Final Exams Scheduling for Universities

by

Mazen Nuhad Timany

**Submitted in partial fulfillement of the requirements
for the degree of Masters of Science**

**Thesis Advisors : Dr. Nash'at Mansour
Dr. Ramzi Haraty
Dr. George Nasr**

**Computer Science Program
Lebanese American University
July 2001**

Lebanese American University

Graduate Studies

We hereby approve the thesis of:

Mazen Nuhad Timany

Candidate for the Master of Science Degree*.

Signed by: Signatures Redacted

Dr. Nasha't Mansour: ______________________________
Associate Professor of computer Science (Chair)

Signatures Redacted

Dr. Ramzi Haraty: ______________________________
Assistant Professor of Computer Science

Dr. George Nasr: Signatures Redacted ______
Associate Professor of Engineering

Date: ______________________

*We also certify that written approval has been obtained for any proprietary material contained therein.

To God.

To all those having Dukkha.

To all those leading the path for the cessation of Dukkha.

To students of Beir Zeit University.

Acknowledgement

This thesis was accomplished with the great help, support and valuable suggestions from Dr. Nash'at Mansour. Faithful and indebted thanks go to the second advisers, Dr. Ramzi Haraty and Dr. George Nasr.

Sincere and profound gratitude is to Mr. Ziad Abou Tarieh, Mr. Mohamed Awad, Mr. Ziad Mikati, Mr. Abbass Tarhini and Mr. Nazih Faraj. Their aid, encouragement, cooperation and suggestions were of enormous benefit in undertaking this thesis.

Special credit and deep appreciation also goes to Dr. Bassem Salha, Mr. Robert Kfoury and Mr. Jihad Njeim who assisted me indirectly in one way or another.

Abstract

Scheduling final exams for large numbers of courses and students in universities, such as the Lebanese American University (LAU), is an intractable problem. In order to solve this problem, the approach must be efficient, flexible and adaptable. Conflicts occur when multiple exams are scheduled for the same student at the same period (simultaneously), and unfairness to a student refers to consecutive exams (two exams directly after each other) or more than two exams on the same day (referred to as multiples). A good exam schedule would aim for minimizing conflicts and the two unfairness factors based on user-assigned weights associated to these three factors and subject them to some constraints. Likewise, since a limited number of rooms are available in each exam period, an additional constraint concerned with room violations is added to achieve the goal of minimizing room violations. All constraints may be violated if necessary, since it is almost impossible in real world situations to find a solution without violating any constraint.

In this work, we first formulate the problem as a modified weighted-graph coloring problem and adapt two natural optimization algorithms: Simulated Annealing and Genetic Algorithm; in addition to a clustering based algorithm (FESP), and a hybrid of natural optimization and clustering based algorithms (FESPSA) for solving the exam scheduling problem taking into account the specific objectives and constraints of LAU. Then, we compare these algorithms with each other as well as with the manual procedure done by the registrar's office. The comparison is done using realistic data taken from LAU for six semesters.

Our experimental results show that simulated annealing gives better exam schedules than genetic algorithms, FESPSA, FESP and manual scheduling. All algorithms were run on different exam days ranging from five to ten. Simulated Annealing stayed to show the best results in all semesters in all days variations. Moreover, Simulated Annealing shares with the Genetic Algorithm more flexibility to accommodate various user constraints. On the other hand, FESPSA showed better results in terms of conflicts and unfairness factors among all exam days when compared with FESP. Moreover, FESPSA also proved to be better than FESP when dealing with room violations. That is, FESPSA minimized the number room violations much better than FESP.

Keywords: Exam scheduling, timetabling, constraint programming, simulated annealing, genetic algorithms, optimization heuristics.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Where scheduling of final exams for large numbers of courses and students in liberal arts universities, such as the Lebanese American University (LAU), is done manually by the Registrar's Office, a lot of students make a large number of complaints about conflicts or unfairness in their exams schedule. Conflicts occur where simultaneous exams are scheduled for the same student, and unfairness to a student refers to consecutive exams or more than two exams on the same day. A good exam schedule at LAU would aim for minimizing conflicts and the two unfairness factors based on user-assigned weights associated to these three factors and subject them to some constraints within a given and predefined number of days. Moreover, a good exam schedule would take into consideration minimizing the number of room violations since a predefined and fixed number of rooms (i.e. seats) are available in a period.

Several distinct approaches to the solution of this problem have been developed. Some has approached the solution using graph coloring [Balakrishnan, 1991; Carter, Laporte and Chinneck, 1994; Wood, 1968; Erben, 2000; Mehta, 1981, Balakrishnan 1992], extended clique [Carter and Johnson, 2001], integer programming [Arani, Karwan and Lotfi, 1988; Descroshes, Laporte, and Rousseau, 1978], clustering algorithms [Leong and Yeong, 1987; Lotfi and Cerveny 1991], non-classical and natural optimization algorithms [Hertz, 1991; Ergul, 1996; Thomson and Dowsland, 1996; Tarhini and Mansour, 1998; Corne, Fang and Mellish 1993], etc. Pure genetic algorithms for graph coloring are in general outperformed by more conventional methods. Therefore, an alternative solution based on the grouping character of the graph coloring problem was chosen [Erben 2000]. Furthermore, a fitness function defined on the set of partitioned vertices, competently guiding the grouping genetic algorithm, was developed.

Other approaches have developed their solutions to exam timetabling based on Tabu Search [Di Gaspero and Schaerf, 2000; Hertz, 1991; White and Xie, 2000]. Constraints of the problem were classified or divided into "soft" and "hard" constraints [Burke, Bykov and Petrovic, 2000; Di Gaspero and Schaerf, 2000]. Soft constraints were included in the objective function that measures the quality of the solution by evaluating the unfairness factors (consecutives and multiples). These soft constraints are subject to violations in their way to optimal solutions. Conversely, "hard" constraints (simultaneous exams) are less likely to be violated and they have larger weighting factors than soft ones. On the other hand, Tabu relaxation technique can accelerate downhill movement, that is, worsen the optimal solution. Thus, a four phase system using an automated Tabu search has been implemented to the exam scheduling problem such that a frequency based longer term memory mechanism combined with Tabu relaxation technique is used to optimize the examination timetabling problem [White and Xie, 2000]. The longer term Tabu technique diversifies search space efficiently, prevents cycling and consequently produces better results. Furthermore, a multi-criteria approach has been achieved where a number of criteria are defined for the "soft" constraints [Burke, Bykov and Petrovic, 2000]. In the first phase of the multi-criteria approach the goal is to find a high quality sub-optimal solution for each criterion alone. Then in the second (final phase), trade offs among all criteria are performed to reach a compromised final solution with respect to all criteria simultaneously.

One of the recent major directions for the research on the exams timetabling problem seems to be constraint logic programming. Constraint logic programming is capable of handling university timetabling by providing several libraries of constraints solvers. These libraries handle set constraints using set variables ranging over finite set domains specified by lower and upper bounds for set inclusion [Reis and Oliveria, 2000; Lajos, 1995]. University timetabling has been implemented after being modeled as a constraint satisfaction problem using constraint based reasoning that combines logic programming and constraint solving technique based on an arc consistency algorithm [Deris, Omatu and Ohta, 2000]. A few commercial software programs have emerged providing aid ranging from identifying conflicts to full automation of exam timetabling [McCollum and Newall, 2000; Beynon, Ward and Maad 2000; OPTIME; Schedulexpert]. Some of them are even capable of producing individual timetables for each student. These software programs are now looking forward to integrate their system to the web to reduce stationary and to provide easier access to students and teachers [OPTIME]. We note that each of these solution procedures, and in particular the natural optimization algorithms, has dealt with specific objectives and constraints of the exam-scheduling problem.

In this work, we first formulate the problem as a modified weighted-graph coloring problem. Important to mention, each color in the graph is considered to be a period. Then, we adapt two natural optimization algorithms, a simulated annealing algorithm and a genetic algorithm for solving the exam scheduling problem taking into account the specific objectives and constraints of LAU. The simulated annealing algorithm solves the three types of conflicts in addition to room violations by allocating to each of them a weight to be multiplied with. The summations of the resultant multiplications build up the objective function to be minimized in order to produce a good exams schedule. On the other hand, based on a mathematical model presented by Lotfi and Cerveny [Lotfi and Cerveny, 1991] FESP was introduced. The model consists of several mathematical formulations for minimizing the three types of unfairness conflicts. Each formulation is associated with a heuristic algorithm to solve it. But, major variations from Lotfi and Coreney's solutions have taken place in FESP. In FESPSA simultaneous exams are solved using FESP [Mikati, 1999; Lotfi and Cerveny, 1991] algorithm, whereas the unfairness factors and room violations are solved using simulated annealing [Kirkpatrick, Gelatt and Vecchi, 1983] after subjecting them to constraints in an objective function. FESPSA, combination of simulated annealing and FESP, simultaneous exam conflicts were solved alone by the same way as FESP. Whereas, the other constraints namely, consecutives and multiples were solved using simulated annealing. Unlike the simulated annealing code where all three types of conflicts and room violations were solved concurrently, and also unlike FESP, where each type of conflict is solved alone, after the solution is produced, room violations were solved alone using simulated annealing. Further, we compare the two natural algorithms with each other and with a good clustering-based heuristic solution procedure FESP, FESPSA, as well as a manual procedure. The comparison is done using realistic LAU's data from six semesters. Additionally, comparison is done while altering the number of exam periods from twenty (5 days; each day 4 periods) till forty (10 days).

The rest of the thesis is organized as follows. Chapter 2 specifies the problem and gives the modified weighted-graph coloring problem formulation. Chapters 3 and 4 present the simulated annealing and genetic algorithm. Chapters 5 and 6 explain FESP and modifications done on FESP to get to the FESPSA algorithm. The experimental results are presented and discussed in Chapter 7. Chapter 8 contains the conclusion.

