DEVELOPING A COURSE SCHEDULING SYSTEM BY USING GENETIC ALGORITHM

A MASTER'S THESIS

in

Computer Engineering

Atılım University

by

CANSU ÇİĞDEM AYDIN

AUGUST 2008

DEVELOPING A COURSE SCHEDULING SYSTEM BY USING GENETIC ALGORITHM

A THESIS SUBMITTED TO

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

OF

ATILIM UNIVERSITY

BY

CANSU ÇİĞDEM AYDIN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

THE DEPARTMENT OF COMPUTER ENGINEERING

AUGUST 2008

Approval of the Graduate School of Natural and Applied Science, Atılım University.

_____

Prof.Dr. Abdurrahim Özgenoğlu

Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

_____

Prof. Dr. İbrahim Akman

Head of Department

This is to certify that we have read thesis "Developing A Course Scheduling System By Using Genetic Algorithm" submitted by "Cansu Çiğdem Aydın" and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____	_____
(Title and Name)	Asst. Prof. Nergiz Ercil Çağıltay
Co-Supervisor	Supervisor

Examining Committee Members

Asst. Prof. Nergiz Ercil Çağıltay	_____
Asst. Prof. Elif Uray Aydın	_____
Inst. Gül Tokdemir	_____
.....	_____
.....	_____

Date: 29.08.2008

I declare and guarantee that all data, knowledge and information in this document has been obtained, processed and presented in accordance with academic rules and ethical conduct. Based on these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name: Cansu Çiğdem, Aydın

Signature:

ABSTRACT

DEVELOPING A COURSE SCHEDULING SYSTEM

BY USING GENETIC ALGORITHM

Aydın, Cansu Çiğdem

M.S., Computer Engineering Department

Supervisor: Asst. Prof. Nergiz Ercil Çağıltay

August 2008, 54 pages

Course scheduling problem defined as assigning of courses into specific time slots in a week, considering the special constraints. The manual solution of this problem by considering all constraints usually requires a long time and a hard work in order to provide an appropriate solution. In addition, the obtained solution usually is unsatisfactory due to limited human oversight. In this thesis, Genetic Algorithm (GA) is presented for solving the constraint-based university timetabling problem. The developed system includes flexible and intelligent software that automate the course scheduling task and addresses all special features found in the Atılım University's Engineering Faculty.

Keywords: Genetic Algorithm, Course Scheduling, Timetabling

ÖZ

GENETİK ALGORİTMA KULLANILARAK DERS ÇİZELGELEME SİSTEMİ TASARIMI

Aydın, Cansu Çiğdem

Yüksek Lisans, Bilgisayar Mühendisliği Bölümü

Tez Yöneticisi: Yrd. Doç. Dr. Nergiz Ercil Çağıltay

Agustos 2008, 55 sayfa

Ders çizelgeleme problemi, daha önceden belirlenmiş olan kısıtlar göz önüne alınarak belirli zaman aralıklarına derslerin atanması olarak tanımlanır. Bütün kısıtlar dikkate alınarak, problemin elle çözümü, uzun bir zaman ve sıkı bir çalışma gerektirir. Ayrıca, belirlenen çözüm dikkatsizlikten dolayı yeterli olmayabilir. Bu tezde, kısıt tabanlı üniversite ders çizelgeleme probleminin Genetik Algoritma ile çözümü sunulmuştur. Geliştirilen sistem, Atılım Üniversitesi Mühendislik Fakültesi'nin tüm özelliklerini adresleyen esnek ve akıllı ve ders çizelgeleme faaliyetini otomatik olarak gerçekleştirebilen bir yazılım içermektedir.

Anahtar Kelimeler: Genetik Algoritma, Ders Çizelgeleme, Zaman Çizelgeleme

ACKNOWLEDGEMENTS

I would like to thank my thesis examination commitiee for their time, interest, and input in this study. I would especially like to thank my supervisor Asst. Prof. Nergiz Ercil ÇAĞILTAY for her guidance, encourangement and insight throughout the research.

I should also express thanks to Serdar BİROĞUL from Gazi University and Başar ÇETİNKAYA from Computer Center at Atılım University for their helps.

To my family, I offer sincere thanks for their continuous support and patience during this period.

TABLE OF CONTENTS

ABSTRACT iv
ÖZ v
ACKNOWLEDGEMENTS vi
TABLE OF CONTENTS vii
LIST OF TABLES x
LIST OF FIGURES xi
CHAPTER 1 1
INTRODUCTION 1
CHAPTER 2 3
LITERATURE REVIEW 3
2.1 Graph Coloring 3
2.2 Simulated Annealing 3
2.3 Tabu Search 4
2.4 Constraint Logic Programming 4
2.5 Genetic Algorithms 5
2.6 Why is Genetic Algorithm? 6
2.7 Details of Genetic Algorithm 7
2.7.1 Genetic Operators 7
Selection Operator 7
Roulette Wheel Selection 8
Crossover Operator 8
Crossover Types 9
Two Point 9
Mutation Operator 11
Repair Operator 12
Elitism 12

2.7.2 Coding and Chromosome Representation 14
2.7.3 Fitness Function and Cost Function 14
2.7.4 Parameters of GA 15
Crossover and Mutation Probability 15
Other Parameters 15
CHAPTER 3 16
RESEARCH METHODOLOGY 16
3.1 Problem Description 16
3.2 Requirement Analysis 17
3.3 Aim of the Study 18
3.4 System Inputs Based on System Requirements 19
3.4.1 Constraint Object 19
Hard Constraints 19
Soft Constraints 20
3.4.2 Lecturer Object 21
3.4.3 Classroom Object 21
3.4.4 Course Objects 22
3.4.5 Course-Section-Lecturer Triad Object 23
3.5 Applied Software Methodology 24
CHAPTER 4 26
DESIGN AND IMPLEMENTATION 26
4.1 Genetic Representation 26
4.2 Initialization of the Population 28
4.3 Cost Function 28
4.4 Fitness Function 29
4.5 Selection 30
4.6 Crossover and Mutation 30
4.7 Repair Operator 31
4.8 Elitism Operator 31
4.9 System User Interface 32
CHAPTER 5 42
EXPERIMENTS and RESULTS 42
CHAPTER 6 50
DISCUSSIONS&CONCLUSION 50

FUTURE WORK... 51
REFERENCES 52

LIST OF TABLES

TABLE

1. Characteristic of experimental data 43

2. Weights of Penalties for Hard and Soft Constraints 43

3. Parameters of GA 44

4. Experiment Results 44

5. Penalty Values of best solution for each constraints 45

LIST OF FIGURES

FIGURES

1.1 Gen-Chromosomes and Population Structure 6
2. Roulette Wheel 8
3. Crossover of Binary Strings 9
4. Mutation of Binary Strings 12
5. General Structure of Genetic Algorithm 13
3.4.1 Course Time-Slots Groups 20
3.3.2 Data Structure of Lecturer Object 21
3.4.3 Data Structure of Classroom Object 22
3.4.4 Data Structure of Course Object 22
3.4.5 Data Structure of Course-Section-Lecturer Triad Object 24
3.5 Flow Diagram of Waterfall Model 25
4.1 Genetic Representation 27
4.6.1 Process of mutation operator 30
4.6.2 Process of crossover operator 31
4.7.1 After the Repair 31
4.9 Workflow diagram of program 32
4.9.1 Main Window 33
4.9.2 Duplicated Data 34
4.9.3 Courses Tab 35
4.9.4 Classroom Tab 36
4.9.5 Preschedule Tab 37
4.9.6 Lecturer-Course Assignment Tab 38
4.9.10 Population Tab 39
4.9.11 Experimental Results Tab 40
4.9.12 View Schedule Tab 41
5.1 Division of a week into time slots 42
5.3 Fitness Value Propagation for best solution 46

5.4 Course Schedule of Specific Courses in Computer Engineering 47
5.5 Course Schedule of Specific Lecturer in Computer Engineering 48
5.6 Course Schedule of Specific Course 48
5.7 Course Schedule of Specific Classroom 49

CHAPTER 1

INTRODUCTION

Timetabling is the problem of scheduling a set of events to specific time slots. It means that no person or resource is expected to be in more than one location at the same time and with enough space available in each location for the number of people expected to be there [1]. These two main constraints and size of the problem make the timetabling a classically hard problem to solve [2].

As other scheduling problems, course scheduling is also an NP-complete problem [3] and each educational institution faces with course scheduling problem every term. It has been solved by human resources and concerns by scheduling a certain number of resources such as teachers, courses and classrooms to a number of time periods on a daily basis.

In the scheduling process, there are several considerations to be taken into account, such as teaching staff preferences, offered courses, number of classrooms and classroom requirements for each course. For example, some instructors may prefer to teach in a certain day in a week. Similarly, a group of courses may be preferred to be given at the same time or some may be preferred to be given at different time slots.

The manual solution of the problem by considering all constraints usually requires a long time and a hard work in order to provide an appropriate solution. In addition, the obtained solution may be unsatisfactory due to limited human oversight. Our aim in this thesis is to develop a flexible and intelligent software for timetabling that addresses all special features found in Atılım University's Engineering Faculty, and we aim to use genetic algorithm (GA) to solve the constraint-based school timetabling problem.

This study reports the details of the developed software as well as the test results showing how this proposed software can support the scheduling problem solution process, based on Atılım University's requirements as a case study.

CHAPTER 2

LITERATURE REVIEW

In the literature, a number of techniques have been developed to solve scheduling problems [4]. Mathematical methods, constraints logic programming numerical analysis and heuristic methods are used in the solution of these problems. Genetic algorithm, graph coloring, simulated annealing, constraint logic programming and tabu search are the most common heuristic approaches found in the literature. In the following sections, we summarized these approaches.

2.1 Graph Coloring

Graph coloring is the formulation of the problem on the graph by manually. It is not preferred to be used in course scheduling problem. It is also impossible to express complex models on the graph. Therefore, it only works in cases where the course-scheduling problem can be reduced to a simple assignment problem. In graph coloring, every course is assumed as a node, with courses that are likely to be taken concurrently connected by an edge, and then the problem of scheduling reduces to coloring the graph. In this case, each color represents a (disjoint) time to offer a course. Burke [5] has applied graph-coloring method to university scheduling problem. It requires the incorporation of non-academic constraints into the problem formulation, which makes it extremely difficult to implement [13].

2.2 Simulated Annealing

"Simulated annealing is a search strategy that keeps track of one feasible timetable. On each generation, a neighbours is produced after another feasible timetable. A worse neighbour is accepted as the new solution with a probability that decreases as the computation proceeds. If the neighbour has a higher penalty than the others, it may be accepted according to a probability, which is related to a control

parameter called temperature. The temperature and thus the probability of inferior neighbours being accepted, is decreased in each iteration or (more usually) after a particular number of iterations (this number may be constant or it can increase as the temperature decreases). The process is analogous to the cooling process in actual annealing. One drawback with simulated annealing is that cooling process can take a long time in order to achieve good results." [8]. Swansea's TISSUE has successfully applied Simulated annealing to the examinations scheduling system [6, 7].

2.3 Tabu Search

As is the case in simulated annealing, tabu search keeps in memory just one current feasible timetable. The difference is in the method by which moves to new timetables are accepted. A tabu search performs a list of tabu moves represent schedules that having been visited recently, are forbidden in order to prevent the search from staying in the same area. The tabu list size is usually fixed, with the oldest moves being removed as new moves are added. Because tabu moves may deny the search from reaching new improved solutions, an "aspiration level" is often maintained, which represents the best solution visited so far. If a tabu timetable reaches the aspiration level, it may be removed from the tabu list. Tabu search has been applied successfully by Boufflet and N`egre to generate examinations timetables at the University of Technology of Compi`egne [9].

2.4 Constraint Logic Programming

Timetabling can be modeled as a constraint satisfaction problem (CSP). In a CSP, values, which satisfy a set of constraints, must be found for a set of discrete variables with finite domains. Constraint logic programming (CLP) is based on the application of declarative logic programming languages such as Prolog to CSPs. A Prolog program consists of a set of clauses, which in CLP may contain constraints, the satisfiability of which is checked during computation. In CLP, a labeling strategy dictates the order in which the search space is traversed, which is vital to an effective search. There are two orderings; the order in which the variables are instantiated (*e.g.* meetings placed) and the order in which the values (*e.g.* times and rooms) are assigned. A Prolog lecture scheduling system has been developed by White *et al* at the University of Ottawa [10,11]. Constraints are divided into groups as primary and secondary. Primary constraints must always be satisfied, but secondary constraints

may be relaxed. Each course is scheduled in turn. If it is not possible to assign a time and room without violating constraints, the program backtracks, relaxing secondary constraints one by one until the course can be scheduled. If the course still cannot be scheduled, a similar course will be temporarily unscheduled, and the search tried again. Unschedulable courses are left to be manually placed.

Boizumault, Delon and P´eridy have designed an examinations scheduling program [12] in CHIP, a Constraint Logic Programming language based on Prolog, which provides several types of constraint. But the performance of CLP is very sensitive to even minor changes in formulation and locks flexibility.

2.5 Genetic Algorithms

The Genetic Algorithm (GA) is an evolutionary algorithm based on processes and concepts drawn from natural selection and genetics (Goldberg, 1989). Inspired by the principles of natural evolution, Holland (1975) first developed a search technique based on randomly generated points that named as GA. A comparison between GAs and other optimization methods was described by Goldberg [19], and can be summarized as follows: GAs

- follow probabilistic transition rules, not deterministic rules.
- deal with sets of possible discrete solutions, not a single solution.
- directly use the objective function itself, not derivative information.
- work with a coding of the parameter sets, not the parameters themselves.

It is an optimization algorithm, which is used to evolve sets of parameters for a given problem. Each parameter set represents a potential solution to the given problem and is referred to in this thesis as an individual. The collection of all the individuals in a GA simulation at a given point of time is referred to as the population. Problem parameters in GAs are encoded as sets of genes. These sets of genes are usually (but not always) stored within a computer system as an array or string of binary values. This gene structure is typically referred to as a genetic string or a chromosome (Figure 1.1).

