hand institutions where rooms are scarce this would be weighted less importantly.

Hence, from the presentation above, it can be seen that the system has the ability to generate manual timetable from different perspectives as well as generating automated timetables using a constraint-satisfaction based algorithm. In addition to this the report generator allows for the creation of specific management reports that can be produced as and when the need arises. Therefore comparisons carried out between the objectives of the system and the output produced by the system indicate that the requirements have been achieved.

To test that others could also use the system with relative ease a number of university administrators were asked to provide a preliminary evaluation of the system. Four such evaluations were carried out. Due to the size of the system, it was necessary that a user already familiar with the system be present so that the relatively large learning curve required to become familiar with the system’s use could be avoided. Whilst it would have been desirable to have first trained these administrators in the complete use of the system it was not practical, and certainly beyond the scope of this work, to do so.

The results of the four evaluations ranged from very positive to highly enthusiastic. All four administrators complimented the system on its ease of use, simplicity of navigation, and the overall fully integrated functionality. All made comments about the lack of duplication which they have all experienced in the systems that they use on a day-to-day basis in their jobs. More than once the comment “when can we use this system” was made by all four evaluators.

None-the-less, these evaluations were almost totally subjective in nature and the participants were not chosen randomly from the pool of all possible university administrators and timetable officers. This first evaluation was used primarily to eliminate the possibility that the system was not of a user-friendly, too technical, nature.
Again, it is acknowledged that this evaluation is qualitative only, but certainly helped to eliminate the possibility that the result was totally unacceptable to administrators.

This exercise, however, highlighted a deficiency in the tools available to researchers in this area, in the evaluation of timetabling systems. This gap being the need for an objective evaluation of a proposed automated timetabling and timetable information system. No references in the literature could be found that even addressed this deficiency.

**Conclusion and Future Enhancements**

The stated objective to develop a university timetable information system that supported all aspects of other teaching related administrative functions has been achieved. Users of an existing system have enthusiastically received the system. It is noted that this experiment and reception does not represent an objective evaluation of a system. The timetable support functionality, although only evaluated by one expert user, was well received but, again, does not allow any objective conclusion to be drawn regarding the efficiency or effectiveness of the overall timetable solution generator encoded.

However given the enormity of the task already undertaken there is still a significant amount of hack programming required to be completed before a production run of the system for widespread use could be achieved. Nonetheless the current version could be customised for in-house use in any one university in a relatively straightforward manner. The issue of conversion from current systems to this system, though, would require the careful planning and transition that goes with such information system changeovers. On a positive final note there would appear to be almost no training issues to be addressed.

There is a need for the development of a mechanism for the objective evaluation of timetable information systems. This is absolutely necessary when trying to determine if the automated solutions generated represent “better” solutions than those of previous implementations. The next chapter seeks to address this issue by way of developing two tests that can determine whether or not an implementation has successfully generated solutions and identified the best possible solution in a reasonable time frame.
CHAPTER 4
EVALUATING THE EFFECTIVENESS OF TIMETABLE SOLUTION GENERATORS

This chapter investigates the use of benchmarking to provide some measure of the effectiveness of a timetable solution generator. The need for such a benchmark was discussed in the previous chapter. In this chapter we concentrate on the problem of evaluating the solution generator itself. The other aspects of a timetable information system are not considered here.

A benchmark is proposed which consists of a series of two tests. Each test is designed to assist in the evaluation of solution generators at different stages in their research development life cycle. The work presented in this chapter is an attempt at the objective comparison of different timetable solution generators.

The need for the objective evaluation of timetable systems is argued in the first instance from the literature review carried out. This need is discussed in the next section. In particular comparisons with other attempts at providing metrics for automated generators of other problems is included to show that such evaluations have been successfully attempted in these research areas.

The benchmark is then proposed and details of its two tests included. The chapter ends with a general discussion suggesting how the use of this benchmark can be facilitated.

Background

There has been no documented method for the objective comparison of the efficacy and run-time efficiency of any timetable solution implementation to date despite the nearly fifty years over which the literature has developed. As noted in chapter 2 the first half of this literature is liberally sprinkled with claims of successful implementations and/or solution generation algorithms. Although such claims have tended to diminish, researchers have found it especially difficult in recent years to compare the performance of their own particular implementation against any earlier work. The need to demonstrate an objectively superior implementation has become significantly more important considering the subtle speed and other various performance differences between algorithms. Without clearly defined and objective metrics, a heuristic
improvement in an algorithm, for example in the use of a branch-and-bound technique to traverse the search space after the first solution has been found, will be very difficult (if not impossible) to demonstrate.