Chapter 2

Exam Scheduling Problem and its Graph Formulation

The main concern in the exam scheduling problem is to deduce a schedule that assigns exams to exam periods within classrooms in a manner that minimizes conflicts and unfairness facts.

2.1 Exam Scheduling Problem

Given that A exams are to be taken by students over B days, where E exam periods can be done per day, the exam scheduling problem consists of assigning A exams to Π $(B*E)$ exam periods, within specified classrooms. The objective is to minimize the conflict and the unfairness factors, which are:

i) The number of students with simultaneous exams,

ii) The number of students with consecutive exams,

iii) The number of students with two or more exams on the same day, and

iv) The number of rooms violated.

In this work, we assume the following conditions and constraints that apply at LAU:

i) The user should be provided with the flexibility of assigning weights to the four conflict and unfairness factors.

ii) A limited predefined number of exams A.

iii) The number of exam periods, Π, is predefined; Π may be varied.

iv) A limited predefined number of classrooms, R, are available for exams.

v) Room capacity is taken into consideration in assigning exams to rooms. Also, more than one exam/section can be assigned to the same room at the same time if they fit.

vi) The last period of one day is considered to be consecutive to the first period of the next day.

2.2 Modified Graph Coloring Problem Formulation

Scheduling problems can be represented by graphs [Wood, 1968]. Let $G(V, E)$ be a graph in which: vertex $v_i \in V$ represents an exam to be scheduled; vertex weight w_i represents the number of students taking exam v_i; edge $e \in E$ joining two vertices v_i and v_j represents the existence of students taking both exams v_i and v_j; weight of edge e, w_{ij}, represents the number of students taking both exams v_i and v_j. The vertex weight is used to match a room's capacity.

The exam scheduling problem can be expressed as a modified graph coloring problem, where we color the vertices of a graph using a specified maximum number of colors (exam periods), Π, such that the objective function OF1 (Equation 1) is minimized and the constraints (listed in Section 2) are met. A solution to the exam-scheduling problem is henceforth denoted as the configuration C. Note that each color corresponds to an exam period and all vertices having the same color represent the exams that can be assigned to the same period.

Let $c(v)$ be the color of vertex v, and $\xi = \{ c_1, c_2, \ldots c_\pi \}$ be the set of ordered, available colors; that is, $|\xi| = \Pi =$ maximum number of available colors, and $(c_i - c_{i-1}) = 1$ for $I=2, \ldots, \Pi$. The objective function, OF_1, is given in terms of the following factors:

(i) S_{SE}, the total number of students taking conflicting simultaneous exams $= \Sigma w_{ij}$ with $c(i) = c(j)$.

(ii) S_{CE}, the total number of students taking consecutive exams $= \Sigma w_{ij}$ with $|c(i) - c(j)| = 1$.

(iii) S_{ME}, the total number of students taking two or more exams per day $= \Sigma w_{ij}$ with $c(i)$ and $c(j)$ referring to exam periods on the same day.

Specifically,

$$OF1 = \alpha * S_{SE} + \varphi * S_{CE} + \sigma * S_{ME} + \gamma * (\Sigma_{1<=k<=\Pi} \; \Sigma_{1<=I<=R} \; \rho_{ik}) \qquad (1)$$

where α, φ, σ and γ are user-defined weights for simultaneous exams, consecutive exams, multiple exams, and room violations respectively; ρ_{ik} tells if room i is violated in period (k) i.e. if room i was assigned larger number of students than Ψ_i which is the capacity of room i. The inner summation in $(\Sigma_{1<=k<=\Pi} \Sigma_{1<=i<=R} \rho_{ik})$ refers to the total number of rooms violated in a period, whereas, the outer summation refers to the total number of rooms violated in all periods. The useful symbols used are summarized in Table 2.1.

Note that different sets of weights for OF1 are used in this work, which are shown in Table 7.2. These weights are important in producing the exam schedule. They might be contradictory; that is, by increasing one of those weights, say α, the solution will improve in minimizing one kind of conflict (S_{SE}) where it might increase the other factors. These weights will allow flexibility in using the solution procedures to suit the user's particular choices or requirements for different instances of the problem.

Table 2.1 Summary of useful symbols

Symbol	*Meaning*
A	The number of exams taken by students in a semester that are to be scheduled.
B	The number of exam days.
E	The number of exam periods per day.
Π	The maximum number of available exam periods (maximum number of colors).
R	The maximum number of available classrooms.
ξ	The set of available exam periods (available colors); $\lvert\xi\rvert = \Pi$
c_j	An available color in ξ (i.e exam period).
$C(i)$	The period to which exam I is assigned (The color of vertex I).
C	The system configuration, i.e exams schedule.
S_{SE}	The total number of students having conflicting simultaneous exams.
S_{CE}	The total number of students having conflicting consecutive exams.
S_{ME}	The total number of students having two or more exams per day.
ρ_{ik}	Tells if room is violated at period k.
Ψ_i	Capacity of room I; the number of seats in room i.
α	A weighting factor related to the importance of S_{SE} in OF1
φ	A weighting factor related to the importance of S_{CE} in OF1
σ	A weighting factor related to the importance of S_{ME} in OF1
γ	A weighting factor related to the importance of $(\sum\rho_{ik}\ _{1<=k<=\Pi})$ in OF1
w_i	The number of students taking exam i.
W_{ij}	The number of students participating in both exams I and j.

Chapter 3

Simulated Annealing Algorithm

Simulated annealing is based on ideas from statistical mechanics and is motivated by an analogy to the physical annealing of a solid [Kirkpatrick, Gelatt and Vecchi, 1983]. To coerce some material into a low-energy state, it is heated and then cooled very slowly, allowing it to come to thermal equilibrium at each temperature. At each fixed temperature in the cooling schedule, the Metropolis algorithm can simulate the behavior of the system. An iteration of the Metropolis algorithm starts with proposing a random perturbation and evaluating the resultant change in the energy system. If the change is negative, corresponding to a downhill move in the energy landscape, the perturbation is accepted and the new lower energy configuration becomes the starting point for the next perturbation. Zero change is also accepted. If the energy change is positive, corresponding to an uphill move, the proposed perturbation may be accepted with a temperature-dependent probability. The main advantage of this Monte Carlo algorithm is that the controlled uphill moves can prevent the system from being prematurely trapped in a bad local minimum-energy state.

The simulated annealing algorithm (SA) simulates the natural phenomenon by a search (perturbations) process in the solution space (energy landscape) optimizing some cost function (energy). It starts with some initial solution at a high (artificial) temperature and then reduces the temperature gradually to a freezing point. At each temperature, regions in the solution space are searched by the Metropolis algorithm.

In the following subsections, we describe how simulated annealing is adapted for solving the exam scheduling problem; an outline of the SA algorithm is given in Figure 3.1.

```
Initial configuration = Random color assignment;
Determine initial temperature T(0);
Determine freezing temperature $T_f$ ;
save_best_sofar();
while ($T(i) > T_f$) do
 repeat $\Pi * A$ times
    Generate_mechanism i;
 save_best_sofar();
 $T(i+1) = \theta * T(i)$;
endwhile

procedure Generate_mechanism i
perturbi;
if ($\Delta OF1 \leq 0$ ) then
 updatei
else
 if (random() < $e^{-\Delta OF1 / T(i)}$) then
  updatei
 else
  reject_purturbation();
```

Figure 3.1 Outline of the SA algorithm for the exam-scheduling problem.

3.1 Solution representation and energy function

The system to be coerced into a low-energy state in the exam-scheduling problem is represented by the configuration *C*, which is an array of linked lists. The size of the array is equal to *A* and the nodes of the linked lists represent the sections of their corresponding course to be scheduled. That is, the nodes of the linked list in an array entry are graph vertices colored with the same color c_i, where c_i is the color of the first section in the course list. The system energy is given by OF1 (Equation 1). In the annealing process, *C* goes through many changes until it reaches an optimal or sub optimal configuration at freezing temperature. However, the initial configuration is constructed by randomly assigning graph vertices to the linked lists in the array entries, provided that rooms' capacities are not violated except with penalty as shown in OF1 (Equation 1).

3.2 The Metropolis step

The Metropolis step, Generate_mechanism(), consists of a perturbation operation, an accept/reject criterion, and a thermal equilibrium criterion. Perturbation to configuration C is done by randomly selecting a linked list node with color c_i and assigning it another color c_j, with j in the range 1 to Π, provided that if room capacity is violated, OF1 would penalize the resultant solution (i.e. OF 1 would increase).

The acceptance criterion checks the change in OF1 due to the perturbation. If the change decreases the objective function, the perturbation is accepted and C is updated. However, if the perturbation causes the objective function to increase, it is accepted only with a probability $e^{-\Delta OF1 / T(i)}$. Note that for lower temperature values T(i), the probability of accepting uphill moves becomes smaller; at very low (freezing) temperatures, uphill moves are no longer accepted.

The perturbation-acceptance step is repeated $\Pi * A$ times at every temperature after which thermal equilibrium is considered to be reached.

3.3 Cooling schedule

The initial temperature T(0) is the temperature that yields a high initial acceptance probability of 0.93 for uphill moves. The freezing point is the temperature at which such a probability is very small (2^{-30}), making uphill moves impossible and allowing only downhill moves. The cooling schedule used in this work is simple: $T(i+1) = \theta * T(i)$, with $\theta = 0.95$.

As the annealing algorithm searches the solution space, the best-so-far solution (with the smallest OF1) found is always saved. This guarantees that the output of the algorithm is the best solution it finds regardless of the temperature it terminates at.