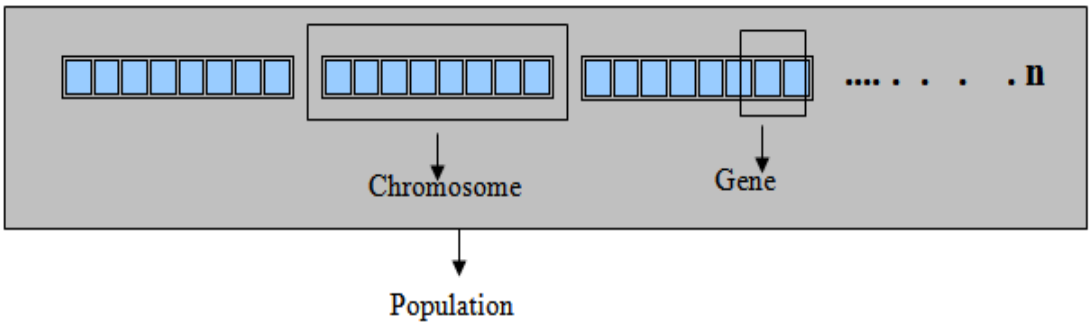


Figure 1.1 Gen-Chromosomes and Population Structure

After an initial population is randomly generated, the algorithm evolves the through operators as follows:

- **Selection** which equates to survival of the fitness;
- **Crossover** which represents mating between individuals;
- **Mutation** which introduces random modifications;

And addition to this three operators,

Repair which fix invalid solution in chromosome.

2.6 Why is Genetic Algorithm?

Numbers of technique have been developed to solve timetabling problem. The oldest and most popular technique is the graph-coloring algorithm. However, this method has one major disadvantage in that it requires the incorporation of non-academic constraints into the problem formulation, which makes it extremely difficult to implement [13].

Some researcher used constraints logic programming (CLP). However, the performance of CLP is very sensitive to even minor changes in formulation and locks flexibility. Tabu search and simulated annealing are still among in popular heuristic algorithms. Colorni [18] report that GAs and tabu search produces better timetables according to their experiments. Between all this algorithms, GAs are rapidly finding application to the solution of scheduling problem [13].

A comparison between GAs and other optimization methods was described by Goldberg [19], and summarized as follows:

- GAs follow probabilistic transition rules, not deterministic rules,
- GAs deal with sets of possible discrete solutions, not a single solution,
- GAs directly use the objective function itself, not derivative information,
- GAs work with a coding of the parameter sets, not the paramaeters themselves.

2.7 Details of Genetic Algorithm

In this section, information about genetic operators will be provided.

2.7.1 Genetic Operators

The three basic operators used in GAs are selection, crossover, and mutation. The applications of these operators are determined by user selectable probabilities that greatly affect the efficacy of the GA.

Selection Operator

Selection operator gives preference to better individuals, allowing them to pass on their genes to the next generation. The goodness of each individual depends on its fitness. Fitness can be determined by an objective function. According to it, new generation is created with offsprings having maximum fitness. Roulette selection is the method mostly used. There are many other methods that are used for how to select the best chromosomes such as Boltzman selection, tournament selection, rank selection, and steady state selection.

Roulette Wheel Selection

In this method, Parents are selected according to their fitness. The better the chromosomes are, the more chances to be selected they have. Chromosome with bigger fitness will be selected more times.

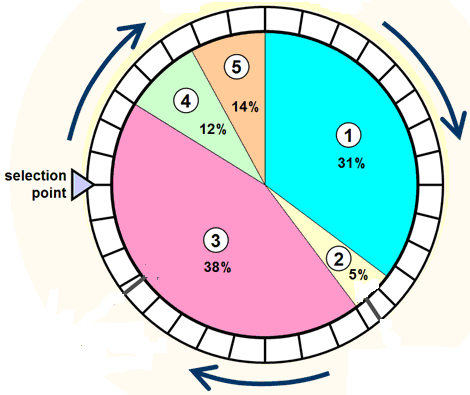


Figure 2: Roulette Wheel

This can be simulated by following algorithm.

1. **[Sum]** Calculate sum of all chromosome fitnesses in population - sum ***S***.
2. **[Select]** Generate random number from interval ***(0,S)*** - ***r***.
3. **[Loop]** Go through the population and sum fitnesses from ***0*** - sum ***s***. When the sum ***s*** is greater then ***r***, stop and return the chromosome where you are.

Crossover Operator

Sometimes called recombination, the crossover operator takes two offspring from the population randomly or according to their fitness. Two individuals are chosen from the population using the selection operator. A crossover site along the bit strings is randomly chosen. The values of the two strings are exchanged up to this point. If S1=110010011 and S2=000100111 and the crossover point is 4 then S1'=110000111 and S2'=000110011

The two new offspring created from this mating are put into the next generation of the population.

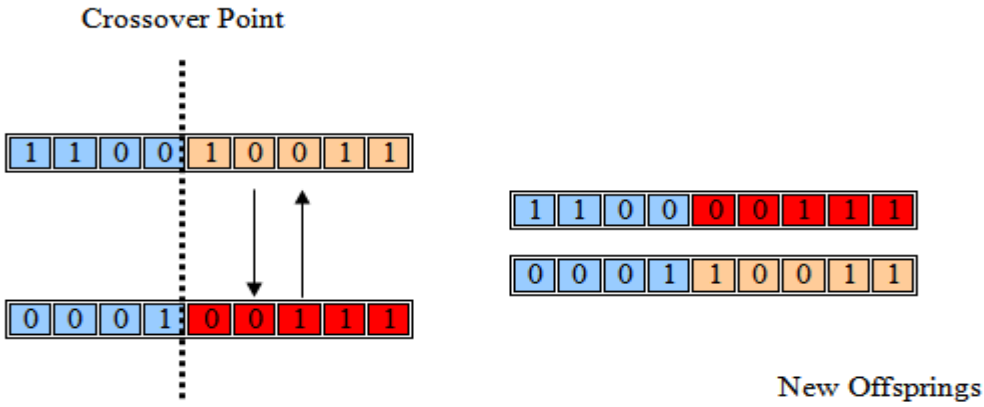


Figure 3: Crossover of Binary Strings

Crossover Types

There are many kinds of crossover [14]. Some of the important ones are:

One Point

A crossover operator that randomly selects a crossover point within a chromosome then interchanges the two parent chromosomes at this point to produce two new offspring.

The "|" symbol indicates the randomly chosen crossover point.

Parent 1: 11001|010

Parent 2: 00100|111

After interchanging the parent chromosomes at the crossover point, the following offspring are produced:

Offspring1: 11001|111

Offspring2: 00100|010

Two Point

A crossover operator that randomly selects two crossover points within a chromosome then interchanges the two parent chromosomes between these points to produce two new offspring.

Parent 1: 110|010|10

Parent 2: 001|001|11

After interchanging the parent chromosomes between the crossover points, the following offspring are produced:

Offspring1: 110|001|10

Offspring2: 001|010|11

Multi-Point Crossover

A crossover operator that randomly selects *n* crossover points within a chromosome then interchanges the two parent chromosomes between these points to produce two new offspring.

Parent 1: 1|10|010|10

Parent 2: 0|01|001|11

After interchanging the parent chromosomes between the crossover points, the following offspring are produced:

Offspring1: 0|10|001|10

Offspring2: 1|01|010|11

Arithmetic

A crossover operator that linearly combines two parent chromosome vectors to produce two new offspring according to the following equations:

Offspring1 = a * Parent1 + (1- a) * Parent2

Offspring2 = (1 – a) * Parent1 + a * Parent2

where *a* is a random weighting factor (chosen before each crossover operation).

Consider the following 2 parents (each consisting of 4 float genes) which have been selected for crossover:

Parent 1: (0.3)(1.4)(0.2)(7.4)

Parent 2: (0.5)(4.5)(0.1)(5.6)

If $a = 0.7$, the following two offspring would be produced:

Offspring1: (0.36)(2.33)(0.17)(6.86)

Offspring2: (0.402)(2.981)(0.149)(6.842)

Heuristic

A crossover operator that uses the fitness values of the two parent chromosomes to determine the direction of the search. The offspring are created according to the following equations:

Offspring1 = BestParent + r * (BestParent – WorstParent)

Offspring2 = BestParent

Where r is a random number between 0 and 1.

It is possible that *Offspring1* will not be feasible. This can happen if r is chosen such that one or more of its genes fall outside of the allowable upper or lower bounds. For this reason, heuristic crossover has a user settable parameter (n) for the number of times to try and find an r those results in a feasible chromosome. If a feasible chromosome is not produced after n tries, the *WorstParent* is returned as *Offspring1*.

Mutation Operator

It is a random alteration of a gene, or genes, in a chromosome. In the genetic algorithm, mutation involves selecting a random point, or points, in the chromosome and substituting the value found at that point with an appropriate value that has been randomly generated.

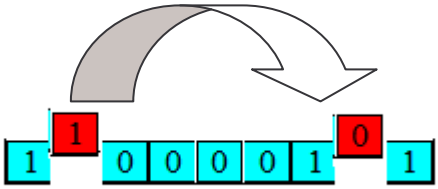


Figure 4: Mutation of Binary Strings

Repair Operator

Repair operator is used to fix some with invalid solution for the problem. Because, the new gene structures formed after crossover and mutation operates turn into unsuitable gene structures. This operator use problem specific knowledge about the problem domain to repair missing, lost or extra lessons in timetable.

Elitism

When creating new population by crossover and mutation, we can loose the best chromosome. Elitism is name of method, which first copies the best chromosome (or a few best chromosomes) to new population. The rest is done in classical way. Elitism can very rapidly increase performance of GA, because it prevents losing the best found solution. Figure 5 illustrates general structure of genetic algorithm.

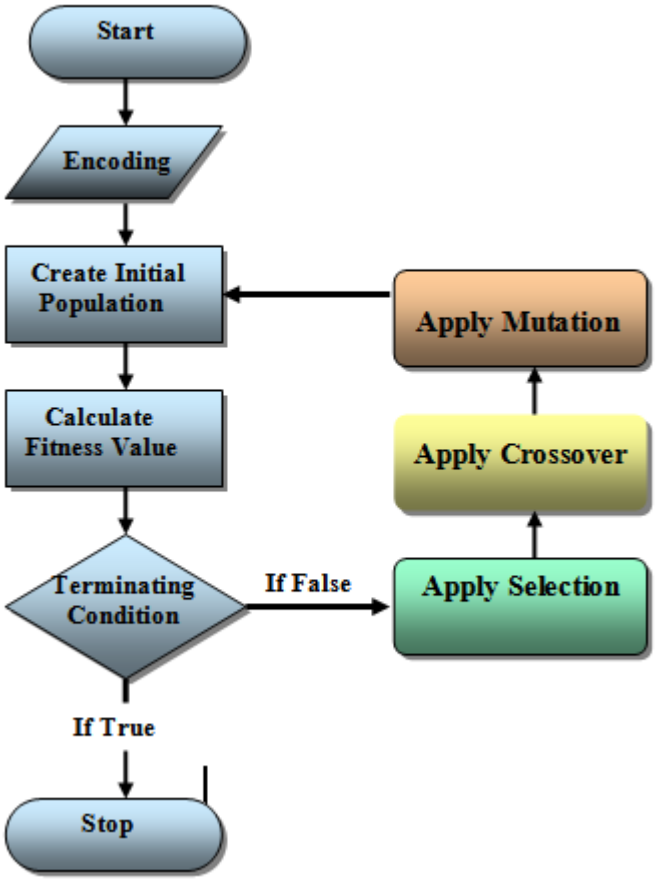


Figure 5: General Structure of Genetic Algorithm

Typically, a genetic algorithm consists of following steps:

1. Determine a Coding Scheme for possible solution.
2. Determine the population size.
3. Create an initial population.
4. Choose how you want to select the parents(According to fitness)
5. Determine a Crossover Operator and the probability of crossover ($P_{crossover}$)
6. Choose a Mutation Operator and a probability of mutation(P_{mutate})
    a. Determine whether or not you want to use a Mutation Operator
    b. Determine which offspring should be kept.

7. Determine what you want to do with the new offspring. Some choices include:

Placing the new offspring in a new population. When the new population has reached a given size (usually the size of the initial population) you can replace members of the current population with the chosen offspring. Possible replacements include

2.7.2 Coding and Chromosome Representation

Coding of chromosomes is one of the problems, while starting to solve problem with genetic algorithm and it depends on problem type. In a classical genetic algorithm, chromosomes are represented as binary, non-binary fixed, and variable length bit strings.

According to Wilhelm Erben and Jürgen Keppler [15], problem specific knowledge should be incorporated in the representation of solutions, and the chromosome representation should be 'natural', it should contain all the relevant information and close to the original problem. According to Beasley, Bull and Martin [16] the efficiency of the genetic algorithm mainly depends on the used coding method and fitness function. The encoded chromosomes may not always correspond to a feasible solution, and genetic operators may produce illegal chromosomes as well.

2.7.3 Fitness Function and Cost Function

Cost function is used for the value of each chromosome and allows us to distinguish good solution from bad. It guides to the better solutions. In course-scheduling problem, cost function will be the sum of the penalties given to the constraints violations.

Fitness function plays a very important role in genetic algorithm. It is used for the survival of the certain chromosome. It is generally inversely proportional to the cost function since worse solutions have greater cost.

2.7.4 Parameters of GA

There are some main parameters of GA such as crossover and mutation probability and they control the precise operation of the genetic algorithm.

Crossover and Mutation Probability

Crossover probability says how often will be crossover performed. If there is no crossover, offspring is exact copy of parents. If there is a crossover, offspring is made from parts of parents' chromosome. If crossover probability is 100%, then all offspring is made by crossover. If it is 0%, whole new generation is made from exact copies of chromosome from old population.

"Crossover is made in hope that new chromosomes will have good parts of old chromosomes and maybe the new chromosomes will be better. However, it is good to leave some part of population survive to next generation . Mutation probability says how often will be parts of chromosome mutated. If there is no mutation, offspring is taken after crossover (or copy) without any change. If mutation is performed, part of chromosome is changed. If mutation probability is 100%, whole chromosome is changed, if it is 0%, nothing is changed. Mutation is made to prevent falling GA into local extreme, but it should not occur very often, because then GA will in fact change to random search" [17].

Other Parameters

There are also some other parameters of GA. One also important parameter is population size. Population size says how many chromosomes are in population (in one generation). If there are too few chromosomes, GA has a few possibilities to perform crossover and only a small part of search space is explored. On the other hand, if there are too many chromosomes, GA slows down due to demanded more computation and memory. Research shows that after some limit (which depends mainly on encoding and the problem) it is not useful to increase population size, because it does not solve the problem faster.

