A comparative study of genetic algorithms and it's applications in secondary schools

By
Othman Omran Bulgahsaim

Supervisor:
Dr. Abdalhamed Abdalkafi

A thesis submitted to the department of Computer Science in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

Al-Tahaddi University
Faculty of Science
Department of Computer Science

Sirite, G. S. P. L. A. J.

Academic year 2009-2010

الرقم الإشاري ج ث س/ك02/ 152//2010 م

التاريخ

الموافق 2010/3/30 ميلادي

Faculty of Science

Department Of Computer Science

Title of Thesis

A Comparative Study of Genetic Algorithm with Application to School Timetable

By

Othman Omran Belgasem

Approved by:

Dr. Addlhamed M. Abdalkafi
(Supervisor) .....

Dr. Nage E. Bazina
(External examiner) .....

Dr. Zakaria Suliman Zubi
(Internal examiner) .....

Countersigned by:

Dr. Ahmed Farag Mhgoub

(Dean of Faculty of Science)

30 / 3 / 2010

Dedication

To My Parent souls, my wife and my Sons

(Who much suffered and stand beside me to fulfill this research) and to

my family and Friends

Acknowledgement

First and foremost, I would like to thank Allah for his mercy and kind to help me in completing this work. Next , I wish to express my sincere gratitude to Dr. Abdalhamed Abdalkafi for his supervision ,continued support , valuable information , precious guidance , and patience towards my long time taken in carrying out this work. Thanking his for useful discussions, and arguments which increased my analytical thinking and experience to accomplish this thesis. I would like to express my thanks to all staff in the Al-Tahadi University for their help and furnishing all facilities to complete this research I would also like to thank Mr.Mousa Ali for offering useful information in programming language. Also, I wish to thank Mr. Monir.Abd Allah for his valuable help and translations. Also, I would like to thank the responsible in our secondary schools to save the data and information which shared to succeed this work.

Othman O. Bulgasem

Abstract

This study illustrates the use of genetic algorithms (simple and parallel genetic algorithms) in school timetable and a comparative study on genetic algorithms between earlier studies discussed in order to create feasible and efficient timetables for Libyan secondary school timetable for each level alone. Different constraints can be applied to produce different timetables using single and multi crossover point. Earlier studies used GAs in solving school timetabling problems in different methods based on the genetic algorithm operators *(reproduction, crossover and mutation)* in order to reach a near optimal solution. The major difference between these studies is the chromosome encoding which is one of the most significant factors in order to coverage satisfactory and fast to a near optimal solution . Comparing these studies with the educational system in *Libyan secondary school* , it is convenient to design and implement the simple genetic based algorithm to obtain a school timetable for Libyan secondary school , where students groups, classes , time slots are fixed, and scheduling a set of subjects *(teachers)* over a set of time periods, while satisfying a wide range of constraints , the hard and soft constraints . In the suggested design more than one teacher can teach the same subject to different groups in different time slots avoiding clash problems *(subject clash problem or class clash problem)*. The GA operators will be applied for producing a new chromosome *(timetable)* in order to reach a near optimum solution

Table of contents

Dedication	ii
Acknowledgment	iii
Abstract	iv
Table of contents	v
List of figures	ix
List of tables	xi
CHAPTER 1: INTRODUCTION	1
1.1 Introduction	1
1.2 Overview of timetabling problem	2
1.2.1 Definition	2
1.2.2 Educational timetabling	3
1.2.3 School timetable	3
1.2.4 Timetabling difficulties	3
1.2.5 Traditional timetabling methods	4
1.3 Overview of genetic algorithms	4
1.3.1 Genetic algorithm in brief	5
1.3.2 How genetic algorithm works	5
1.4 Reproduction	7
1.4.1 Crossover	7
1.5 Applications of Genetic Algorithms	8
1.6 Using GAs in timetabling	9
1.7 Objectives of the study	9
1.7.1 The statement of the problem	10
1.8 Organization of study	11
	12
CHAPTER 2: LITERATUER REVIEW	13
2.1 Introduction	13

2.2 Simple genetic algorithm (SGA) 13
2.3 Parallel genetic algorithm (PGA) 14
1. Global PGA 14
2. Coarse-grained PGAs 14
3. Fine-grained PGA 15
2.4 Timetabling problem
2.4.1 School Timetabling 16
2.4.2 Timetabling Constraints 17
1. Hard Constraints 17
2. Soft Constraints 17
2.5 GAs Biological Terminology, operators and parameters 18
2.5.1 Biological Terminology 18
2.5.2 Operators and parameters 19
1. Encoding of Chromosomes 19
2 Fitness function 19
3. Selection operator 20
3.1 Selection Conceptual 20
4 Crossover operator 20
.4.1 One-point crossover 21
4.2 Two-point crossover 21
4.3 Uniform crossover 22
5 Mutation 23
6. Population Size 23
2.6 Repair Strategies 24
2.7 Difference between GAs and Traditional Methods 24
2.8 Genetic Algorithm in Timetabling 24
2.9 Comparative study of GA application in earlier study 27
CHAPTER 3: ANALYSIS SURVEY AND METHODOLOGY 28
3.1 Introduction 28
3.1.2 Nature of the problem 28
3.1.3 Current system 30
3.1.4 Requirement of a good timetable 31

3.2 Methodology 32
3.2.1 Problem definition 32
1. Hard constraints 33
2. Soft constraints 33
3. Constraint data 33
4. Clash problem 34
5. Parameters in representations 35
3.2.2 The Algorithm 35
1.Genetic algorithm steps 35
2. Mathematical model 36
3. Chromosome encoding 38
4.Constraints 41
4.1 Hard constraints 41
4.2 Soft constraints 41
4.3 Expectations and Preferences for the Students 41
4.4 Expectations and Preferences for teachers 43
4.5 Constraints satisfaction 43
5 .The repair-function 44
6. Evaluation 45
6.1 The fitness-function 45
1. How to calculate (RC) 46
7. Selection
8. Crossover 48
1. One-point crossover 49
2 .Two –point crossover 50
3. Multi point crossover 50
9. Mutation 51
10. Population size 52
CHAPTER 4: SYSTEM DESIGN
4.1 Introduction 53

4. 2 Design of LSTT_GA system
1. The use case diagram 54
1.1 Data entry 55
1.2 Create initial population 55
2. The class diagram 56
3. The sequence diagrams 63
4. 3 Initialization 65
4. 4 Calculating the clashes 67
CHAPTER 5: IMPLEMENTATION 68
5.1 Genetic Algorithm Modules 68
1. Selection Module 68
1.1 Random selection 68
2. Crossover Module 69
2.1.One Point Crossover 69
2.2.Multi point crossover 70
3. Mutation Module 71
5.2 User interface 72
1. School information interface 72
2 . constrains value interface 73
3. Subjects information interface 73
4. Main interface 74
5. Selection interface 74
6. Evaluation values interface 75
5.3. Weekly timetable Report 75
CHAPTER 6: EXPERIMENTS AND RESULTS 76
6.1 Introduction 76
6.2 System Requirement and Program language 76
6.3 Evaluation Strategy 77
6.4 Problem description 78
6.5 Experiments
6.6 Evaluation results 79
6.7 Population size 79

6.8 crossover 80
6.9 Best solution 80
6 .10Conclusion 87
6 .11 Future Work 87
Reference 88
Appendix

List of figures

Figure		Page
1.1	Evolution flow of the Basic genetic algorithm	6
1.2	One point crossover	8
1.3	Two point crossover	8
2.1	One point crossover [29]	21
2.2	Two Point Crossover [29]	22
2.3	Parameterised uniform crossover [29]	22
3.1	Result real example of chromosome	41
3.2	One point crossovers process	49
3.3	Two point crossover process	50
3.4	Multi point crossover process	51
3.5	Mutation (selected random gene)	52
4.1	The Use case diagram	54
4.2	The Class diagrams for LSTT_GA SYSTEM	61
4.3	The class diagram with relationships for LSTT_GASYSTEM	62
4.4	The Sequence diagrams for data entry	63
4.5	The Sequence diagrams for create initial population	64
4.6	The Sequence diagrams for generate new population	64

4.7	Result of initialization chromosome	65
5.1	Random selection algorithm	68
5.2	One point crossover algorithm	69
5.3	Multi point crossover algorithm	70
5.4	Random mutation algorithms	71
5.5	School information interface	72
5.6	Soft constrains interface	73
5.7	Subjects information interface	73
5.8	Main interface	74
5.9	Selection interface	74
5.10	Evaluation values interface	75
5.11	Weekly timetable Report for one group interface	75
6.1	best fitness of population size (50,100,200,500) and generation 40	80
6.2	The best chromosome implementation (20 generation, population 100)	81
6.3	The best chromosome implementation (40 generation, population 100)	82
6.4	The best chromosome implementation (65 generation, population 100)	83
6.5	chromosome implementation (100 generation, population 100)	84
6.6	result of comparative between one & multi point crossover with (20 , 40 , 65 and 100) of population size 100	85
6.7	The weekly school timetable obtained from different generations (20 , 40 , 65 and 100) of population size 100	86

List of tables

Table		Page
3.1	The result of survey in details	29
3.2	The percentage of constraint	31
3.3	Results total periods per a week	32
3.4	Relationship between subject and teacher	34
3.5	Class clash problem	35
3.6	Chromosome representation	39
3.7	The real example of chromosome representing	40
3.8	Students Expectation and Preferences for the Timetable	43
3.9	Teachers Expectation and Preferences for the Timetable	44
3.10	Table (3.10) illustrates how to calculate (R_C)	46
3.11	Table (3.10) illustrates how to calculate (P_C)	47
3.12	Table (3.12) shows different fitness	48
4.1	The relationship between subjects and classes	66
4.2	Relationship between teachers, subjects and classes	66
6.1	Total periods per week and subjects used in the experimental	78
6.2	The best fitness of population size (50,100,200.500) and generation 40	80
6.3	The result of one & multi point crossover with generation 20 and population 100	82
6.4	The result of one & multi point crossover with generation 40 and population 100	83
6.5	The result of one & multi point crossover with generation 65 and population 100	84
6.6	The result of one & multi point crossover with generation 100 and population 100	85
6.7	The result of competition crossover schemes	86

CHAPTER 1: INTRODUCTION

This chapter provides an overview of the timetabling problem, its difficulties, the traditional methods used, and an overview of the Genetic Algorithms (GAs), their history and literature, their successful use and their operators and parameters and how the timetabling problem is solved using GAs.

1.1 Introduction

Timetabling problem is one of the most difficult practical optimization problem [25]. Timetabling problem is a scheduling problem, which shows what, where, and when event takes place. A practical timetabling problem usually involves many interacting constraints which must be satisfied and it is often impossible to specify there constraints and their interactions to obtain near optimal solutions for this kind of problem using a genetic algorithms (GAs) [2, 5, 29 and 4]. Genetic algorithms first specified by John Holland in the1960s, 1970s, were very successful in solving many complicated optimization problems [9, 25, and 1]. From that time many research have attempted to develop powerful techniques for solving such kind of problems depending on the basic genetic algorithms operators reproduction, crossover and mutation. The timetabling problem for educational organizations is usually refers to exam or course timetabling at a school or a university, there are three basic constituent parts of educational timetabling: course timetabling, exam timetabling and school timetabling [3,20] . This study shows comparative study on genetic algorithms used in school timetabling to depict a simple algorithm which can be applied on secondary Libyan school timetable.

1.2 Overview of timetabling problem

This section provides a brief overview of timetabling problems in organizations such as schools colleges and universities.

1.2.1 Definition

[3] defines timetabling problem as; *"The timetabling problem consists in fixing a sequence of meetings between teachers and students in a prefixed period of time(typically a week), satisfying a set of constraints of various types ".* Timetabling is a special case of scheduling.

"Timetabling is the allocation, subject to constraints, of given resources to objects being placed in space- time, in such a way as to satisfy as nearly as possible a set of desirable objectives". As Wren defines [20].

[13] defines timetabling problem as a form of scheduling problem. *"It involves processes which occur in almost all areas of our daily life, such as daily timetabling, school timetabling, or activities timetabling. Computer-based timetabling methods however, concern themselves more with simply finding the shortest timetable that satisfies all the constraints".*

1.2.2 Educational timetabling

Educational timetabling is the sub-class of timetabling for which the events take place at educational institutions. A.schaerf [3] classifies the timetabling problems into three main classes;

School timetabling: The weekly scheduling for all the classes of a school, avoiding teachers meeting two classes at the same time, and vice versa;

Course timetabling: The weekly scheduling for all the lectures of a set of university courses, minimizing the overlaps of lectures of courses having common students;

Examination timetabling: The scheduling for the exams of a set of university courses, avoiding overlap of exams of courses having common students, and spreading the exams for the students as much as possible.

1.2.3 School timetable

Timetabling problem is an example of complex optimization problem where the traditional search algorithms are not very effective in solving this problem. Usage of genetic algorithm in process of finding a "correct timetable" for an educational institution is one solution of this problem. Given sets of: *subjects S, teachers T, classes C and (hard constraints H, soft constraints, S)* [5, 8 and 12].

1.2.4 Timetabling difficulties

They are a large number of combination in which events (times, places and people) can be combined for even quite small constraints and produce an optimal solutions. In general there are difficulties shared among all kinds of timetabling problems [13], they are

1) A general algorithm for solving a polynomial time algorithm cannot be achieved easily.

2) The constraints and the conditions differ in each particular problem.

3) Some conditions and conditions in the real world-timetabling problem cannot be presented precisely.

4) Designing effective heuristics for optimal solution is very difficult.

1.2 .5 Traditional timetabling methods

For producing a timetable there are a number of ways and techniques some of these are [26, 28]:

- Graph coloring techniques.
- Direct Heuristic.
- Hybrid methods.
- Constraints based approach.
- Logic programming approach.

1.3 Overview of genetic algorithms

This section introduces genetic algorithm in brief, how it works and application specially in timetabling problem.

1.3.1 Genetic algorithm in brief

Genetic algorithms are search algorithm based on the mechanics of natural selection and natural genetic [7]. Genetic algorithms have been developed by John Holland, his colleagues and his students at the University of Michigan , Holland's monograph *"adaptation in natural and artificial system"*, published in 1975 is regarded as the

seminal work on genetic algorithms [30, 24 and 23]. Some of the key ideas introduced were:

- A population – based algorithm.
- Crossover, mutation inversion as operators.

Goldberg [7] bridged the gap between the simple genetic algorithm used in computer science and the set of very complex natural processes on which the genetic algorithm are based.

Mitchell [21] introduced the use of genetic algorithm as the search algorithm use in engineering and computer science.

1.3.2 How genetic algorithm works

In its standard form a simple GA that yield good results in many practical problems is composed of three operators [28].

1. Reproduction selection.
2. Crossover.
3. Mutation.

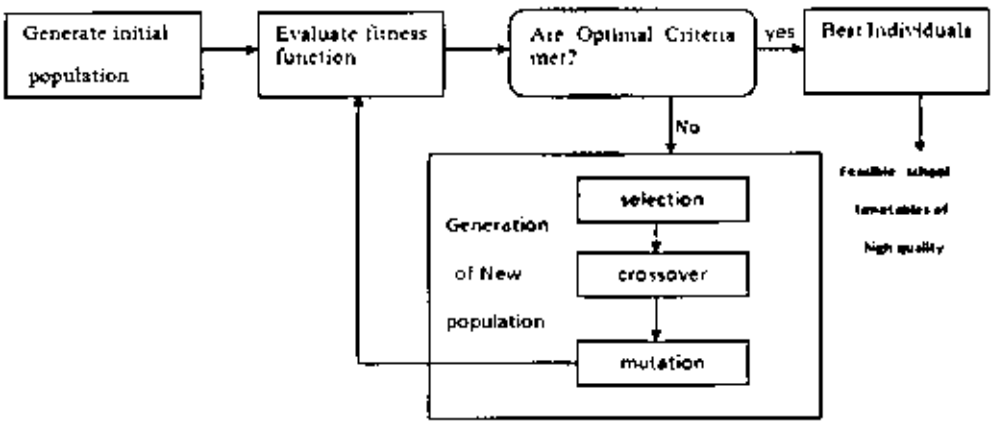


Figure (1.1) Evolution flow of the Basic genetic algorithm[12]

The basic algorithmic process of GAs goes as follows:

1) [Start] Generate random population of n chromosomes (suitable solutions for the problem).

2) [Fitness] Evaluate the fitness f(x) of each chromosome x in the population.

3) [New population] Create a new population by repeating following steps, until the new population is complete

a) [Selection] Select two parent chromosomes from a population

b) [Crossover] With a crossover probability, crossover the parents to form a new offspring (children).

c) [Mutation] With a mutation probability

the parents. This gives each individual a chance of passing on its genes without the disruption of crossover.

For binary encoding the crossover can look like this (↓ is the crossover point) .The following: Figures 1.2 and 1.3 shows one point, two point crossover works.