Chapter 4

Genetic Algorithm

Genetic Algorithms are based on the mechanics of natural evolution [Goldberg, 1989; Davies, 1991]. They mimic natural populations reproduction and selection operations to achieve efficient and robust optimization. Through their artificial evolution, successive generations search for beneficial adaptations in order to solve a problem. Each generation consists of a population of chromosomes, also called individuals, and each chromosome represents a possible solution. The initial generation consists of randomly created individuals. The Darwinian principle of reproduction and survival of the fittest and the genetic operations of recombination (crossover) and mutation are used to create a new offspring population from the current population. The reproduction operation involves selecting, in proportion to fitness, an individual from the current population of individuals, and allowing it to survive by copying it into the new population. Then, two mates are randomly selected, and crossover is carried out to create two new offspring individuals. The offspring population replaces the parent population, and the process is repeated for many generations with the aim of maximizing the fitness of the individuals.

In the following subsections, we describe how a classical genetic algorithm (GA) is adapted for solving the exam-scheduling problem. An outline of this GA is given in Figure 4.1. This GA was developed by [Tarhini and Mansour, 1998; Awad, 2001].

```
Random generation of initial population, size POP;
Evaluate fitness of individuals;
repeat
   save_best_sofar();
   Rank individuals and allocate reproduction trials;
   for I = 1 to POP step 2
Randomly select two parents from list of reproduction trials;
        Apply crossover and mutation;
   endfor
   Evaluate fitness of offspring;
until  convergence;
```

Figure 4.1 Outline of the GA algorithm for the exam scheduling problem.

4.1 Chromosomal representation, and fitness

GA's population is an array of POP individuals. An individual in the population is encoded as an A-element vector (c(1), c(2), ... , c(A)) that corresponds to a candidate exam schedule. Recall that we have A exams and c(i) takes a color-value (period) assigned to a graph vertex (exam) i. The initial population of individuals is randomly generated. But, the colors, c_k, assigned to c(i), i=1, 2, ..., A, in each individual are selected (randomly) from the set ξ.

We use 1/OF1 as the fitness of an individual that is required to be maximized.

4.2 Reproduction scheme and Convergence

The whole population is considered a single reproduction unit within which random selection is performed. Our reproduction scheme involves elitist ranking, followed by random selection of mates from the list of reproduction trials (or copies) assigned to the ranked individuals. In the ranking scheme [Baker, 1985], the individuals are sorted by fitness values. After sorting, each individual is assigned a rank based on a scale of equidistant values for the population. The ranks assigned to fittest and least-fit individuals are 1.2 and 0.8, respectively. Individuals with ranks greater than 1 are first assigned single copies. Then, the fractional part of their ranks and the ranks of the lower half of individuals are treated as probabilities for random assignment of copies.

Elitism is used to exploit good building blocks and to ensure that good candidate solutions are preserved. This is done by replacing the least-fit individual with the best-sofar individual if the latter is better than the current-fittest. Convergence is detected when the best-sofar candidate solution does not change its OF1 value for 20 generations.

4.3 Genetic Operator

The genetic operators employed in GA are 2-point crossover and mutation at the rates 0.75 and 0.01 [Grefenstette, 1986], respectively. The application of the operators starts with randomly selecting pairs of individuals from the mating pool. Each pair of these chromosomes undergoes crossover, where positions k and l along the chromosome are selected at random between 1 and N, and all genes between k and l are swapped to create two new chromosomes. Then, mutation is applied to randomly selected genes, c(i), where its value is randomly changed from c_j to c_k ($\in \xi$).

Chapter 5

FESP Algorithm

Lotfi and Cerveny [Lotfi and Cerveny, 1991] proposed a heuristic algorithm named Final Exam Scheduling Process FESP for exam scheduling. FESP can be viewed as a set of phases where each phase solves a part of the problem until the final solution for the whole process is reached. We use FESP to compare our algorithms to it.

FESP algorithm is divided into four phases, shown in Figure 5.1. In the first phase, all exams are grouped into Π sets called blocks where these resultant blocks have the least number of simultaneous exams. The second phase consists of assigning the resultant Π blocks from the first phase to exam days in a way that minimizes the number of multiple conflicts, where a student gets two or more exams in the same day. The third phase arranges exam days and exam blocks within each day to minimize the number consecutive conflicts. Days are arranged in this phase to minimize the number of consecutive exams in the same day or in two consecutive days since the last period in an exam day is considered to be consecutive with the first period in the next day. In the fourth phase exams are assigned to classrooms by maximizing the space utilization (i.e. diminishing or minimizing room violations).

In the classical FESP algorithm Phase I is solved by assigning the first Π exams to Π blocks. Then the next exams are put in these Π blocks where there exists no conflict between the exam to be assigned and the block in which the exam is to be inserted. If the exam to be inserted has conflicts with all Π blocks, then the exam is inserted in the block having the least conflict with the exam. Phases two and three were treated separately. Phase II is solved alone by formulating it as a Traveling Salesman Problem (TSP) [Lucena, 1990]. In this formulation, the last period of a certain day is considered to be adjacent to the first period in the next day. Moreover, the TSP cities represent exam blocks and an optimal salesman tour constitutes the arrangement of the Π exam blocks with a minimum number of students having multiple exams (two or more exams in a day).

Then, Phase III is solved by taking the resultant Π exam blocks arrangement from Phase II and tries to rearrange exam blocks in order to minimize the number of consecutive exams. Phase III starts by rearranging the resultant Π exam blocks within exam days. After this part is done, exam days are rearranged in a way to minimize consecutive exams formed from blocks that are adjacent while belonging to two successive exam days. (Last block in a day is considered consecutive with the first block in the succeeding day). This third phase also may be formulated as two separate Traveling Salesman Problems (TSP). The first part of this third phase is viewed as B Traveling Salesman Problems each with E cities where E is the number of exam periods per day (Π / B), and B is the number of exam days. The second part of the third phase is viewed as B Traveling Salesman Problems.

In Phase IV, classical FESP used a greedy algorithm to solve this phase. For each period, rooms are sorted by the order of their capacities then for each exam belonging to that specified period, available rooms are searched until first room is encountered whose *remaining* capacity is greater than or equal to the specified exam enrolments.

Phase I: Assign courses to Π blocks in a way minimizing simultaneous exams.

Phase II: Arrange the Π blocks in a way minimizing multiple exams.

Phase III_1: Arrange blocks within the same day to minimize consecutives.

Phase III_2: Arrange exam days to minimize consecutive exams resulting from exams belonging to the last period in a certain day and the first period in the next day.

Phase IV: Assign exams to rooms.

Figure 5.1 Outline for the FESP Algorithm.

Chapter 6

FESPSA Algorithm

The following subsections describe how FESP is modified by adding the simulated annealing algorithm to phases two, three and four to produce improved exam schedules. It is important to mention that unlike FESP algorithm, phases two and three were done concurrently by considering them as a single traveling salesman problem with Π cities. Also, unlike the simulated annealing algorithm where simultaneous exams, consecutive exams, multiple exams and room violations were included in the objective function and thus solved concurrently, FESPSA is split into three phase. In the first phase, simultaneous exams are solved alone, then in phase two consecutive exams and multiple exams are solved alone by being introduced in an objective function, and finally in phase three, room violations are solve alone.

6.1 Assigning Exam Blocks to Exam Days and Arranging Exam Days and Blocks within Days (Phases II and III).

In the newly developed approach, simulated annealing is adopted to solve these two phases (II and III) by considering the whole problem of multiple and consecutive conflicts a single Traveling Salesman Problem with Π cities. Our aim in this case is to minimize the objective function OF2:

$$OF\,2 = \varphi_2 * S_{CE} + \sigma_2 * S_{ME} \qquad (2)$$

where φ_2 represent the weight factor for consecutives and σ_2 represent the weight factor for multiples.

In this approach, we take as input the resultant Π blocks list produced by phase I. From these blocks a conflict block graph is built, which is actually an array of linked lists. The array length is Π. In this block graph, each block node in the array extends a list that consists of each other block having conflict with the block node.

From this array, a block matrix [Π, Π] is produced. Using that matrix, the Π blocks are first randomly arranged to have an initial configuration. Then, pairs of Π blocks are randomly swapped (A*Π) iterations to determine initial temperature. Then, in each perturbation, the old consecutives and multiples are decreased while the new ones are added. Keeping in mind that our goal is to arrange the Π blocks in a way that constructs a minimal OF2 at each temperature, if the resultant is negative, that is, new objective function is better than the saved (best so far) objective function, the solution is accepted and the new objective function value and exams schedule are saved. Clearly, the output from this phase would be an exam schedule having the least objective function reached among all temperatures.

In this experiment the weight factor value of φ_2 *is* taken equal to φ and σ_2 is equal to σ which were used by the simulated annealing algorithm to unify approaches as much as possible, although this is not necessary. Moreover, the same simulated annealing algorithm shown in Figure 3.1 was used here as well.

6.2 Assigning Scheduled Exams to Classrooms (Phase IV)

The greedy algorithm used in classical FESP showed to be inefficient when the number of student enrolments in a semester increases and a number of room violations appeared. Similarly, when the number of exam days was decreased a considerable number of room violations also emerged since in these circumstances the number of room seats: $\Sigma_{1<=k<=\Pi} \Sigma_{1<= \mathrm{i}<=\rho} (\Psi_i)$

where Ψ_i is the capacity of room i would surely decrease (as Π decreases) while the number of needed student exam seats would still be the same.

The input for this phase is the resultant exam schedule attained in the previous phase. In addition, rooms' list is built classifying rooms' names, capacities (Ψ_i), and the remaining vacancy in each one at each period. Of course, initially all rooms have remaining vacancies equal to their capacities at all periods. In other words, the list embeds in each of its room nodes an array of size Π used to store the remaining vacancy of the concerned room in each period.

