

Application of Genetic Algorithms in Optimizing the Distribution of Examinations in an Examination Scheduling Problem

by

Yousef Moh'd Ahmad Kilani

Thesis submitted in fulfillment
of the requirements for the degree
of Master of Science

September 1996

PROCES
4558

ACKNOWLEDGEMENTS

First of all I would like to take this opportunity to convey my sincere thanks and deepest gratitude to my supervisor: **Dr. Ahamad Tajudin Khader**, who has introduced me to the fascinating and challenging world of Artificial Intelligence. I am very grateful for him and for all the help and valuable guidance, fruitful discussions, patience and continued encouragement provided to me at every stage of this thesis. I consider myself privileged to have had the opportunity to work under his guidance.

I am also very grateful to the external examiner **Professor Jerzy Korczak** from Université Louis Pasteur-France and for the internal examiner **Dr. Bahari Belaton**.

I would like to convey my appreciation and thanks to the Dean of the School of Computer Science **Prof. Zaharin Yusoff** and to the staff of the Artificial Intelligence, Unit Terjemahan Melalui Komputer (UTMK), Artificial Intelligence Lab, Institute of Postgraduate Studies, University Library and Universiti Sains Malaysia for providing assistance. Also, I wish to thank Dr. Quah Soon Hoe, and Prof. Kuan Kee-Sin from Statistics Department and special thanks to the system administrators: Zulkifli Shaari and Ayub Ismail.

I wish to thank my fellow graduate students in the Postgraduate lab for their encouragement and whole hearted support; thanks to Ayoub El-Feghieh, Ibrahim Al-Atoum, Luai Al-Shalabi, Feras Dabous, Khalid Al-Tahat, Shireen, Chandra, Vasoky, Saira, Seetha, Mosleh AL-adhaileh and Parimala.

Moreover, I wish to thank the members of football team, Prof. Ibrakhimov, Prof. Victor Malyshkin, Dr. khaild, Dr. Salam AL-Salheen, Harry, Dr. Ibrahim El-Yaseeri, Ali El-Badri, Yousef Taher, Ibrahim Al-Atoum, Luai Al-Shalabi, Ayoub El-Feghieh, Ahmad AL-Essa, Nassir Saeed and Hamza-Khreem.

Finally and most important of all, I would like to express my most sincere and warmest gratitude to my father, mother, brothers, sisters, nephews and nieces for their prayers, love, generous moral and financial support during my study.

All praises for the Almighty, without whose will everything would cease to be.

بسم الله الرحمن الرحيم

إلى هدا : :

إلى والدي و والدي، وإخواني
وإخواناتي أهدي هذا الجهد المتواضع.

يوسف محمد أحمد كيلاني

Table of Contents

Acknowledgments.....	ii
Dedication.....	iv
Table of contents.....	v
Abstrak (Bahasa Malaysia).....	ix
Abstract (English).....	x

CHAPTER 1:.....Scheduling Problems..... 1

1.1 Introduction.....	1
1.1.1 Job-Shop Scheduling.....	3
1.1.2 Open-Shop Scheduling.....	4
1.2 Timetabling.....	5
1.3 Examination Timetabling.....	7
1.4 Some Terminology	7
1.5 Scope and Objectives of Thesis	8
1.6 Summary of Chapters	8

CHAPTER 2:.....The State of the Art..... 10

2.1 Introduction.....	10
2.2 Heuristic Approaches.....	11
2.3 Graph Colouring Approaches	13
2.4 Mathematical Programming Approaches	14
2.5 Constraint Logic Programming Approaches.....	14
2.6 Simulated Annealing Approaches.....	15
2.7 Tabu Search Approaches.....	16
2.8 Practical Application.....	17

CHAPTER 3: Genetic Algorithms	19
3.1 Introduction	19
3.2 History	20
3.3 Features of Natural Evolution	21
3.4 Anatomy of Genetic Algorithm	22
3.4.1 The Evaluation Module	22
3.4.2 Population Module	23
3.4.3 Reproduction Module	24
3.4.3.1 Crossovers Operators	24
3.4.3.2 Mutation Operator	27
3.5 Genetic Algorithm in Action	27
3.6 Schemata	28
3.7 Applications of Genetic algorithm	32
3.8 The Use of Genetic Algorithms in Timetabling Problems	32

CHAPTER 4: Domain Analysis and Implementation Details	35
4.1 Introduction	35
4.2 Problem Description	35
4.3 Problem Representation	37
4.4 The life of Genetic Algorithm	39
4.5 Data Structure for an Individual	40
4.6 Steps of GA	41
4.6.1 The Initialization	42
4.6.2 Evaluation Function	44
4.6.2.1 Objective Function	44
4.6.2.1.1 Handling the First Type of Constraints	46
4.6.2.1.2 Handling the Second Type of Constraints	47