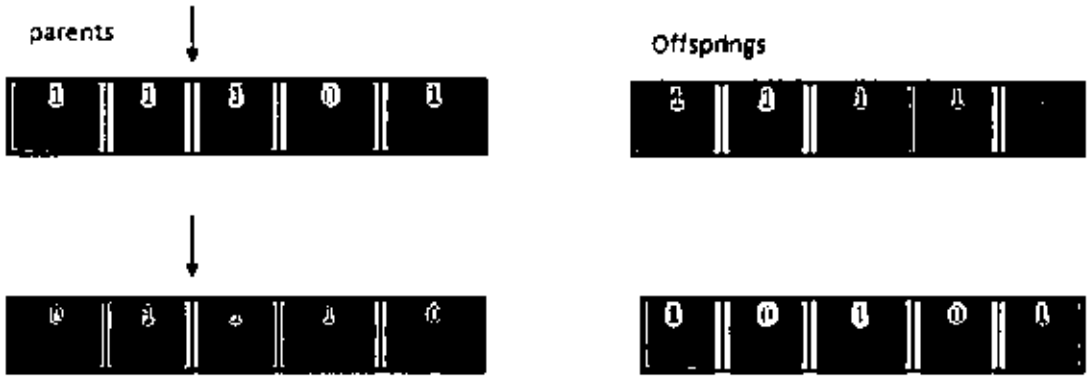


Figure (1.2) One point crossover

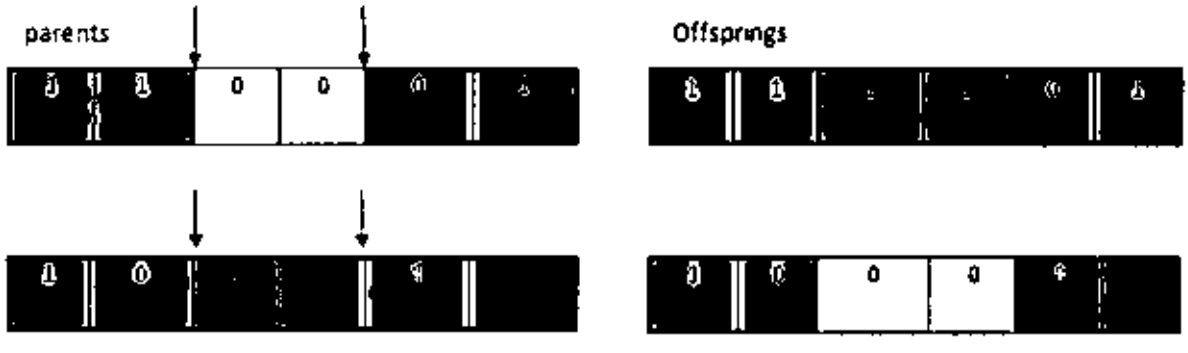


Figure (1.3) Two point crossover

1.5 Applications of Genetic Algorithms

A Simple Genetic Algorithm is very simple, but variations on this basic theme have been used in a large number of scientific and engineering problems and models [22]. Genetic algorithms are used in solving problems in many areas such as [28]:

- Distribution computer network topologies.
- Electronic circuit design.
- Mobile communications.
- Scheduling applications.
- Medical.
- Control.
- Telecom network.
- Robotics design.
- Strategy planning.

1.6 Using GAs in timetabling

From the definition of timetabling which is a problem of scheduling the events (times, places and people) & constraints, a timetabling problem can be thought of as a search space of optimal solution in the search space of possible timetables in this case a timetable [29,7].

1. Can be represented as a chromosome.
2. The timetabling fitness can be used to distinguishing the good timetable from the worse ones.

3. The timetabling problem is very large and multi-model search space.

For these reasons, the timetabling problem can be treated as an optimization problem, for these GAs optimization techniques can be used to solve it.

1.7 Objectives of the study

The aims of this study is to investigate the use of different *GA* operators in secondary school timetabling problem the purpose of this study are:

- A detailed introduction to the topics of GA (history, methods, variations ...)
- To introduce a comparative study of the earlier researches to obtain the optimal design for GA timetable problem.
- The design of a simple genetic algorithm *(SGA)* module for Libyan secondary school timetable problem *(LSSTTP)*.

1.7.1 The statement of the problem

- Collect the information about subjects, teachers, lab rooms and year of study from different secondary schools.
- Analyzing the collected information to specify the hard and the soft constraints.
- Define the relationship between the components of the problem (such as the relationship between subject and classes, subjects and teachers, etc.).
- Chromosome Representation.

- Design a questionnaire to obtain the degree of preferences of teachers and students.
- Design a repair function to repair any infeasible chromosome and make feasible.
- Design a fitness function to evaluate the timetable.
- Using a parent selection technique for selecting the parents from the population and then define the crossover module and mutation module.

1.8 Organization of study

Chapter One

Introduction

This chapter gives a general introduction and background about timetable problem and a brief overview about GAs their history, and their operators and parameters.

Chapter Two

Literature review

This chapter introduces the earlier and current works in school timetable.

Chapter Three

Analysis survey and methodology

This chapter illustrates the timetabling problem in Libyan secondary school's it is divided into three sections: Nature of the problem, Current system, Requirement of a good timetable.

Chapter Four

System design

This chapter demonstrates the design of Libyan school timetable using *GA* system *(LSTT_GA)* and discusses the algorithm used in addressing the evaluation of the timetabling problem with some detail and introduces in brief the method used in the analysis and design of the system *(UML)*.

Chapter Five

Implementation

This chapter illustrates the Genetic Algorithm Modules used for producing Libyan school timetable using *GA* system *(LSTT_GA)* .The modules used are selection module (random selection), crossover module *(One Point Crossover , multi point crossover)* and mutation module .

chapter Six

Experiments and results

The chapter present the experimental results obtained from using GAs to solve a real life Libyan secondary school timetable problem. The operators and the parameters used in the experiments were one point, multi point's crossover, mutation, and random selection technique and population size with different generation number.

CHAPTER 2: LITERATUER REVIEW

2.1 Introduction

This chapter introduces the recent studies used genetic algorithms (simple SGA and parallel PGA) for solving timetabling problem.

2.2 Simple genetic algorithm (SGA)

The Simple genetic algorithm (SGA) is easy to understand and to implement. The SGA in the standard form (using fixed undirected mutation, generational reproduction and crossover) is the algorithm that can be used against the other variations of GAs to compare [22]. Some of the main points of difference are noted below:

- Non-binary alphabet: The use of a non-binary alphabet to represent possible timetables as chromosomes necessitated some changes to the algorithm. Each P position on the chromosome represents an event (e.g. a subject). An integer representing the time and place in which the event has been scheduled is allocated to each slot. If there are C classes and T timeslots the value that can be assigned to a gene is an integer in the range {0,(C*T)-1}. This range of all possible gene values is called TP[29].

- The population (of size N) is initialized by producing N chromosomes with randomly chosen values from TP for each gene. A mutation of a particular gene is modeled as a change in the gene value to a randomly chosen integer in TP.

- Several crossover schemes, population sizes, and mutation strategies can be used.

2.3 Parallel genetic algorithm (PGA)

Evolution is a highly parallel process yet the traditional genetic algorithm is a serial one. As [7] notes:

"In a world where serial algorithms are usually made parallel through countless tricks and contortions, it is no small irony that genetic algorithms (highly parallel algorithms) are made serial through equally unnatural tricks and turns."

An interest in the parallel nature of adaptation and evolutionary algorithms dates back to Holland's earliest work. In recent years there have been attempts to produce parallel models for GAs which reflect some of the complexity of natural systems. The three main kinds of parallel genetic algorithm . The classification of PGAs presented below is based on that of [29].

1- **Global PGA**: Timetabling using cellular GAs with adaptive mutation operators . In this model the evaluation of the fitness of individuals and the application of genetic operators is explicitly parallelized, but the population is not partitioned in any way. That is to say each individual has the opportunity to mate with any other. In this sense the mating scheme is no different to that of the SGA – there is random mating.

2- **Coarse-grained PGAs**: The population is divided into sub-populations. The number of such populations is quite low (compared with the fine-grained model)

hence 'coarse-grained'. Each sub-population evolves independently using an SGA, with individuals being exchanged (randomly or using a selection method) between subpopulations in a process called 'migration'. The two main models for the coarse-grained PGA are:

i) **The island model:** The population is subdivided into sub-populations and migration can occur between any of the subpopulations.

ii) **The stepping stone model:** Population is partitioned in the same way as for the island model, but the concept of distance between sub-populations and consequently 'neighborhood's is introduced. Migration is restricted to neighboring sub-populations.

3- **Fine-grained PGA:** The population is divided into a large number of small populations, which may overlap. Although selection and mating occur within sub-populations, overlap between them allows high-fitness schemata to disseminate across the entire population. Fine-grained PGAs have been shown to form a sub-class of cellular automata and are also called 'cellular PGAs'.

2.4 Timetabling problem

The aim of timetabling problem is the assignment of classes to rooms and timeslots (periods) to satisfy the hard constraints and taking into consideration soft constraints.

"It is difficult to make a clear-cut distinction between acceptable and not acceptable timetables. Because of the large diversity in acceptance criteria, realistic timetable

construction problems are multidimensional; each dimension may introduce its own characteristic aspects that add to the complexity of the problem" [27].

In the 1990's a number of variants of the timetabling problem have been proposed, which differ from each other based on the type of organization involved (university or school) and distinct constraints. Manual solution may take days or weeks to solve the problem .

The timetabling problem is a form of scheduling problem. It involves processes which occur in almost all areas of our daily life , such as educational timetabling , school timetabling University / course timetabling , examination timetabling . Most of the early techniques and approaches were based on a simulation of the human way of solving the problem. However, computer – based timetabling methods concern themselves move with simply finding the shortest timetable that satisfies all the constraints [13].

2.4.1 School Timetabling (STTP)

The STTP is essentially the weekly scheduling of lessons in a school. Also known as the class/teacher model, the problem consists in assigning classes and teachers to periods in such a way that no teacher or class is scheduled more than once in the same time period provided the teaching requirements are satisfied. The teaching requirements here constitute the number of lessons per week that each teacher teaches each class. Each class consists of a set of students who follow a common curriculum. Many computer programs have been developed to solve the STTP but they perform well only under certain conditions.

2.4.2 Timetabling Constraints

We define the STTP as the problem of scheduling a set of lessons over a set of time periods, while satisfying a wide range of constraints.

There are two types of constraints, the soft and the hard constraints [5, 4 and 12]. The hard constraints must be satisfied to obtain the feasible timetable and are therefore compulsory restrictions. The soft constraints are not mandatory restrictions in that they do not cause the timetable to be unfeasible but represent those that would satisfy the class or the teacher. A good quality timetable should satisfy as many soft constraints as possible. From our study, we identified the following hard and soft constraints for the school:

1. Hard Constraints

- No participant (e.g. class, teacher) can be assigned to different events at the same period of time, i.e. no clashes can be tolerated.
- All lessons must be scheduled.
- There should be no periods in the class timetable when no teacher has been assigned to a class.

2. Soft Constraints

- No student meets more than three consecutive courses.
- There must not be spaces between lessons of the same subject for a class.

- For each teacher, the teaching requirements are specified, i.e. the maximum number of teaching hours is specified.
- For each teacher, the number of teaching hours per day should not exceed a specified limit.
- Teachers should not teach more than three consecutive periods in one day.
- For each subject, the lessons should be spread out evenly.
- Some lessons, e.g. Physical Education and certain laboratory classes are to be held early in the morning.

2.5 GAs Biological Terminology, operators and parameters

2.5.1 Biological Terminology

All living organisms consist of cells. In each cell there is the same set of chromosomes. Chromosomes are strings of DNA and serve as a model for the whole organism [4]. A chromosome consists of genes, blocks of DNA. Each gene encodes a particular protein. Basically, we can be said, that each gene encodes a trait, for example color of eyes. Possible settings for a trait (e.g. blue, brown) are called alleles.

Each gene has its own position in the chromosome. This position is called locus. Complete set of genetic material (all chromosomes) is called genome. Particular set of genes in genome is called genotype. The genotype is with later development after birth base for the organism's phenotype, its physical and mental characteristics, such as eye color, intelligence etc.

2.5.2 Operators and parameters

There are three major operations involved in evolving the population of a typical GA. In no particular order, these are selection, crossover and mutation [2, 22].

1. Encoding of Chromosomes

There are many ways to represent one timetable (chromosome), but to have following properties:

1- It has to contain complete information about teachers, groups, subjects.

2- It has to be possible to efficiently traverse the structure in order to check existence of constraint violations

3- It has to support efficient constructing and exchange of elements because of genetic operations crossover and mutations.

The chromosome encoding is one of the most important factors in order for the GA to converge satisfactorily and fast to a (near) optimal solution. Every chromosome must encode all the information needed for the description of a timetable. Encoding scheme assists the preservation of as many characteristics as possible of the timetable concerning hard constraints. [29]

2. Fitness function

The fitness function is the one of the most significant parts of the GA [12]. It decodes the chromosome into a timetable and returns a value representing the fitness of each specific chromosome. The fitness function, this is the function which checks if hard constraints and most of soft constraints are satisfied by each chromosome (school timetable). Each

constrain has value where this value represent how important this constrain to be in the chromosome. This value can be positive for constrains that are required to be in the chromosome and negative for constrains that are not preferred to be in the chromosome, the chromosome with the greater value represents the better a solution.

3. Selection operator

The question of how parents are selected from the population is fundamental to the operation of genetic algorithms. As with many other aspects of GAs there is no hard and fast rule regarding the choice of selection scheme.

3.1 Selection Conceptual

This is the procedure for choosing individuals (parents) on which to perform crossover in order to create new solutions. The idea is that the 'fitter' individuals are more prominent in the selection process, with the hope that the offspring they create will be even fitter still. Two commonly used procedures are 'roulette wheel' and 'tournament' selection. In roulette wheel, each individual is assigned a slice of a wheel, the size of the slice being proportional to the fitness of the individual. The wheel is then spun and the individual opposite the marker becomes one of the parents. In tournament selection several individuals are chosen at random and the fittest becomes one of the parents

4. Crossover operator

Offspring are formed by crossover of the genes from each of the parents. This traditionally uses two parents to produces two children. There are three common forms of crossover: One-point crossover, two-point crossover, uniform crossover [9, 25 and 16].

4 .1 one-point crossover

One-point crossover is the simplest form of crossover. A point is chosen (typically at random) on the parents and the segments of the chromosomes after the point are exchanged. This is best understood by means of a diagram [29]:

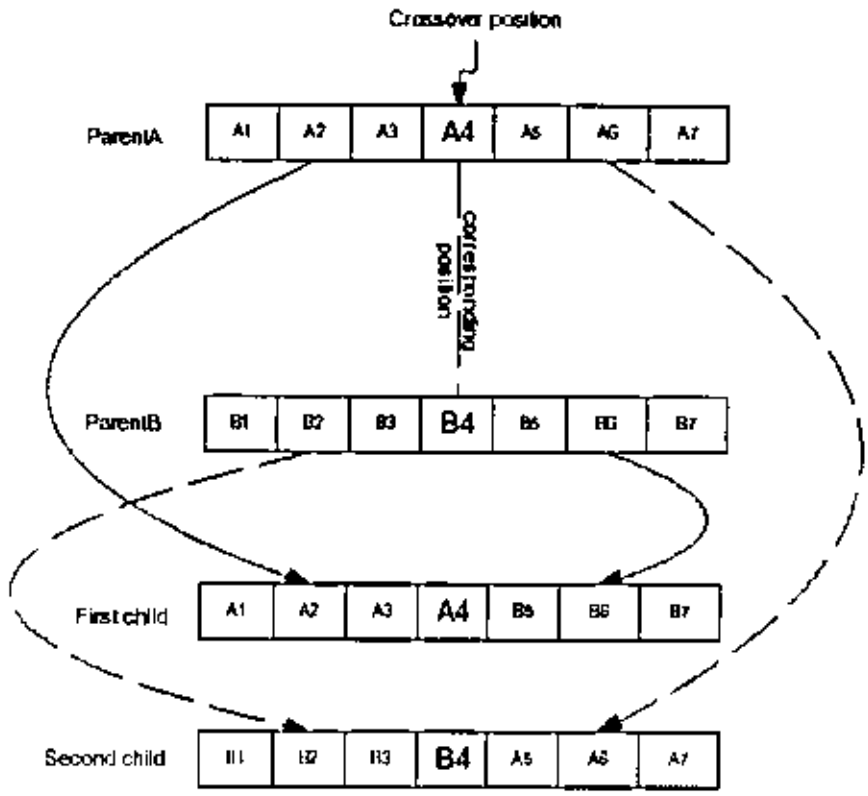


Figure (2.1): One Point Crossover[29]

4 .2 two-point crossover

Two points are chosen on the parent chromosomes at random and the segments between them exchanged [29]:

4 .3 Uniform crossover

Uniform crossover is the most open of the schemes: exchange takes place at every position on the chromosomes with a probability(p)that the genes will be exchanged ('Parameterised uniform crossover') [29].