The objective function in this phase is:

$$\mathrm{OF3} = \gamma_3 * (\Sigma_{1<=k<=\Pi} \Sigma_{1<=i<=R} \rho_{ik}) \quad (3)$$

where γ_3 is the weight factor for the total number of rooms violated in all periods.

Scheduled exams are first assigned to random rooms. An initial temperature is deduced and the current solution is saved as the best so far. In each perturbation, rooms of all courses belonging to the same period are randomly changed. If the change resulted in a better OF3, changes are accepted otherwise, old values are returned. At each decrease in temperature ($A*\Pi$) perturbations are done and if the resulting objective function OF3 showed to be better than the best so far, the best so far is replaced with the new emerging solution.

Chapter 7

Experimental Results

In this chapter, the results deduced from the Simulated Annealing program (Number of Rooms Violated Included) with two suggested versions would be shown. These results would be compared with those of the Genetic Algorithm, classical FESP program, FESPSA program and with those of the manual schedule prepared and applied by the registrar's office.

In the following subsections experimental setup, deduced results and discussion of these results would clarify accomplished achievements.

7.1 Experimental Setup

All programs FESP, FESPSA, simulated annealing and genetic algorithm were run and tested on six realistic subject problems from the Lebanese American University for five semesters. All subject problems used periods ranging from 20 till 40 with 21 available classrooms. Table 7.1 summarizes the subject problems values. The subject problem T4 is the same as T1, but a number of course sections were treated as different courses. In addition, programs were implemented in C++ and run on the same computer having 350MHZ processor and 64MB RAM. Since the average number of exams per student is approximately four, all conflict values resulting from all programs are divided by four. In other words, all results appearing in Tables 7.3 till 7.50 represent the number of students having conflicts.

The two suggested versions used in SA, GA and FESPSA are shown in Table 7.2.

Table 7.1 Subject Problems T1 till T6.

Subject Problem	# of exams	# of Potential Conflicts	# of Enrolments
T1	336	16079	9550
T2	357	15237	9735
T3	359	6990	10836
T4	336	16079	9550
T5	426	18007	12275
T6	477	9272	12406

Table 7.2 Different versions using different weights in OF1, OF2 and OF3

Versions	α	φ	σ	γ
Version 1	100	5	0.2	200
Version 2	100	1	1	200

It is important to mention that various enhancements have been tried to the simulated annealing program before this final "best" program has been reached that is used also in the improved FESP phases as well. For example, inserting the number of rooms violated in the objective function was the best way to get rid of room violations after inserting a suitable coefficient for rooms violated. On the contrary, due to classical FESP greedy behavior whenever a room violation is encountered, it would be stuck in an infeasible solution.

To test the robustness or the sensitivity of Simulated Annealing, Genetic Algorithm, FESP and FESPSA as well, the number of exam days were decreased and increased by changing exam periods by multiples of four (four exam periods are allowed in a day).

7.2 Experimental Results for SA and GA

Manual results were available only for subject problems T2 and T3. Tables 7.3 till 7.8 summarize the results of applying SA, GA and FESP to subject problems T1 till T6 on periods of 32 referring to 8 days by using the two different versions shown in Table 7.1. Tables 7.9 through 7.50 summarize the results of applying SA and GA to subject problems T1 till T6 on different periods ranging from 20 till 40 referring to 5 till 10 days respectively (four periods per day).

Table 7.3 Results of SA and GA for Subject Problem T1 on period equals 32

T1				
Algorithm	**Simultaneous**	**Consecutives**	**Multiples**	**Rooms Violated**
FESP	0	221	286	0
SA V1	0	62	268	0
GA V1	0	65	247	0
SA V2	0	81	203	0
GA V2	0	86	203	0

Table 7.4 Results of SA and GA for Subject Problem T2 on period equals 32

T2				
Algorithm	**Simultaneous**	**Consecutives**	**Multiples**	**Rooms Violated**
Manual	5	251	310	0
FESP	0	183	264	0
SA V1	0	34	206	0
GA V1	0	23	207	0
SA V2	0	41	136	0
GA V2	0	67	150	0

Table 7.5 Results of SA and GA for Subject Problem T3 on period equals 32

T3				
Algorithm	Simultaneous	Consecutives	Multiples	Rooms Violated
Manual	4	115	146	0
FESP	0	67	98	0
SA V1	0	16	80	0
GA V1	0	17	95	0
SA V2	0	17	52	0
GA V2	0	11	50	0

Table 7.6 Results of SA and GA for Subject Problem T4 on period equals 32

T4				
Algorithm	Simultaneous	Consecutives	Multiples	Rooms Violated
FESP	0	194	277	0
SA V1	0	77	273	0
GA V1	0	74	249	0
SA V2	0	103	226	0
GA V2	0	89	202	0

Table 7.7 Results of SA and GA for Subject Problem T5 on period equals 32

T5				
Algorithm	Simultaneous	Consecutives	Multiples	Rooms Violated
FESP	0	242	368	0
SA V1	0	36	243	0
GA V1	0	40	264	0
SA V2	0	68	179	0
GA V2	0	73	173	0

Table 7.8 Results of SA and GA for Subject Problem T6 on period equals 32

T6				
Algorithm	Simultaneous	Consecutives	Multiples	Rooms Violated
FESP	0	100	157	2
SA V1	0	14	100	0
GA V1	0	17	102	0
SA V2	0	19	72	0
GA V2	0	18	61	0

Table 7.9 Simultaneous exam results of SA and GA for subject problem T1, with 5-10 days

T1 Simultaneous						
Algorithm\Days	5	6	7	8	9	10
FESP	41	5	2	0	0	0
SA V1	6	2	0	0	0	0
SA V2	3	2	0	0	0	0
GA V2	4	1	0	0	0	0

Table 7.10 Consecutive exam results of SA and GA for subject problem T1, with 5-10 days

T1 Consecutives						
Algorithm\Days	5	6	7	8	9	10
FESP	368	300	241	221	138	133
SA V1	261	203	126	62	34	18
SA V2	336	194	150	81	52	30
GA V2	308	220	143	86	58	24

Table 7.11 Multiple exam results of SA and GA for subject problem T1, with 5-10 days

T1 Multiples						
Algorithm\Days	5	6	7	8	9	10
FESP	577	466	374	286	221	181
SA V1	572	433	339	268	205	152
SA V2	572	384	300	203	140	116
GA V2	537	420	265	203	162	106

Table 7.12 Simultaneous exam results of SA and GA for subject problem T2, with 5-10 days

T2 Simultaneous						
Algorithm\Days	5	6	7	8	9	10
FESP	22	1	1	0	0	0
SA V1	3	0	0	0	0	0
SA V2	2	0	0	0	0	0
GA V2	2	0	0	0	0	0

Table 7.13 Consecutive exam results of SA and GA for subject problem T2, with 5-10 days

T2 Consecutives						
Algorithm\Days	5	6	7	8	9	10
FESP	332	264	212	183	165	97
SA V1	224	136	65	34	21	8
SA V2	252	200	77	41	19	15
GA V2	226	143	92	67	26	9

Table 7.14 Multiple exam results of SA and GA for subject problem T2, with 5-10 days

T2 Multiples						
Algorithm\Days	5	6	7	8	9	10
FESP	526	422	344	264	216	168
SA V1	489	336	326	206	157	92
SA V2	478	310	203	136	80	65
GA V2	412	293	190	150	90	62

Table 7.15 Simultaneous exams results of SA and GA for subject problem T3, with 5-10 days

T3 Simultaneous						
Algorithm\Days	5	6	7	8	9	10
FESP	5	1	0	0	0	0
SA V1	2	0	0	0	0	0
SA V2	1	1	0	0	0	0
GA V2	1	0	0	0	0	0

Table 7.16 Consecutive exams results of SA and GA for subject problem T3, with 5-10 days

T3 Consecutives						
Algorithm\Days	5	6	7	8	9	10
FESP	136	106	77	67	54	49
SA V1	73	44	18	16	7	2
SA V2	104	52	26	17	7	4
GA V2	108	66	41	11	10	8

Table 7.17 Multiple exams results of SA and GA for subject problem T3, with 5-10 days

T3 Multiples						
Algorithm\Days	5	6	7	8	9	10
FESP	216	165	117	98	71	69
SA V1	171	155	76	80	72	41
SA V2	187	112	74	52	36	27
GA V2	190	134	94	50	44	25

Table 7.18 Simultaneous exams results of SA and GA for subject problem T4, with 5-10 days

Simultaneous T4						
Algorithm\Days	5	6	7	8	9	10
FESP	37	5	2	0	0	0
SA V1	4	1	1	0	0	0
SA V2	5	2	0	0	0	0
GA V2	4	1	0	0	0	0

Table 7.19 Consecutive exams results of SA and GA for subject problem T4, with 5-10 days

Consecutives T4						
Algorithm\Days	5	6	7	8	9	10
FESP	379	303	242	194	144	171
SA V1	286	187	135	77	49	26
SA V2	323	208	154	103	56	31
GA V2	294	230	157	89	67	35

Table 7.20 Multiple exams results of SA and GA for subject problem T4, with 5-10 days

Multiples T4						
Algorithm\Days	5	6	7	8	9	10
FESP	592	466	372	277	227	197
SA V1	571	442	337	273	225	181
SA V2	544	387	271	226	147	114
GA V2	519	397	310	202	173	108

Table 7.21 Simultaneous exams results of SA and GA for subject problem T5, with 5-10 days

Simultaneous T5						
Algorithm\Days	5	6	7	8	9	10
FESP	32	8	3	0	0	0
SA V1	6	1	0	0	0	0
SA V2	2	1	0	0	0	0
GA V2	3	0	0	0	0	0

Table 7.22 Consecutive exams results of SA and GA for subject problem T5, with 5-10 days

Consecutives T5						
Algorithm\Days	5	6	7	8	9	10
FESP	431	304	250	242	162	152
SA V1	259	140	89	36	20	6
SA V2	370	223	111	68	43	11
GA V2	293	186	112	73	38	25