4.6.2.2 Fitness Function	56
4.6.3 Reproduction.....	57
4.6.3.1 Operators Selection	58
4.6.3.2 Parents Selection.....	59
4.6.3.2.1 Roulette Wheel Parent Selection.....	60
4.6.3.2.2 Parent Selection By Fitness Ratio	61
4.6.3.3 Genetic Operators.....	63
4.7 Exclusive Example.....	68
4.8 Summary of Chapter	69

CHAPTER 5: Experiments and

Discussions.....	74
5.1 Introduction.....	74
5.2 The Traditional Genetic Algorithm.....	75
5.3 The Algorithm GA1 with Crossovers.....	82
5.4 Convergence.....	85
5.5 The algorithm GA2 with Crossovers.....	89
5.6 Elitism.....	89
5.7 The Algorithm GA3 with Crossovers.....	91
5.8 Inserting Random-Initialized Chromosomes.....	94
5.9 Further Discussion About Fitness Function.....	94
5.10 Population Size.....	96
5.11 Further Discussion About Initialization.....	102
5.12 Comparisons with Constraints Satisfaction.....	104
5.13 Importance of Constraints.....	107
5.14 Further Discussion About Mutation.....	108
5.15 Summary of Chapter.....	117

CHAPTER 6:....Conclusions and Future Work	118
6.1 Introduction.....	118
6.2 Summary.....	118
6.3 Future Work.....	119
6.4 Conclusion.....	120
 Appendix: Input/Output.....	 129
A.1 Introduction.....	129
A.2 Data Processing.....	129
A.3 Constraints Entry and Representation.....	132

Aplikasi Algoritma Genetik Untuk Pengoptimuman Taburan Peperiksaan Dalam Masalah Penjadualan Peperiksaan

ABSTRAK

Penjadualan waktu peperiksaan bagi universiti dan institusi sudah lama diakui sebagai suatu masalah yang berkekangan tinggi. Kedua-dua jenis kekangan yang lemah dan kuat harus dipenuhi. Tambahan pula, peningkatan bilangan pelajar-pelajar dan kursus-kursus dari setahun ke setahun juga meninggikan kesukaran dalam penjadualan waktu peperiksaan.

Pelbagai pendekatan sudahpun digunakan untuk meneliti masalah penjadualan waktu peperiksaan ini. Pendekatan-pendekatan ini termasuk: pendekatan heuristik, pendekatan matematikal dan pendekatan gelintiran secara rawak. Di antara pendekatan gelintiran secara rawak, terdapat pula algoritma genetik. Projek ini menyelidik penggunaan algoritma genetik dalam penyelesaian masalah penjadualan waktu peperiksaan untuk Universiti Sains Malaysia. Algoritma yang diterangkan dalam tesis ini meningkatkan pemeriksaan ujian-ujian yang berturutan masanya. Tambahan pula, tesis ini menerangkan prestasi banyak algoritma genetik dan melihat pada kemungkinan yang berlainan untuk menunjukkan perlakuan dan prestasi algoritma genetik. Perbandingan dilakukan dengan cara semasa dan suatu pendekatan hibrid ditunjukkan untuk menjanakan keputusan yang lebih baik.

ABSTRACT

The scheduling of exams in universities and institutions is known to be a highly constrained problem. Soft and hard constraints need to be satisfied. The increasing number of students and courses offered from year to year increases the difficulty of scheduling the exams.

Different approaches have been used to tackle the exam timetabling problem. These approaches include: heuristic approaches, mathematical approaches and random search approaches. Among the random search approaches is the genetic algorithms. This project investigates the use of genetic algorithms on the problem of scheduling exams for the Universiti Sains Malaysia. The algorithm described in this thesis improves the checking of consecutive exams . Moreover, the thesis investigates the performance of many genetic algorithms and it looks at different possibilities to show the behavior and performance of the genetic algorithm. Comparison is done with the current method and a hybrid approach is shown to generate better result.

CHAPTER 1

Scheduling Problems

1.1 Introduction

Scheduling is how to assign a set of events to a set of resources within a limited time, with respect to various constraints; on the events, resources and time. Since these resources are limited, it is paramount that they are used efficiently.

Scheduling and planning are difficult problems with a long and varied history in the areas of operation research and artificial intelligence, and they are considered particularly challenging for both areas.

Scheduling and planning are difficult problems for two reasons, as described by [Davis, 1991] :

- They are computationally complex problems, described in computer science term as NP-complete*. This means that optimal solutions or sometimes near optimal solutions cannot be obtained within a polynomial time. Search techniques that deterministically and exhaustively search the whole space of possibilities will probably fail because of time requirements. In addition, search techniques that use heuristics to prune the search space will not be guaranteed to find an optimal (or even good) solution.