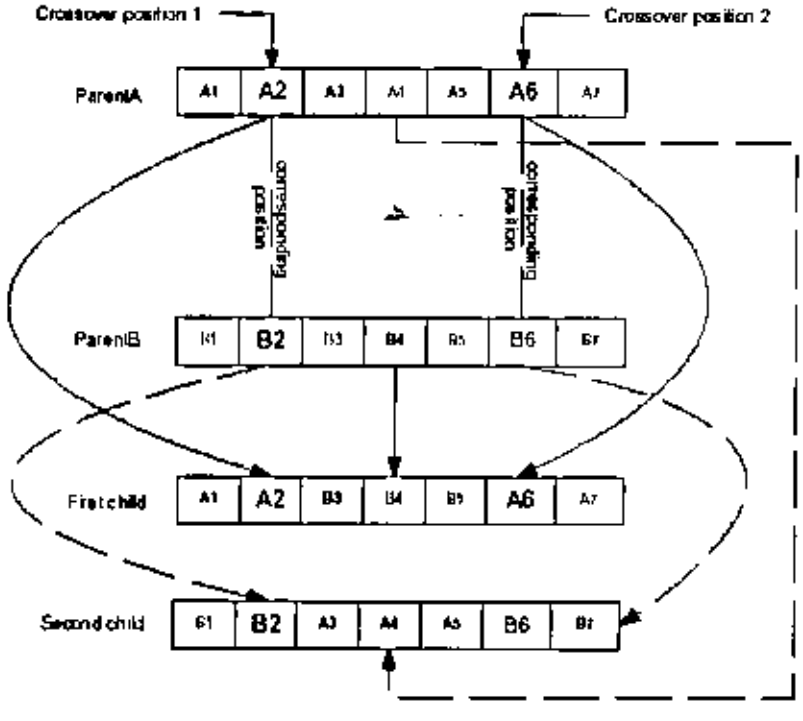


Figure (2.2): two Point Crossover[29]

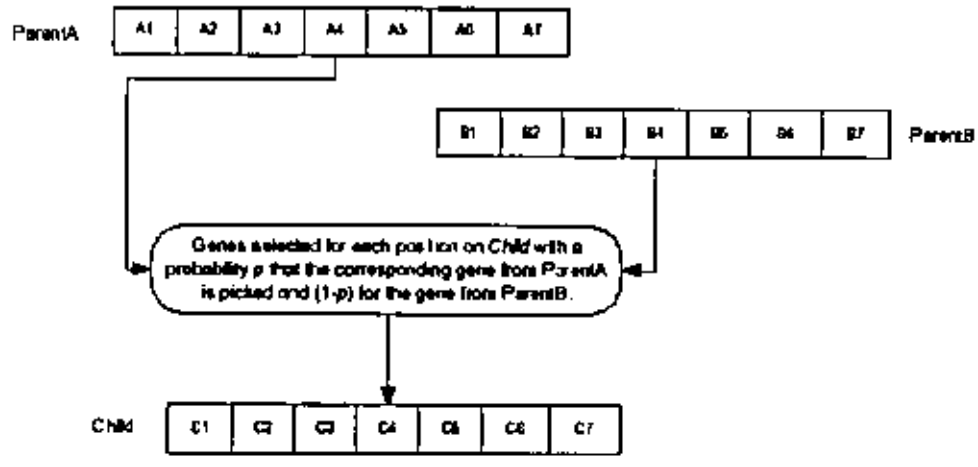


Figure (2.3): Parameterized Uniform crossover[29]

5. Mutation

After crossover, each bit of the string has the potential to mutate, based on a mutation probability (Pm). In binary encoding mutation involves the flipping of a bit from 0 to 1 or vice versa.

6. Population Size

The population size is the number of candidate solutions in any one generation. In natural evolution the total population size is governed by what is sustainable by the environment and similarly in GAs the larger the population size the more computationally intensive (in terms of memory requirement) is the search. In nature, the bigger the gene pool the more

2.6. Repair Strategies

Some representations of chromosomes can cause offspring outside the search space. In this case, a particular repair function is needed to fix that chromosome to make it more suitable for the search space.

2.7 Difference between Genetic Algorithms (GAs) and Traditional Methods

The main differences between them are [17]:

- Traditional methods work on point-by-point basis.
- They start with an initial guess and a new solution is found iteratively.
- GAs work with coding of the parameter set, not the parameters themselves. Advantage of working with a coding of variable space in GAs is that the coding discreteness the search spaces even though the function may be continuous.
- GAs search from a population of points, not single point so it is very likely that the expected GAs solution maybe a global solution
- GAs use objective function values and not derivatives.

2.8 Genetic Algorithm in Timetabling

- A simple genetic algorithm was applied to high school timetabling. The representation used was a 5-tuple of time-intervals (hours), resources (teachers), lessons, a matrix R representing a timetable and a fitness-function. The matrix R had rows representing teachers – 'row constraints' and columns representing hours

column-constraints'. Their representation was such that row constraints were always satisfied, but were unable to produce optimal timetables [1].

- A simple genetic algorithm applied to the school timetabling problem using a representation very much like an actual timetable. Entries in a matrix corresponded to time slots and tupelos of rooms, teachers and classes were allocated to the slots [6].
- The use of genetic algorithms on the problem of scheduling exams timetable at the university [13].
- Investigating the use of genetic algorithms (GAs) for solving arrange of timetabling and scheduling problems .
- A genetic algorithm used to solve a small lecture timetabling problem. An 'assignment' was defined as a 4-tuple of event, time, places and 'agents' a term used to describe the people required for an event to take place.
- Were able to achieve better performance than traditional genetic algorithm in some problems through using direct mutation as a main operator[13] .
- An example usage of Genetic Algorithms (GAs) for finding optimal solutions to the problem of Lecture Timetabling at a large university was explained [19].
- A possible way to generate timetable using genetic algorithms (GA), for implementation uses special mutation: repair" mutation not only serves to leave local optimum, but it moves GA to more promising areas in search space.
- A genetic algorithm was used to produce timetables for school . A problem specific chromosome representation and the use of a repair algorithm after the genetic operators avoid searching through illegal timetables. We also tested the use

of different fitness functions and present results obtained with our prototype timetabling system implemented in C [5].

- The use of heuristic method of Genetic Algorithm (GA), especially designing the GA for timetabling type of problems [2].
- A genetic algorithms used in a university timetabling. Results show directed mutation offers significant gains in speed and reliability over the fixed mutation operator of the canonical algorithm. Directed mutation has been found to reduce the time taken to find an optimal solution by more than 50% (for a 65-event problem) and 35% for a 130-event problem. The cellular PGA was found to significantly outperform the SGA, confirming results from (Payne 98) [29].
- An adaptive algorithm based on evolutionary computation techniques, designed, developed and applied to the timetabling problem in order to create feasible and efficient timetable for high schools in Greece. Simulation results showed that the algorithm is able to construct a feasible and very efficient timetable more quickly and easily compared to other techniques, thus preventing disagreements and arguments among teachers and assisting each school to operate with its full resources it can be used each time satisfying different specific constraints, in order to lead to timetables, thus meeting the different needs that each school may have. [11].

2.9 Comparative study of GAs application in earlier study

Earlier studies used GAs in solving school timetabling problems in different methods based on the genetic algorithm operators *(reproduction, crossover and mutation)* in order to reach a near optimal solution. The major difference between these studies is the

chromosome encoding which is one of the most significant factors in order to coverage satisfactory and fast to a near optimal solution. The term encoding is used to describe a way of representing a weekly timetable. Comparing previous studies with the educational system in *Libyan secondary school* , we have chosen the SGA for designing the timetable problem , it is convenient to design a school timetable where students groups, classes , time slots are fixed, and scheduling a set of subjects *(teachers)* over a set of time periods, while satisfying a wide range of constraints , the hard and soft constraints . In the suggested design more than one teacher can teach the same subject to different groups in different time slots avoiding clash problems *(subject clash problem or class clash problem)*. The GA operators will be applied for producing a new chromosome *(timetable)* in order to reach a near optimum solution. This will be discussed in the design and implementation chapter.

CHAPTER 3: ANALYSIS SURVEY AND METHODOLOGY

3.1 Introduction

This chapter illustrates the timetabling problem in Libyan secondary school's it is divided into three sections: Nature of the problem, Current system, Requirement of a good timetable

3.1.1 Nature of the problem

To clarify the school time table problem the survey asked the following question:

How many periods per week assigned for each subject?

What is the maximum number of periods a week?

How long is each period?

Are there different break times for the different year in the school?

What is the maximum capacity of a classroom?

How many students are in the school?

How many teachers are in the school?

How many sections are in the school?

Table (3.1) below illustrates the result of survey in details , where

N- the number of schools

Min – the minimum value given from the schools for each question

Max- the maximum value given from the schools for each question

Mean – the mean value (the sum of answers for each question divided by the number of schools)

questions	N	Min	Max	mean
What is the maximum number of period a week?	9	27	35	30.4
How long is each period?	9	40	50	44.664
Are there different break times for the different year in the school?	9	1	1	1
What is the maximum capacity of a class, room?	9	30	37	33.55
How many student are in there school?	9	120	350	225.33
How many teachers are in there school?	9	12	42	18.88

Table (3.1) the result of survey in details

For the construction of the timetable to the started all the necessary input data must be available to provides the necessary information about the relations between teachers, classes and subjects. In example of an input data *(necessary information)* is presented as follows:

The total number of class`s in the specific example there are 18, the total number of teacher's 36, the total number of days per week that courses take place (in the specific example this number equals 5) which is a typical value for Libyan secondary schools, and

the total number of period per day that courses take place *(in the specific example this number equals 6)* which is a typical value for Libyan secondary schools.

3. 1.2 Current system

During the survey a list of questions prepared about the methods used by school to produce school timetable functionality, usability and an associate difficulties to produce school timetabling system

- Is a computer used at any phase in the timetabling development?
- Is a novel timetable generated each year?
- What is/are the most important cause(s) of change?
- Is manual manipulation of the timetable allowed?
- How would you rate the timetables formed by your system?
- What is the maximum amount of time available between receiving the data and having to produce the timetable?
- How long does it typically take to construct the timetable from receiving the data to printing the final report?
- What are the major reasons for this change?

But the answers of first question shows that all schools don't use computer at any stage in the timetabling process, so some of these questions were useless. As any manual work there are many problems in producing the tables. So the survey will try to solve these manual problems.

3. 1.3 Requirement of a good timetable

A good school timetable means a timetable which fulfill the hard and soft constraints In order to keep the timetable valid .The results of survey show the importance of hard and soft constraints in school timetabling for example table (3.2) shows (percentage of constraint) the important of the first constraint to the school timetable, it illustrates constraint (1) in details , first column criteria of constraints , second column represents the frequency criteria (the number of schools apply the specified constraint) and the third column represents the percentage (frequency / number of schools *100) of the current criteria

constraint	Frequency	percent
Valid required	6	66.7
optional	1	11.11
not required	1	11.11
Total	8	88.89
Missing System	1	11.11
Total	9	100.00

Table (3.2) the percentage of constraint

Table (3.2), shows that 6 school of 9 answered questions 1 as required, 1 as optional, 1 as not required and 1 no answer. This means that this question has higher priority for school timetable.

3.2 Methodology

In this study the algorithm works on a population of individuals, random selection scheme, applying crossover *(one-point or two-point)* and a simple mutation operator to them.

3. 2.1 Problem definition

The school timetable problem can be defined as a class teacher timetabling problem. The weekly timetable in *Libyan secondary school* is divided into 6 , 45 minutes periods (time slots) which results in a total of 30 periods. From 0 -34, as can be seen in Table (3.3). Each lesson must be assigned to a time period in such a way that a number of constraints (requirements) are met.

Time	San	Mon	Tues	Wed	Thurs
08:00 - 08:45	0	7	14	21	28
08:45 - 09:30	1	8	15	22	29
09:30 - 10:15	2	9	16	23	30
10:15 - 11:00	3	10	17	24	31
11:00 - 11:45	4	11	18	25	32
11:45 - 12:30	5	12	19	26	33
12:30 - 01:15	6	13	20	27	34

Table (3.3) results total periods per a week

The school timetable is affected by many parameters and must satisfy a large number of requirements. This requirement is known as hard and soft constraints.

1. Hard constraints

- No subject can be assigned to two different classes at the same time.
- All lessons must be scheduled.
- In addition to the above constraints good timetables satisfy as many of the following Soft constraints.

2. Soft constraints

- Each subject should not be consecutive periods in each day.
- Some subjects are to be held early in the morning.
- The number of periods per day for each subject should not exceed two periods.
- Lesson continuity (students and teachers don't like timetables with gaps).
- A breakfast break must be scheduled (30 minutes) between (10:00 - 11:00)

3. Constraint data

It includes the hard constraints data used to test each hard constraint *(information about teachers and their subjects and classes).*

4. Clash problem

Class clash problem will happen when a teacher is assigned to more than one class at the same time and teacher clash problem will happen when more than one teacher is scheduled to teach the same class at the same time slot.

In our system the teacher and the subject are related with each other, so a class clash can be occurred when a subject is assigned to more than one class in the same time slot. Tables (3.4) and (3.5) show an example case where the class1 , class2 and class5 are clashed because these classes have the same subject at the same time ,this means that teacher clash will happen too , because the teacher and the subject are related with each other .

subject no	teacher name	subject name	periods per week
1	Salem,shda	AsL	3
2	Hammed, salmen	Eng	3
3	Mhfoth , same	Arab	3
4	Aldrsee , mohna	Math	4
5	Azeet , adeel	Chm	3
6	Mhdee , soad	Phy	4
7	Alsnose , kaled	Comp	3
8	Salah , hnan	Drw	3
9	Moftah	Pol	2
10	Ali	Mil	2

Table (3.4) relationship between subject and teacher

Class	class1	class2	Class3	Class4	Class5
Class1					
Class2					
Class3					
Class4					
Class5					

Table (3.5) Class clash problem

5. Parameters in representations

In this study the *Libyan secondary school* time tabling problem is considered school timetabling problem is a problem of assigning teachers, students, classes, time slot and rooms. In our problem we don't consider teachers, students and rooms because they are fixed by the school management. Our problem now is minimized to assign subjects to certain classes and time slots.

3.2.2 The algorithm

Recently GA is used to solve complex real-world problem. There are variants of techniques can be used to meet fast implementations. The algorithm used in our system is the simple GA will be discussed in this chapter.

1. Genetic algorithm steps:

Below, the steps of GA used in the evaluation:

Step1: Initialize the first population randomly; either randomly creates the first population using Random-Function, or read the first population from (*secondary storage)*

Step2: Use the Repair-Function to make all infeasible chromosomes *(timetables)* feasible according to the *(hard constraints)*.

Step3: Use the Fitness-Function to determine the fitness of each chromosome in the population.

Step4: Use the random selection schema to select the parents

Step5: Choose one of the crossover schemes *(one-point, multi point)* to combine the chosen parents to produce new individuals *(offspring)* to the new population.

Step6: Repeat the crossover of parents until whole the population has been formed.

Step7 use the random mutation to alter the genes in the new individuals. Repeat the mutating of the individual until whole the population has been mutated *(new generation)*.

Step8: Replace the current generation with the new one.

Step9: Return to '2' and repeat until the end of generation number to obtain the optimal or near optimal solution *(high best fitness)*.

2. Mathematical model

In our problem teacher is assigned to teach one subject to different groups for each school phase (level) , so the necessary parameters and data sets needed for the problem's model definition are the following:

- S is the set of subjects.
- G is the set of groups.

- **D** is the set of days in the timetable.
- **H** is the set of teaching hours.
- L is the set of lessons.
- subjects _ group _Hour_Day[s][g][h][d] is defining if subject s is assigned to time slots group g at hour h in day d.
- group _Lesson_Hour_Day[g][l][h][d] is defining if at group g lesson l is taught at hour h in day d. If group _Lesson_ Hour_Day[g][l][h][d] equals 1 this means that group g is taught lesson l at hour h in day d, otherwise if group _ Lesson_Hour_Day[g][l][h][d] equals 0 this means that group g is taught no lesson (empty period) at hour h in day d.
- subjects _Total_Hours[s] is the total hours assigned to subjects $s \in S$
- group _Total_Hours[g] is the total hours assigned to group $g \in G$.

Except for that, the following costs are defined:

- cost1: The empty periods of group (they must be as few as possible).
- cost2: The periods of each subject not uniformly distributed (they should be uniformly distributed).

So, the mathematical model of the problem can be expressed as follows:

$\min(cost1 + cost2)\ \forall s \in S, g \in G, l \in L, h \in H, d \in D$

under the following constraints:

1. ∀ s ∈ S , gi, gj ∈ G, hi, hj ∈ H, di, dj ∈ D ,∉ (subject_ group _Hour_Day[s][gi][hi] [di],subject_ group _Hour_ Day[s][gj][hj][dj]) such that gi ≠ gj and hi = hj and di = dj.

2. ∀ g ∈ G, li, lj ∈ L, hi, hj ∈ H, di, dj ∈ D∉ (group _Lesson_Hour_Day[g][li][hi][di], group _Lesson_Hour_ Day[g][lj][hj][dj]) such that li ≠ lj and hi = hj and di = dj .

3. ∀ s ∈ S , subject _Total_Hours[s] = subject _Hours[s], where subject _Hours[s] is the total hours for each subject according to law.

4. ∀g ∈ G, group _Total_Hours[g] = group _Hours[g], where group _Hours[g] is the total hours for each group .

5. ∀g ∈ G, l ∈ L, h ∈ H, d ∈ D∉ group _Lesson_Hour_Day[g][l][h][d] = 0 such that h is not the last teaching hour of a day.

6. ∀ si, sj ∈ S, gi, gj ∈ G, hi, hj ∈ H, di, dj ∈ D, ∉ subject _ group _ Hour _Day [si][gi][hi] [di] . subject _ group _ Hour_Day[sj][gj][hj][dj] such that gi = gj and hi = hj and di = dj and subjects si and sj are not assigned at the same period to group gi = gj .

3. Chromosome encoding

Before using a GA to solve the timetable problem a chromosome representation must be given. School timetabling is a process of assigning teacher, student, classes time and rooms. Encoding the chromosome have a great impact on the efficiency performance of the GA.