Table 7.23 Multiple exams results of SA and GA for subject problem T5, with 5-10 days

Multiples T5						
Algorithm\Days	5	6	7	8	9	10
FESP	688	541	379	368	277	226
SA V1	575	415	337	243	169	118
SA V2	606	386	269	179	121	79
GA V2	549	382	265	173	114	75

Table 7.24 Simultaneous exams results of SA and GA for subject problem T6, with 5-10 days

Simultaneous T6						
Algorithm\Days	5	6	7	8	9	10
FESP	9	1	0	0	0	0
SA V1	2	1	0	0	0	0
SA V2	2	1	0	0	0	0
GA V2	2	0	0	0	0	0

Table 7.25 Consecutive exams results of SA and GA for subject problem T6, with 5-10 days

Consecutive T6						
Algorithm\Days	5	6	7	8	9	10
FESP	195	130	118	100	64	51
SA V1	98	69	39	14	11	2
SA V2	130	68	35	19	14	6
GA V2	117	77	44	18	20	8

Table 7.26 Multiple exams results of SA and GA for subject problem T6, with 5-10 days

Multiples T6						
Algorithm\Days	5	6	7	8	9	10
FESP	316	227	175	157	116	89
SA V1	252	179	110	100	63	58
SA V2	222	144	87	72	49	29
GA V2	255	158	117	61	50	36

7.3 Experimental Results for FESPSA Versus FESP

Tables 7.27 till 7.32 summarize the results of applying FESPSA and FESP to subject problems on periods of 32 referring to 8 days. Tables 7.33 till 7.50 summarize the results of applying FESPSA and FESP to subject problems on different periods ranging from 20 till 40 referring to 5 till 10 days respectively (four periods per day). Tables 7.51 till 7.53 summarize room violations results of applying FESPSA and FESP on subject problems T3, T5 and T6 on different periods ranging from 20 till 40. In the other subject problems the number of room violations were equal to zero.

Table 7.27 Results of FESPSA and FESP for Subject Problem T1 on period equals 32

T1				
Algorithm	Simultaneous	Consecutives	Multiples	Rooms Violated
FESP	0	221	286	0
FESPSA	0	103	256	0

Table 7.28 Results of FESPSA and FESP for Subject Problem T2 on period equals 32

T2				
Algorithm	**Simultaneous**	**Consecutives**	**Multiples**	**Rooms Violated**
FESP	0	183	264	0
FESPSA	0	96	241	0

Table 7.29 Results of FESPSA and FESP for Subject Problem T3 on period equals 32

T3				
Algorithm	**Simultaneous**	**Consecutives**	**Multiples**	**Rooms Violated**
FESP	0	67	98	0
FESPSA	0	27	93	0

Table 7.30 Results of FESPSA and FESP for Subject Problem T4 on period equals 32

T4				
Algorithm	**Simultaneous**	**Consecutives**	**Multiples**	**Rooms Violated**
FESP	0	194	277	0
FESPSA	0	103	249	0

Table 7.31 Results of FESPSA and FESP for Subject Problem T5 on period equals 32

T5				
Algorithm	Simultaneous	Consecutives	Multiples	Rooms Violated
FESP	0	242	368	0
FESPSA	0	120	309	0

Table 7.32 Results of FESPSA and FESP for Subject Problem T6 on period equals 32

T6				
Algorithm	Simultaneous	Consecutives	Multiples	Rooms Violated
FESP	0	100	157	2
FESPSA	0	39	134	0

Table 7.33 Simultaneous exam results of FESPSA and FESP for subject problem T1, with 5-10 days

T1 Simultaneous						
Algorithm\Days	5	6	7	8	9	10
FESP	41	5	2	0	0	0
FESPSA	41	5	2	0	0	0

Table 7.34 Consecutive exam results of FESPSA and FESP for subject problem T1, with 5-10 days

T1 Consecutive						
Algorithm\Days	5	6	7	8	9	10
FESP	368	300	241	221	138	133
FESPSA	262	200	149	103	76	59

Table 7.35 Multiple exam results of FESPSA and FESP for subject problem T1, with 5-10 days

T1 Multiple						
Algorithm\Days	5	6	7	8	9	10
FESP	577	466	374	286	221	181
FESPSA	511	406	325	256	218	192

Table 7.36 Simultaneous exam results of FESPSA and FESP for subject problem T2, with 5-10 days

T2 Simultaneous						
Algorithm\Days	5	6	7	8	9	10
FESP	22	1	1	0	0	0
FESPSA	22	1	1	0	0	0

Table 7.37 Consecutive exam results of FESPSA and FESP for subject problem T2, with 5-10 days

T2 Consecutive						
Algorithm\Days	5	6	7	8	9	10
FESP	332	264	212	183	165	97
FESPSA	236	174	131	96	72	39

Table 7.38 Multiple exam results of FESPSA and FESP for subject problem T2, with 5-10 days

T2 Multiple						
Algorithm\Days	5	6	7	8	9	10
FESP	526	422	344	264	216	168
FESPSA	471	376	300	241	197	150

Table 7.39 Simultaneous exam results of FESPSA and FESP for subject problem T3, with 5-10 days

T3 Simultaneous						
Algorithm\Days	5	6	7	8	9	10
FESP	5	1	0	0	0	0
FESPSA	5	1	0	0	0	0

Table 7.40 Consecutive exam results of FESPSA and FESP for subject problem T3, with 5-10 days

T3 Consecutive						
Algorithm\Days	5	6	7	8	9	10
FESP	136	106	77	67	54	49
FESPSA	88	63	39	27	19	16

Table 7.41 Multiple exam results of FESPSA and FESP for subject problem T3, with 5-10 days

T3 Multiple						
Algorithm\Days	5	6	7	8	9	10
FESP	216	165	117	98	71	69
FESPSA	195	149	118	93	72	54

Table 7.42 Simultaneous exam results of FESPSA and FESP for subject problem T4, with 5-10 days

T4 Simultaneous						
Algorithm\Days	5	6	7	8	9	10
FESP	37	5	2	0	0	0
FESPSA	37	5	2	0	0	0

Table 7.43 Consecutive exam results of FESPSA and FESP for subject problem T4, with 5-10 days

T4 Consecutive						
Algorithm\Days	5	6	7	8	9	10
FESP	379	303	242	194	144	171
FESPSA	265	206	160	103	80	57

Table 7.44 Multiple exam results of FESPSA and FESP for subject problem T4, with 5-10 days

T4 Multiple						
Algorithm\Days	5	6	7	8	9	10
FESP	592	466	372	277	227	197
FESPSA	548	405	327	249	221	189

Table 7.45 Simultaneous exam results of FESPSA and FESP for subject problem T5, with 5-10 days

T5 Simultaneous						
Algorithm\Days	5	6	7	8	9	10
FESP	32	8	3	0	0	0
FESPSA	32	8	3	0	0	0

Table 7.46 Consecutive exam results of FESPSA and FESP for subject problem T5, with 5-10 days

T5 Consecutive						
Algorithm\Days	5	6	7	8	9	10
FESP	431	304	250	242	162	152
FESPSA	307	230	178	120	84	69

Table 7.47 Multiple exam results of FESPSA and FESP for subject problem T5, with 5-10 days

T5 Multiple						
Algorithm\Days	5	6	7	8	9	10
FESP	688	541	379	368	277	226
FESPSA	586	465	372	309	249	196

Table 7.48 Simultaneous exam results of FESPSA and FESP for subject problem T6, with 5-10 days

T6 Simultaneous						
Algorithm\Days	5	6	7	8	9	10
FESP	9	1	0	0	0	0
FESPSA	9	1	0	0	0	0

Table 7.49 Consecutive exam results of FESPSA and FESP for subject problem T6, with 5-10 days

T6 Consecutive						
Algorithm\Days	5	6	7	8	9	10
FESP	195	130	118	100	64	51
FESPSA	129	95	56	39	29	22

Table 7.50 Multiple exam results of FESPSA and FESP for subject problem T6, with 5-10 days

T6 Multiple						
Algorithm\Days	5	6	7	8	9	10
FESP	316	227	175	157	116	89
FESPSA	285	201	160	134	104	82

Table 7.51 Room violations results of FESPSA and FESP for subject problem T3, with 5-10 days

T3 Room Violations						
Algorithm\Days	5	6	7	8	9	10
FESP	2	0	0	0	0	0
FESPSA	1	0	0	0	0	0

Table 7.52 Room violations results of FESPSA and FESP for subject problem T5, with 5-10 days

T5 Room Violations						
Algorithm\Days	5	6	7	8	9	10
FESP	7	2	1	0	0	0
FESPSA	3	1	0	0	0	0

Table 7.53 Room violations results of FESPSA and FESP for subject problem T6, with 5-10 days

T6 Room Violations						
Algorithm\Days	5	6	7	8	9	10
FESP	5	2	3	2	2	1
FESPSA	3	1	2	0	1	0