* a problem is NP-complete if no polynomial-time-bound algorithm has been found for that class of problem

- Scheduling problems are often domain-dependent, complicated by the details of that domain and cannot be easily adapted to solve any other scheduling problems.

The main objective in solving scheduling problems is to ensure that the resources are shared efficiently, so that all requirements and constraints are fulfilled within a limited time. There are many examples of scheduling problems such as :

- Sport activities scheduling [Schreuder, 1992],
- Meeting scheduling [Cadas, 1989],
- Airline gate assignment [Brazile and Swigger, 1988],
- Open-shop problem [Fang et al., 1994],
- Job-shop problem [Taillard, 1993],
- Traveling-salesman problem [Whitley et al., 1989],
- School timetabling [Khader, 1993],
- Course timetabling [Erben, 1995], and
- Exam timetabling [Fang, 1992].

These problems have been formulated by many different ways such as mathematical methods, graph theory and network-based methods. Different methods have been used to solve these problems, such as heuristics, mathematical programming approaches (such as integer linear programming), logic programming, knowledge based methods and evolutionary algorithms. The last three approaches are in the field of artificial intelligence.

In the following two subsections we are going to describe two scheduling problems: job-shop and open-shop, as examples of scheduling problems. We have chosen these problems since they are common in the scheduling field beside timetabling problems. After that, a general description of timetabling problems and a description of examination timetabling problems are outlined. The work of this thesis is in the area of scheduling problems using genetic algorithms, with an emphasis on timetabling problems; in

particular, exam timetabling.

1.1.1 Job-Shop Scheduling.

Job-shop scheduling problem (JSSP) is a common industrial and practical problem, known to be extremely hard [Fang et al., 1993] and the worst member of the class of NP-complete problems [Gray and Johnson, 1979]. The difficulty of the general JSSP makes it very hard for conventional search-based methods to find near-optimal solution in reasonable times.

In general, JSSP involves a collection of $\underline{m}$ machines and a collection of $\underline{J}$ jobs, each job comprises a set of operations (sometimes called tasks) which must each be done on a different machine for different specified processing time. An operation is an ordered pair $(\underline{a}, \underline{b})$ in which $\underline{a}$ is the machine on which the operation must be performed, and $\underline{b}$ is the time it will take to process this operation on machine $\underline{a}$. Table 1.1 shows a standard 6×6 benchmark problem ($\underline{J}=6$, $\underline{m}=6$) as taken from [Fang et al., 1993]. In this table, each job comprises 6 operations, for example, job 1 must go to machine 3 for 1 unit of processing time, second operation of the same job must go to machine 1 for 3 units of processing time, and so on. With the condition that the order in which the operations for every job are to be processed is important. A legal schedule must be created to satisfy the following conditions:

- Operations sequence on each machine such that each job's operation order is preserved.
- A machine does not process two different jobs at once.
- Different tasks of the same job are not simultaneously being processed on different machines.

The goal is to minimize the total elapsed time between the beginning of the first operation and the completion of the last operation.

	(m,t)	(m,t)	(m,t)	(m,t)	(m,t)	(m,t)
Job 1	3,1	1,3	2,6	4,7	6,3	5,6
Job 2	2,8	3,5	5,10	6,10	1,10	4,4
Job 3	3,5	4,4	6,8	1,9	2,1	5,7
Job 4	2,5	1,5	3,5	4,3	5,8	6,9
Job 5	3,9	2,3	5,5	6,4	1,3	4,1
Job 6	2,3	4,3	6,9	1,10	5,4	3,1

Table 1.1 : The 6×6 benchmark problem.

1.1.2 Open-Shop Scheduling

The open-shop scheduling problem (OSSP) is similar to the JSSP, with the exception that there is no a priori ordering on the tasks within a job. The main objective is the same as JSSP that is to generate a schedule with makespan as short as possible. The OSSP has considerably larger search space than the JSSP, and seems to be less heavily addressed in the literature. Table 1.2 is an example of 5×5 taken from [Fang et al., 1993].

machines	1	2	3	4	5
job 1	64	66	31	85	44
job 2	7	69	68	14	18
job 3	74	70	60	1	90
job 4	54	45	98	76	13
job 5	80	45	10	15	91

Table 1.2: A 5×5 benchmark OSSP.

In the above table, operation 1 of job 1 must go to machine 1 for 64 units of processing time, operation 2 of the same job must go to machine 2 for 66 processing time, and so on ,with no restrictions on the order in which the tasks for any job are to be processed. The goal is to generate a valid schedule with minimal makespan.