Of course, these kinds of benchmarking problems exist for all ongoing research problems and are not restricted to timetable automation. Consequently many of the remarks and suggested techniques have quite broad implications and applications. A number of closely related attempts at benchmarking have been used as the prototype for the current benchmark (ref Spring, Berghel, Harris and Forster, 1992; Harris, Forster and Sandhu, 2001).

It is generally well known that benchmarks are not extensively published in the computer literature. Due to this paucity the state of benchmarking is now examined as a background for the currently proposed timetable solution generator benchmark. Consequently, a computerised literature research has yielded very few references, most of which are related to benchmarking in only the most marginal of terms. However the few which are related are very detailed in their focus and can be categorised into the following three classes:

1. Hardware and/or software benchmarks
2. Database performance benchmarks
3. Research

The first category contains the largest body of published work to date. The Database Performance benchmarks (the second category) were developed by the Transaction Processing Performance Council (TPC) in an attempt to reduce the sheer number of conflicting implementation claims (Moad, 1992). In contrast solely research orientated benchmark work is very scant indeed.

**Hardware and/or Software Benchmarks**

Raising questions regarding standard units of measurement, Houston (1984) discussed the different inherent problems with benchmarking hardware and software. He also raised the problems associated with allowing for technological development. The results of such published benchmarks can lead to erroneous conclusions about performances being drawn (Fleming and Wallace, 1986). Unintentionally Flemming
and Wallace have presented the elementary nature and lack of noteworthy work in this area by discussing the need to be wary in using the arithmetic mean in benchmarks. It is also suggested that improved statistical analyses of benchmark performances need to be considered. Discussing the use of rather more complex but not overly sophisticated statistical models, Berry (1992) more correctly evaluates overall IT performance, which allows for more robust benchmarks. Discussed by Salamone (1990) is the need for LAN benchmarks but also considered is the need for benchmarks in other Information Technology areas (Eckerson, 1991; Bentley, Kernigham and Van Wyk, 1991; Moad, 1992).

The performance of benchmarks that result from automatic parallelisation is discussed by Blume and Eigenmann (1992). Some 60,000 lines of Fortran source code were tested in the engineering and scientific computing areas. They successfully demonstrated the importance of benchmarks specifically designed for parallel technology. The measurement of performance gains in “real” programs was also discussed. This is of special interest in the timetable literature given not only the increased availability of parallel processing hardware but more importantly due to the number of publications over the last decade which have claimed improved in algorithms from parallelisation.

Serlin (1989) attempted to develop a benchmark for transaction processing systems so as to disembark from the rather simplistic hardware based tests that had currently existed. This was the first real attempt at benchmarking algorithms. The specific aim of the work being to move towards a reduced number of conflicting performance claims that often arise in relation to hardware based tests.

Database Performance Benchmarks

The establishment of the TPC benchmarks (Dietrich, Brown, Cortes-Rello and Wunderlin, 1992) was created by the need for a benchmark for database intensive commercial systems. Both Sivula (1990) and Magney (1992) offer evidence that, given the commercial factors involved, it is impossible to develop a truly independent matrix of performance benchmarks that could be generally accepted. It was argued that this is due to hardware and software vendors specifically tuning their implementations to perform far better than the published benchmarks. Nevertheless, considerable interest was generated in the TPC-A, B and C benchmarks (Serlin, 1989; Moad, 1992; Dietrich et al, 1992; Li, Wendel and Held, 1993; Nash, 1993; Burgess, 1993). Shown by
Johnson and Gray (1993) were the very practical benefits that accrue from having widely accepted benchmarks. They compliment their conclusions with a very detailed analysis of all the TPC benchmarks.

Research Benchmarks

The first paper to develop a benchmark from a purely academic perspective was published by Berghel and Rankin (1990) and was created for the implementation of Crossword puzzle solution generation algorithms. An improved benchmark was developed for an associated, and much more complex problem, i.e. the automation of the Crozzle (Spring, 1993) which based upon a completely new set of tests. The publications in this area suggest that even for well-defined problems (eg. variants of word problems) the creation of benchmarks is problematic and therefore cannot be a straightforward task. In the case of the Crossword and Crozzle problems it is much more difficult considering that the Crozzle is competitive and has its own metric of success i.e. the number of points scored by valid completed solutions varies with the highest points score being the winner (Spring, 1993). Taking this into context, the benchmark proposed here is less concerned with speed, and more concerned with the ability to generate high quality solutions.