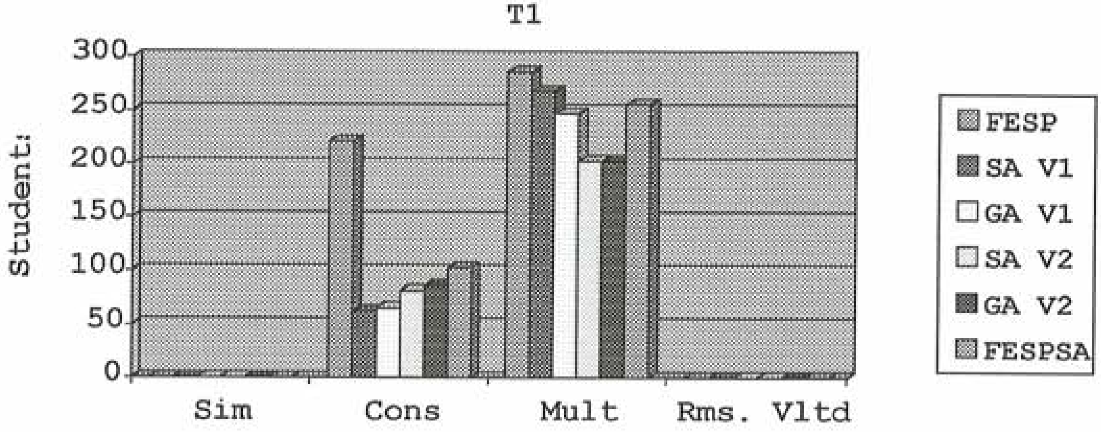

Figure 7.1 Results of SA and GA for Subject Problem T1 on period equals 32

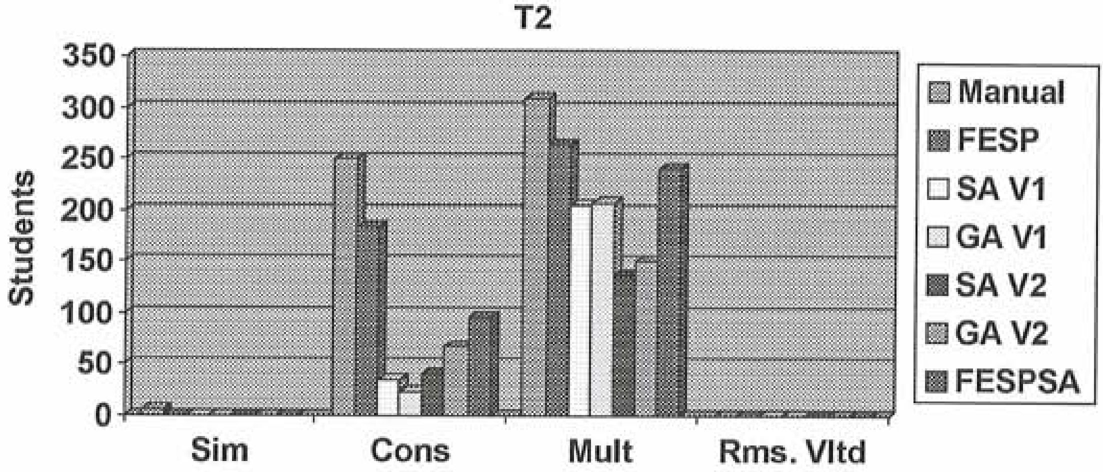

Figure 7.2 Results of SA and GA for Subject Problem T2 on period equals 32

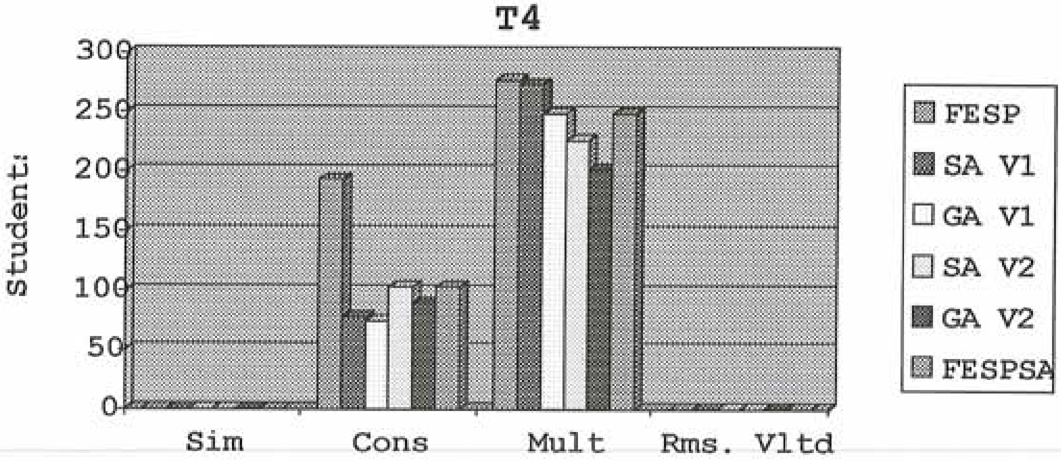

Figure 7.3 Results of SA and GA for Subject Problem T4 on period equals 32

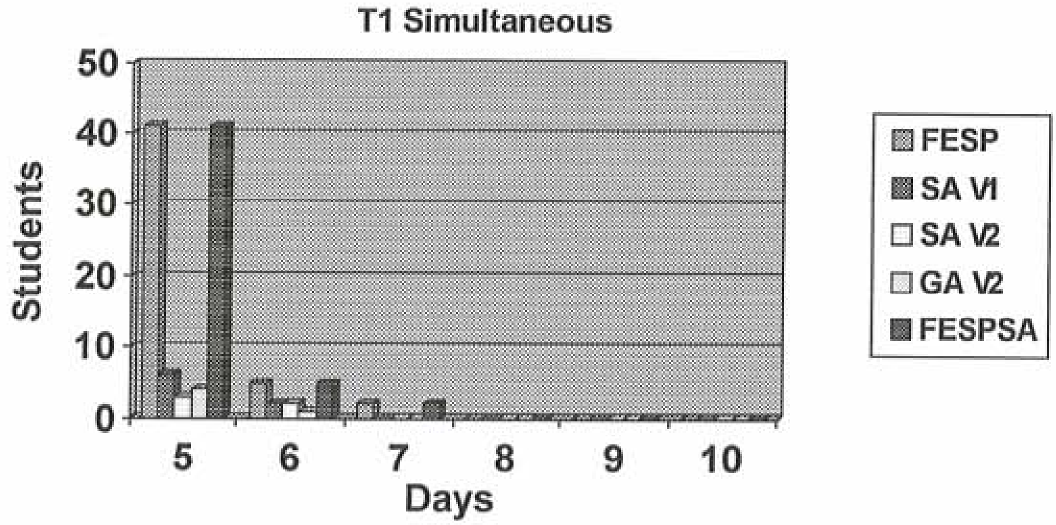

Figure 7.4 Simultaneous exam results for subject problem T1, with 5-10 days

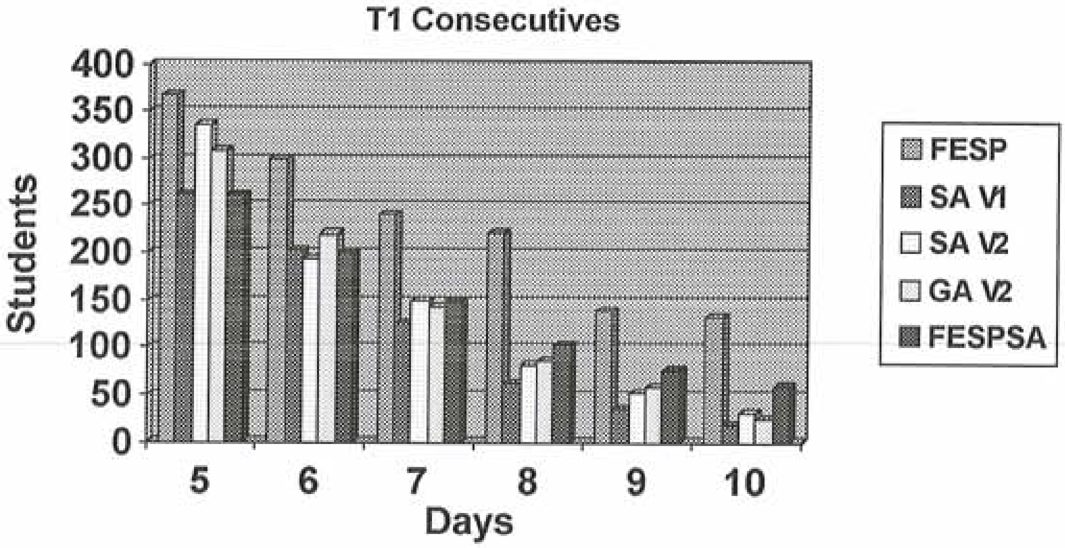

Figure 7.5 Consecutive exam results for subject problem T1, with 5-10 days

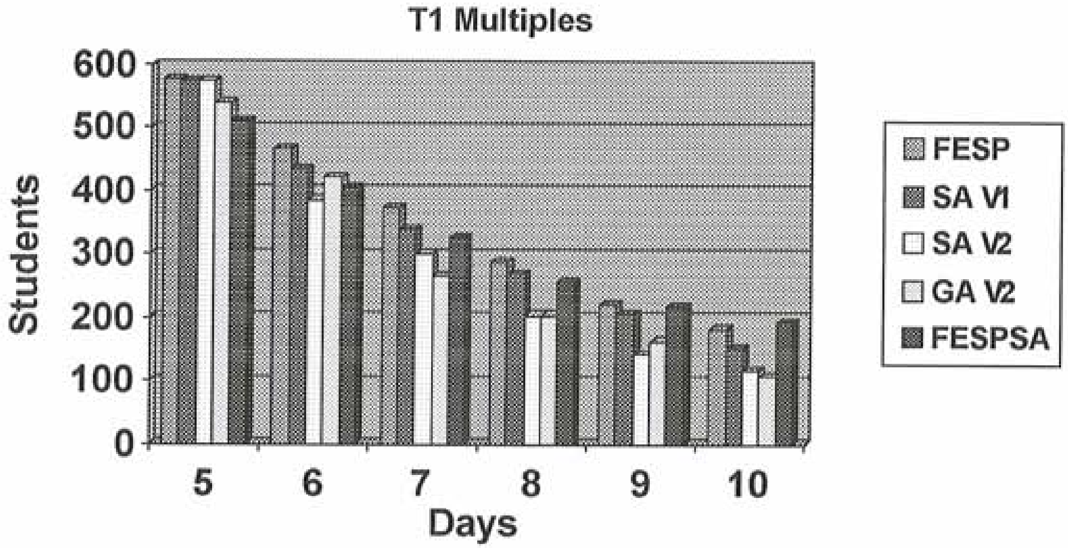

Figure 7.6 Multiple exam results for subject problem T1, with 5-10 days

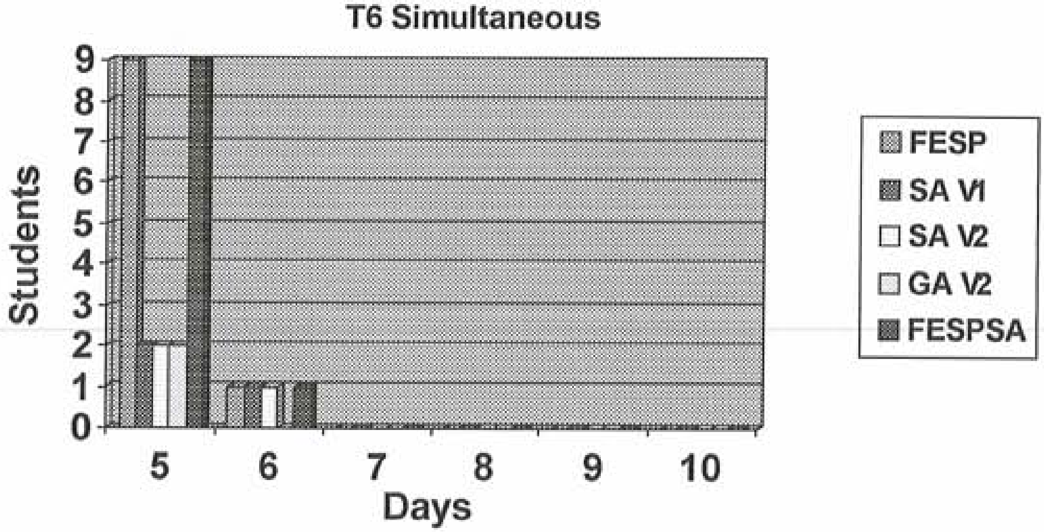

Figure 7.7 Simultaneous exam results for subject problem T6, with 5-10 days

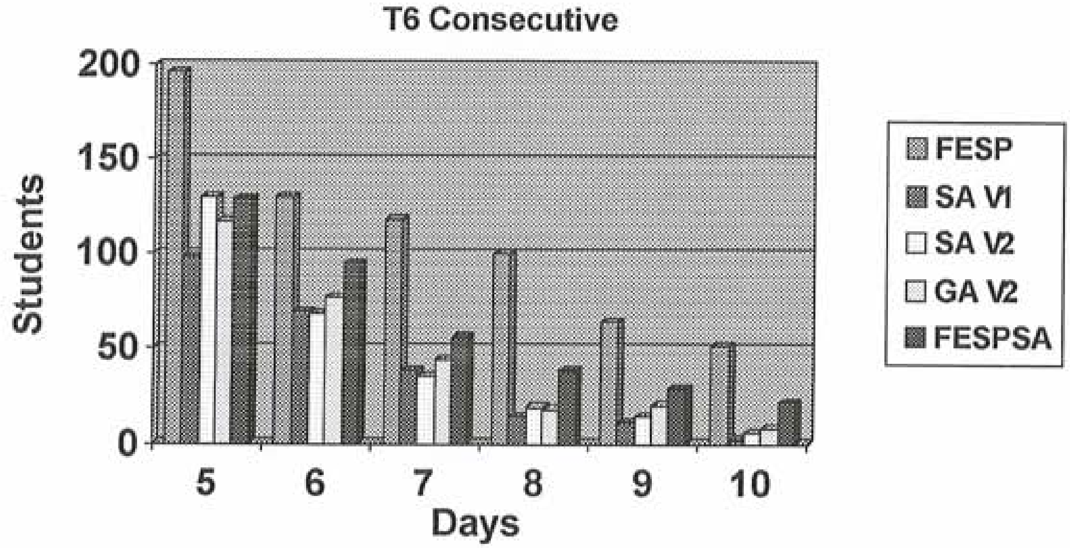

Figure 7.8 Consecutive exam results for subject problem T6, with 5-10 days

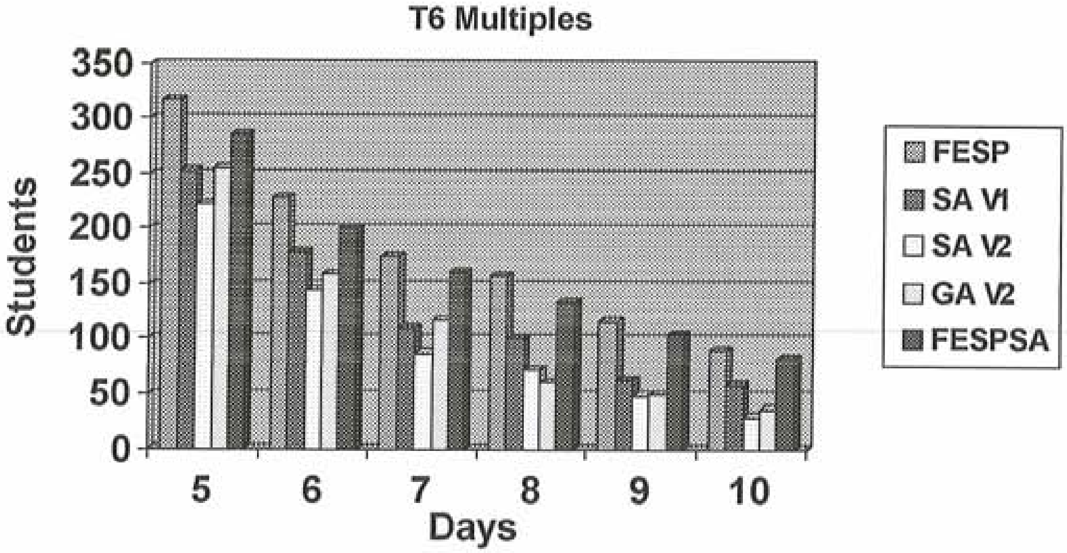

Figure 7.9 Multiple exam results for subject problem T6, with 5-10 days

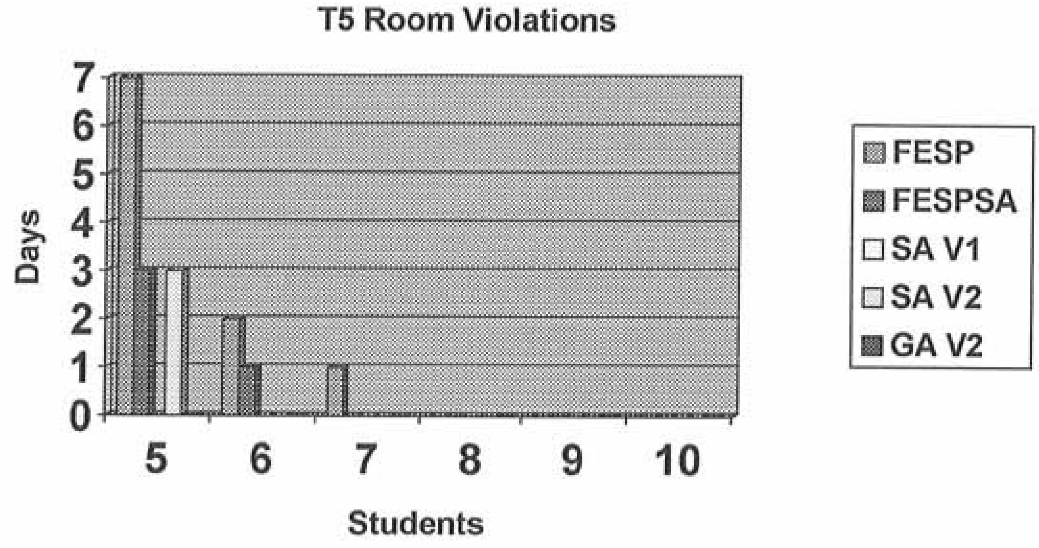

Figure 7.10 Room violations results for subject problem T5, with 5-10 days

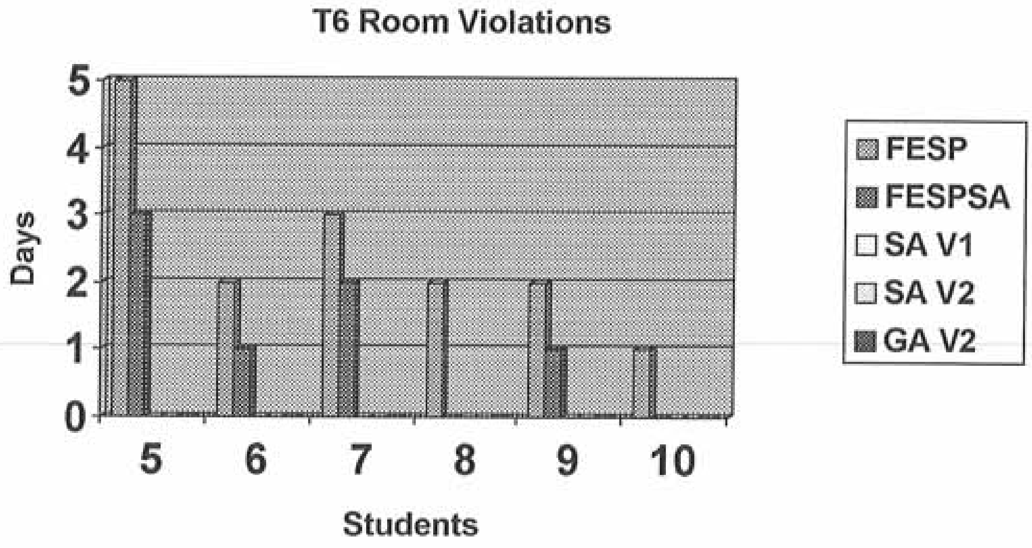

Figure 7.11 Room violations results for subject problem T6, with 5-10 days

7.4 Discussion of Results

Tables 7.3 till 7.50 show that SA and GA give better results in all versions, subject problems and period variations to those of FESP and FESPSA. Tables 7.3 till 7.26 also show that SA gives competitive and almost half the times better results than GA in terms of solution quality in much less execution times. However, the Genetic Algorithm showed to be competitive and even better than Simulated Annealing in fifty-two percent of the cases. Furthermore, Simulated Annealing remained to show better results than FESP after decreasing exam days by one i.e. SA on x-1 days showed better results than FESP in x days. Nevertheless, in one subject problem namely T6, using version 2 simulated annealing remained to show better results than FESP after decreasing exam days by two. On the other hand, FESPSA showed to be much better than FESP in terms of solution quality. Tables 7.51 till 7.53 also reveals the improvements of FESPSA concerning the number of room violations. Simulated Annealing, Genetic Algorithm and FESPSA stayed robust in dealing with simultaneous, consecutive and multiple conflicts in parallel with room violations even after decreasing exam days from eight to five. Similarly, increasing exam days using Simulated Annealing decreased the number of conflicts dramatically.

Further, the results in Tables 7.9 through 7.50 reveal that: (i) SA is competitive with GA and better in forty-eight percent of the cases; (ii) All algorithms improve the unfairness and conflict factors when exam days are increased; (iii) as α decreases with respect to φ and σ in OF1, the number of conflicts rises and unfairness drops. This means that SA, and GA are flexible in allowing the user to choose alternative schedules based on alternative requirements; (iv) φ and σ in OF2 for FESPSA may also be varied so that the number of consecutive and multiples would vary accordingly. This means that FESPSA, unlike classical FESP is also flexible in allowing the user to choose alternative schedules based on alternative requirements, but limited (no simultaneous variations); similarly (v) γ may be increased or decreased as required with penalty in OF1 and OF3 to minimize room violations in contrast with increasing it enormously to avoid room violations while producing worse solutions than with lower values of γ.