1.2 Timetabling

Timetabling problem is an example of scheduling problems. It is used in every area of human life, be it in watching television, following a daily timetable, doing a degree at a university, or just eating food. All of these may involve consulting some kind of timetable or schedule. Our interest in this research is on the kind of timetables that are concerned with arranging exams and courses for students in schools and universities. Henceforth, when the word **timetable** is mentioned, it means a timetable that is concerned with arranging exams and courses for students in schools or universities.

Timetabling problem is known to be NP-hard [Lajos, 1995]. It is not new and it has been a topic of research since the sixties. The progress in this area in the early years was hampered by the lack of computing power to handle large real problems, so much of the early work was theoretical or conceptual with limited implementation or testing, and it was only on problems of a small scale and limited realism. Work in timetabling has progressed along with advances in computers and even personal computers nowadays have the capacity to handle some limited timetabling programs. Most of the work has been in school timetabling and university timetabling. School timetabling is usually thought of as the most difficult of these, due to the compact nature of the required timetable; all the lessons must be scheduled within a specific range of time with little or no free time in the timetable for students [Khader, 1993].

There are two types of university timetabling:

- Course timetabling [Erben, 1995] and
- Examination timetabling [Fang, 1992].

These problems are related to each other but there are, in fact, many differences between them such as:

- There is usually only one exam in each subject while there are several lectures in a course.
- More than one exam may be held in each exam hall at any particular time. Whereas it would be extremely unlikely that any institution would allow two courses to take place in the same room.
- In a course schedule, clashes are not allowed, while in exam timetable it may happen*.

There are differences between course timetables for universities and course timetables for schools; such as no free time in course timetables for schools while in course timetables for universities it is allowed. Another difference is that, in a school it can usually be assumed that classes are roughly the same size and that all rooms in the school are sufficiently large enough to house any class, however, a university class can range from four or five students to many hundreds.

Even among schools, there may be differences between different levels in a school and there are differences among schools in different countries. The basic difference is between class-based and student-based timetabling [Khader, 1993]. In class-based timetabling, students are grouped into classes and the lessons are conducted on that basis, while in student-based timetabling, students have to choose lessons which they would like to attend within given guidelines.

* Some universities allow the exams clash and they postpone the exams for the effected students and some universities do not allow the exams clashes.

The focus of this thesis is on exam-based timetabling problem which is usually found in universities (the next section describes this).

1.3 Examination Timetabling

Most educational institutions must schedule a set of examinations at the end of each semester or year. Every year/semester a new timetable must be produced to take account of staff, student and course changes, causing a necessarily large amount of work. Examination timetabling can be defined as assigning a set of examinations to a fixed number of time periods so that no student must be in more than one place at once (no conflicting exams), and there must be sufficient seats to house all students present (no overfull rooms). No timetable is accepted if it violates these two fundamental constraints. Many other constraints may be considered desirable but are not essential, such as no student is to have exams in two adjacent periods. This may be highly desirable but may be ignored.

1.4 Some Terminology

The thesis uses some terminology which may need clarification. This is specially due to specialized use of some of these terms in the timetabling field. The following is list of such terms:

- A period usually refers to a unit of time where exams could be assigned.
- A time slot/slot usually refers to a specific interval of time where an exams can be assigned. Sometimes the word period is used to carry the same meaning as slot.
- The degree of an exams is the number of other exams that conflict with it.
- Conflict/clash : we say that two exams clash/conflict if these exams have at least one student in common, and they are scheduled at the same time slot.
- Exams in a row when there is no time slot empty between these exams. These

exams comprise at least one student in common. The last slot of any day and the first slot of the next day are considered in a row.

- Consecutive exams: the same as exams in a row. In this thesis, we mostly use the term 'consecutive exams' for two exams, and the term 'exams in a row' for more than two exams.
- Room size is the maximum number of students that a room can accommodate.
- Exam size is the number of students attending this exam.
- Slot total is the number of students having exams on this time slot.
- Candidates means the same as students.
- Overflow: the number of students who has no available seat to take their exams.

1.5 Scope and Objectives Of Thesis

The scope of the work covered in this thesis is aimed at investigating the use of genetic algorithms in exam timetable for Universiti Sains Malaysia (USM), and the use of different techniques to show the behavior and performance of the genetic algorithm. The exam timetable of USM is currently being done by using constraints satisfaction approach. The resultant timetable for USM still violates some constraints, like the students having exams in a row. So the genetic algorithm described in this thesis improves the resultant timetable by minimizing the number of constraints violations. All the experiments have been done by using real data from this university. Lastly, we shall show how much improvement has been achieved over the constraint satisfaction approach which is currently used to generate the exam timetable for USM.

1.6 Summary of Chapters

Chapter 1 gives an introduction to scheduling problems. It describes three examples of these problems: open-shop, job-shop and timetabling .

Chapter 2 gives a literature review of the approaches which were used and which are currently being used to tackle the exam timetabling problem.