CHAPTER 3

RESEARCH METHODOLOGY

In this chapter, more detail information about the implementation of GA into scheduling problem is presented. Then, the solution of the course-scheduling problem is presented for the Engineering Faculty of Atılım University using genetic algorithm approach. The information entered at the input stage of the system and data structures that store this information and problem description have been discussed.

3.1 Problem Description

The problem of constructing course timetables for Atılım University consists of allocating the set of courses offered by the Engineering Faculty to time periods and classrooms in such a way that no teacher, student or room is used more than once per period and that room capacities are not exceed.

The problem addressed here deals with the assignment of the each course to the defined time periods for undergraduate program.

The Engineering Faculty building has only 34 rooms. Only 3 of these rooms don't have projector and 14 rooms are reserved from Faculty of Arts & Sciences and Management. Availability of classrooms is limited and this is the main reason for extending teaching hours using lunch break.

In Atılım University, undergraduate education is completed on semester basis. In this thesis, Engineering Faculty is examined for course scheduling program. There are 8 departments in Engineering Faculty. Each student can start their faculty program in any semester, after passed English Proficiency Exam. For this reason, each course can be opened in each semester. Curriculum consists 8 semesters. Dean's Office authorizing a time tabler schedules these courses. According to

curriculum of the program, a regular student enrolls 5 to 8 curriculum courses in one semester.

Courses are grouped as elective and must. Elective courses are grouped as non-technical and technical. Some courses consist class and lab hours, others only class hour. Course hours are divided into 2 or 3 blocks of 2 hours (sometimes 3 or 1 hours) during the week and specific class can be assigned to some courses. For example, IE206 and EE208 courses are lectured only in Ergonomi Lab. and EE Lab. B There are full-time and part-time lecturer in the university. Usually, part-time lecturers have strict time and day constraints.

Courses are given in 9 hours per day and 5 days a week (45 time slots). Depend on the density of courses, lunch breaks can be used as a time slots for a course. Class hours of a course can be grouped into blocks, forcing each block to be scheduled in different days. It is obvious that schedule of courses in a term curriculum and time slots assigned for a lecturer mustn't conflict.

3.2 Requirement Analysis

Before designing a course scheduling program, in order to better understand the current scheduling problem and design an effective system, we conducted several interviews with the administrative and academic staff who have the responsibility of course scheduling making process.

The administrative staff provided us sample scheduling reports of all departments and documentation from current system. Administrative staff talked about problems of scheduling process and how prepare schedule of university. He report that making schedule of part time lecturers is the hardest part of the scheduling process because of their strict time and day restiriction Allocation of service courses have also problematic structure because those course are offered to several departments. He expends 60-70 hours to prepare schedule of Engineering Faculty in a single term. Though this hard work, academic staff said that there was still confliction in classroom assignment. Administrative staff also have provided some information about the current system and it can be summarized as follows.

- The Dean office determines offered courses and number of students in each classroom.
- Department determines number of groups of a course.
- Information about who gives which course, course hour,course code, number of section, special classroom needs, given the administrative staff before the schedule prepared.
- In every term, number of section of a course is changed, so it is not possible to use old schedules.
- Initial scheduling data given by department has criticial value.
- Because of limited number of classrooms in the Engineering Faculy and high amount of course hours, the scheduling process becomes more complex.

3.3 Aim of the Study

The manual solution of the problem considering all constraints defined in Problem Description part usually requires several days or weeks. In addition, the solution obtained can be unsatisfactory due to the limited human oversight. Our aim in this thesis is to develop a flexible, intelligent software for timetabling that addresses all special features found in the Atılım University's Engineering Faculty, and to achieve this we used an evolutionary approach, namely Genetic Algorithm (GA).

From the problem description above, the created course scheduling software should satisfy the following requirements;

1. Lecturer can offer more than one courses, and these courses mustn't be scheduled at the same time
2. The courses must not be scheduled at the same time with other required courses in the same class.
3. Only one course can be assigned to a classroom at any point of time.
4. All allocated classrooms should be large enough to hold the students and the classroom and course type should match.
5. Lecturer can demand a lesson in some specific time slots or days.
6. Course hours can be grouped into blocks and each block of course must be in different days.

Moreover the system can be used as a decision support tool that provided near optimal solutions quickly, and rather than producing optimum solutions with any cost.

3.4 System Inputs Based on System Requirements

The system contains four kinds of objects during the input stage of the program. These are also predefined information of the scheduling system.

- Constraints Objects
- Lecturer Object
- Classroom Object
- Course Object
- Course-Section-Lecturer Triad Object

3.4.1 Constraint Object

In the scheduling process, there are several considerations to be taken into account such as course, student constraints and teaching staff preferences. In practice, automated time tabling problems have much more complex constraints and structures. In most of implementations, constraints are classified as hard and soft constraints [3].

While hard constraints provide operational feasibility of the schedule, soft constraints provide quality of the schedule. There are several criteria to measure the quality of a schedule and changes according to customer demand. It is not essential to make a good timetable. However, if it is considered, the better timetable will be produced. On the other hand, hard constraints include conditions that every acceptable timetable must satisfy. In this thesis, we allow both types of constraints.

Hard Constraints

These constraints must be fully satisfied for a timetable to be feasible. These are;

Q1. Lecturer can give more than one course, and these courses should not be scheduled at the same time.

Q2. Only one course can be assigned at the same time in the same class.

Q3. Courses for the same year students of a department cannot take place at the same time.

Q4. All allocated classrooms should be large enough to hold the students and the classroom and course type should match.

Q5. Lecturer can demand a lesson in some specific time slots or days (Part-time or administrative personnel)

Soft Constraints

These constraints are less strict constraints, which should be satisfied as far as possible, but if not satisfied they do not affect feasibility of the solution.

Q6. A course having no blocks should not be split into different days

Q7. Course hours can be grouped into blocks and each block of course must be in different days. For example, a course of 4 hours can be lectured in two blocks as in 2+2, 3+1 or in some situations as in one 4 hours.

Courses are formed in Engineering Faculty as below;

Total Hours of Courses	Groups
1	1
2	2 1+1
3	3 2+1
4	4 2+2 3+1
5	2+2+1
6	2+2+2 3+2+1
7	2+2+1+2

Figure 3.4.1 Course Time-Slots Groups

Q8. At noon between 12:30 and 13:30, at least one time period for each class should be available for the lunch

3.4.2 Lecturer Object

A lecturer is an entity, where all the information about each academic staff is stored. The predefined information about the lecturer is:

- Lid (auto increment value)
- Lecturer Name (Lname)
- Lecturer Department (Ldepartment)

Field	Type
Lid	int(3)
Lname	varchar(20)
Ldepartment	varchar(35)

Indexes:

Keyname	Type	Cardinality	Field
PRIMARY	PRIMARY	127	Lid
Lname	UNIQUE	127	Lname

Figure 3.3.2: Data Structure of Lecturer Object

3.4.3 Classroom Object

A classroom is an entity, where all information about it is stored. The following list shows these predefined information:

- Classroom ID (Clid)
- Classroom Name (Clno)
- Classroom Equipment(Projector/Tepegöz) (Clequipment)
- Classroom Capacity (Clcapacity)

Field	Type
Clid	int(3)
Clno	varchar(25)
Clequipment	varchar(35)
Clcapacity	int(3)

Indexes:

Keyname	Type	Cardinality	Field
PRIMARY	PRIMARY	50	Clid
Clname	UNIQUE	50	Clno

Figure 3.4.3: Data Structure of Classroom Object

3.4.4 Course Objects

A course is an entity, where all the information about courses is stored. The following list shows these predefined information:

- Course ID (cid)
- Course Name (cname)
- Course Code (ccode)
- Department of Course (cdepartment)

Field	Type
cid	int(3)
cname	varchar(60)
ccode	varchar(11)
cdepartment	varchar(35)

Indexes:

Keyname	Type	Cardinality	Field
PRIMARY	PRIMARY	166	cid
cname	UNIQUE	166	cname
ccode	UNIQUE	166	ccode

Figure 3.4.4: Data Structure of Course Object

3.4.5 Course-Section-Lecturer Triad Object

"Course-Section-Lecturer" object is an entity, where stores schedule specific course-section-lecturer-classroom information. . The following list shows these predefined information:

- Class
- CourseName
- CourseHour
- Classroom
- Lecturer
- Lwithid
- Lecturerid
- Courseid
- Classroomid
- Section
- check_bit
- prePreferredDay
- prePreferredHourStart
- GroupCourse

Field	Type
Class	int(11)
CourseName	varchar(40)
CourseHour	int(1)
Classroom	varchar(25)
Lecturer	varchar(20)
Lwithid	int(3)
Lecturerid	varchar(15)
Courseid	int(3)
Classroomid	int(3)
Section	int(1)
check_bit	int(1)
prePreferredDay	char(10)
prePreferredHourStart	time
GroupCourse	varchar(1)

Indexes:

Keyname	Type	Cardinality	Field
PRIMARY	PRIMARY	82	Lwithid
Group	FULLTEXT	None	GroupCourse

Figure 3.4.5: Data Structure of Course-Section-Lecturer Triad Object

3.5 Applied Software Methodology

Waterfall

While software is developed, waterfall methodology was applied to the study. In waterfall model, the following steps are followed in order:

1. Requirements specification
2. Design
3. Implementation
4. Verification
5. Maintenance

In the requirements specification phase, in order to better understand the course scheduling problem in the Atılım University, interviews have been conducted with people who have been involved in the course schedule making process of the university. According to their contributions, system requirements were determined and software was designed. In the developed software, timetabling problem in the Department of Computer Engineering at Atılım University was used as a pilot study and it was tested. According to software testing outputs, all department in Engineering Faculty were added to scope of the study.

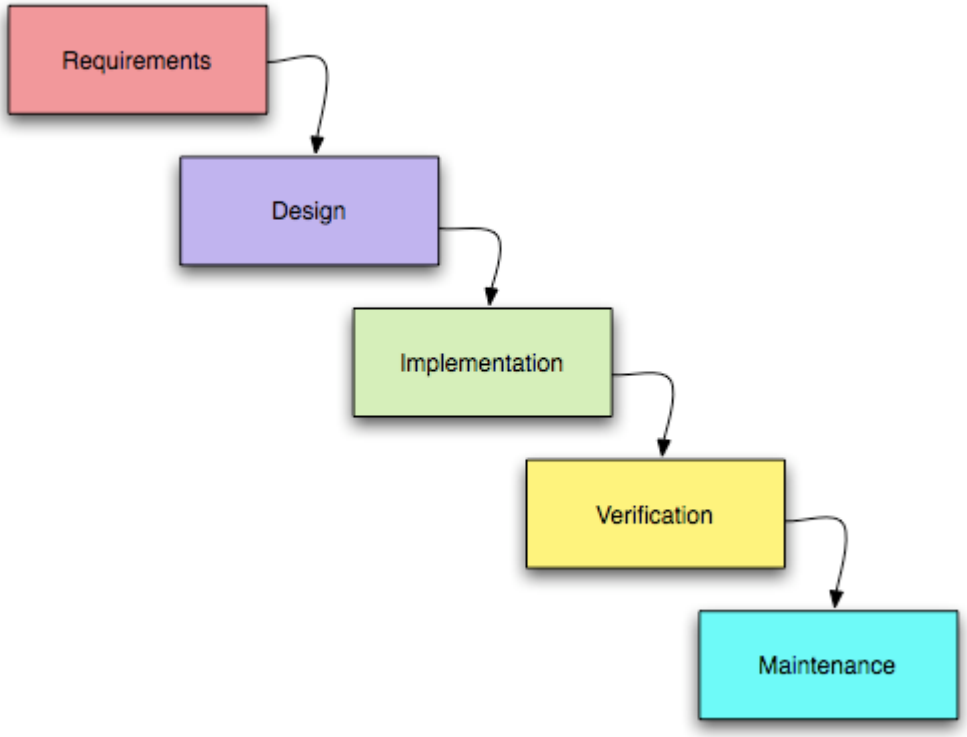


Figure 3.5: Flow Diagram of Waterfall Model

CHAPTER 4

DESIGN AND IMPLEMENTATION

4.1 Genetic Representation

GA work with a population of strings or chromosomes. The idea is to create population of individuals, each individual representing a possible timetable. In our implementation an individual represents a school timetable that consists of a sequence of all classes timetable of the school, as shown in Figure 4.1.

These chromosomes are constituted by genes, which consist of *Check_Bit, Class, Course_Hour, Section, Lesson_ID, Room_ID and Teacher_ID*. Genes include decimal strings and alphabetic characters and used for each activity in the timetable and stored in two-dimensional array. Our chromosome and gene structure is shown in Figure 4.1.

Figure 4.1 shows the genetic representation of our implementation, which is described as two-dimensional array. In this array, each of 40 x (z, k, t, r, m) elements represents course schedule of one department for one class, and each element stores series of section. Department courses are grouped according to their class and section. So, in the chromosome, 'section 1' part includes courses opened in only one section. If one course is opened in two sections, genetic representation for second section of these courses is written into 'Section 2' part of chromosome. In addition, if a course has four hours a week and it needs to be divided into 2+2, its genetic representation has placed 2 different genes in the chromosome.