Very few other publications on benchmarks could be located. Indeed not one paper discussing timetable benchmarks could be sourced. Consequently, the relative performances of algorithm implementations has not been compared to date. The proposed benchmark in this chapter is intended as a starting point for such discussions in the area of benchmarking timetable solution generation algorithms but, it is hoped, it holds lessons for benchmarking in general.

Data requirements

Before any objective evaluation of a system can be performed it is necessary that a test data set be compiled. As with any data to test any system, be it computerised or otherwise, it is necessary to obtain a representative coverage of the final data that the system is to operate on. In this case the data for one complete university would represent a good test data set. However there were many issues in attempting to acquire such a data set.
Firstly the sheer size of the data to be gathered. In the current institution there are over 60 schools spread across 6 geographically separate campuses with in excess of 2,000 teaching staff servicing over 20,000 students enrolled in over 400 courses. Further difficulty was experienced, as the data was not all collated in a single database. Although most of the data could have been collated from material published on the web, there were still an issue of privacy. However this issue was finally resolved and all data used is Griffith University data.

To ensure that a representative profile of the data requirements was produced meant that it was necessary to evaluate the current and historical data held in university publications and reports. After studying these reports it was decided that the following parameters would be used in collating and generating the data set.

Where possible complete data profiling the overall university structure would be gathered in full. This included the following:

- **Campuses** – all campuses would be entered to ensure the system could support a multi campus set up.
- **Faculties** – all faculty details would be entered as they could be cross campus and thus represent an aspect of any system that purports to support real world functionality.
- **Facilities** – all buildings would be entered for one campus and all rooms in all buildings.
- **Departments** – as this represents a level of specialisation in teaching only the data for one department would be entered but would be complete in all aspects including number and teaching responsibilities of all staff members, and thus all subjects taught.
- **Student enrolment data** – This was sourced from Griffith data and represents enrolments for semester 2 2002.
- The other data such as personal details and results are inconsequential to the timetable generation aspect of the system but are significant in a completeness sense for the test of any integrated student information system. However issues of privacy prevented access to this personal data. As this data is only descriptive it was decided to generate it using lists and a random number based selection algorithm.
The lists were drawn up of street names, and suburbs. Using a random number generator, random selections were made from these lists to create a database of student personal details that were randomly matched to existing Griffith students. Enrolments, however, were sourced from real data.

- Subject details were sourced from the web. These documents contained enough public information to ensure a virtual 100% accuracy of the representation of the staff, teaching duties, and responsibilities as well as the subject timetabling requirements.

- The courses these subjects belonged to were also able to be sourced in the same manner and their details, although not relevant to timetabling, were entered for the sake of completeness.

Consequently and in summary a reasonable data set adequate for the purposes of system coverage as well as timetable generation was gathered and generated. All data used for the purpose of timetable generation was real Griffith data. The only data that needed to be generated artificially was student personal data. This data was presented merely for completeness of the system. Using this approach one could, with enough resources, reasonably profile an entire institution’s enrolments for the purposes of testing such systems. Certainly, though, the current data is more than adequate for the purposes of generating benchmark results.

This benchmark data has now been fully compiled and is available upon request from the author. It has also been suggested (Havens, 1997) that the data proposed for use be made available on a Web site to facilitate ongoing research by others in this area. The administrative processes necessary to make this a reality have been initiated and are expected to result in a publicly available benchmark data set in the near future.

**Proposed Benchmark**

The benchmark put forward here for use in evaluating timetable solution generators is necessarily comprised of two tests. These tests are designed to enable the objective measurement of the implementations during those three phases that these systems go through. Each test will be described in detail and separately along with the rationale for the inclusion of the test. These tests are chosen to fit the model proposed by Berghel and Rankin (1990) in their first benchmark developed for a similar problem, crossword generation. As with their first benchmark, this benchmark is comprised of just 2 tests.
Test 1

The first phase of any timetable generation system involves determining the correctness of the implementation. That is, given the algorithm is an intangible set of instructions, how does one determine that the implementation is carrying out the rules and directions of the algorithm? Due to the nature of the timetable problem it is not possible to come up with a specific benchmark test for this situation. Thus it is proposed that this first test consists of the ability of the solution generator to determine the optimal solution for a timetable situation in which there are no clashes. Furthermore each solution will be of equal weight. Thus for this test, all costs must be zero and to successfully pass this test an implementation is timed to determine how long it takes to not only generate a solution but to then deduce that that solution is optimal.