Furthermore, SA and GA shows flexibility in accommodating various user constraints such as i) forcing diverse exams to be assigned to a certain period; ii) forcing a variety of exams to certain days without specifying the period. Such constraints would make manual scheduling much more difficult whereas FESP and FESPSA do not include a straightforward mechanism to incorporate them.

SA reveals the least execution time among all algorithms, FESPSA showed better execution times than FESP whereas GA showed the highest execution times. SA took an execution time ranging from 1.5 to 2.4 minutes, FESPSA took an execution time between 2.3 and 3.2 minutes and FESP took an execution time between 2.4 and 4 minutes. However, GA took an execution time between 1.34 and 2.51 hours.

Chapter 8

Conclusion

The experimental results show that SA gives better exam schedules than manual scheduling, the clustering heuristic method FESP and FESPSA in all periods and all subject problems. It is also competitive with the GA algorithm in terms of conflicts, and is enormously better in terms of execution time. SA and GA are flexible to accommodate various user constraints. SA, GA and FESPSA show smoothness and robustness when various periods are applied to the algorithm, whereas FESP algorithm produced less competitive solutions (high number of conflicts and unfairness) in almost all subject problems especially when the number of periods or exam days decreased. In SA and GA exams may be easily prefixed with minimal effect on the resultant exam schedules while FESP is expected to give worse results since more constraints are added initially to its greedy procedures. FESPSA results showed to be better than those of FESP taking less execution times. Moreover, FESPSA was able to minimize room violations more efficiently than FESP.

References

Arani, T. Karwan, M. and Lotfi, V. (1988) A Langragian relaxation approach to solve the second phase of the exam scheduling problem, *Euro. J. Operational Research*, 34 (3), pp. 372-383.

Awad, M. (2001) Incremental and Classical Genetic Algorithms. Unpublished MSc Dissertation, LAU.

Baker, J. (1985) Adaptive Selection Methods for Genetic Algorithms, in J. Greffenstette (Ed.), *Proceedings of First International Conference on Genetic Algorithms*, Lawrence Erlbaum Associates, Hillsdale, NJ, pp. 101-111.

Balakrishnan N., Lucena A., and Wong R., (1992) Scheduling Examination to Reduce Second-Order Conflicts. *Computer Operations Research*, 19 (5), pp. 353-361.

Balakrishnan, N. (1991) Examination Scheduling: a Computerized Application *Omega*, 19 (1), pp. 37-41.

Beynon, M. Ward, A. and Maad, S (2000) The Temposcope: A Computer Instrument for the Idealist Timetabler. *Proceedings of the Third International Conference on the Practice and Theory of Automated Timetabling*. Germany, pp. 153-175.

Burke, E. Bykov, Y. and Petrovic, S. (2000) Multicriteria Approach To Timetabling Problems. *Proceedings of the Third International Conference on the Practice and Theory of Automated Timetabling*. Germany, pp.180.

Carter, M. and Johnson, D. G. (2001) Extended Clique Initialization in Examination Timetabling. *Journal of the Operational Research Society*, 52 (5), pp. 538-544.

Carter, M. Laporte, G. and Chinneck, J. (1994) A general examination scheduling system, *Interfaces*, 24 (3), pp. 109-120.

Corne, D. Fang, H. L. and Mellish, C. (1993) Solving the Modular Exam Scheduling Problem with Genetic Algorithms. *Proceedings of the Sixth International Conference in Industrial and Engineering Applications of Artificial Intelligence and Expert Systems.* Edinburgh, Scotland, pp. 370-373.

Davies, L (Ed.), (1991) *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York.