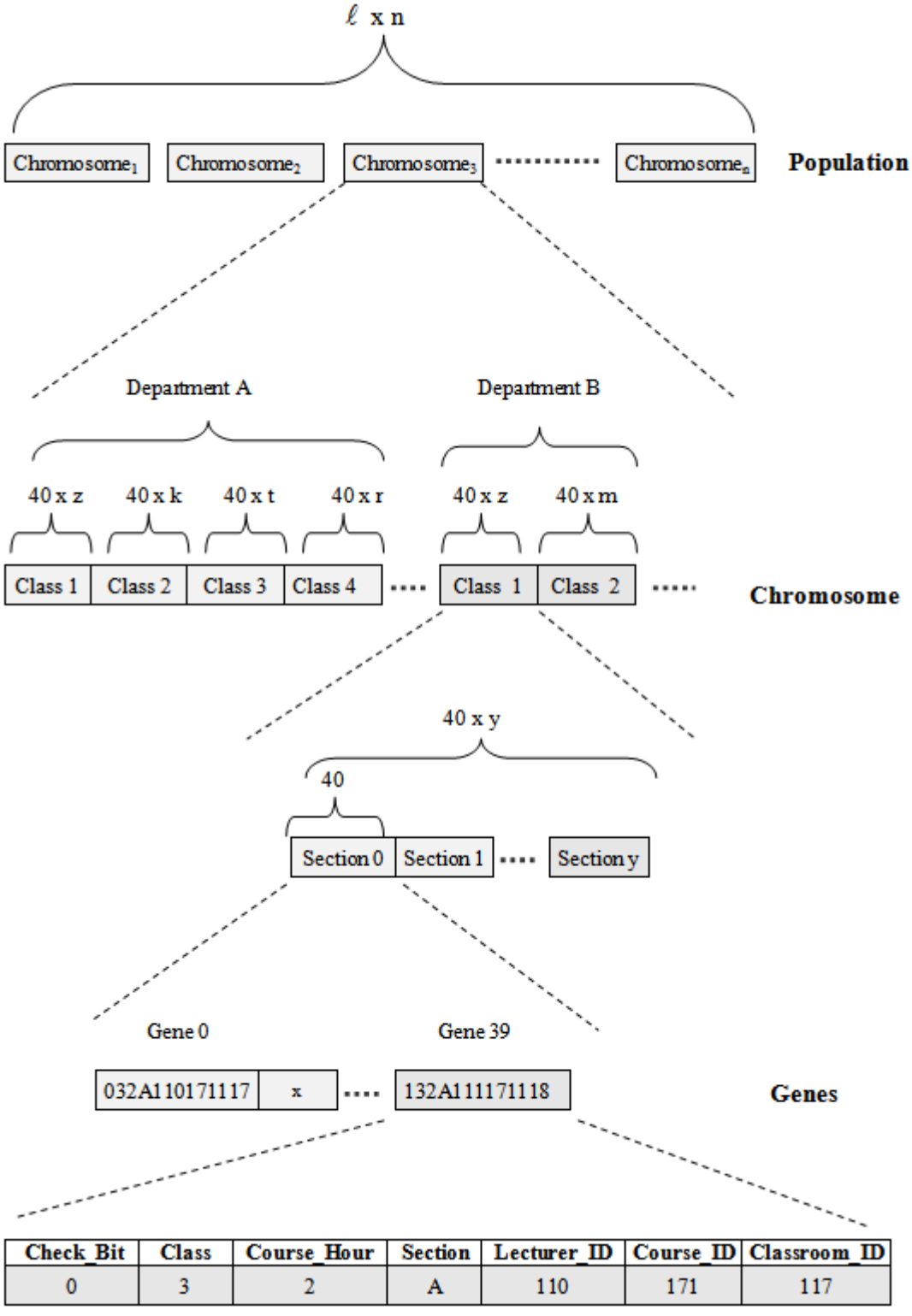


Figure 4.1: Genetic Representation

4.2 Initialization of the Population

In general, there are two issues to be considered for population initialization of GAs: the initial population size and the procedure to initialize the population [19]. Recent studies have shown that satisfactory results can be obtained with a much smaller population size. While population size increases, demanding memory and time also increase [20].

Secondly, there are two ways to generate the initial population: heuristic initialization and random initialization. Although heuristic initialization finds solutions faster, it just explore a small part of the solution space and never find global optimal solutions because of the lack of diversity in the population [21]. Therefore, random initialization is used in this thesis.

Therefore, the random initialization determines population of candidates' solution and chooses genes from the database in a random manner during the encoding process. Total chromosome length depends on number of department and number of section for each course of a department.

A=Total Chromosome Length

B= Daily Course Hour

C= x Number of Total Section

$$A = B \times C \times (5\ Days)$$

$$Number\ of\ Total\ Section = \sum_{n=1}^{n} D_n$$

$$D_n = Number\ of\ Section\ in\ n^{th}\ Department$$

Total chromosome length consists of daily 8 hours and 5 days.

4.3 Cost Function

Cost function value is calculated by giving penalties to the assignments, which not fit defined constraints, and it will be the sum of the penalties given to the constraints violations. As the constraints are satisfied, the cost function decreases. It cannot be zero, because soft constraints cannot be fully satisfied.

Cost function used for this problem is following:

$$\sum_{k=0}^{t}\sum_{i=0}^{n}\sum_{j=0}^{l} P_k * C_{ij}$$

Notation

l = Number of gene in chromosome

P_k = Penalty values of k^{th} constraint

C_{ij} = Gene of j which penalized in chromosome i

n = Number of chromosome in population

t = Number of constraints

4.4 Fitness Function

The penalty values of hard constraints are higher than from soft constraints. Thus the optimization process tries to generate a feasible solution with as few soft constraints violations as possible.

The fitness function is formed as follows;

$$(f) = \frac{1}{1+\left(\sum_{k=0}^{t}\sum_{i=0}^{n}\sum_{j=0}^{l} P_k * C_{ij}\right)}$$

Fitness of j^{th} chromosome in current generation as follows;

$$(f_j) = 1 - \frac{Total\ Penalty\ Value\ of\ Chromosome\ j}{\sum_{k=0}^{t}\sum_{j=0}^{l} P_k * C_{ij}}$$

4.5 Selection

We used Roulette-Wheel selection method while forming the next generation and following steps applied;

- Fitness of all the current generation members was summed up(f_{sum})
- A random number R was chosen between 0 and f_{sum}
- Fitness of the current generation members was added together one by one until the value of random number R is reached in between the two values of the $f_{sum.}$
- The last added individual was selected and copied to the new generation.
- This selection mechanism was continued until N individuals have been selected.

4.6 Crossover and Mutation

Multi-point crossover has been used in this study, where each class timetable is selected randomly with probability α from one parent or the other with. Parent (parents1 and parents2) chromosomes exchange their genetic information to form new generation's offspring [Figure 4.6.2]. For mutation, two genes have been swapped for each class in a random position between n and n+40 with probability β [Figure 4.6.1].

α =Crossover rate (taken from user interface)
β =Mutation rate (taken from user interface)

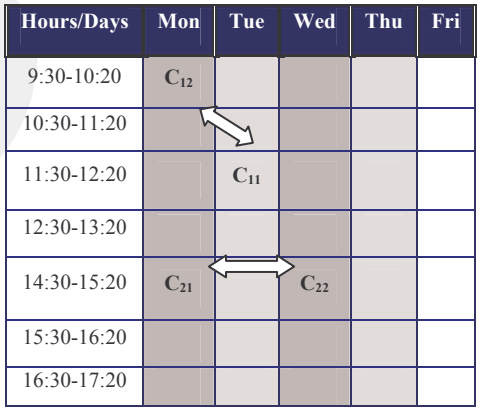


Figure 4.6.1: Process of mutation operator

Parent 1 **Parent 2**

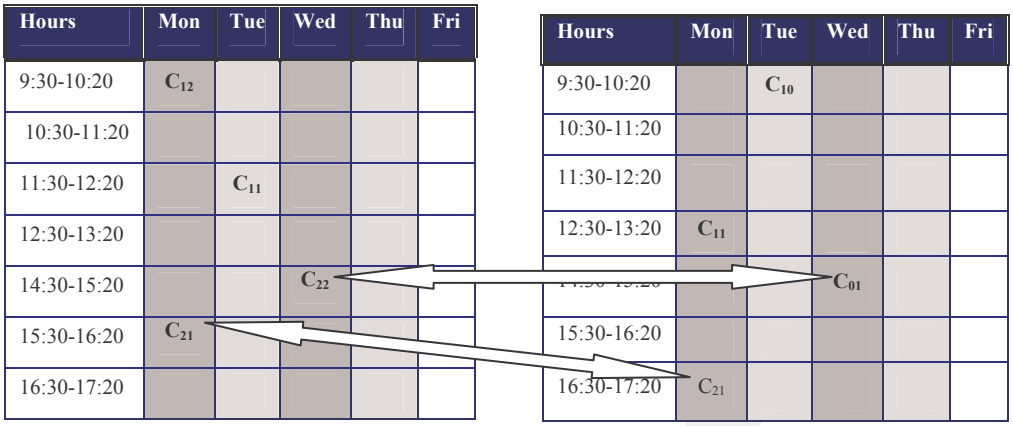


Figure 4.6.2: Process of crossover operator

4.7 Repair Operator

Repair operator is used to fix some with invalid solution for the problem. Because, the new gene structures formed after crossover and mutation turn into unsuitable gene structures. This operator use problem specific knowledge about the problem domain to repair missing or extra genes in chromosome.

Hours/Days	Mon	Tue	Wed	Thu	Fri
9:30-10:20	C_{12}				
10:30-11:20					
11:30-12:20		C_{11}			
12:30-13:20					
14:30-15:20	C_{21}		C_{22}		
15:30-16:20					
16:30-17:20					

Figure 4.7.1: After the Repair

Hours/Days	Mon	Tue	Wed	Thu	Fri
9:30-10:20	C_{12}				
10:30-11:20					
11:30-12:20		C_{11}		~~C_{11}~~	
12:30-13:20				~~C_{11}~~	
14:30-15:20	C_{21}		C_{22}		
15:30-16:20	~~C_{32}~~				
16:30-17:20					

Figure 4.7.2: Before the Repair

4.8 Elitism Operator

With this operator, the best chromosome in the current generation is preserved in the next generation. First, the chromosomes are sorted according to their fitness and the best one is chosen as next generation's first chromosome.

4.9 System User Interface

Graphical User Interface (GUI) and GA part of the program are developed using Java programming language. It interact with the MYSQL database. There are 16 tables in the databases.Figure 4.9 shows workflow diagram of program. All GA parameters and inputs are entered in to the system using corresponding GUI. Then program is work for finding the optimum timetable with the given inputs using GA.

Program allows user to enter the following GA parameters: the size of population, mutation rate, crossover rate, weight of soft and hard constraints.

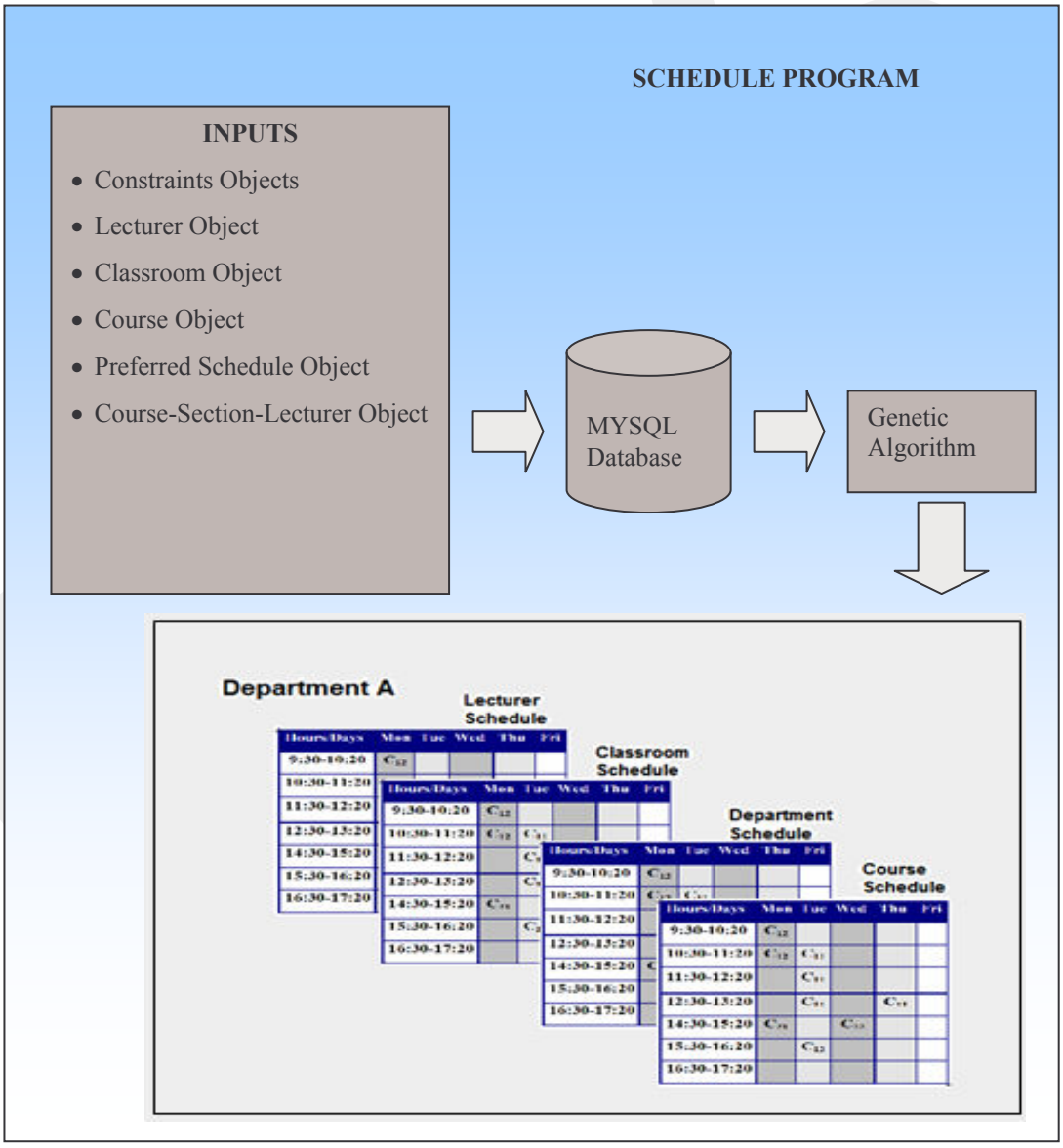


Figure 4.9: Workflow diagram of program

The system has developed using java programming language. The user interface is designed with eight different windows, first five tab for input parameters, sixth and seventh tab for parameter settings and running the program and last for viewing the results as timetables.