Chapter 3 begins with a general description and history of the genetic algorithms, after that it describes the component of the genetic algorithm, and some general terms used in genetic algorithms. At the end of this chapter a brief description of the other work in exam timetabling by using genetic algorithm approach.

Chapter 4 provides the domain analysis of the exam timetable for USM. It introduces the constraints which are required by USM, and it shows how the genetic algorithm can tackle these constraints. Further more, this chapter shows the problem representation and the implementation details.

Chapter 5 illustrates the experiments, tests, and results for the algorithm described in chapter four. It looks at different possibilities to improve the performance of the GA. Performance of the different components are also investigated.

Chapter 6 gives the conclusion of our work and other work that we suggest to be done in the future.

CHAPTER 2

The State of the Art

2.1 Introduction

In this chapter, we are going to describe the early work in exam timetabling, the improvements which have been achieved and the approaches which are currently used to make the exam timetable for some universities. In the last section of this chapter is a survey in British universities in 1995 to show the computer usage in examination timetabling problem.

Different approaches are used to tackle the exam timetabling problem, these approaches include:

- Heuristic approaches [Laporate and Destroches, 1984],
- Graph coloring approaches [Burke, 1994a],
- Mathematical programming approaches [Arani et al., 1988],
- Constraint logic programming approaches [Azevedo and Barahona, 1994],
- Simulated annealing approaches [Thompson and Adownslan, 1995],
- Tabu search approaches [Boufflet and Negre, 1994], and
- Genetic algorithms [Fang, 1992].

In the following sections, we are going to describe briefly about these approaches. The use of the genetic algorithms and constraint logic programming is fairly recent.

2.2 Heuristic Approaches

Much early work was an attempt to imitate the manual timetabling procedure, using heuristics methods which are still present in timetabling work. A heuristic method is a form of "guided trial and error" and is a search technique used to prune the search space of possibilities. Since timetabling problems are combinatorial in nature, have too many possibilities to be evaluated which takes an unacceptable amount of time to find good solutions. If there are e exams to be scheduled on t time slots there are t^e possible candidate timetables, which vary in optimality, for example, if there are 25 time slots and 500 exams, there are 25^{500} candidate timetables. So looking for a good timetable in this space is like looking for a needle in a haystack. Because of this, the heuristic methods have been used to avoid passing all this space.

Heuristics have long been used in timetabling. One of the earliest published examples of exam timetabling was presented by [Broder, 1964]. His stated objective was to minimize the number of student conflicts. His algorithm was basically a "largest degree first" method; each course in the list is randomly assigned to one of the time periods that creates fewest number of conflicts. The algorithm was run several times with a different random selection. The best run was selected. He observed that his algorithms easily adapted to minimize the number of times a student has two final examinations in the same day, three final examinations in a row, three final examinations in two days, and four final examinations in three days[Carter, 1986].

Cole [Cole, 1964] published an algorithm that had been implemented successfully at Leicester University in England. For each period, examinations requested to satisfy any consecutive exams constraints are scheduled first (e.g. if one of consecutive examinations is already scheduled in the previous period). Then the algorithm proceeded by using the "largest degree first" rule, except that examinations are bypassed if they violate any of the constraints. Cole's algorithm introduced the following constraints:

- (i) Consecutive exams.
- (ii) Precedence ordering for some exams (e.g. exam e must come before i exam).
- (iii) Space constraints on room size.
- (iv) Certain examinations to be scheduled only in the morning.

The algorithm that had been implemented successfully at Leicester University in England in 1964 (the data was of 34 subjects requiring 57 examination periods because some examinations require two periods. Each subject conflicts with an average of 11 other courses). Cole [Cole,1964] used a small-store computer, with the emphasis on economy of storage. A binary matrix was used to indicate whether pairs of subjects conflicts. Element (i,j) is "1" if courses i and j have any conflicting and "0" otherwise.

Wood [Wood, 1968] published an algorithm that had been implemented at the university of Manchester in England. Wood [Wood, 1968] primary concern was that examinations had to be scheduled into a set of designated rooms with limited capacities. So, he sorted the courses according to the size of room required. Within each group size, he was choosing each course -according to the "largest degree first" rule - to search for a feasible period with:

- (i) no adjacent conflict (consecutive exams), or, if none;
- (ii) no conflict on the same day (2 exams on the same day), or, if none;
- (iii) the least number of students taking another examination on the same day.

Having selected a period for a particular subject, the next step is to choose a room. This is done by selecting the room with the least number of vacant seats above the number required (e.g. if rooms with 100, 68 and 30 seats are available a subject with 65 candidates is put in the 68 seat room).