Deris, S. Omatu, S. and Ohta H. (2000) Timetable Planning Using the Constraint Based Reasoning. *Computer and Operations Research*, 27 (9), pp. 819-839.

Descroches, S. Laporte, G. and Rousseau, J. (1978) HOREX: a computer program for the construction of examination schedules, *INFOR*, 16 (3), pp. 294-298.

Di Gaspero, L. and Schaerf, A. (2000) Tabu Search Techniques for Examination Timetabling. *Proceedings of the Third International Conference on the Practice and Theory of Automated Timetabling.* Germany, pp.176-179.

Erben, W. (2000) A Grouping Genetic Algorithm for Graph Coloring and Exam Timetabling. *Proceedings of the Third International Conference on the Practice and Theory of Automated Timetabling.* Germany, pp.397-421.

Ergul, A. (1996) GA-based examination scheduling experience at Middle East Technical University, in E. Burke and P. Ross (Ed.), *The Practice and Theory of Automated Timetabling*, Lecture Notes in Computer Science 1153, Springer-Verlag, pp. 212-226.

Goldberg, D. (1989) *Genetic Algorithms in Search,* Optimization and Machine Learning, Reading, MA, Addison-Wesley.

Grefenstette, J. (1986) Optimization of Control Parameters for Genetic Algorithms, *IEEE Transactions on Systems*, Management and Cybernetics 16 (1), pp. 122-128.

Hertz, A. (1991) Tabu search for large scale timetabling problems, *Euro. J. Operational Research*, 54, pp. 39-47.

Kirkpatrick, S. Gelatt, C. and Vecchi, M. (1983) Optimization by simulated annealing, *Science*, 220, pp. 671-680.

Lajos, G. (1995) Complete University Modular Timetabling Using Constraint Logic Programming. *Proceedings of the First International Conference on the Practice and Theory of Automated Timetabling.* United Kingdom, pp. 364-375.

Leong, T. and Yeong, W. (April 1987) Examination scheduling: a quadratic assignment perspective, *Proc. Int. Conf. On Optimization: Techniques and Applications*, Singapore, pp. 550-558.

Lotfi, V. and Cerveny, R. (1991) A final-exam-scheduling package, *J. of the Operational Research Society*, 42, pp. 205-216.

Lucena, A. (1990) Time Dependent Traveling Salesman- The Delivery Man Case. *Networks* 20, pp. 753-763.

McCollum, B. and Newall, J. (2000) Introducing Optime: Examination Timetabling Software. *Proceedings of the Third International Conference on the Practice and Theory of Automated Timetabling.* Germany, pp.485-490.

Mehta, N. (1981) The Application of a Graph Coloring Method to an Examination Scheduling Problem. *Interface*, 11 (5).

Mikati, Z. (1999) Exam Scheduling Algorithms. Unpublished MSc Dissertation, LAU.

OPTIME, http://www.asap.cs.nott.ac.uk/optime

Reis, L. P. and Oliveira, E. (2000) Examination Timetabling Using Set Variables. *Proceedings of the Third International Conference on the Practice and Theory of Automated Timetabling.* Germany, pp. 181-183.

Schedulexpert, Http://www.schedulexpert.com

Tarhini, A. and Mansour N., (1998) Natural Optimization Algorithms for Exam Scheduling, *Proceedings of International Conference on Computer Systems,* 124-127, IASTED, ACTA Press, March 30-April.

Thompson, J. and Dowsland, K. (1996) General cooling schedules for a simulated annealing based timetabling system, in E. Burke and P. Ross (Ed.), *The Practice and Theory of Automated Timetabling*, Lecture Notes in Computer Science 1153, Springer-Verlag, pp. 345-363.

White, G and Xie B. S. (2000) Examination Timetables and Tabu Search With Longer Term Memory. *Proceedings of the Third International Conference on the Practice and Theory of Automated Timetabling.* Germany, pp.184-201.

Wood, D. (1968) A system for computing university examination timetables, *Computer Journal*, 11, pp. 41-47.