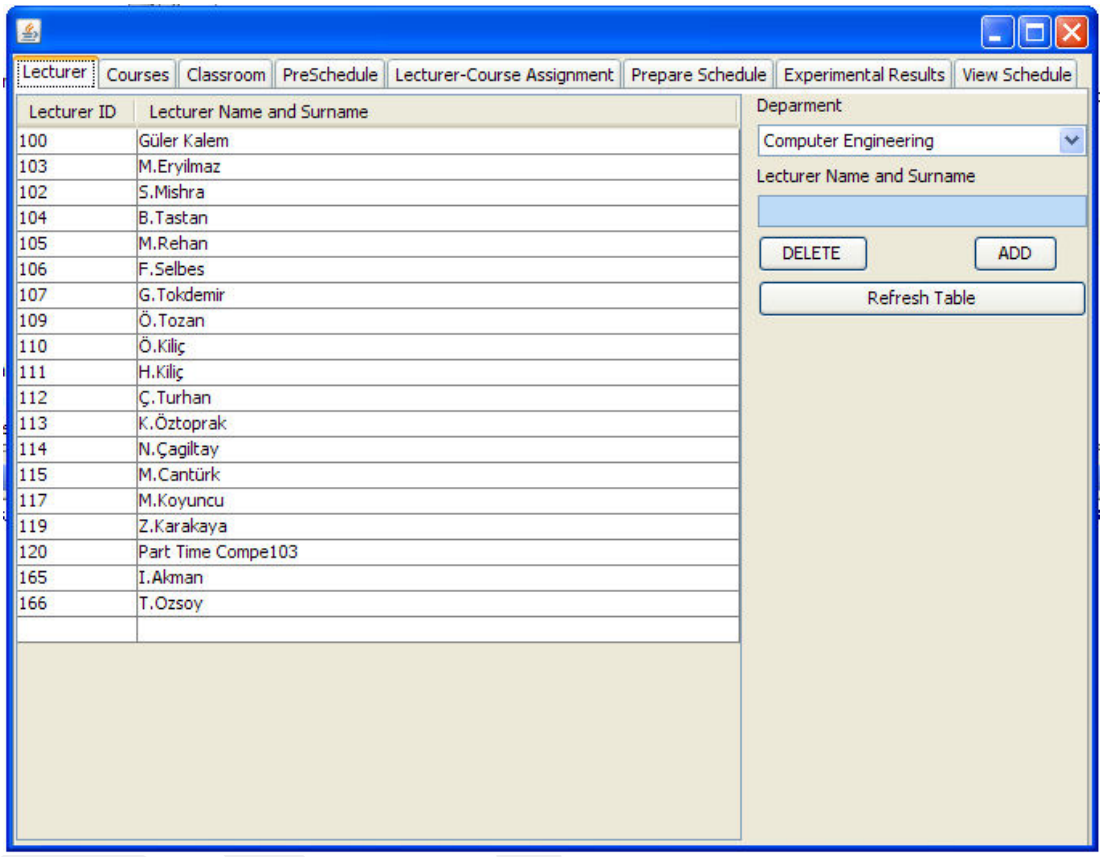


Figure 4.9.1: Main Window

Windows for input parameters have 5 tab sheets ,namely “Lecturer”, ”Courses”, ”Classrooms”, ”Preschedule”, ”Lecturer-Course Assignment”.

With the “Lecturer” tab sheet (Figure 4.9.1) name of the lecturer and department information is entered. To delete a lecturer, it must be selected from the table and “DELETE” button must be pressed. Also, some checking parameters have been added the program to prevent inputting duplicated values.(Figure 4.9.2). Lecturers can be grouped according to their department by changing department information in department combobox.

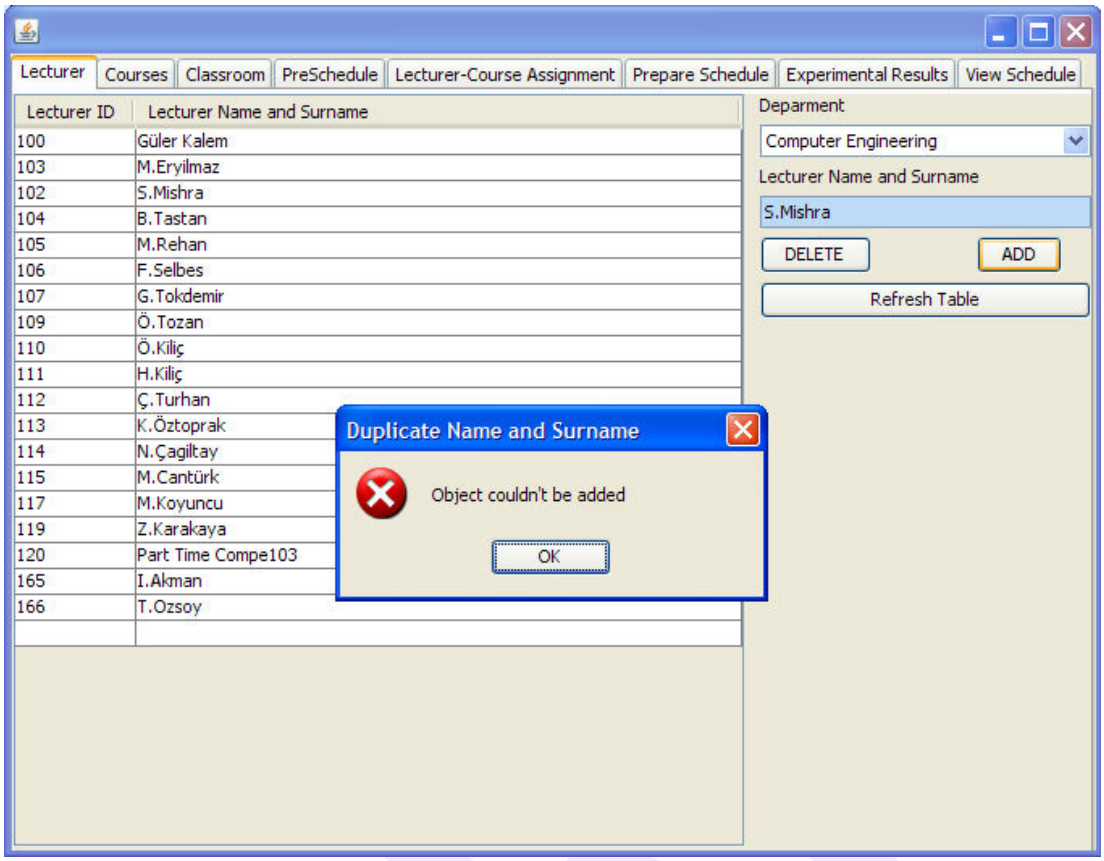


Figure 4.9.2: Duplicated Data

With the "Courses" tab sheet (Figure 4.9.3) Course Name, Course Code and Department information is entered. Delete and adding process are the same as with "Lecturer" tab. Courses can be grouped according to their department by changing department information in department combobox.

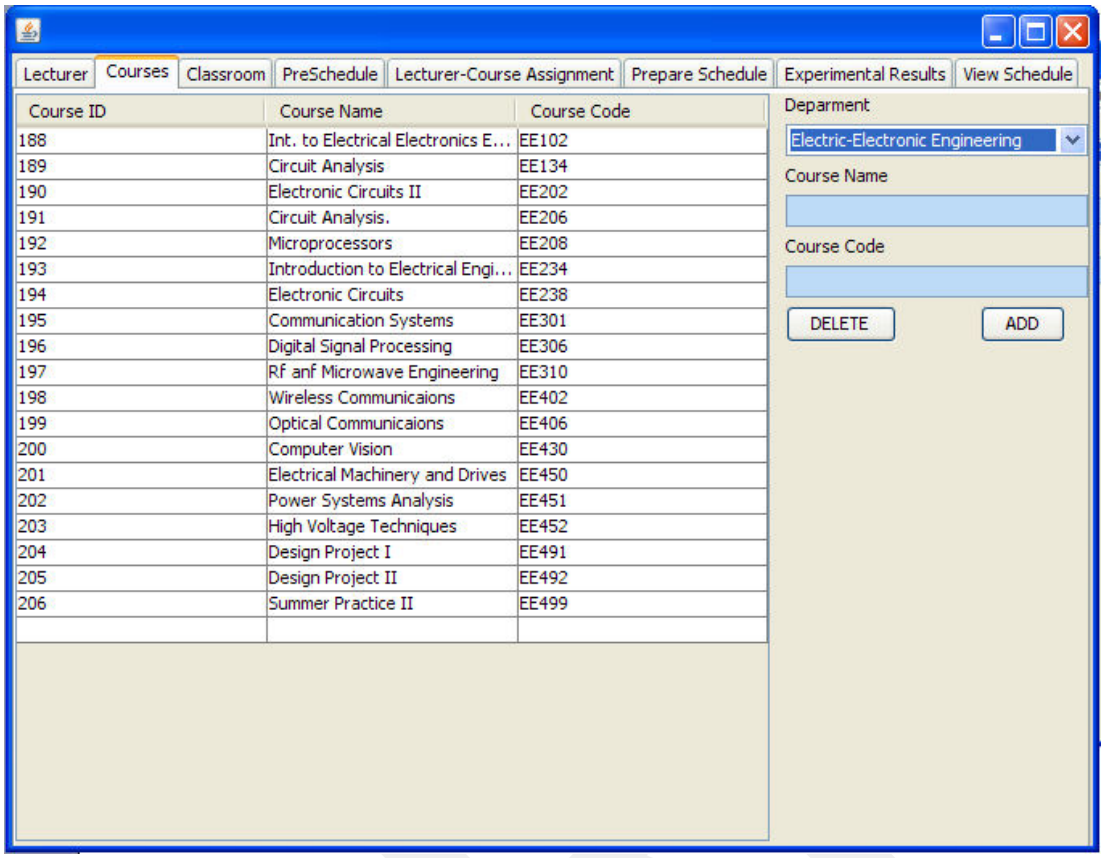
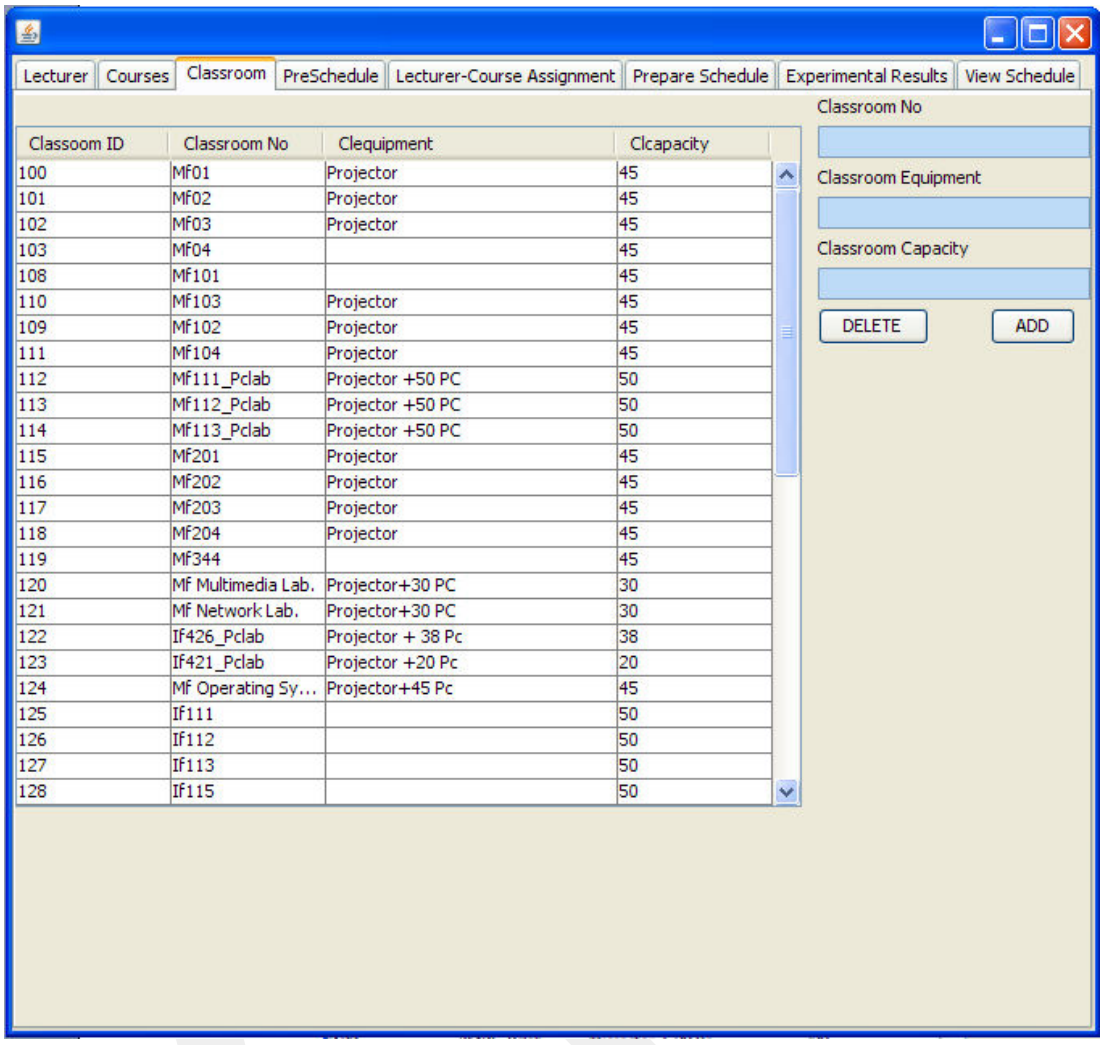


Figure 4.9.3: Courses Tab

With the "Classrooms" tab sheet (Figure 4.9.4) Classroom No, Classroom Equipment and Classroom Capacity is entered. Delete and adding process are the same as with other tab. In addition, classroom object has checking mechanism to prevent inputting duplicated values.



Classoom ID	Classroom No	Clequipment	Clcapacity
100	Mf01	Projector	45
101	Mf02	Projector	45
102	Mf03	Projector	45
103	Mf04		45
108	Mf101		45
110	Mf103	Projector	45
109	Mf102	Projector	45
111	Mf104	Projector	45
112	Mf111_Pclab	Projector +50 PC	50
113	Mf112_Pclab	Projector +50 PC	50
114	Mf113_Pclab	Projector +50 PC	50
115	Mf201	Projector	45
116	Mf202	Projector	45
117	Mf203	Projector	45
118	Mf204	Projector	45
119	Mf344		45
120	Mf Multimedia Lab.	Projector+30 PC	30
121	Mf Network Lab.	Projector+30 PC	30
122	If426_Pclab	Projector + 38 Pc	38
123	If421_Pclab	Projector +20 Pc	20
124	Mf Operating Sy...	Projector+45 Pc	45
125	If111		50
126	If112		50
127	If113		50
128	If115		50

Figure 4.9.4: Classroom Tab

With the "Preschedule" tab sheet (Figure 4.9.5); prescheduled course-lecturer assignment is entered according to preferred day (Preferred Day), preferred time (Preferred Hour), classroom and lecturer, department, section and class information of courses. Therefore, lecturer can demand a lesson in some specific time slots and days. Delete and adding process are the same as with other tab.