In our problem *(Libyan Secondary School Timetabling) student* and classes are prefixed, so they are not included directly into the representation. The problem now is limited in a singing subject to certain classes and period (time slot). Our weekly timetable problem

(Libyan Secondary School Timetabling) is 5 days each day is divided into 6 time slots, each period is 45 minutes, resulting in a total of 30 periods a week numbered from 0-29.

Day	Day 1				.....	N Day		
Period / groups	P1	P2	.....	P7		P1	.....	Pm
Group 1.1	Subject							
.....								
.....								
Group n								

Table (3.6) chromosome representation

The school timetable shown in table (3.6) is represented as a matrix of i × j Where: i (rows) represent groups, j (columns) represent periods (time slots), n the number of days in a week and m: number of periods in one day. Table (3.7) shows real example of chromosome representing . Each chromosome represented as a matrix of G x P size. where G represents Groups (G1, G2 ….Gn) and P represents timeslots (P1, P2… P6). The intersection point of G , P represents subjects (lesson and whether this lesson lab or not).

This means that if the weekly timetable is divided into five days and each day can have at most six time slots per class (*Libyan Secondary School Timetabling*) the number of

cells of the array equals 30. In every cell a number is stored, representing a subject assigned to specific class at specific time period. This chromosome encoding is shown in Table (3.7). The position of each cell shows implicitly to which class and time period each subject is assigned.

Day	Sunday						Monday			...	Thursday					
period / Groups	P1	P2	P3	P4	P5	P6	P1	P2	....	....	P1	P2	P3	P4	P5	P6
Groups 1.1	6	9	3	3	4	2					7	7	9	10	4	1
Groups 1.2	3	7	6	6	5	10					5	8	10	4	7	3
...																
...																
Groups m																

Table (3.7) shows real example of chromosome representing

Figure 3.1 the real example of chromosome representing secondary school timetable. In this study chromosome, for example subject 7 is repeated 3 times (3 periods per a week for each group) can be taught by one or more than one teacher without clashing.

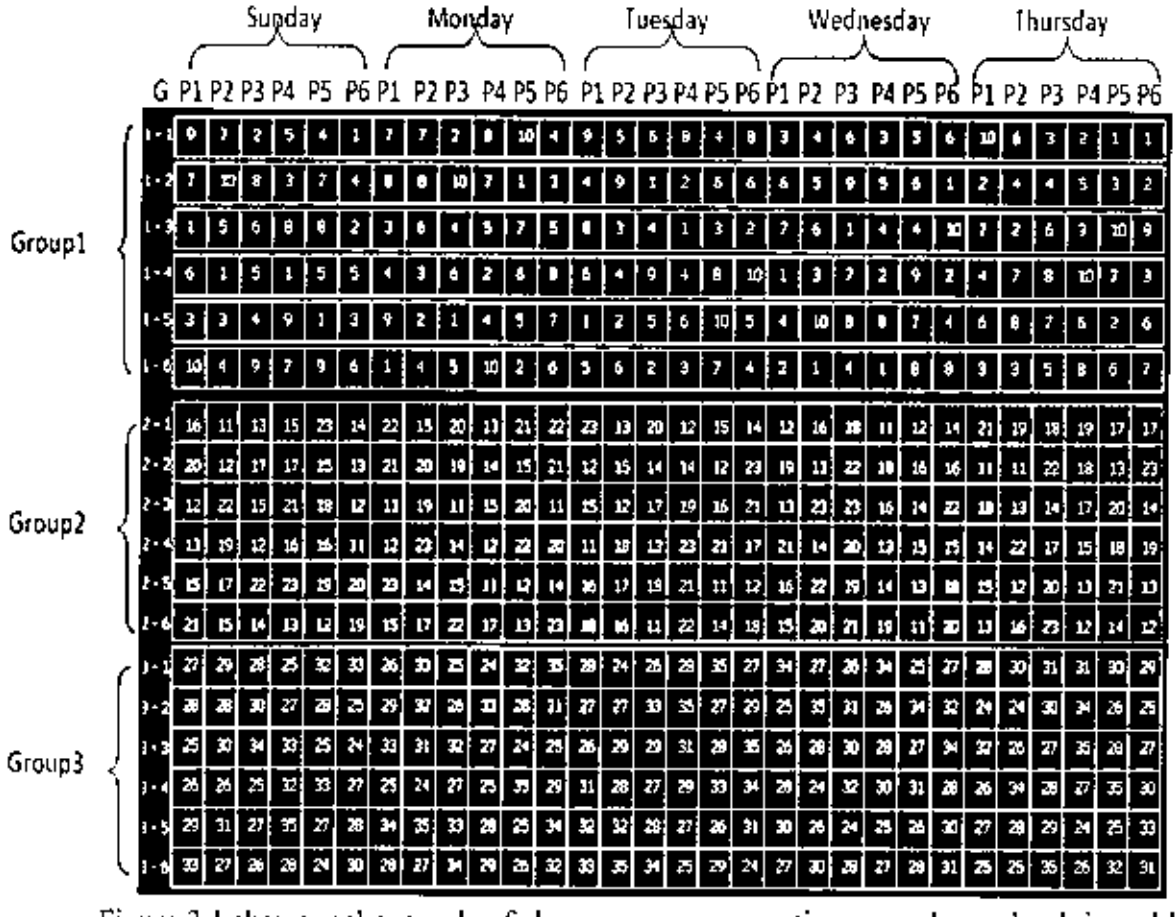


Figure 3.1 shows real example of chromosome representing secondary school timetable

4. Constraints

As mentioned before timetable must be scheduled in a way that suits constraints *(soft and hard constraints)*. These constraints are the fundamental base for the fitness-function, which is the GA most critical operator and it has to be defined before the GA can start evaluation. Therefore, the constraints need to be defined and represented.

4.1 Hard constraints

The hard constraints are constraints that cannot be violated in any timetable *(their violation renders the timetable infeasible).*

4.2 Soft constraints

The soft constraints are constraints which may be violated, but, any violation must be minimized *(the high violations turns the solution to be not-optimal but feasible).* A questionnaire has been made in order to come out with the expectation and preferences for both teachers and students in the secondary school. The expectations and preferences are provided in tables 3.8 and 3.9 the information in both figures is considered as the soft constraints. It reflects the opinions of the majority of the students and teachers who have answered the timetable questionnaire. Each question has a value (the value reflects the degree of preference *(- 5) strongly dislike , while (5) strongly like)*

4.3 Expectations and Preferences for the Students

The students prefer the follows:

- Some lessons are to be held early in the morning.
- Lesson continuity (students don't like timetables with gaps between lessons).
- A breakfast break must be scheduled (30 minutes)
- Number of subjects they prefer in one day is between 6 to 7 subjects
- To have more subjects on some days, in order to have a day without subjects
- The practical subjects (periods) after the break time

For some. wish to have all their subjects to be scheduled in consecutive periods

The next table provides the preferences of subject's time according to the majority of the student. The value reflects the degree of preference (the mean value of the students answers).

time / day	08:00 08:45	08:45 09:30	09:30 10:15	10:15 11.00	11:00 11:45	11:45 12:30	12:30 01:15
SUN	3	5	5	-2	5	4	3
MON	3	5	5	-2	5	4	4
TUE	3	5	5	-2	5	4	3
WED	3	5	5	-2	4	3	2
THU	3	4	5	-2	3	2	2

Table (3.8) Students Expectation and Preferences for the Timetable

4.4 Expectations and Preferences for teachers

The teachers prefer the follows:

- To have more lessons on some days, in order to have a day without lessons.
- Lesson continuity (teachers don't like timetables with gaps between lessons).
- Each teacher should have consecutive periods in each day.
- Some lessons are to be held early in the morning.
- The number of teaching hours per day for each teacher should not exceed four periods.
- A breakfast break must be scheduled (30 minutes) between (10:00 - 11:00)

The next table provides the preferences of subject time made by the majority of the teacher. The value reflects the degree of preference (the mean value of the teachers answers).

time / day	08:00 08:45	08:45 09:30	09:30 10:15	10:15 11.00	11:00 11:45	11:45 12:30	12:30 01:15
SUN	2	4	5	5	5	4	3
MON	2	4	5	5	5	4	3
TUE	2	4	5	5	5	4	2
WED	2	4	5	5	5	2	2
THU	2	4	5	5	5	-4	-4

Table (3.9) teachers Expectation and Preferences for the Timetable

4.5 Constraints satisfaction

Different instances of timetables are distinguished by the degree of constraints satisfaction, which typically make many (or even all) of the same possible timetables poor or unacceptable.

5 . The repair-function

For producing (resulting) chromosomes (timetables) to be valid they have to be repaired. The Repair-Function is meant to make the infeasible timetable feasible according to the imposed constraints. Repairing chromosomes consists of the changing of gene values to valid values closest to the original ones. This is done by first finding the free positions *(positions unoccupied by other subjects or classes)* common to the timetable of both the class and the teacher teaching uncertain subject. The free position closest to the original gene value is then chosen and removed from the list of free positions of the class and the teacher [5].

The Repair-Function used is meant to ensure that all the constraints (hard constraints) were satisfied.

6. Evaluation

After creating the random population, each chromosome evaluated individually based on the set of soft constraints. Each constraint must given a value a according to its property (influence) in the school timetable; the higher penalty value is assigned for the most unwanted violations. The evaluation process is done by evaluation function (fitness-function), which calculates the fitness value of the chromosome.

6.1 The fitness-function

The most important component of any GAs is the fitness function. It used for determining the value of each chromosome and allowing us to distinguish between good and bad ones, and in this way leading the GA to better solution.

Each chromosome must be evaluated using the imposed constraints, to determine which chromosome is better than others. The chromosome with the greater value represents the better a solution. In this study the used fitness functions to evaluated one chromosome is :

$$f = \sum_{i=1}^{n} R_c - P_c \qquad \text{----------(1)}$$

Where

F: fitness function to represent the value of each chromosome.

N: the number of constraints.

Rc: rewards for met soft constraint.

Pc: penalties for violations of soft constraints

1. How to calculate (R_C)

Each soft constrain has value where this value represent how important this constrain to be in the chromosome. This value can be positive for constrains that are required to be in the chromosome . To calculate (R_C) we use the following formula.

$$R_C = \sum_{i=1}^{N-1} (C_{Vi} * R_{Vi}) \qquad \text{---------- (2)}$$

C_{Vi} the value of constraint required and accomplished (rewards)

R_{Vi} number the repetition of the subject in the chromosome (timetable) and which realize the required constrain. Table (3.10) illustrates how to calculate (R_C) for different soft constraints .

i	R_{Vi}	C_{Vi}	$Rci=(C_{Vi} * R_{Vi})$
1	40	60	2400
2	19	50	950
3	25	40	1000
4	5	40	200
5	1	50	50
			Rc = 4600

Table (3.10) illustrates how to calculate (R_C)

$$\sum_{i=1}^{n-1} (R_{Ci}) = 4600$$

To calculate (P_C) we use the following formula.

$$P_C = (C_{vj} * R_{ij}) \qquad \text{----------} (3)$$

C_{vj} the value of constraint not required , but accomplished (penalties)

R_{vj} number the repetition of the subject in the chromosome (timetable) and which realize the not required constrain.

Table (3.11) illustrates how to calculate (P_C) for one soft constraint .

j	C_{vj}	R_{vj}	(C_{vj}* R_{vj})
1	- 40	4	-160

C_{ij} = - 40 (the negative sign means penalties for the not required constrain in the chromosome).

R_{ij} = 4 (the number of repetition for the not required constrain in the chromosome)

P_C = -160 (negative sign means penalty , so when calculate the fitness function (formula (1)) use the absolute value of Pc) .

In this case the fitness function (formula (1)) for the chromosome one

F = 4600 – 160 = 4440

Table (3.12) shows different fitness values for the chromosomes . where the greater value represents the better solution (school timetable) .

population size	Rc	Pc	F (fitness function)
1	4600	160	4440
2	3500	250	3250
3	5300	200	5100
⋮			
39	4000	300	3700
40	4800	140	4660

Table (3.12) shows different fitness values for the chromosomes

7. Selection

The essential operation to GA is selecting the parents from the population. In general there is no hard and fast rule regarding the choice of selection scheme. According to Darwin's evolution theory the best ones should survive and create new offspring [4]. In this study random selection is used in which a certain number of the fittest chromosomes are retained in the next generation while replacing the rest with the offspring. The random selection used in this system is generating randomly a number between 1 and total population size, this operation is preformed two times; for selecting parent1 and parent2 numbers.

8. Crossover

The primary genetic operator is considered to be the crossover. The crossover operator generally will combine genes from two parents to produce a child chromosome without any changes to the values of those genes (does not create new genes but, combines the existing genes to form new individuals). There are three significant crossover schemes commonly used (one point, two points and multi point crossover).

1. One-point crossover

One-point crossover in its standard form is the simplest form of crossover [29]. Crossover point is randomly chosen to occur somewhere in the chromosome. All the genetic material from before the crossover point is taken from one parent, and all the material after the crossover point is taken from the other. As shown in Figure (3.2)

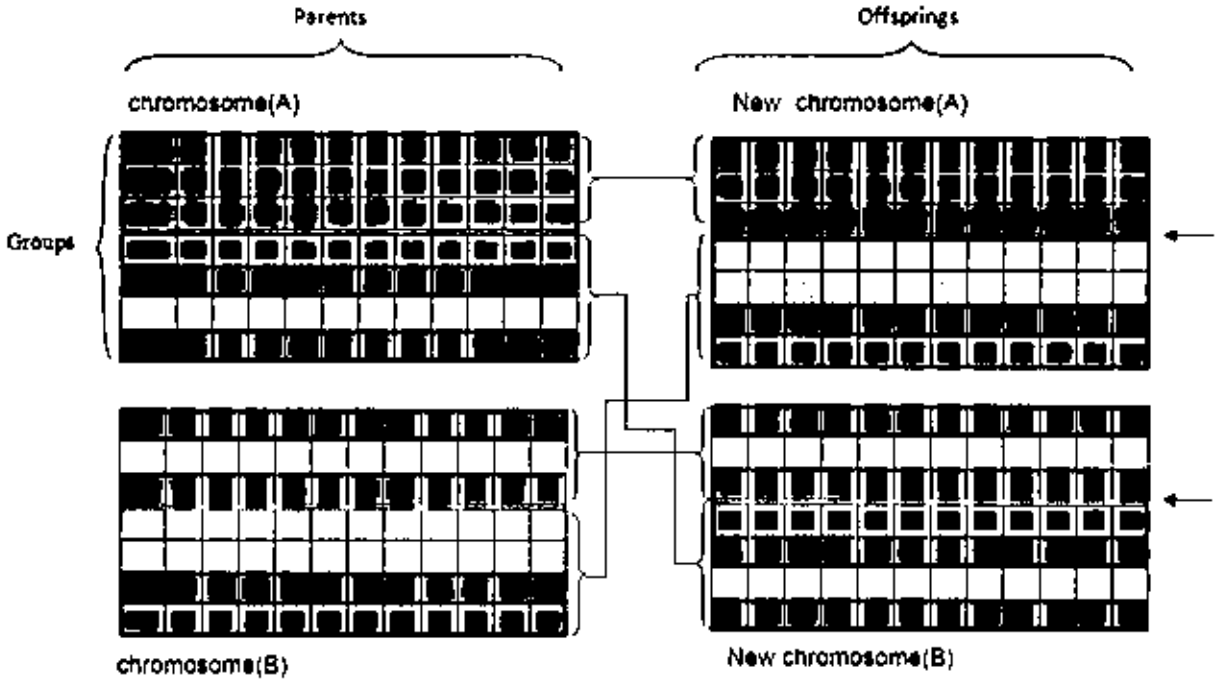


Figure (3.2) One point crossover process

2. Two –point crossover

Two – point crossover process can be performed with two points chosen on the parent chromosomes at random and the segments between them exchanged [29]. As shown in Figure (3.3).