IDesroches et al [Desroches et al., 1978] presented the algorithm HOREX and they solved the exams in a row constraints by using a branch and bound code for integer linear programming. White and Chan [White and Chan, 1979] published an algorithm similar to

HOREX, that had been in use at the university of Ottawa for several years. Work of the similar nature had been followed by [Laporate and Destroches, 1984].

2.3 Graph Colouring Approaches

A graph is a convenient way to represent timetabling problems. Welsh and Powell [Welsh and Powell, 1967] pointed out the similarities between timetabling problem and the one of colouring the vertices of a graph. Here, the vertices are taken to be equivalent to examinations, and the arcs between them represent conflicts. The degree of a vertex is the number of edges incident with that vertex, colouring the graph amounts to placing examinations in appropriate periods. Two vertices are said to be adjacent if they have a common edge.

A partition of a graph is the division of the set of vertices into subsets such that no two vertices in the same subset are adjacent. And this can be done by colouring the vertices such that no two adjacent vertices are colouring the same colour. Many algorithms for graph colouring have been found such as [Wood, 1968], [Dempster, 1969], [Brelaz, 1979] and [Burke, 1994a].

The main disadvantage of the graph colouring approach is the difficulty and sophistication required for incorporating practical constraints into the problem formulation [Lajos, 1995].

The minimum number of colours required to colour a graph so that no two adjacent vertices are coloured the same is known as the chromatic number of that graph. Welsh and Powell [Welsh and Powell, 1967] gave an algorithm for colouring a graph, in which they found out a bound for the number of colours needed.

2.4 Mathematical Programming Approaches

Timetabling problems may be formulated using mathematical programming models [Schmidt and Strohlein, 1979]. There are several algorithms for mathematical programming. The major drawback of this approach is that the number of variables and constraints becomes unmanageably large for practical problems [Kang and White, 1992].

Mathematical programming such as integer (linear) programming, is used in linear programming problems where some or all variables are integer rather than continuous. [Arani et al, 1988] presented an integer programming formulation for the second phase (consecutive conflicted exams) of exam timetabling problem and then they solved the problem by using a lagrangian relaxation approach.

Both graph colouring algorithms and mathematical programming are approaches from the operation research area.

2.5 Constraint Logic programming Approaches

Constraint logic programming (CLP) is an extension to logic programming, which is a framework that enhances logic programming with constraint satisfaction techniques. It aims at expressing constrained decision problems declaratively and still solving them efficiently [Azevedo and Barahona, 1994].

CLP program is a set of clauses, as in logic programming, but each may now contain a constraints body. The satisfiability of these constraints is then checked at each computation step. CLP supports a methodology for formulating combinatorial problems in terms of variables, their domain and constraints that govern the assignment of values to variables drawn from their explicitly stated domains.

Recently, CLP for examination timetabling problems by using the finite domains of CHIP [Dincbas et al., 1988] (which is the most widely used CLP language) has been used at the West Catholic University in Angres [Boizumault et al., 1995].

2.6 Simulated Annealing Approaches

Khader [Khader, 1993] defined annealing as:

"Annealing is a physical process of heating a substance and then cooling it gradually. The cooling needs to be gradual and to allow all the atoms in the substances to settle in their proper place. If the cooling is too fast, the atoms may not have found the proper place and end up frozen at an unsuitable place. "

In an exam timetabling case at Swansea University reported by [Thompson and Adownsland, 1995], the timetabling problem was solved in two phases. In the first phase, a feasible solution which satisfies all the binding constraints was achieved (binding constraints which involved, no clashes, certain pairs of exams to be scheduled at different times, certain groups of exams to be scheduled in order, no more than 1200 student to be involved in any one session). Phase two searches the space of feasible solutions (after obtaining all the binding constraints) with the aim of optimizing the secondary objectives constraints (secondary objectives constraints involved minimizing the number of exams with over 100 student scheduled after the tenth period, minimizing the number of consecutive conflicted exam). They implemented the model by using simulated annealing solution to graph-colouring problem; they define the solution space as the set of partitions of vertices into k colours, and a **neighborhood move** involves changing the colour of a single vertex. The cost is given by the number of edges between vertices in the same colour. They penalized the other constraints in the cost function. An initial temperature and cooling rate for the temperature is set. The temperature is used to

control the probability of accepting an increase in cost . Once a zero cost solution has been found in phase one, phase two commences from the final solution of phase one and proceeds in the same way as in phase one. Neighborhood move involves swapping groups of vertices in two colours classes, which insures that the colouring remains feasible. The cost function was computed as in phase one.

2.7 Tabu Search Approaches

Tabu search (TS) was initially suggested by [Glover, 1985] for specific applications and later developed into a more general framework. TS is one of the most efficient heuristics for handling large optimization problems [Hertz, 1991]. A general description of the method has been given by [Glover, 1989] and [Glover, 1990].