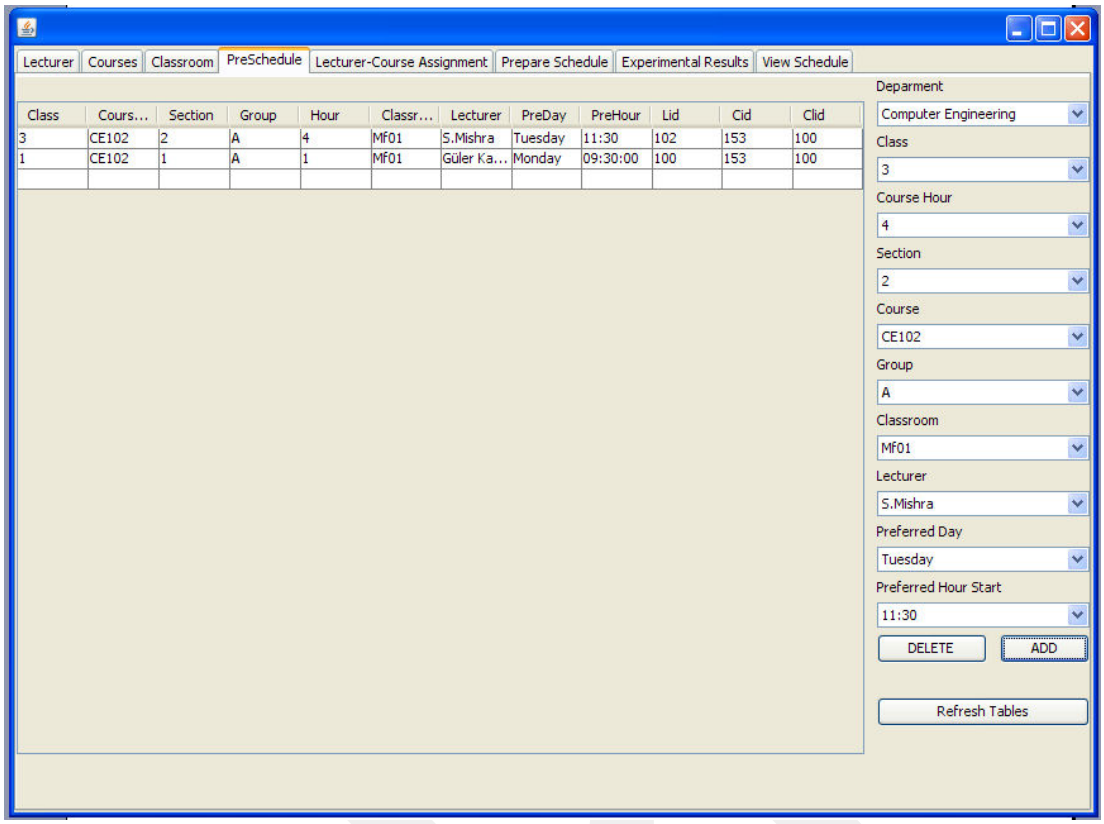


Figure 4.9.5: Preschedule Tab

With the "Lecturer-Course Assignment" tab sheet (Figure 4.9.6); course-lecturer assignment is entered according to lecturer, course, group, classroom, course hour, department, section and class information of courses. Delete and adding process are the same as with other tab.

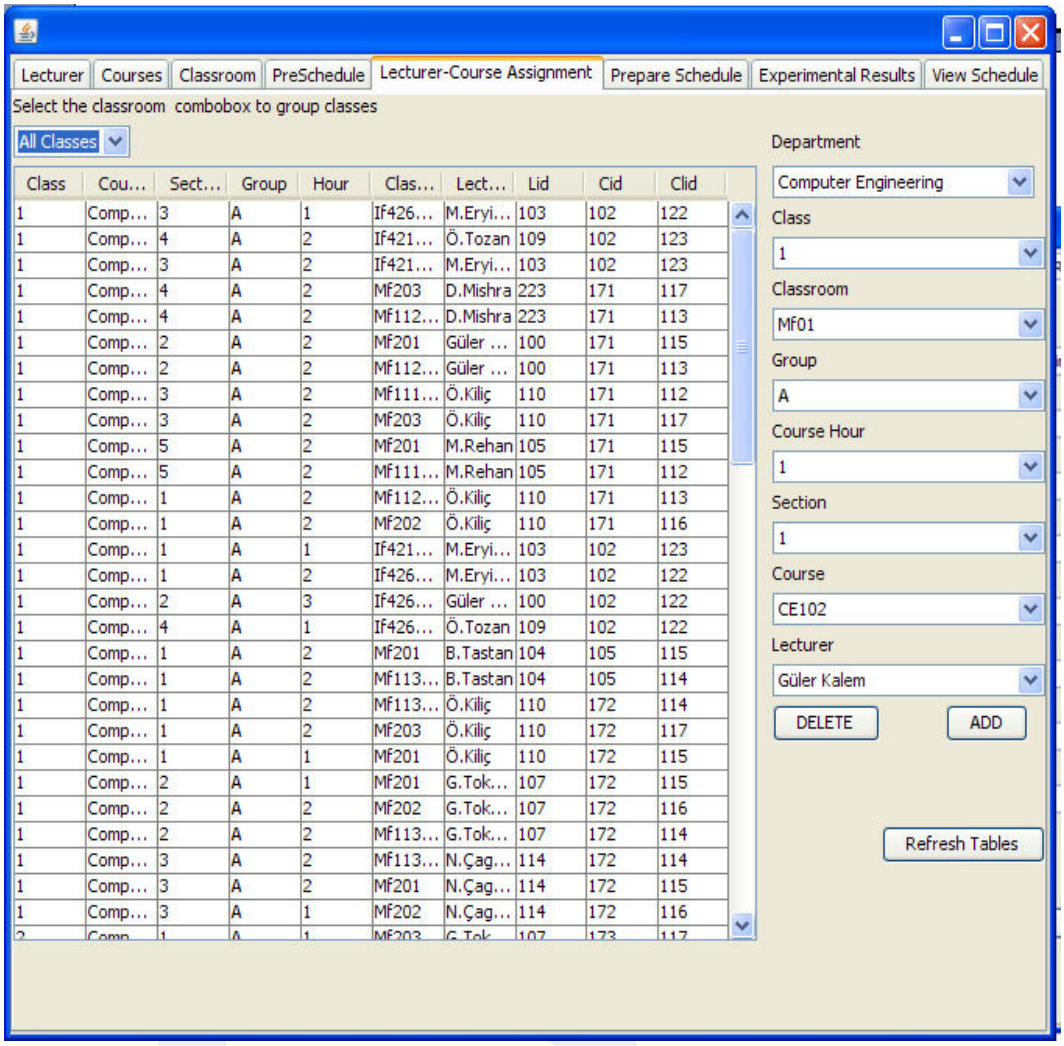


Figure 4.9.6: Lecturer-Course Assignment Tab

With the "Prepare Schedule" tab (Figure 4.9.10), parameters of the algorithm are entered and can be modified. The execution of the genetic algorithm is started in this tab. The algorithm can be restarted with the new values by clicking "Restart with New Values" button. "Select Most Fit" button is used to stop algorithm and to select most fit solution. With the"Penalty Value" and "Fitness" charts, changes in fitness and penalty values can be seen graphically.

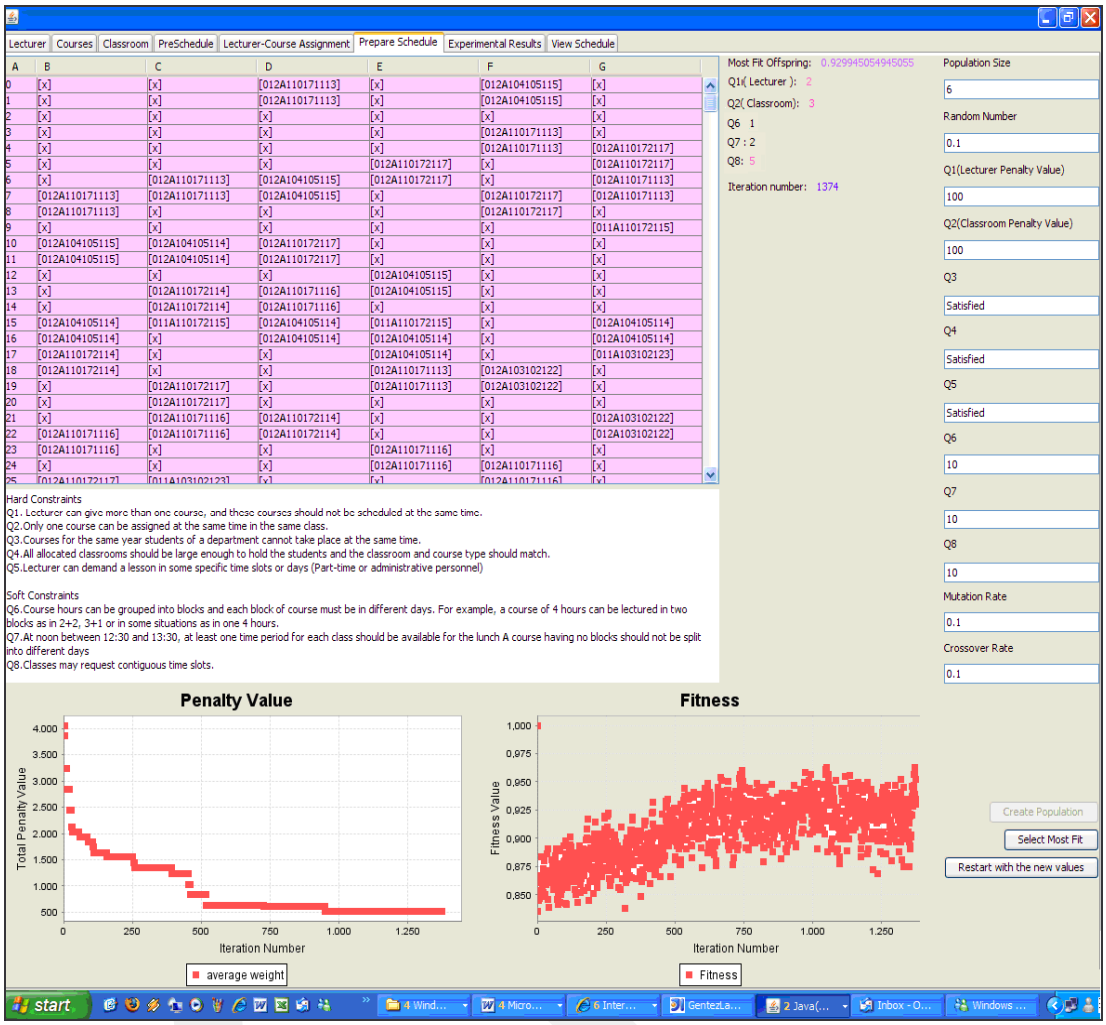


Figure 4.9.10: Population Tab

With the "Experimental Results" (Figure 4.9.11) tab penalty values of each spring for first generation, penalty values of each spring for last generation and array list of last generation can be seen in table.

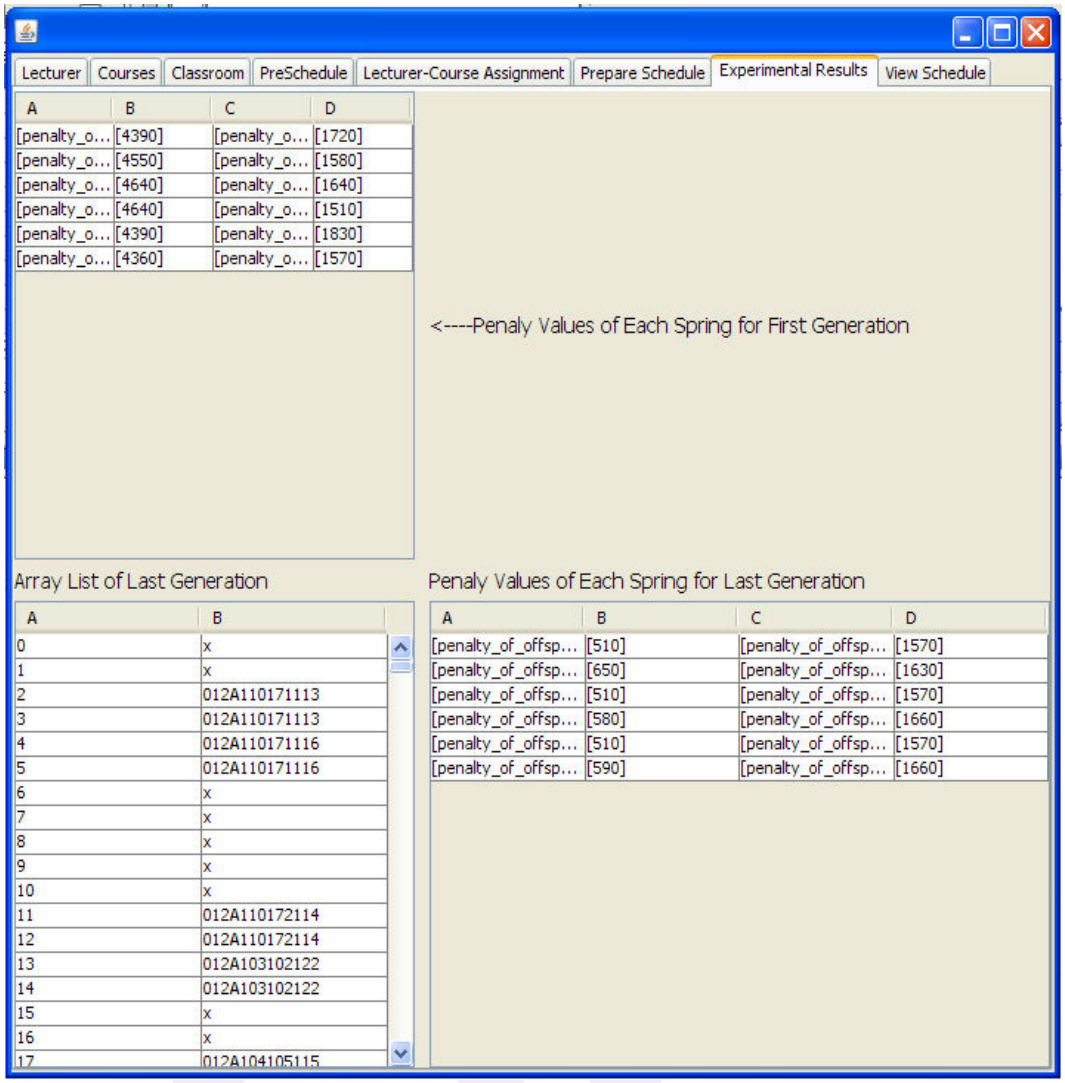


Figure 4.9.11: Experimental Results Tab

"View Schedule" tab (Figure 4.9.12) can be named as schedule generating module of the program and produces the following reports:

- The schedule for each department depending on its term and section
- The classroom's schedule
- The course's schedule
- The teacher's schedule

By clicking "Schedule ..." button, schedule report can be seen in the related table. In addition, if a course, lecturer or classroom has a confliction; conflicted item can be seen in the text pane with its conflicting item.