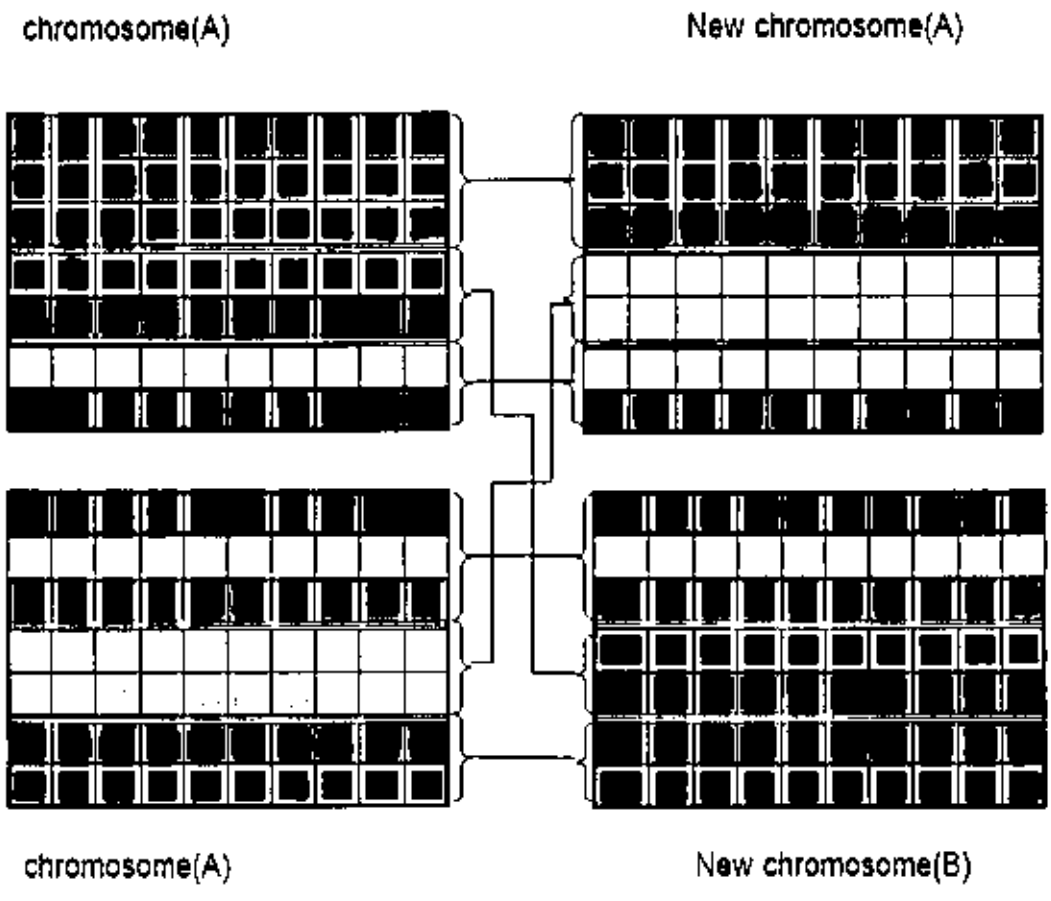


Figure (3.3) Two point crossover process

3. Multi point crossover

In this study a multi crossover points are been developed, in which each group is represented as a layer and groups in one year represent a chromosome . The multi point's crossover is achieved by choosing a single cross over point for each layer, where the head of layer1 from parent1 is jointed by the tail of layer1 from parent2 to produce offspring layer1 and this scheme is preformed for all layers. Each layer has its own cross point as shown in the Figure (3.4).

The inspiration of this scheme comes from the reality where each group weekly timetable is not depended on other groups considering subjects clashes.

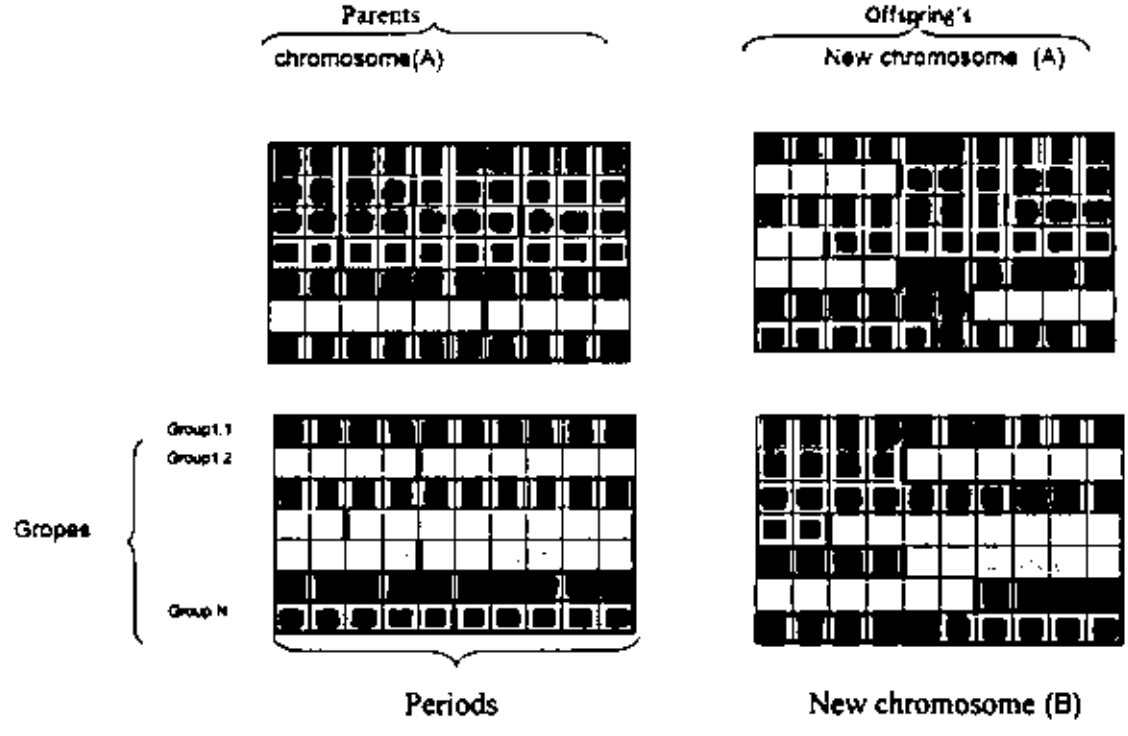
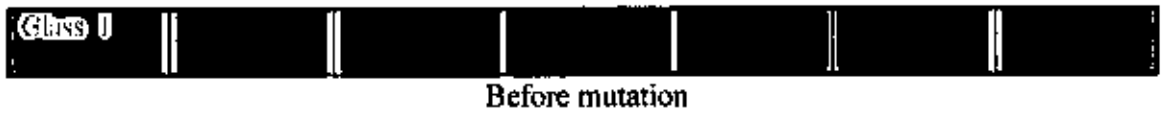


Figure (3.4) Multi point crossover process

9. Mutation

Mutation takes place after the crossover is performed. The mutation should be random, and yet it must not assign any time slot not allowed for that particular class. This is solved by first reading in all the allowable time slots for each class and mapping the random gene to the length of that array and then assigning the content to the selected random gene as shown in the Figure (3.5) .



Before mutation

After mutation

Figure (3.5) mutation (selected random gene)

Mutation, like crossover, must ensure that a timetable still satisfies hard constraints after its action. It cannot shift any subject to another period at random, since this may cause a conflict between the moved subjects.

10. Population size

The first step into initializing an entire population of chromosomes is the determination of the population size, which depends on the techniques used and the problem. In this study different population sizes have been evaluated and tested.

CHAPTER 4: SYSTEM DESIGN

4.1 Introduction

In this chapter we will demonstrate the design of *Libyan School Timetable* using *GA* system *(LSTTGA)*. *(UML)* is a Language for specifying , visualizing, constructing, and documenting the artifacts of software systems

4. 2 Design of LSTT_GA system

The system Libyan school timetable using GA system *(LSTT_GA)* consist of two parts The first part is initial population and evaluate fitness , the second part is reproduction , The reproduction include three modules (selection ,crossover , mutation) these modules can executed jointly for GA system. The user will have to provide information about teachers, subjects, lab room and year of study. And as mentioned in the first chapter, students and classrooms will be prefixed all the time.

In the development of our system we have adopted a graphical notation known as Unified Modeling Language the *(UML)* is a Language for specifying , visualizing, constructing, and documenting the artifacts of software systems [10, 15]. The *UML* divided into two general sets: structural and behavioral diagrams. Structure diagrams: show the static structure of the objects in a system. Behavior diagrams: depict the behavioral features of a system or business process. In our system is described below into three types of *UML* diagrams namely:

- Use Case Diagram.
- Class diagram.
- Sequence Diagram.

1. The Use Case diagram

The use case diagram is used to identify the primary elements and processes that form the system. The Use case diagram of our system (LSTT_GA SYSTEM) is described in figure (4.6).

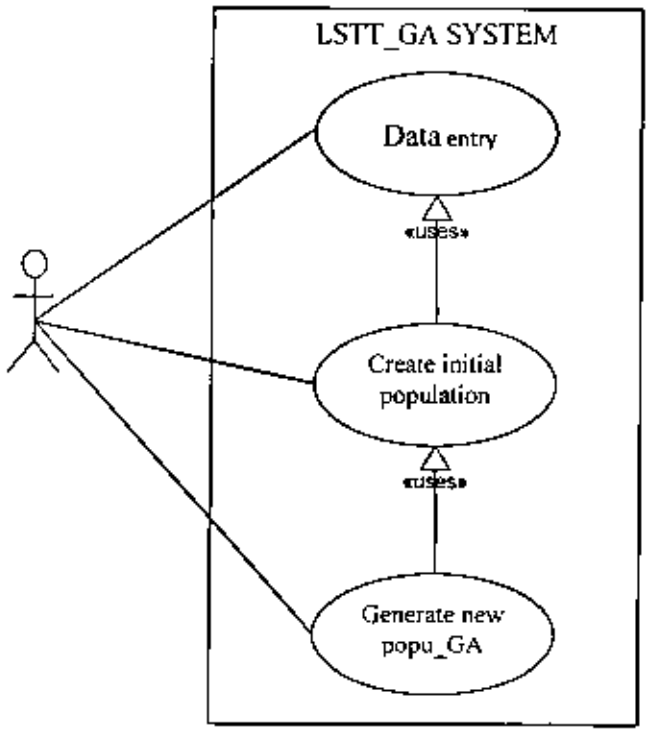


Figure (4.1) The Use case diagram

1.1 Data entry

The user has to supply the system with the required data which are the inputs *(teacher data, subject data and classes data)*. If these data where stored the user has to specify the name and the location of the database.

Outputs

The system generates the data and transfers it to a suitable form.

1.2 Create initial population

Inputs

The user supplies the system with the initial population according to specific conditions if these population where stored the user has to specify the name and the location of the database.

Outputs

The system has to give the first timetable according to the specified condition.

1.3 Generate New Population_GA

If the first timetable was not acceptable the user has to supply the system with the new population and if the population was stored the user has to specify the name and the location of the database.

Outputs

The outputs will be a (near) optimal timetable.

2. The Class diagram

A Class diagram gives an overview of a system by showing its classes and the relationships among them. Class diagrams are (static) they display what interacts but not what happens when they do interact. All the classes contain two common methods: i) the deconstructor method that call the **Finalize** or **dispose** class. It is used to dissolve an unneeded object. The unneeded object, in turn, takes up memory space and slows down the process; and ii) the constructor method which call the **New** or **InitializeComponent** class. It initializes the objects parameters when it is created. Figures 4.7 and 4.8 depict the class diagrams of our system containing the following classes:

Crossover Class: It has the necessary methods to apply the crossover. The methods of this class are as follows:

- **One_Point** : to call following methods
- **FS_One_Point, SS_One_Point and TS_One_Point:** these methods used for one point crossover on the first, second and third secondary year.
- **Mult_Point** : to call the following methods
- **FS_Mult_Point, SS_Mult_Point and TS_Mult_Point:** These methods used for multi point crossover on the first, second and third secondary year.
- **Randomize:** to generate randomly between one and the maximum number of periods per a day.

Generate_Initial_Pobulation Class: It has the necessary methods to generate the initial population. The methods of this class are as follows:

- **intial_pop** : this method is used to generate the initial population for the chromosome (first , second and third secondary year)
- **fill_subjects** : it is a method for getting the data subject

Soft_Constrains Class: this class deals with the data entry of the soft constraints. The methods of the Soft_Constrains class are as follows:

- **Early_Slots_Save, late_slots_save , Not_Same_Slots_Save, Same_Slots_Save , Not_Serial_Slots_Save , Serial_Slots_Save** : these methods used to store the data entry (early , late , not same , same , not serial and the serial) slots (or subject) for the soft constraints .
- **update_constrain** : it is a method used to update the soft constraint values
- **DEL_subj** : this method used to delete the soft constraint values

Evaluate Class: It has the necessary methods dealing with evaluation. The methods of this class are as follows:

- **eval_offspring:** it is a method for evaluating the offspring.
- **CTime , NoCTime , early_subjects , Late_subjects , Same_Slots, Serial_Slots , NOSame_Slots and NOSerial_Slots** :these methods used for evaluating the soft constraints (same time , no same time , early , late , same , serial , not same and no serial) slots or subject .

MainWindow Class: It has several methods to utilize the buttons that are used in the system. The methods of this class are as follows:

- **draw_genes:** it is a method to draw the timetable on the mainwindows form.

- **intiial_pob** : this method is used to generate the initial population for the chromosome (first , second and third secondary year)
- **Last_Table, Next_Table and Previous _Table:** this method is applied for showing the last, next or the pervious timetable on the mainwindows form.
- **Load_Timetable and Save_Timetable:** they are the main methods for reading and writing the timetable from / to the database file.
- **Select_school:** this method used for selecting a school.
- **Showtable:** this method used for showing the current table.

Selection Class: This class is used to select the parent from the population. It contains the following methods:

- **FirstSelection, SecandSelection and ThirdSelection:** these methods are applied for selecting the parent for the first, second and the third secondary year.

Schools Class: It has the necessary methods to deal with data entry of the school. The methods of this class are as follows:

- **Delete_Click, Edit_Click, Find_Click and Save_Click:** these methods for deleting, editing, finding and saving the school data.

fill_schools: this method for reading the stored school data

Subjects Class: It has the necessary methods to deal with data entry of subject. The methods of this class are as follows:

New_Click, Delete_Click, Edit_Click, Find_Click and Save_Click : these methods for adding ,deleting , editing , finding and saving the subject data .

- **fill_subjects** : this method for reading the stored subject data
- **load_schools** : this method for reading the school data

Optimum_value Class: It has the necessary methods to get the optimal value for each chromosome. The methods of this class are as follows:

- **fill_s_name:** this method used for getting the soft constraint .
- **find_subj_repeat:** this method is applied for getting the subject and its number of periods in a week.
- **Load_optimum_value:** this method used for loading the optimal value.

Module1 Class: It the main entry point for the application. Its one necessary method is specified below:

- **MAIN:** it is a method for initiating and creating the connection with the database file.

Decision Class: It has the necessary methods to decide wherever the offspring is accepted or rejected. It contains one main method.

- **decision_offspring:** this method is applied to return accept or reject value depending on the optimal value of the offspring's.

Teachers Class: It has the necessary methods to deal with data entry of the teacher. The methods of this class are as follows:

New_Click, Delete_Click, Edit_Click, Find_Click and Save_Click: these methods for adding, deleting, editing, finding and saving the teacher data.

- **fill_teachers** : this method for reading the stored teacher data
- **load_schools** : this method for reading the school data

RepairFunction Class: It has the necessary and important method which is :-

repair_offspring: it is a method for repairing the offspring or parent depending on the hard constraints

Mutation Class: It has the necessary methods to deal with gene of the chromosome. The method of this class is as follows

- **Gens_replacement:** it replaces the gene with another one in same the chromosome (timetable).

CachedCrystalReport1 Class: It has the necessary methods to create report .The method of this class is as follows:

- **CreateReport** : this method is used to create the current timetable report

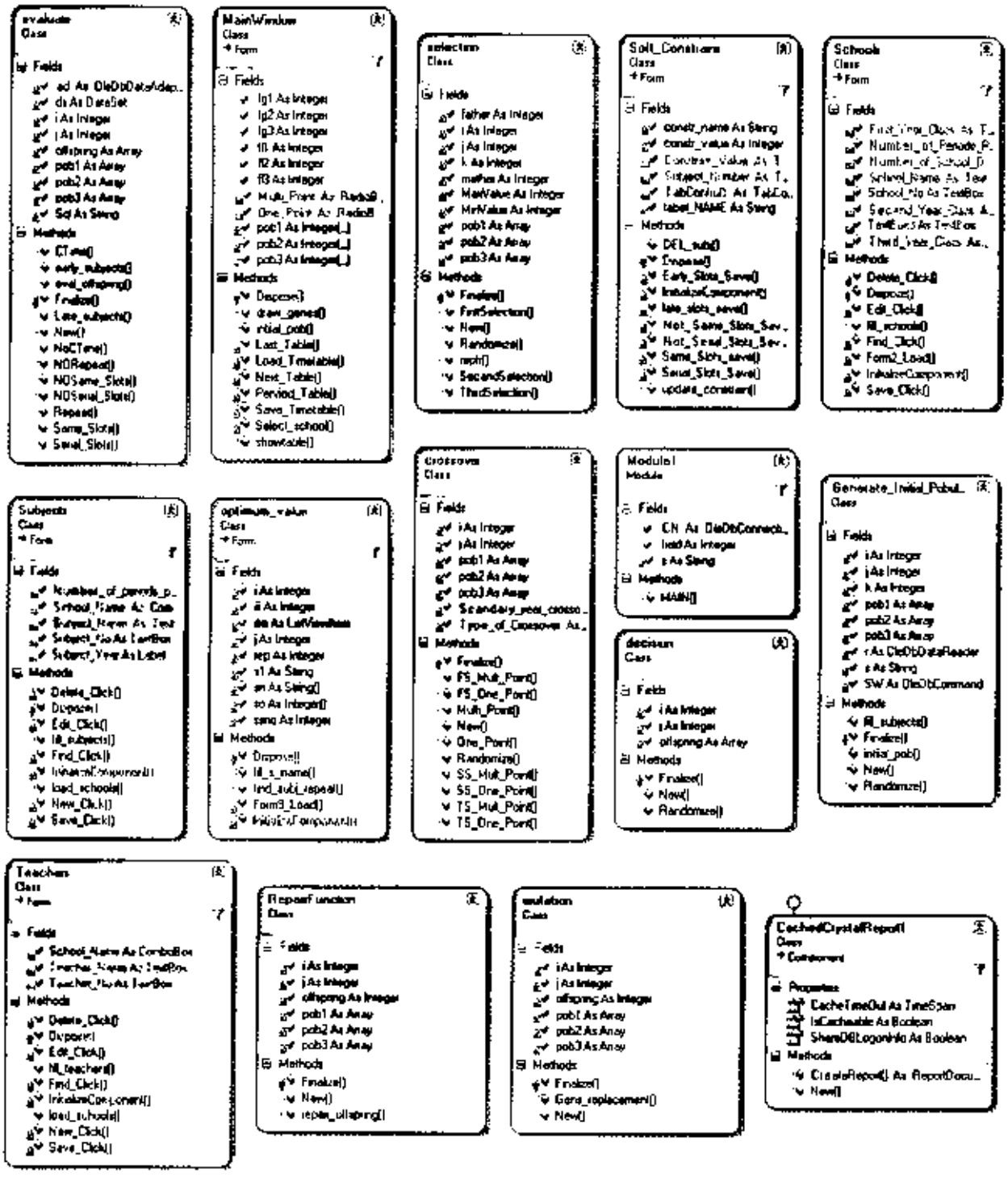


Figure (4.2) class diagrams for (LSTT_GA SYSTEM)