TS basically moves step by step from an initial solution of a combinatorial optimization problems towards a solution which is expected to be optimal or near-optimal. The search process starts with an initial feasible solution S , and the definition of a neighborhood $V(S)$ is defined which consists of solutions reachable in one step from S . The basic step is to move from the current Solution S to the best solution S^* of $V(S)$, even if it is not better than S . A tabu list T stores descriptions of the last moves which were used to avoid cycling. When exploring $V(S)$ to find S^* , T is scanned to avoid the so-called tabu moves which could bring the search back to previous iteration. This iterative process stops after a maximum number of iterations has been completed and it outputs the best solution found. Boufflet and Negre [Boufflet and Negre, 1994] implemented such a method for examination planning. They represented the problem by graph coloring and they solved it by using a TS. In which each vertex is given one of the P allowed colors. The idea is to minimize the conflicts by exploring the neighborhood of the initial solution.

2.8 Practical Application

A good survey of practical applications of examination timetabling algorithms from the early work up to 1986 was written by [Carter, 1986]. This section, is a survey of examination timetabling in British universities which had been done recently by [Burke et al., 1995]. It shows the output of the following question:

How widespread is computer aided timetabling?

The survey had covered 59% of the whole universities in Britain, 56 out of 95 universities (56 universities replied to the survey).

Fifty eight percent of those universities that responded use a computer at some stage in the timetabling process. Of these, 21% (11 universities) have the facility to perform scheduling, although these may, in some cases, still require a certain amount of manual input and previous knowledge of the particular timetabling problem. Two of the eleven scheduling systems are commercial and the rest have been developed in house.

Four universities stated that they either currently developing their own or customising a commercial package.

Two timetabling approaches were given without scheduling aid. The first approach is to take draft examination timetables as produced by each school or department and then merge them centrally into a final timetable. This uses the assumption of independence between departments. Many universities are now heading towards modular schemes which make such an assumption very tenuous. The second approach is to tweak the teaching timetable.

Burke et al. [Burke et al., 1995] says:

" Given the size of the timetables involved it is surprising that so many do not use any form of automation at all. This may in some cases, be because the timetable does not change significantly from year to year. However, of those universities that do not schedule by computer, the split between those that use last year's timetable and those that construct a complete new timetable each year is exactly even. It is interesting to note, however, that in every case, those that do not use the pervious year's or a computer take at least four weeks to produce a timetable. Where a scheduling system is used, virtually all (10/11) take no account of the pervious year's timetable"

In this chapter we gave a survey about the methods which had/have been used to tackle the exam timetabling problems. In the next chapter we are going to describe the method we used "**the genetic algorithms**" in details and what have been done and what we are going to do.

CHAPTER 3

Genetic Algorithms

3.1 Introduction

The best description of genetic algorithms I have ever found has been given by [Goldberg, 1989]:

"Genetic algorithms are search algorithms based on the mechanics of natural selection and natural genetics. They combine "Survival of the fittest" among string structures with a structure yet randomize information exchange to form a search algorithm with some of the innovative flair of human search. In every generation, a new set of artificial creatures (strings) is created using bits and pieces of the fittest of the old; an occasional new part is tried for good measure. While randomized, genetic algorithms are no simple random walk. They efficiently exploit historical information to speculate on new search points with expected improved performance."

GAs were invented to mimic some of the processes observed in natural evolution. Evolution takes place on the chromosomes which are the organic devices for encoding the structure of living beings. Grefenstette [Grefenstette, 1993] described the work of GAs:

"Genetic algorithms solve problems by using principles inspired by natural population genetics: they maintain a population of knowledge structures that represent candidate solutions, and then let that population evolve over time through competition and controlled variation. Genetic algorithms are being applied to a wide range of optimization and learning problems in many domains".

GAs mimic the process of natural selection and can be used as a technique for solving complex optimization problems which have very large search spaces [Abramson and Abela, 1992]. Unlike many schemes which only have one sub-optimal solution at any time, GA maintains many individual solutions in the form of a population.

Evolution in nature is not a purposive or directed process. That is, there is no evidence to support the assertion that the goal of evolution is to produce mankind. Indeed, the processes of nature seem to boil down to different individuals competing for a limited resources in the environment. Some are better than others. Those which are better, are more likely to survive and propagate their genetic material.

3.2 History

Evolutionary algorithms (EA) are based upon biological observations, that dates back to Charles Darwin's discoveries in the 19th century; the means of natural selection and survival of the fittest, e. g. the "Theory of evolution". The inspired algorithms are thus termed EA. The features of natural evolution intrigued John Holland in the early 1970's. Holland believed that, appropriately incorporated in a computer algorithm, they might yield a technique for solving difficult problems in the way that nature has done through evolution, so he began work on algorithms that manipulated strings of binary digits-1's and 0's-that he called chromosomes. Like nature, his algorithms solved the problem of finding good chromosomes by manipulating the material in the chromosomes blindly. Holland [Holland, 1975] published the book "adaptation in natural and artificial systems". Beginning in 1985, the general approach began to receive wide attention. Two formal GA conferences "International conference on genetic algorithms and their applications" are held every two years from 1985, and the conference of "Foundations of genetic algorithms and classifier systems" are held every two years from 1990. There are also many papers published in machine learning and neural network.