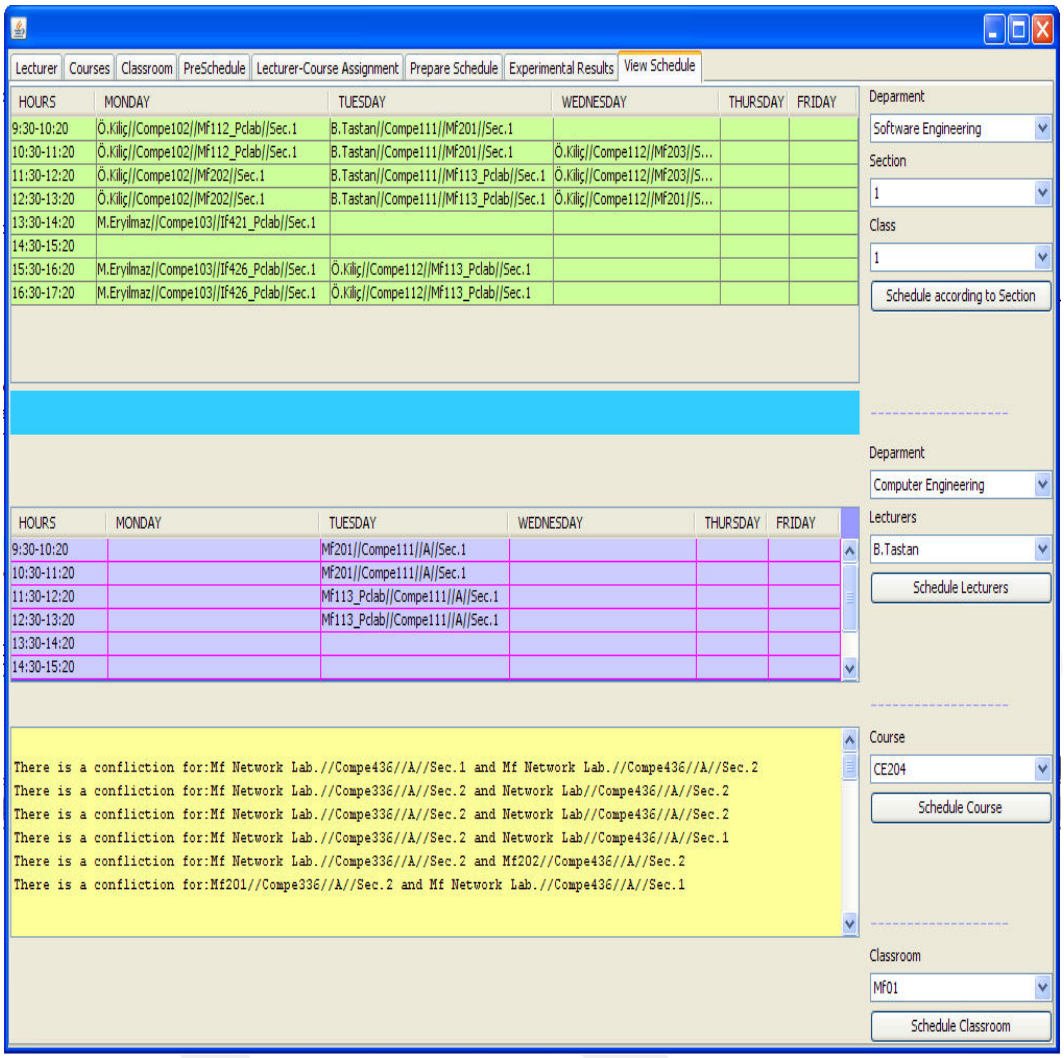


Figure 4.9.12: View Schedule Tab

CHAPTER 5

EXPERIMENTS and RESULTS

All tests were carried out on an Asus Workstation- Intel Pentium Core Duo-1 Gb RAM system running under Windows XP. The algorithm was first applied to the real data of 2007-2008 year 2nd semester of Computer Engineering Department. After several test runs, developed program found the optimum solution for Computer Engineering Department. In all test runs, the weights of penalties given to constraints, mutation rate, crossover rate and random number were the same with the data used in scheduling for all departments. Penalty Values of best solution for each constraint is showed in Table 5.

Then the algorithm was applied to 2007-2008 year 2nd semester of all departments in Engineering Faculty without considering service courses .In input stage of the test, all courses in the curriculum of Engineering Faculty were entered. Totally, there were 288 offered courses (145 sections), 127 lecturers, 50 classroom and 40 class hours to be assigned to timetable slot. Summary of GA input parameters is given in Table 1.

Hours	Mon	Tue	Wed	Thu	Fri
9:30-10:20	1	9	17	25	33
10:30-11:20	2	10	18	26	34
11:30-12:20	3	11	19	27	35
12:30-13:20	4	12	20	28	36
13:30-14:20	5	13	21	29	37
14:30-15:20	6	14	22	30	38
15:30-16:20	7	15	23	31	39
16:30-17:20	8	16	24	32	40

Figure 5.1: Division of a week into time slots

Each day of the week is divided into 8 hours consisting of eight 50 minutes time slots

as shown in Figure 5.1. There should be 10 minutes breaks in between courses.

Resources	Value
Lecturers	127
Department	8
Classrooms	50
Number of Offered Courses	288
Weekly time-slot of Timetable	40
Total time-slot of Timetable	320

Table 1: Characteristic of experimental data

The weights of penalties given to constraints are listed in Table 2. Fitness function uses these values. Penalties given to the hard constraints were chosen high so that search forces the solution to go to the feasible value as far as possible. By changing the penalties, constraints to be satisfied can be prioritized. We observed that genetic algorithm attempts to satisfy constraints having higher penalty values among the others. Constraint Q3, Q4, Q5 (genes having values starts with 1) was embedded into the program and always satisfied in all generations.

Hard Constraints	Value
Q1*	**100**
Q2*	**100**
Q3*	**Satisfied**
Q4*	**Satisfied**
Q5*	**Satisfied**
Soft Constraints	**Value**
Q6*	**10**
Q7*	**10**
Q8	**10**

Table 2: Weights of Penalties for Hard and Soft Constraints

* Look at Soft and Hard Constraints section in Chapter 3

To determine the optimum parameter settings of the algorithm a number of tests were performed. The parameter settings used in all test are listed in Table 3. Best performance observed in these settings.

Parameters	Value
Population size	10
Chromosome size	2040
Mutation rate	0.05
Crossover rate	0.2
Random number (used in initialization)	0.1

Table 3: Parameters of GA

Experiment ID	Number of Generation	Total Penalty Value	Computation Time
1	11783	1050	623
2	11783	2500	620
3	11783	1400	678
4	11783	2070	658
5	11783	1200	623
6	11783	2050	670
7	**11783**	**740**	**654 minute**
8	11783	1300	695
9	11783	870	659
10	11783	800	666
11	11783	1020	655
12	11783	1000	672
13	11783	2070	645
14	11783	850	648
15	11783	930	685
16	11783	920	628
17	11783	1350	635
18	11783	780	616
19	11783	920	678
20	11783	770	659

Table 4: Experiment Results

Constraint	Value
Q1	100
Q2	500
Q3	0
Q4	0
Q5	0
Q6	40
Q7	30
Q8	70
For All Department	

Constraint	Value
Q1	0
Q2	0
Q3	0
Q4	0
Q5	10
Q6	0
Q7	10
Q8	
Only Computer Engineering	

Table 5: Penalty Values of best solutions

Graphical form of the total penalty value propagation belonging to Exp ID#7 (best result) is given in Figure 5.2.

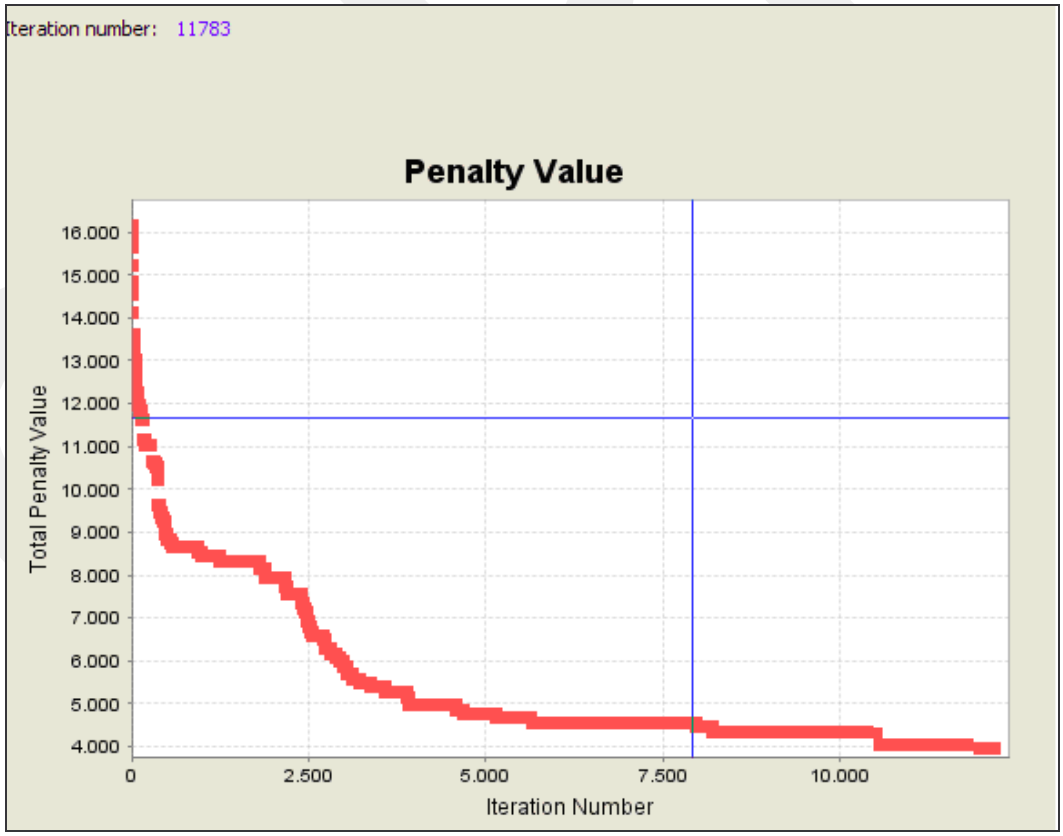


Figure 5.2: Total Penalty Value propagation in successful run (Exp. id#7)

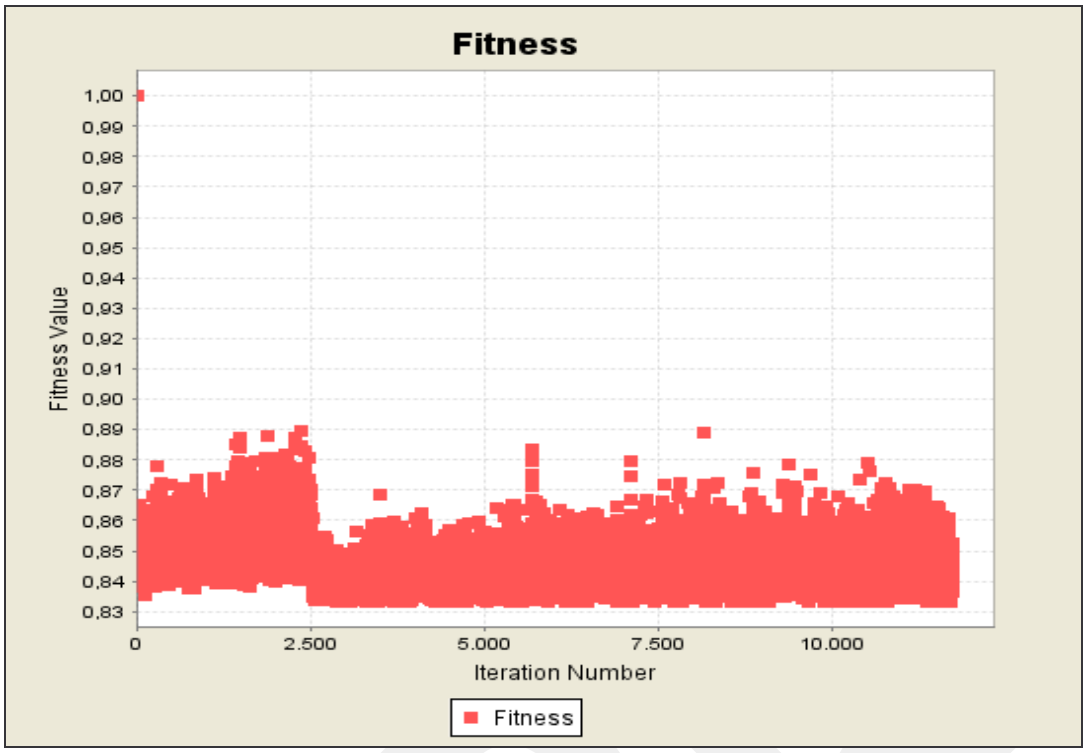


Figure 5.3: Fitness Value Propagation for best solution

When the results are analyzed, the hard constraint except Constraint Q2 (classroom conflict) are fully satisfied in the best solutions.

The best solution violated some soft and hard constraints and it is listed in Table 2.

Number of classroom is the one of the reason of violation of Constraint Q2. There are not enough classrooms in Engineering Faculty and it decreases the search space for optimum solution.

When the manually prepared timetable is analyzed, there are some lecturer and classroom conflicts despite of all care. It realized in term beginning and causes a lot of problem. But, in this study, automatically prepared schedule prevent this problem and give attention about conflicts after the schedule prepared.