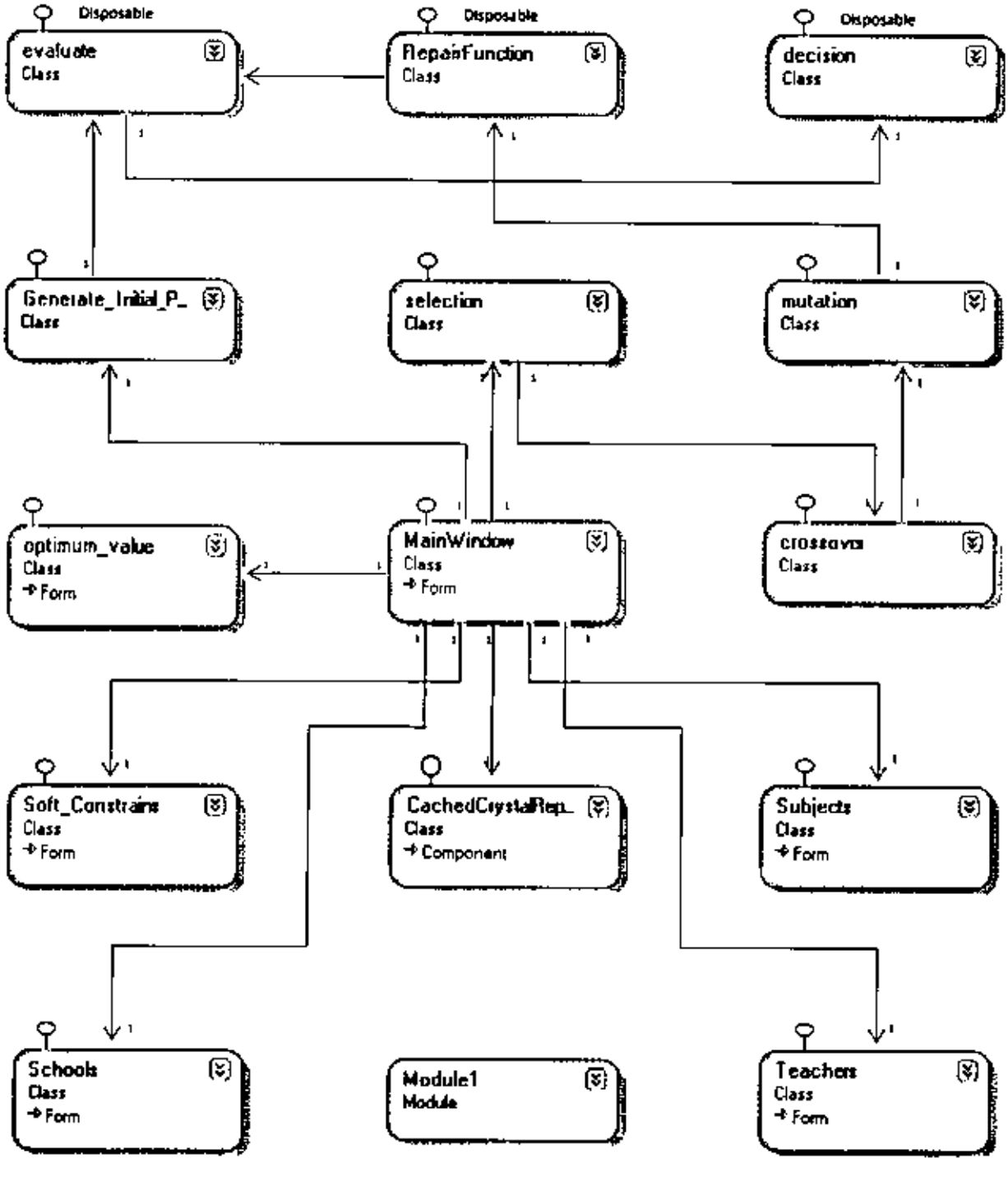


Figure (4. 3) the class diagram with relationships for (LSTT_GASYSTEM)

3. The Sequence diagrams

Sequence diagrams: are structured representations of behavior as a series of sequential steps over time. They are used to depict work flow, message passing and how elements in general cooperate over time to achieve a result

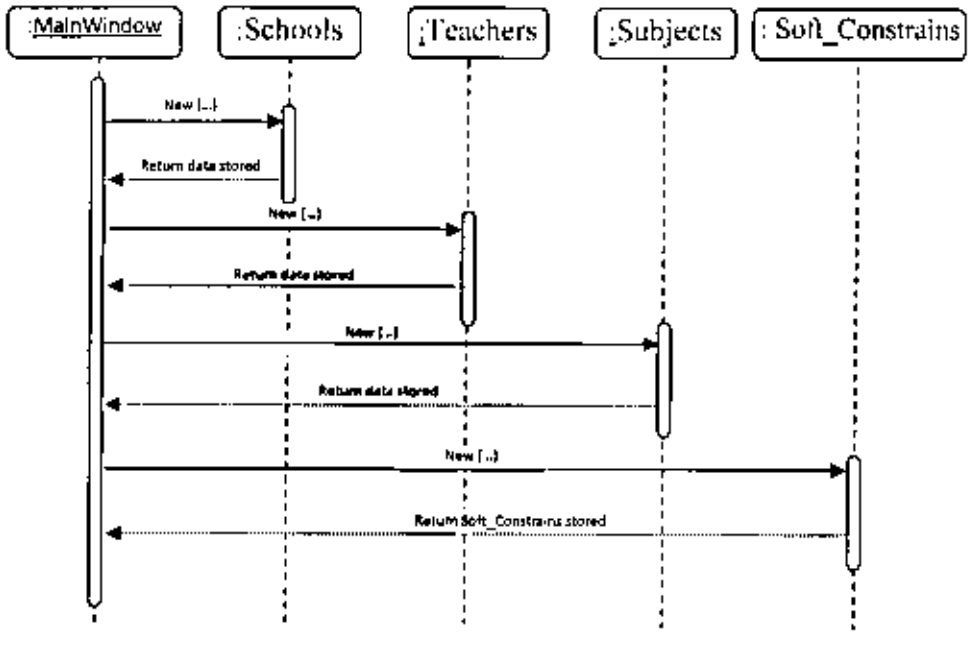


Figure (4. 4) The Sequence diagrams for data entry

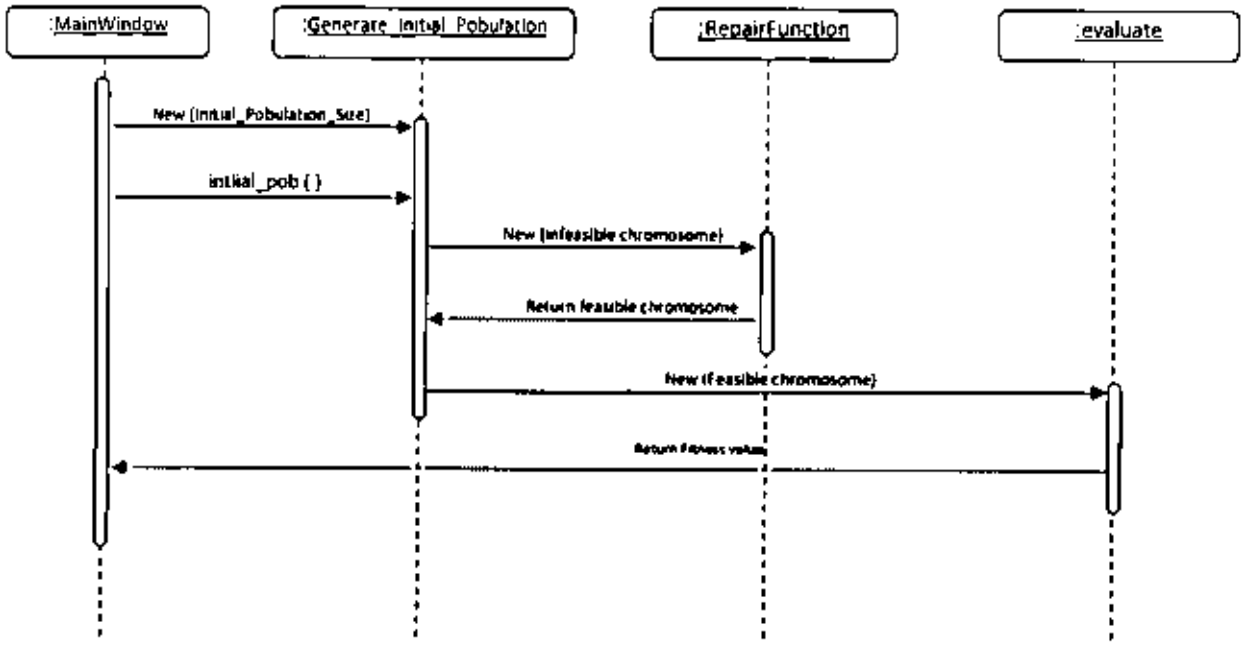


Figure (4.5) The Sequence diagrams for create initial population

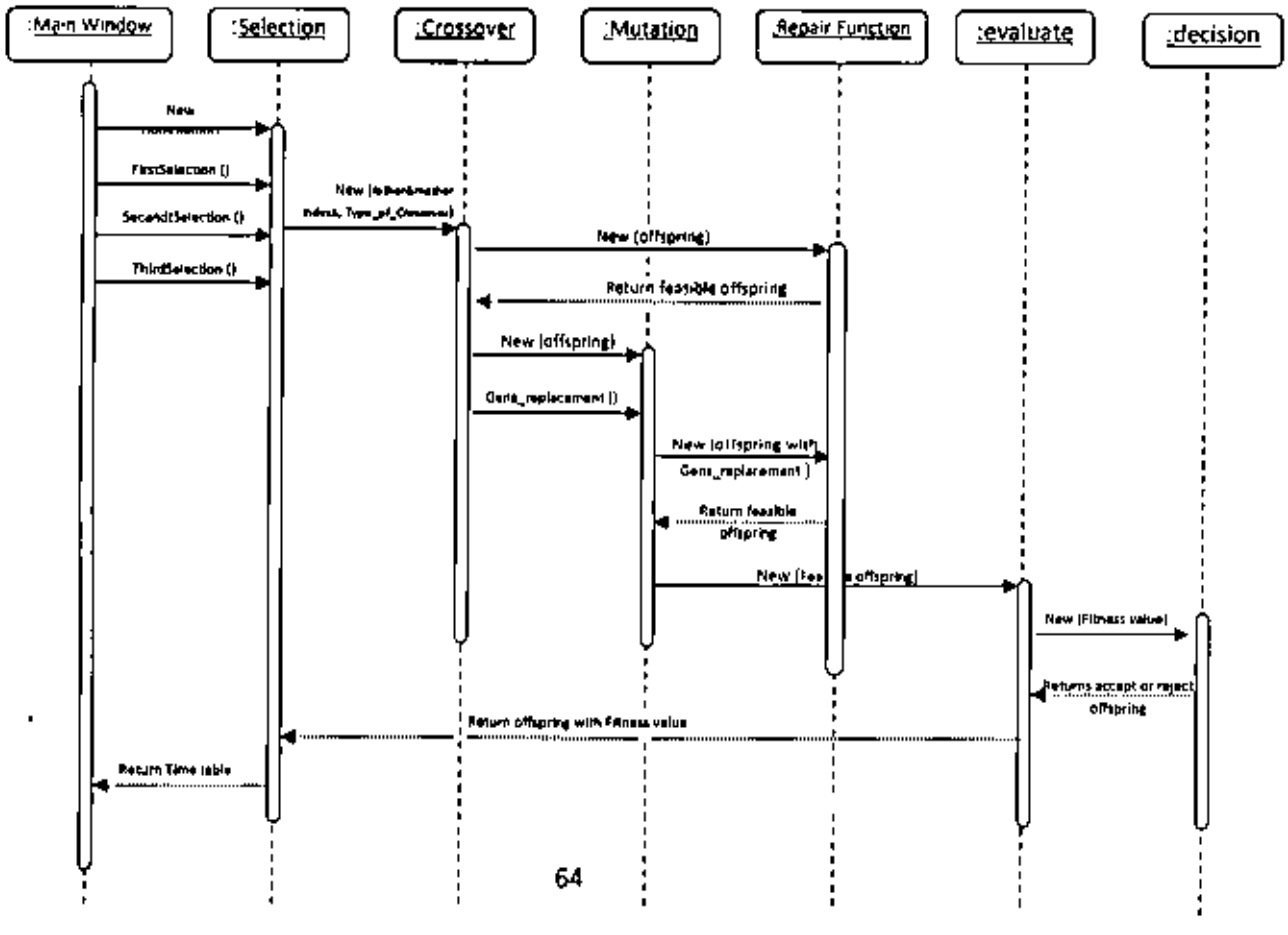


Figure (4.6) The Sequence diagrams for generate new population

4.3 Initialization

The input data of the system *(subject database, teacher database)* must be entered to make a timetable that satisfies the hard constraints and without clashes and then map it to formatted chromosome to specific period to specific class. The initialization algorithms are as follows:

For each chromosome in the population and for each class assign a subject which refers to a specified teacher without causing a violation to the hard constraint to the period figure 4.12 shows the result of initialization. It represent one day with 6 periods for 6 classes. The user decides the number of generations.

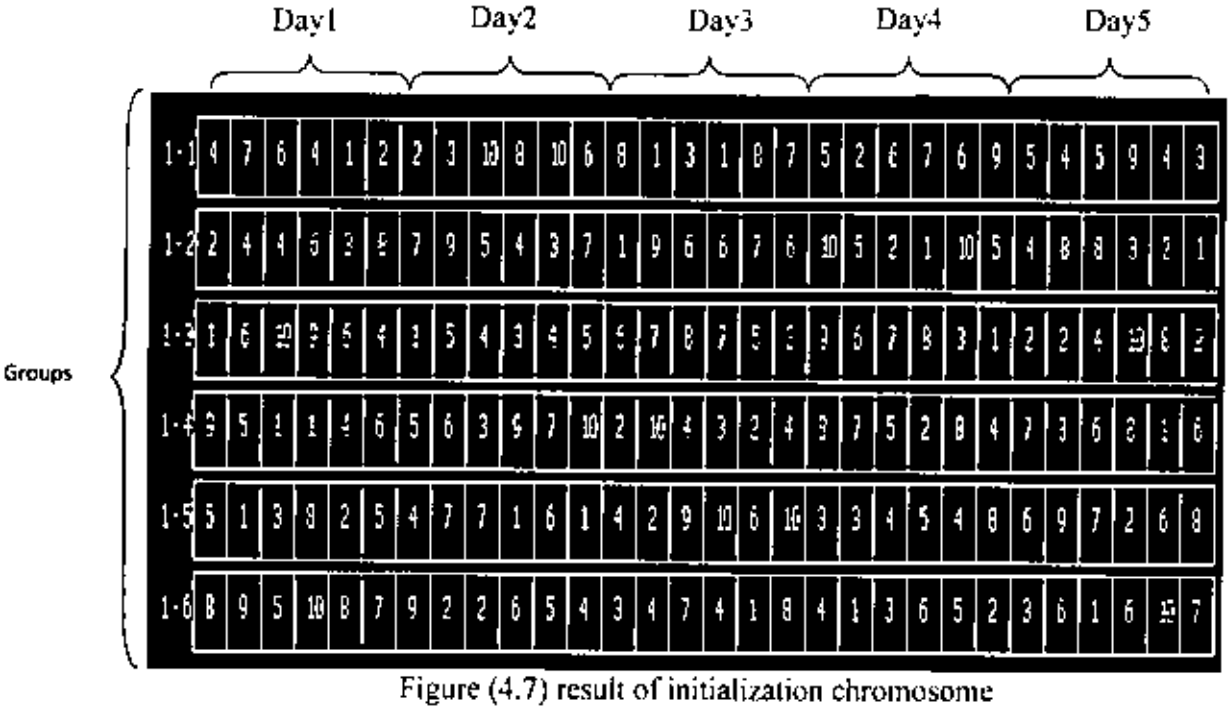


Figure (4.7) result of initialization chromosome

Table (4.5) shows the relationship between subjects and classes. Table (4.6) shows the relationship between teachers, subjects and classes.

class	subjects									total periods
1/1	Math	Eng	Arab	Asl	...	...	...	...	...	30
1/2	...	...	...	...	...	...	...	...	...	30
1/3	...	...	...	...	...	...	...	...	...	30
1/4	...	...	...	...	...	...	...	...	...	30
1/5	...	...	...	...	...	...	...	...	...	30
1/6	...	...	...	...	...	...	...	...	...	30
										180

Table (4.1) the relationship between subjects and classes

subject-no	teacher	subject	class
1	Salem, shta	Asl	1/1,1/2,1/3,1/4,1/5,1/6
2	Hammed, salme	Eng	...
3	moftah, same	Arab	...
...	...	...	...