3.3 Features of Natural Evolution

The simulation of evolution allows the principle of survival of the fittest to be applied to optimization problems, where the goal is to find an individual which is very fit.

The processes of chromosome encoding and decoding are not fully understood, but there are some general features of the theory that are widely accepted [Davis, 1991]:

- Evaluation is a process that operates on chromosomes rather than on the living beings they encode.
- **Natural selection** is the link between chromosomes and the performance of their decoded structures.
- The process of reproduction is the point at which evolution takes place.
- Biological evolution has no memory. Whatever it knows about producing individuals will function well in their environment is contained in the gene pool and structure of the chromosome decoders.

In natural selection, individuals are born (children) with various attributes of their parents, and therefore inherit properties from their parents. These attributes are indicated by particular genes, which are combined into chromosomes. If the combined attributes of a child make it more suited to the environment, then the probability that the child will survive, and breed, is increased. Consequently, the desirable attributes of the child will be passed on to future generations. However, if a child is born with undesirable attributes, which make it less able to handle the environment, then it is likely that the child will die before reproducing, and the less desirable attributes will be removed from future generations. The net effect of survival of the fittest is that the average fitness of the population increases with each generation.

3.4 Anatomy of Genetic Algorithm

There are many components of GA. These components are collected into three modules (classified in this way by [Davis, 1991]). Figure 3.1 shows a general components found in most GAs. Let us see the working of the three modules in detail.

Evaluation-Module: Evaluation function.
Population-module: Representation techniques. Initialization techniques. Reproduction techniques. Deletion techniques. Parent selection techniques. population members.
Reproduction module: operator selection technique: Mutation opertator. Crossover operator.

Figure 3.1: modules of GA.

3.4.1 The Evaluation Module

The evaluation module contains an **evaluation function** that measures the worth of any chromosome on the problem to be solved. The **evaluation function** is the link between the GA and the problem to be solved. So it takes a chromosome as input and returns a number that is a measure of the chromosome's performance on the problem to be solved. Each member (chromosome) of the population is evaluated and assigned a

measure of its fitness as a solution.

The **evaluation function** comprises two functions: **objective function** and **fitness function**. The **objective function** gives a number for every member, which is a measure of the rules/constraints (in the environment or in the problem to be solved) violations. The **fitness function** provides a fitness value for every member based on this number. This fitness is inversely proportional to the constraints/rules violations of each chromosome.

The **evaluation function** plays the same role in GAs that the environment plays in natural evolution. The interaction of an individual with its environment provides a measure of its fitness, and the interaction of a chromosome with an evaluation function provides a measure of fitness that the GA uses when carrying out reproduction.

3.4.2 Population Module

The population module contains a set of candidate solutions to the problem being tackled and techniques for creating and manipulating them. Each candidate solution is called individual or member. Each individual/member contains a chromosome, positions in the chromosomes are called genes and the value a gene takes is called an allele. In this thesis, whenever one of the words member, individual or chromosome is mentioned, it means the candidate solution to the task being tackled.

In the **representation technique**, the solutions to the problem should be represented as chromosomes. The technique for encoding solutions may vary from GA to GA and no one technique works best for all problems. The early work in GAs used binary representation, in which the chromosomes are strings of binary digits.

The **initialization technique** initializes the starting population of the GAs. Some GAs create its starting population by filling it with random solutions, and others by taking the results of another methods to the same problem.

The **reproduction technique** replaces the old population by the new one. Each replacement of the old population by the new one is called a generation. The **deletion technique** decides which members should be deleted and replaced by the new children.

The **parent selection technique** selects parents from the current population. The purpose of parent selection in GAs is to give more reproduction chances, on the whole, to those population members that are the most fit.

The population of GAs comprises many candidate solutions to the problem being tackled. The number of population members (candidate solutions) is called population size. The size of the population should be greater than one.

3.4.3 Reproduction Module

The reproduction module includes the operators (sometimes they are called "search operators" or "genetic operators") for creating children that are different from their parents. In the field of GAs, there are two kind of operators crossover (or recombination) and mutation. One important feature of the operators is that it can produce children that are different from their parents.

3.4.3.1 Crossovers Operators

Crossover is an extremely important component of a GA. Crossover occupies a special place in the hearts of GA practitioners. In nature, crossover occurs when two parents