Some automatically prepared Schedule examples are given in Figure 5.4, 5.5, 5.6, 5.7.

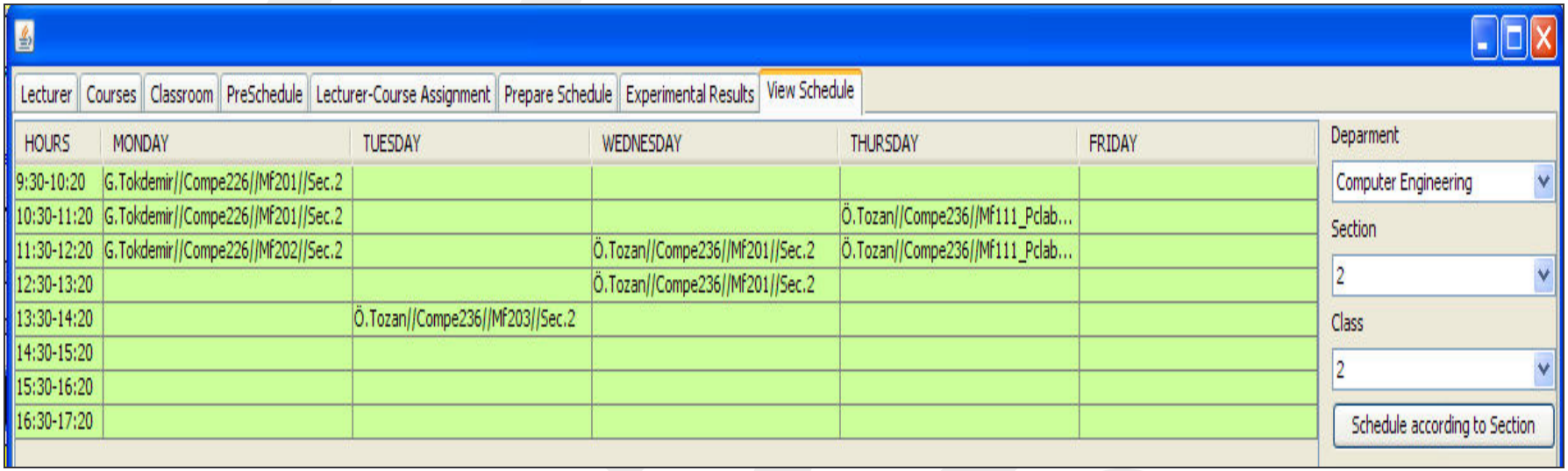


Figure 5.4: Course Schedule of Specific Courses in Computer Engineering (Class: 2, Section: 2)

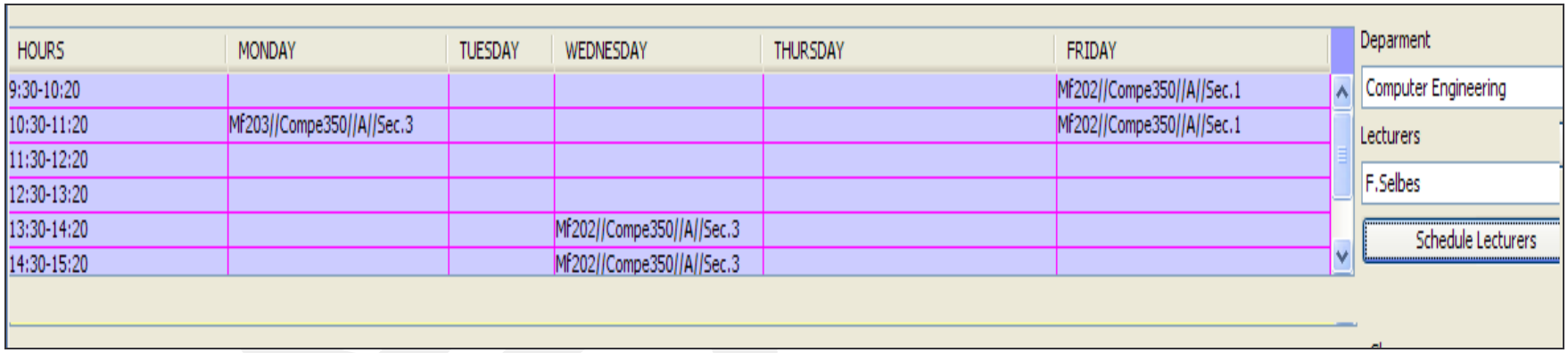


Figure 5.5: Course Schedule of Specific Lecturer in Computer Engineering

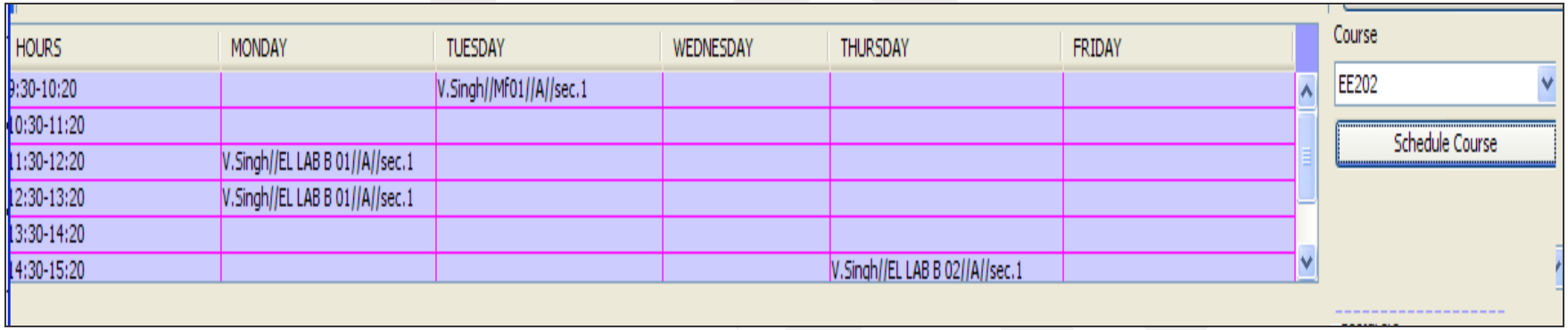


Figure 5.6: Course Schedule of Specific Course

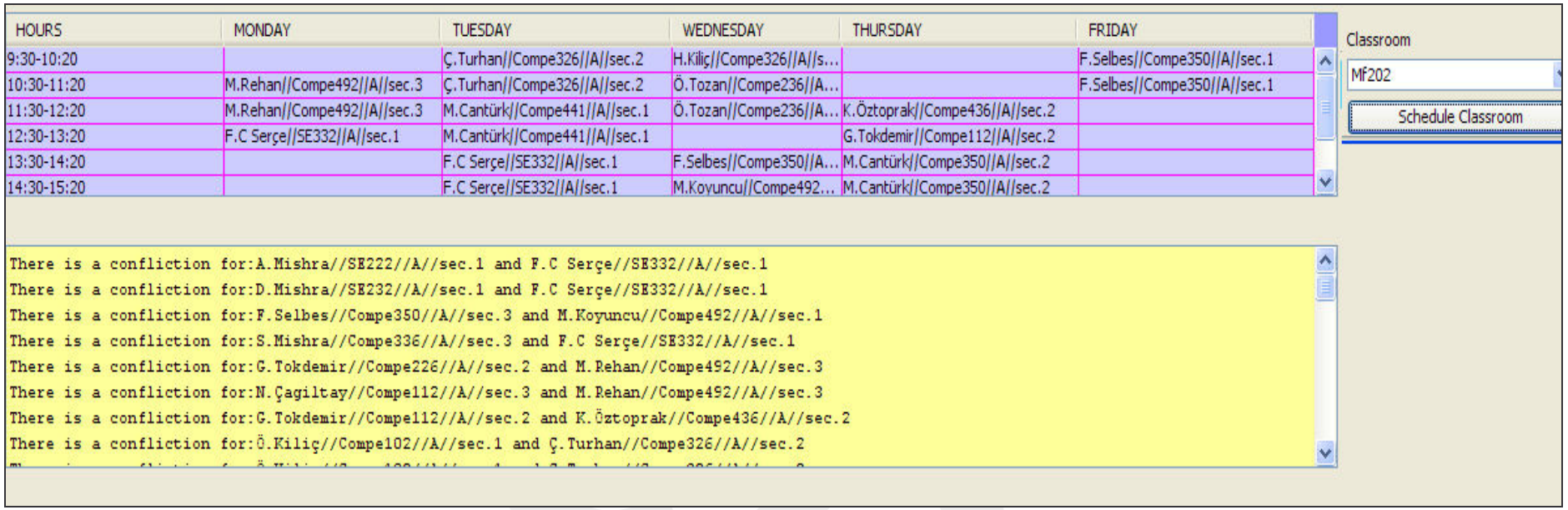


Figure 5.7: Course Schedule of Specific Classroom

CHAPTER 6

DISCUSSIONS & CONCLUSION

In this thesis, genetic algorithm was employed to solve course-scheduling problem. The components of genetic algorithm such as chromosome representation, mutation, crossover, selection, fitness function and convergence criteria are all designed and implemented successfully. In addition, a new operator named repair operator are developed to fix some invalid solution for the problem.

Altough the specific requirements of Atılım University, a general architecture was provided for solving any university scheduling problem. Developed program can be used also in course scheduling of any educational organization.

The timetables generated by developed algorithm are dependent on constraint defined in the problem. The hard constraints can be mostly satisfied and a feasible solution can be reached but satisfaction of the soft constraints is not quaranteed. “Lecturer can demand a lesson in some specific time slots or days” constraint can always satisfied.

We proposed that the developed system can be used as a decision support tool that provided near optimal solutions quickly and user have chance to select the most suitable timetable among the set of solutions.

FUTURE WORK

During and after the test runs, we have discovered some practical problems, which are to be implemented. As a future work, we have noted:

1. Schedule input and reports module can be modified to be user friendly. Excel output can be generated.
2. New constraints can be added algorithm to increase quality of solution.
3. Program can be modified to make classroom assignment automatically.
4. Distinct blocks of a course can be inputted automatically.

REFERENCES

[1] Burke, E., & Elliman, D., *Specialised recombinative operators for timetabling problems*, Lecture Notes in Computer Science, Springer Berlin / Heidelberg, Volume 993/1995,pp. 75-85, 2006.

[2] S. Even, A., & Shamir, A., *On the complexity of timetable and multicommodity flow problems*, MathSciNet, pp. 691–703, 1976.

[3] Cooper, T.B., & Kingston, J.H., *The Complexity of Timetable Construction Problem,* Proceeding Of the First International Conference on The Practice And Theory Of Automated Timetabling, pp. 283 – 295, 1995.

[4] MirHassania S.A., Carter,M.W., Laporte, G., *Improving paper spread in examination timetables using integer programming*, Recent Developments in Practical Examination Timetabling, Lecture Notes in Computer Science, Volume 179, Issue 2, pp 702-706, 2006.

[5] Burke E. K., Elliman D. G., & Weare R. F., *A University Timetabling System based on Graph Colouring and Constraint Manipulation* , Journal of Research on Computing in Education, pp. 1-18, 1994.

[6] Thompson & KA Dowsland, Variants of Simulated Annealing for the Examination Timetabling Problem, Annals of Operations Research, pp. 105-128, 1995.

[7] J Thompson and KA Dowsland, *General Cooling Schedules for a Simulated Annealing based Timetabling System*, in the Practice and Theory of Automated Timetabling, Springer-Verlag (Lecture Notes Computer Science), pp. 345-363, 1996.

[8] Burke E. K., Jackson K., Kingston J., Rupert W., *Automated University Timetabling: The State of the Art* ,University of Nottingham, University of Sydney,Computer Journal, pp 565-571, 1997.

[9] Boufflet J.P. & N`egre S., *Three Methods used to solve an Examination Timetable Problem*, in the Practice and Theory of Automated Timetabling, ed. EK Burke and P. Ross, Springer-Verlag (Lecture Notes in Computer Science), pp. 327-344, 1996

[10] Kang L. & White G.M., *A Logic Approach to the Resolution of Constraints in Timetabling*, European Journal of Operational Research, vol. 61, pp. 306-317, Elsevier, Science Publishers, 1992.

[11] Cheng C., Kang L., Leuung N. & White G.M., *Investigations of Constraint Logic Programming Approach to University Timetabling,* In the Practice and Theory of Automated TimeTabling EK Burke and P Ross, pp. 112-129, Springer-Verlag (Lecture Notes in Computer Science) , 1996

[12] Boizumault P., Delon Y., & P´eridy L., *A CLP Approach for Examination Planning*, Constraint Processing: Selected Papers, Lecture Notes in Artificial Intelligence, pp. 85-101, Springer-Verlag, 1995.

[13] Enzhe Y. & Sung K., *A Genetic Algorithm For A University Weekly Courses Timetabling Problem*, International Federation of Operational Research Societies, Journal of Blackwell S, pp.703, 2002

[14] NeuroDimension, 2002
(Available at http://www.nd.com/products/genetic/crossover.htm)

[15] Erben W. & Keppler J., *A Genetic Algorithm Solving a Weekly Course-Timetabling Problem*, Lecture Notes In Computer Science; Vol. 1153, Springer-Verlag , London

[16] Beasley D., Bull D.R.& Martin R.R., *Complexity Reduction using Expansive Coding in Evolutionary Computing*, Ed. T. C. Fogarty, 304-319, Springer-Verlag, 1994.

[17] Czech Technical University, September 1998, Marek Obitko.
(Available at http://www.obitko.com/tutorials/genetic-algorithms/parameters.php)

[18] Colorni A., Dorigo M., & Maniezzo V., *A Genetic Algorithm to Solve the Timetable Problem*, Politecnico di Milona, Italy, pp 90-95, 1990

[19] Goldberg, D.E., *Genetic Algorithm in Search, Optimization, and Machine Learning*, Addison- Wesley Publishing Company Inc, pages 372, Canada (1989)

[20] Hue X., *Genetic algorithms for optimization: Background and applications* Edinburgh Parallel Computing Centre, Scotland, pp 121-128 1997

[21] Harik G., Cant˘u-Paz E., Goldberg D. E., & Miller B. L., *The Gambler's ruin problem, genetic algorithms, and the sizing of populations*, pp. 231–253, 1999.

[22] de Werra D., *An introduction to timetabling*, Europan Journal Of Operations Research 19,pp. 151-162 1985