Table (4.2 relationship between teachers, subjects and classes

4.4 Calculating the clashes

The main objective of this procedure it to calculate the clashes among the subjects as follows:

1. Read each gene (subject) on each raw on the timetable.
2. Compare read gene with others on the same column. Place that gene if there is no identical one.
3. If so, try with next gene.
4. Repeat it until the timetable is full and usable.

CHAPTER 5: IMPLEMENTATION

5.1 Genetic Algorithm Modules

The operators used to implement the simple GA in this project are (selection, crossover and mutation) have been described in details in previous chapters. In this study we chose the random selection, one point crossover, multi point crossover and random mutation.

1. Selection Module

1.1 Random selection

Random selection is used in which a certain number of the fittest chromosomes are retained in the next generation while replacing the rest with the offspring. In this system random selection is used for generating randomly a number between 1 and total population size, this operation is preformed two times; for selecting parent1 and parent2 numbers. The following pseudo code represents random selection algorithm. Show figure (5.1).

1- Set i=0 *(where i is number of random generate)*

2- Randomly generate a number X > 1 and X <= population size

3- Increase i

4- Sign x as parent (i) number

Figure (5.1): random selection algorithm

2. Crossover Module

2 .1 One Point Crossover

As mentioned earlier in chapter three, two chromosomes are chosen randomly. In one point crossover operator implementation, swap the first part of the first chromosome with the last part of the second chromosome and vice versa. The following pseudo code represents one point crossover algorithm sees figure (5.2).

```
{
    • Determine one random point to be chosen for cross over N < number of
      gens (chromosome length).
    • For the first point until N Choose genes from parent 1
    • For N to last gene choose genes from parent2.
    • Swap the gene between parents.
}
```

Figure (5.2): One Point Crossover Algorithm

2.2 Multi point crossover

Each layer has a single cross point . the same point is used to produce the offspring layer The following pseudo code represents multi point crossover algorithm see figure (5.3) .

1. Counter =0
2. Calculate number of layers L
3. Randomly generate a number N >0 and N < layer length (total number of period each week for one group)
4. For the first point until N Choose genes from parent 1
5. For N to last gene choose genes from parent2.
6. Swap the gene between parents.
7. Counter +=1
8. While not counter < L go to step 3

Figure (5.3) multi point crossover algorithm

3. Mutation Module

In the mutation operator, the gene representing subject is replaced randomly by another gene if the predefined probability rate is passed in the given chromosome. The following pseudo code is the mutation algorithm see figure (5.4).

```
Mutation
{
    • Choose two genes randomly.
    • Swap  gene values
    • Check layer for feasibility if not  go to step 1
}
```

Figure: 5.4 Random mutation algorithms

5.2 User interface

1. School information interface

Figure (5.5) interface is used to enter the School information.

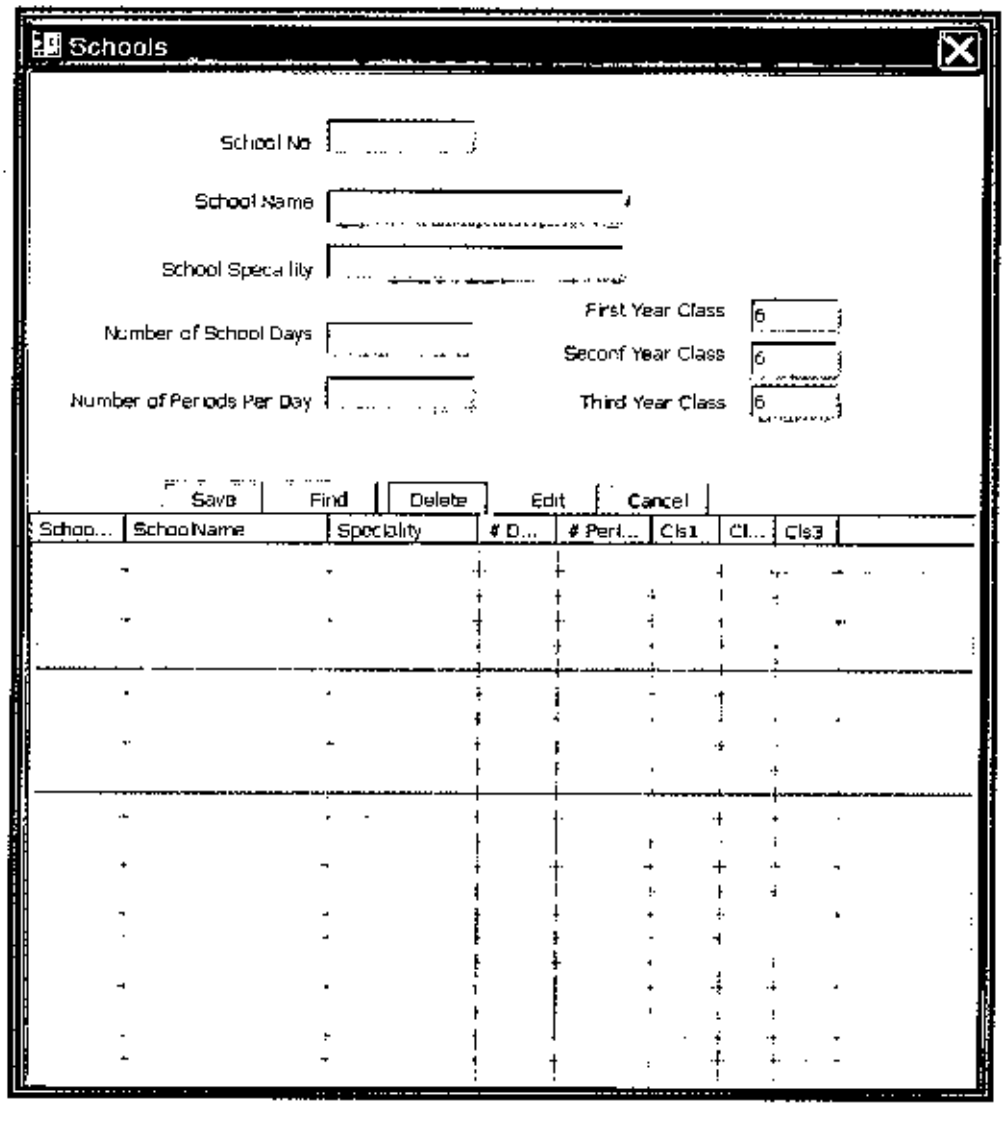


Figure (5.5) School information interface

2. constrains value interface

Figure (5.6) interface is used to enter the constrains value .

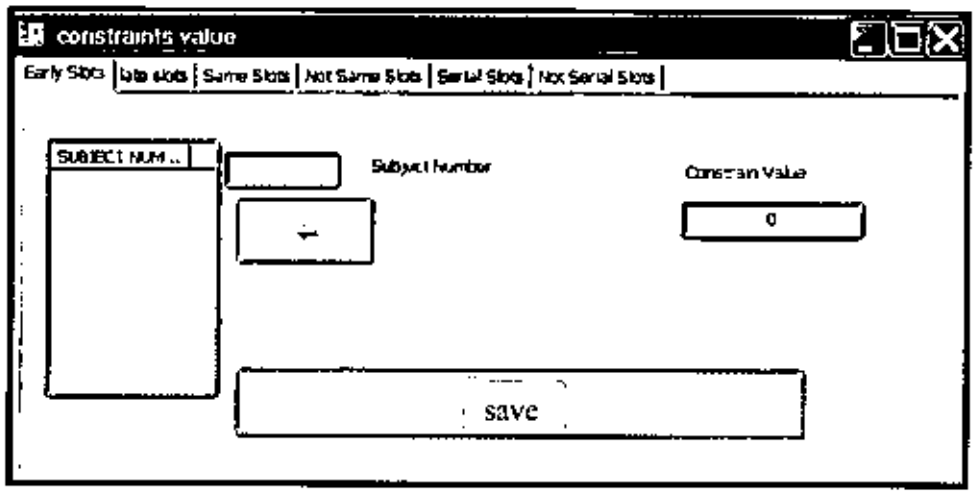


Figure (5.6) shows the form that the user can change the constrains value through.

3. Subjects information interface

Figure 5.7 interface is used to enter the Subjects information.

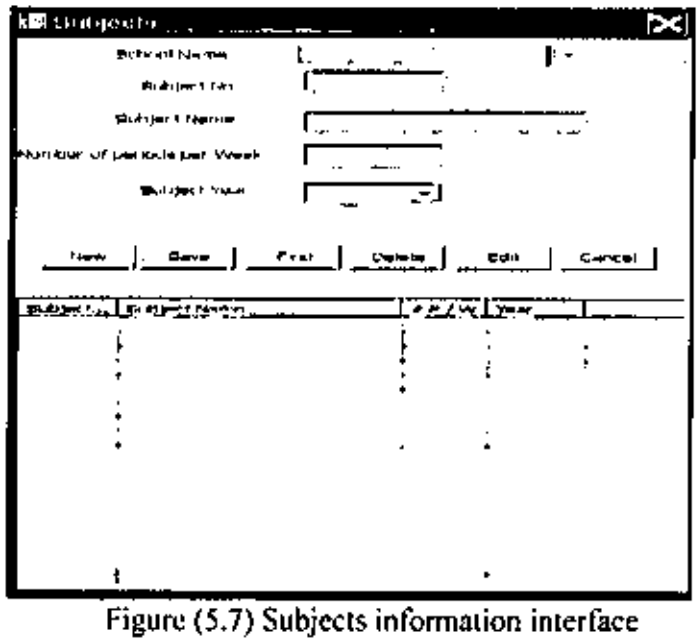


Figure (5.7) Subjects information interface

4. Main interface

Figure 5.8 is main interface which is used for running the system.

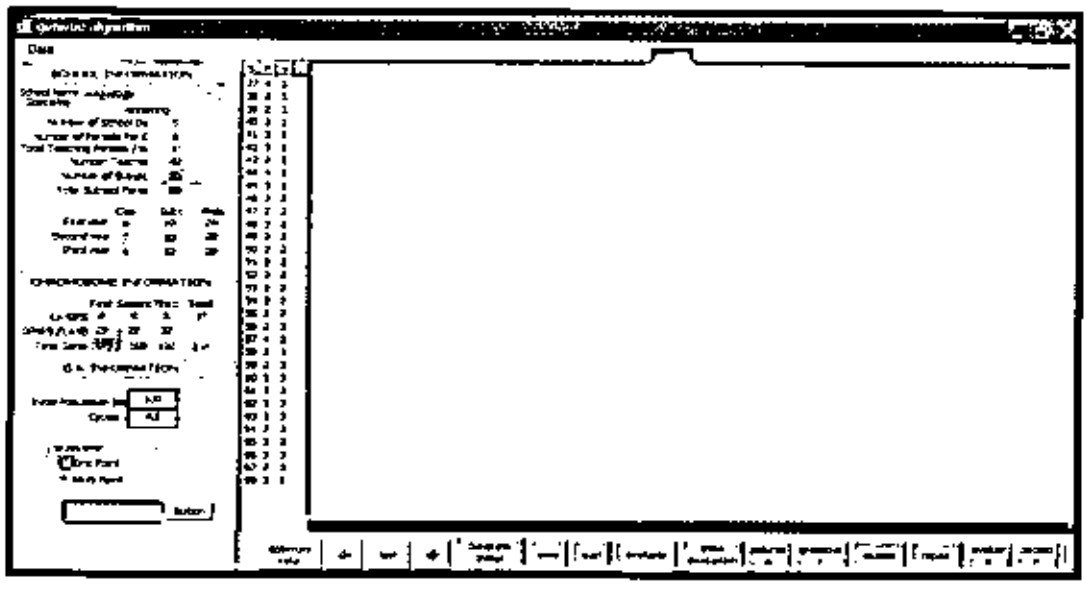


Figure (5.8) Main interface

5 . Selection interface

Figure (5.9) is selection interface shows how the selection occurs.

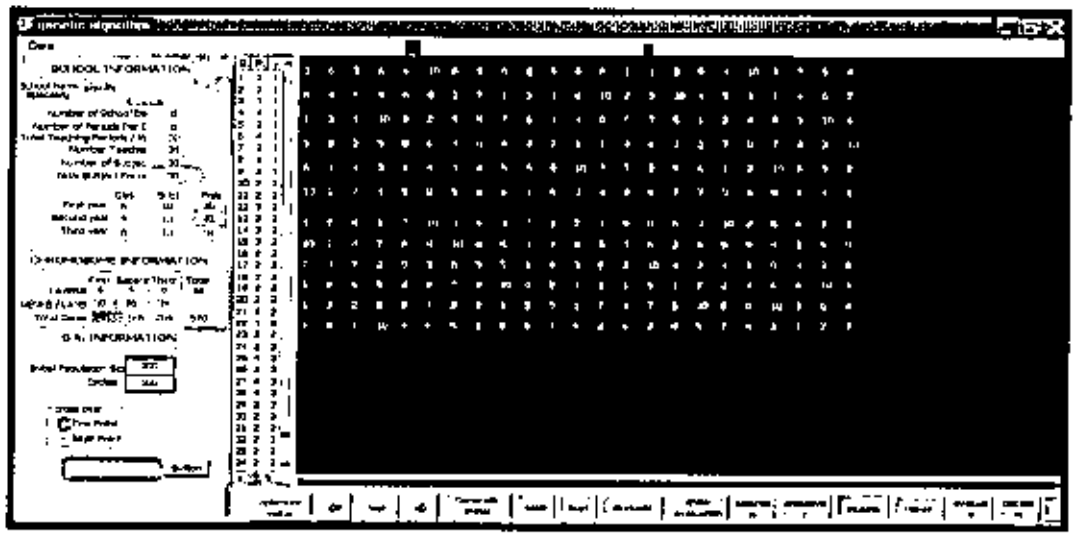


Figure (5.9) Selection interface

6. Evaluation values interface

Figure (5.10) shows an example valuation values interface.

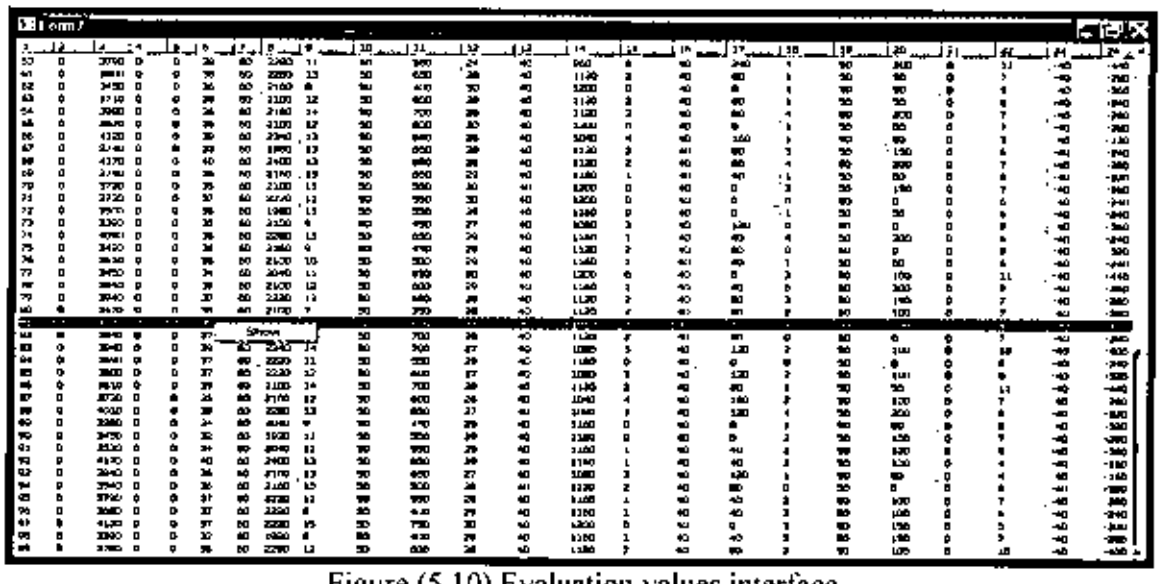


Figure (5.10) Evaluation values interface

5.3 Weekly timetable Report

Figure (5.11) shows an example of the weekly time table report for one group obtained from the system.