This test is of most benefit to the CSP approaches. For these algorithms, because there are no constraints, they will quickly go to the bottom of the search tree and begin their backtracking. The effectiveness of the test for the CSP implementations is thus the efficiency of the branch-and-bound techniques used to trim the remainder of the solution phase space.

However, for all other algorithms except LP/IP, this test is subtly different. By their very nature these other algorithms that use random starting points will have valid solutions to start from. Consequently this test then determines a measure of their ability to realise that their valid starting points represent optimal solutions.

Test 2

Having determined the correctness of an implementation in absence of constraints, and thus the ability to generate valid solutions, the next phase involves a measurement of the quality of the solutions generated. In this second test the value of the objective function is now taken into account. In particular, it is proposed that any implementation, which cannot generate solutions, whose objective function value is not statistically discernible from a randomly generated solution has no value.

For this second test it is necessary that a number of solutions be generated by an implementation. These solutions are then to have their “fitness” evaluated. An equal number of randomly constructed solutions obeying hard constraints be also generated. A simple differences of two means statistical test is then to be performed to enable, at
99% confidence, a statement to be made as to whether the average quality of the output from the implementation is statistically significantly different from the average output of the randomly generated solution set.

This test again is subtly different between the two classes of solution generators discussed in Test 1 above. For the CSP based solution generators it measures the ability of hybrid approaches to quickly arrive at good solutions as opposed to measuring the ability of the other approaches’ iterative improvement techniques. The difficulty in the current test is the differentiation between hard and soft constraints. A number of papers in the literature (see eg. Schaerf, 1999; Wren, 1996, and appropriate references therein) note that for one algorithm what is a soft constraint is equally well defined as a hard constraint in another approach. In this first benchmark it is left up to the judgment of individual researchers to ensure that they clearly generate their stochastic solution set with a set of hard constraints appropriate to their approach.

**Results and Discussion**

The database described above was used to test the implementation of the timetable information system described in the previous chapter. The results were as follows:

**Test 1**

The data from Griffith University was used for this test. To ensure that the requirement of no clashes was met, the student enrolments were randomly constructed so that each student was enrolled in one subject only. Similarly, each staff member was randomly assigned one class only to teach. No constraints were placed on room requirements. The system successfully trimmed the search space using a standard branch-and-bound technique. In a number of runs, the number of nodes traversed was always less than 2N, where N = the number of classes to be assigned. Due to the use of a CSP-based algorithm, the time taken for each run was less and 1 second with 1000 students enrolled in 100 subjects with 100 teachers to teach each class.

As an extension, the trimming was removed to check if the entire search space was traversed. This was verified by a small test (due to the explosion in the size of the search space) in which all combinations of classes to rooms was generated, counted and checked against the analytically known result of $n!$P$_r$ where n = the number of rooms available and r = the number of classes to be assigned. That is, we want to assign r classes to n rooms where temporal order is important – a classic permutation problem.
Test 2

Using the same dataset, this test requires an appropriate objective function to be determined. For our purposes, and following the rationale used in a number of papers using genetic algorithms, the following function was constructed:

Value of a timetable, \( V(t) \) is given by:

\[ V(t) = N - P - Q \]

where:

- \( N \) = number of classes assigned to their own room (that is, no room clash)
- \( P \) = number of classes with student and/or staff clashes
- \( Q \) = number of classes that simply could not be scheduled (time clashes).

This function, although discrete, will have a maximum value equal to \( C \), the number of class assignments that need to be made. Its minimum value must necessarily be greater than the \(-C\).

When applied to randomly generated timetables the value of the function continually dropped as \( C \) increased. That is, and not surprisingly, the number of clashes with a randomly constructed timetable increases as the number of constraints in the problem decreases.

The implementation described in this chapter, being a CSP algorithm, never generated any value for \( P \) (by design no classes will be assigned that result in staff or student classes). However, as the number of classes to be assigned increased, the algorithm took longer and longer to generate solutions. By forcing large values of \( C \), and introducing a timeout value of 5 minutes for each run (starting from different, randomly selected, class lists), the value for \( Q \) increased and \( N \) decreased (\( C = N + Q \) here as \( P = 0 \)).

Statistically, there was a significant difference between the average result of the randomly generated timetables and those generated by the current implementation. This difference increased as \( C \) increased. For arbitrary low values of \( C \) (found by experimentation to be values of \( C/n < 0.5 \)) there was no significant difference.

Finally, it was interesting to note that even though large values of \( C \) often resulted in the algorithm not finding a complete solution, the algorithm was always able to place more than 50% of the classes. (At 50%, the \( V(t) = 0 \).) A further issue that needs to be addressed is to ascertain the viability of the system.
**System Usability**

In order to draw any viable conclusions on the functionality of the system an objective appraisal on the usability of the system needs to be undertaken. One test as pointed out by Wu and Wu (1994) is to carry out an acceptance test. This test is typically undertaken when the system is essentially implemented in the organization and users actually use the system. In the case of the Timetabling Information System this cannot be carried out, as the system has not been implemented into an organization. Nevertheless other objective tests could be performed. A measure of usability should include a measure of how users apply the system and also the ‘user-friendliness’ of their perception of the system. Users attitude to the system can depend on various factors including their level of technological competency and the echelon of training provided to them. Thus any measure involving users has to take into account the subjective nature of any test that is applied. A broad observation of the system can provide some indication of the usability of the system by the very nature of encompassing most of the important administrative processes and managerial functions pertaining to timetabling. This in turn can point to the ‘user-friendliness’ of the system since all functionality is encapsulated in the one application. Nevertheless these measures are still highly subjective and do not really lead to an objective assessment of the system. Any objective test on usability typically requires a determination of a set of objective usability criteria, a randomly selected sample of users and a data collection process to collect the responses of the sample. The data collected then needs to undergo various statistical tests from which inferences and evaluation of the usability if the system can be drawn. However this is not the main focus of the current work and certainly beyond the scope of the thesis. Nevertheless a simple process to determine some form of objective analysis of the system is required. This analysis should evaluate the correctness of the system that will then provide a measure of the integrity of the system. If the test proves to be valid then it can be argued that the system is usable as it presents results that are free of error and thus can be relied upon.

The process of undertaking this test involves the comparison of results produced by the system against historical results that have been used by the university previously. A number of subjects are selected at random and their timetables are generated. These are then compared with the existing subject timetable for semester 2 2002. The results of this exercise are shown in Appendix 9. It can be seen from these examples that the system is able to generate timetables that are characteristically similar to the actual
timetable that was utilized in the university for the selected subjects. Hence we can objectively infer that the system does produce results that can be verified and thus are usable.

**Future Research**

The need to facilitate at an objective level both the effectiveness of implementations and the claims by researchers in the area has been sorely lacking in the literature to date. The work presented in this chapter represents the first step towards achieving this goal. This chapter has provided the structure for a fair and objective means of promoting and evaluating research into the timetable problem with the provision of the first benchmark for such research. The two tests detailed cover aspects of the research life cycle for this problem. However, there is still a large amount of work, both research and administrative, which needs to be done to make this a reality. Certainly this is beyond the scope of the current thesis. This would include:

- Development of a “common” test database that would be readily available for all researchers to access,
- A multi-level set of objective functions to be optimised, and,
- Customisation of the hardware tools used.

A common and readily sourced test database would enable competing claims of improvements in algorithms and implementations to be objectively settled. This has been sorely missed in the literature to date and is an absolute must for future research. Especially so given the costs associated with the manual construction of timetables.

In line with the diversity of actual timetable problems that exist, a number of objective functions, based on the test database mentioned, will need to be developed. For instance, a regular weekly timetable has different constraints from a 6-monthly exam timetable.

Finally, the huge disparity in computing hardware, plus the increase in speed of the new hardware being released, requires that tests need to be developed that are either hardware independent (as in the ones presented in this Chapter) and/or have an open-ended component which is not diminished by simply porting to a faster hardware platform.
CHAPTER 5

CONCLUSIONS AND FUTURE RESEARCH

This thesis has examined the timetable scheduling problem. It began with a discussion into the terminology and size of the research area. A strong distinction between the terms scheduling and timetabling was made to avoid confusion in the work presented.

An overview of the entirety of the timetable generation literature that could be sourced was presented. The major solution algorithms that have stood the test of time were summarized. This was followed by a detailed examination of the literature. Using the major solution techniques the literature was classified, and a summary presented which demonstrated the temporal trends that have passed through this literature.

From these trends objectives for the research were developed and a proposed information system to support all aspects of university teaching was developed. Its uniqueness, in being derived and built on a foundation of supporting timetable generation, was highlighted. Screen dumps of most of the fundamental data entry screens for an implementation of this system highlighted the seamless integration of an information system that supported not only research activity but also transactional administrative functions. An initial evaluation by a number of interested parties, especially users, although necessarily conducted in a purely exploratory manner, provided positive feedback. This feedback included comments regarding the ease of use of the system. The system that was developed produced outcomes that were expected and hence met the requirements set by the objectives.

A much larger test of the system is suggested and the construction of a hypothesis to enable the statistical determination of the system be carried out in conjunction with a number of commercially available products. Such work is beyond the scope of this thesis.

An issue brought to light in the extensive literature review was ongoing but invalid claims of the effectiveness of solution algorithms and their implementations. In an attempt to address this issue a benchmark was proposed and presented. A detailed discussion of the tests that comprise the benchmark was presented.

A more complete set of metrics is required to enable the benchmark to more fully facilitate the objective comparison of alternative implementations. Given the literature review presented in Chapter 2 the absence of benchmarks has been a significant failing
in the cumulative research effort. The aim of any future test would be to provide an objective means by which comparisons by different researchers can be made. Such a test would require a realistic database, as described in the previous chapter, to be produced and made freely available via an appropriate medium such as the Web. The test requires that the data set be downloaded and be used by an implementation. A readme file would have to be prepared which describes the constraints that exist as information in the database. Implementations have to generate their optimal solution within time frames of one minute, one hour and one day of processing time. This would enables elegant solutions to demonstrate their effectiveness by generating high quality solutions in short time periods while not proving to be a disadvantage to the more brute force applications which are intended to spend some time traversing the solution phase space.

By its very nature the choice of the dataset is contentious. A rationale would have to be developed for the choice of tables to be used in this test. Whilst the choice will undoubtedly be argued against by some in the literature, it nonetheless represents a starting point from which the development of more advanced evaluations and future benchmarks could be made.

It is suggested that future research efforts concentrate on hybrid approaches between CSP and stochastic based solution generators. Although some hybrid generators have been discussed in the literature it is felt that they have not exploited or utilized the iteration between the different solution algorithms made available. It would be interesting to see research in which iterative improvement algorithms generated solutions in which the common assignments were fixed thus reducing the problem size down to a size where an exhaustive examination by a CSP based technique should result in an overall high quality solution. Additionally a significant information system research project should concentrate on efforts in establishing a dynamic objective function formulation that can be manipulated by end users to satisfy individual institutional requirements.

The research literature is scant indeed on the partial timetable problem. These problems occur when a generated timetable needs to be adjusted at a later date, while minimizing disruption to the current in use solution. Anecdotal evidence from the evaluators suggests that this is a difficult problem that has been the focus of much attention by higher-level administrators.
The final avenue for future research is in the application and further development of the benchmark proposed in this thesis. Testing the benchmark was not possible due to time constraints. Consequently the potential exists for research into applying and comparing these tests. Also the existence and application of this benchmark infers that further benchmarks need be developed that examine other aspects of timetable solution generators.
References


boundaries in weekly timetabling Cybernetics.


International Conference. Selected Papers, p. xiii+379.


60. Csima J and Gotlieb CC, 1964. **Tests on a computer method for constructing school timetables** Comm of ACM.


62. de Werra and Krarup, 1982. **Chromatic optimization limitations objectives uses references** EJOR.


Programming Study.


152. Lions J, 1966. **Matrix reduction using Hungarian method for the generation of school timetables.** Comm ACM.


156. Macon N and Walker EE, 1966. **A Monte Carlo algorithm for assigning students to class.** Comm ACM.


158. Makower M. S. and Williamson E., 1975. **Operational Research- Teach Yourself Books**


162. Maxfield, C., 1997. **Genetic algorithms: programs that boggle the mind.** EDN.


164. McIlhagga-M, 1997. **Solving generic scheduling problems with a distributed**


Atlanta, GA, USA; 1980; xii+354 pp.


179. Osterman R and DeWerra D, 1982. **Some experiments with a timetabling system.** OR spectrum.


Selected Papers, p. xiii+379. Springer-Verlag; Berlin, Germany,


196. Sampson-SE and Weiss-EN, 1995. Increasing service levels in conference and
The University Timetable Problem

K. Sandhu


230. Tybon, R & Harris, G. A Benchmark and an Intelligent Scheduling System for the Gowords Crossword Puzzle Problem, 1999 Proceedings of Workshop in AI’ 99 Conference UNSW


239. Wren-A , 1996. Scheduling, timetabling and rostering-a special relationship?