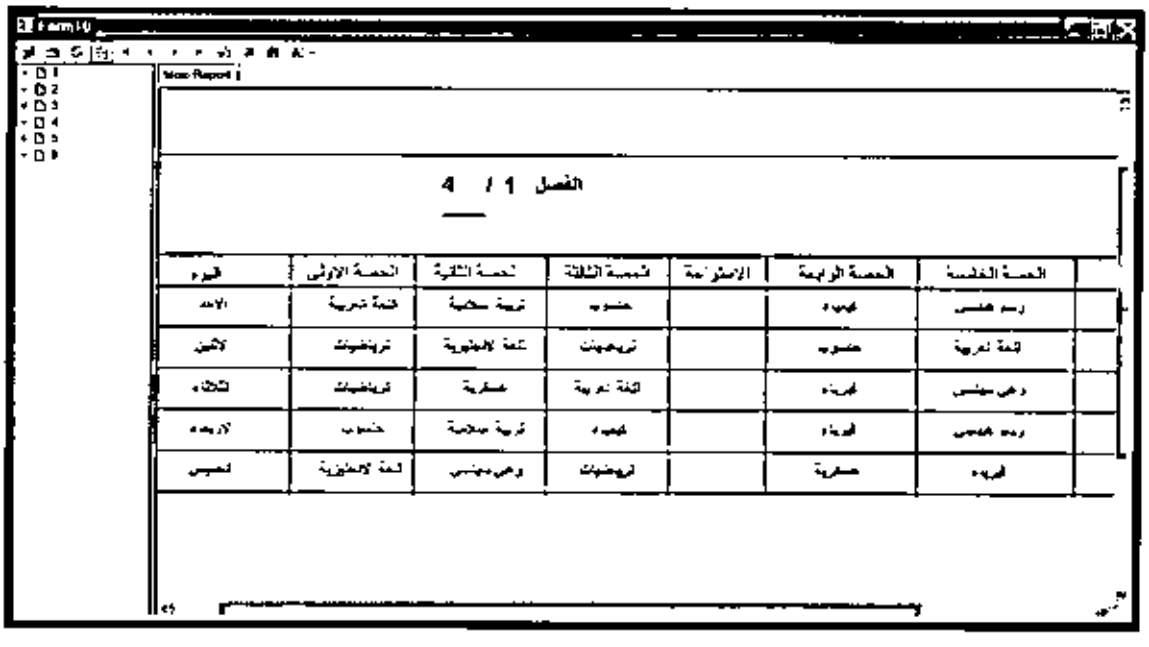


Figure (5.11) Weekly timetable Report for one group interface

CHAPTER 6: EXPERIMENTS AND RESULTS

6.1 Introduction

The chapter present the experimental results obtained from using GAs to solve a real life Libyan secondary school timetable problem. The operators and the parameters used in the experiments were one point, multi points crossover, mutation, random selection technique and population size with different generation number.

6.2 System Requirement and Program language

The experiment was tested on a computer with the following specifications:

OS	Microsoft Windows XP Professional
Version	5.1.2600 Service Pack 2.5.1.2600
Processor	1.8 GHz, Pentium 4
RAM	512.00 MB

The Visual Basic.Net is used to develop the system with Microsoft access [18, 19].

6.3 Evaluation Strategy

A comparison between the operators and different parameters were carried to obtain the optimal or near optimal solution different operators produces different solutions, the best solution is the solutions with higher fitness value.

6.4 Problem description

The problem can be described as follows;

There is no difference between class room size, so the class rooms is fixed for each groups

- Number of subjects (10) for the first year, for each subject there is different number of periods in a week.
- Number of teacher (18)
- Total Number of periods (30) per week and
- Each teacher is related to the subject in many groups.

Table (6.1) shows total periods per week and subjects used in the experimental

subject no	subject name	periods per week
1	AsL	3
2	EnG	3
3	Arab	3
4	Math	4
5	Chm	3
6	Phy	4
7	Comp	3
8	Drw	3
9	Pol	2
10	Mil	2
total periods per week		30

6.5 Experiments

Real data is used with these experiments (see Table (4.5), Table (4.6) and Table (6.1)). The final result the weekly school timetable obtained (see Figure (6.7)) . The experiments focus on the best fitness that could be obtained with fixed population size and different generation iterations. The result of implementing this experiment will be represented by curves. Each curve has two axes X and Y (X represents fitness values and Y represents generation iterations) .

The experiments are tested with total number of periods (30) per week , (18) teachers and number of subjects (10) for the first secondary year. Events are lessons, labs and groups. Events are based on data collected from some different Libya secondary

schools. The penalty function and set of constraints are defined in Chapter 4. All the chromosomes passed to GA are feasible solution all the time. The user determines the number of chromosomes that needs to be initialized. These chromosomes are members of first generation. These chromosomes will be saved by the system (see Figure (5.8)) Then these members will be used to produce a child for the next generation. User can choose the population size , type of crossover (one or multi crossover point) and the number of generations , which will be 20, 40, 65 and 100 in this experiment to find out in what generation the fittest chromosome can be obtained. After generating the required numbers, the system will choose the fittest member among members of generations (see Figure (5.10)) . The output is the weekly school timetable. (see Figure (6.7) .

6.6 Evaluation results

All parameters and operators were evaluated with respect to each other.

6.7 Population size

Different population size were evaluated (50,100, 200 and 500) using multi points crossover , random mutation ,the number of generation was (40) for each run the result is shown in table (6.2) and figure (6.1) . The result shows that the large population size doesn't have large influence on the fitness value, but it requires more time to reach the best value. The figure (6.1) shows that in case of population size (50) the best fitness value was lower than best fitness of population size (100) but bigger than the fitness value of the population size 200 and 500 and this is referred for using random selection.

initial population	Generation	crossover	fitness value	
			best	avr
50	40	multi point	3990	3143
100	40	multi point	4160	3704
200	40	multi point	3750	3138
500	40	multi point	3880	3141

Table (6.2) the best fitness of population size (50,100,200,500) and generation 40

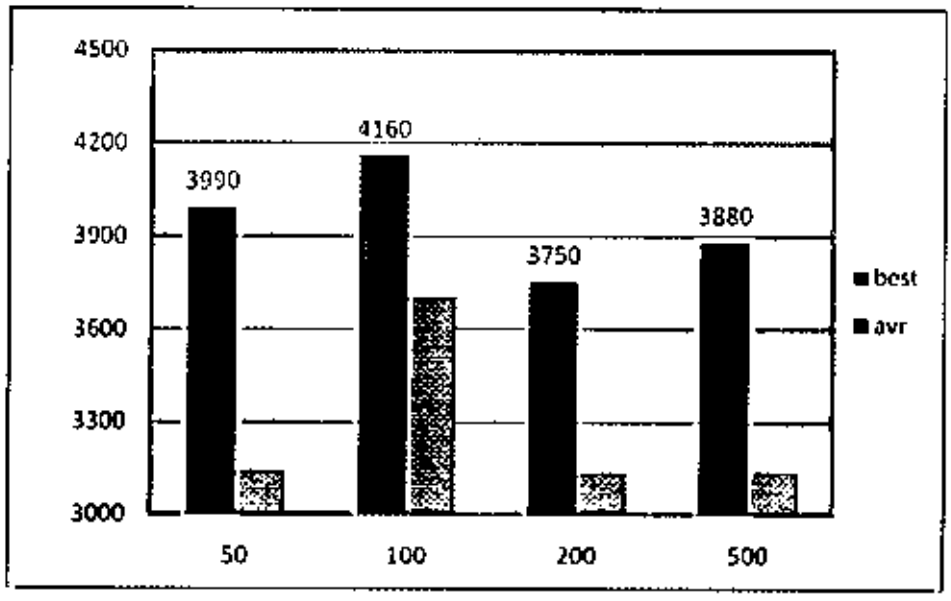


Figure (6.1) best fitness of population size (50,100,200,500) and generation 40

6.8 Crossover

The experiment was carried on two type of crossover schemes (one point crossover and multi points) in order to find their influence on the results (best solution) the results are provided below.

6.9 Best solution

The two types of crossover schemes (one point and multi point) used to produce the best solution (the largest fitness value) using random mutation schemes. The experiment was carried for a number of generation (20,40,65 and 100) with fixed population size 100 .

the following tables ((6.3),(6.4),(6.5),(5.6)) and figures ((6.2),(6.3),(6.4),(5.5)) show the results obtained and used to compare the ability of the crossover scheme to produce the best solution with the same initial population within each run . The ability to obtain the best value (the output is the weekly school timetable see Fig (5.11) section 5 chapter 5).

The chart in figure (6.2) shows the best chromosome implemented (the output is the weekly school timetable) with (20) generation and population size 100.

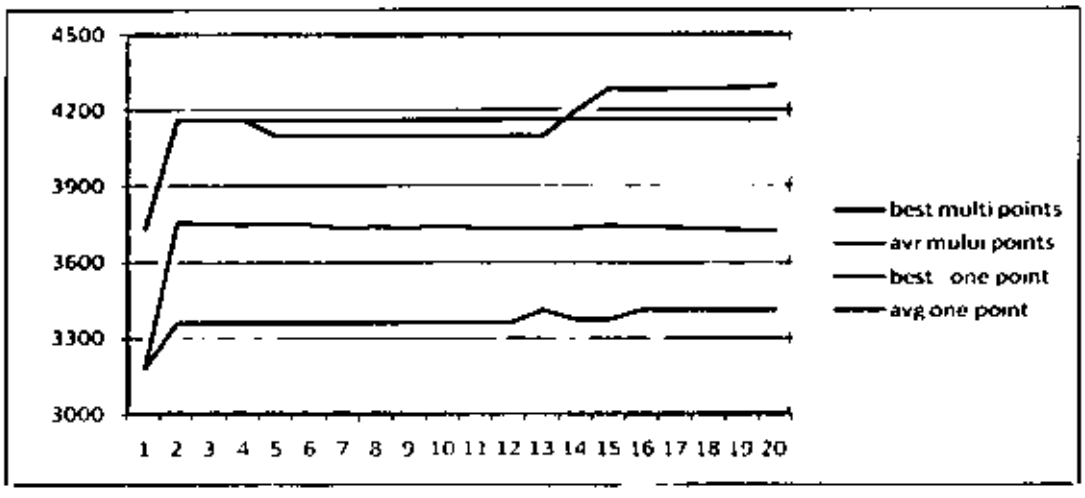


Figure (6.2) the best chromosome implementation (20 generation, population 100)

A sample result is shown in table (6.3) of generation 20 and population size 100.

Mutation Scheme	Generation No	multi points Crossover		one point Crossover	
		Best	Average	Best	Average
Random Mutation	1	3730	3183	3730	3183
	2	4160	3758	4160	3360
	3	4160	3750	4160	3360
	4	4160	3748	4160	3360
	5	4100	3747	4160	3360
	6	4100	3749	4160	3360
	7	4100	3737	4160	3360
	8	4100	3738	4160	3360
	9	4100	3736	4160	3360
	10	4100	3743	4160	3360
	11	4100	3734	4160	3360
	12	4100	3729	4160	3360
	13	4100	3731	4160	3410
	14	4200	3734	4160	3370
	15	4285	3744	4160	3370
	16	4281	3741	4160	3410
	17	4282	3734	4160	3410
	18	4283	3729	4160	3410
	19	4289	3725	4160	3410
	20	4295	3723	4160	3410

Table (6.3) the result of one & multi point crossover with generation 20 and population 100

The chart in figure 6.3 shows the best chromosome implemented (the output is the weekly school timetable) with (40) generation and population size 100.

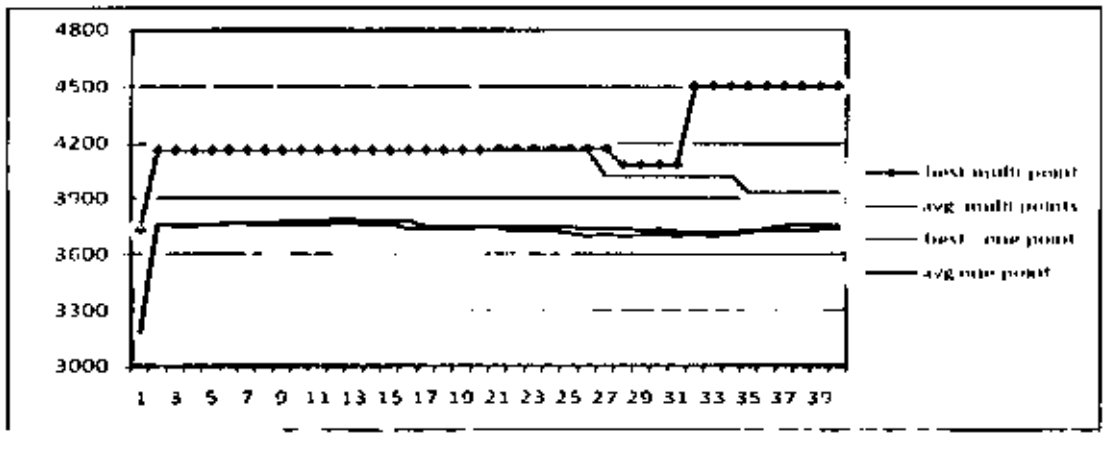


Figure (6.3) the best chromosome implementation (40 generation, population 100)

A sample result is shown in table (6.3) of generation 40 and population size 100.

Mutation Scheme	Generation	multi points Crossover		one point Crossover	
		Best	Average	Best	Average
Random Mutation	30	4060	3710	4020	3732
	31	4060	3698	4020	3724
	32	4500	3710	4020	3718
	33	4500	3698	4020	3717
	34	4500	3710	4020	3716
	35	4500	3719	3930	3728
	36	4500	3738	3930	3727
	37	4500	3754	3930	3730
	38	4500	3760	3930	3726
	39	4500	3754	3930	3732
	40	4500	3754	3930	3740

Table (6.4) the result of one & multi point crossover with generation 40 and population 100

The chart in figure (6.4) shows the best chromosome implemented (the output is the weekly school timetable) with (65) generation and population size 100.

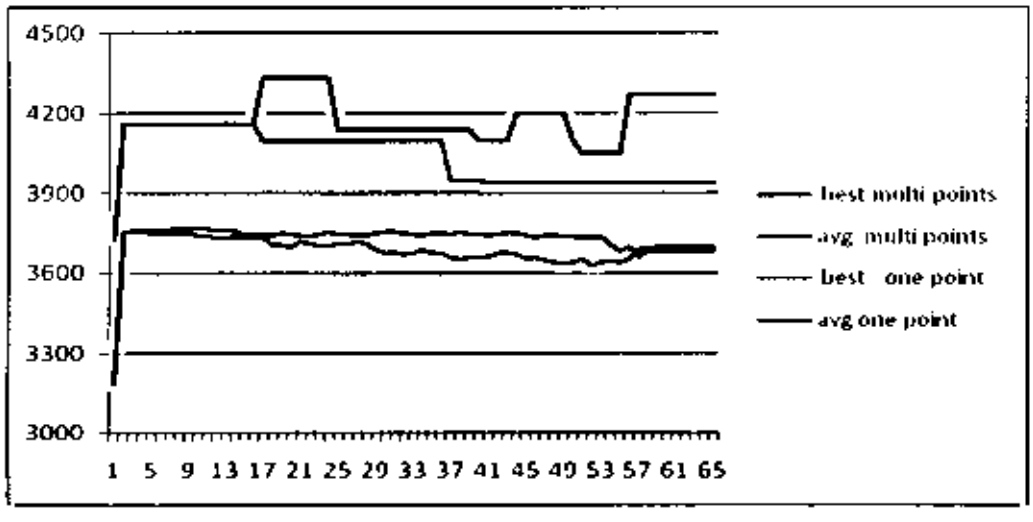


Figure (6.4) the best chromosome implementation (65 generation, population 100)

A sample result is shown in table 6.5 of generation 65 and population size 100.

Mutation Scheme	Generation No	multi points Crossover		one point Crossover	
		Best	Average	best	Average
Random Mutation	10	4160	3742	4160	3768
	11	4160	3741	4160	3769
	12	4160	3732	4160	3761
	13	4160	3732	4160	3758
	14	4160	3737	4160	3761
	15	4160	3738	4160	3746
	16	4160	3732	4160	3747
	17	4330	3738	4100	3742
	18	4330	3706	4100	3744
	19	4330	3707	4100	3752
	20	4330	3697	4100	3748

Table (6.5) the result of one & multi point crossover with generation 65 and population 100

The chart in figure (6.5) shows the best chromosome implemented (the output is the weekly school timetable) with (100) generation and population size 100.

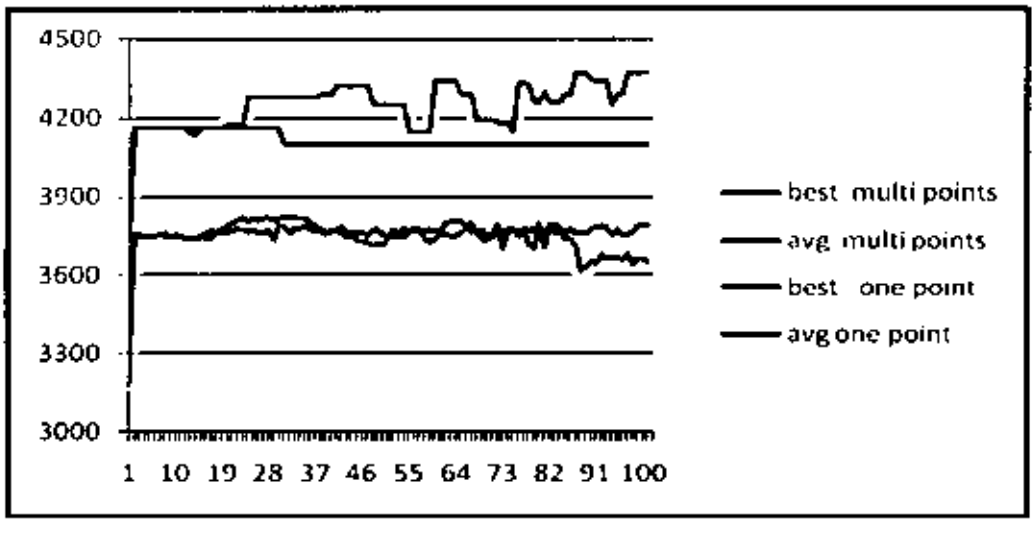


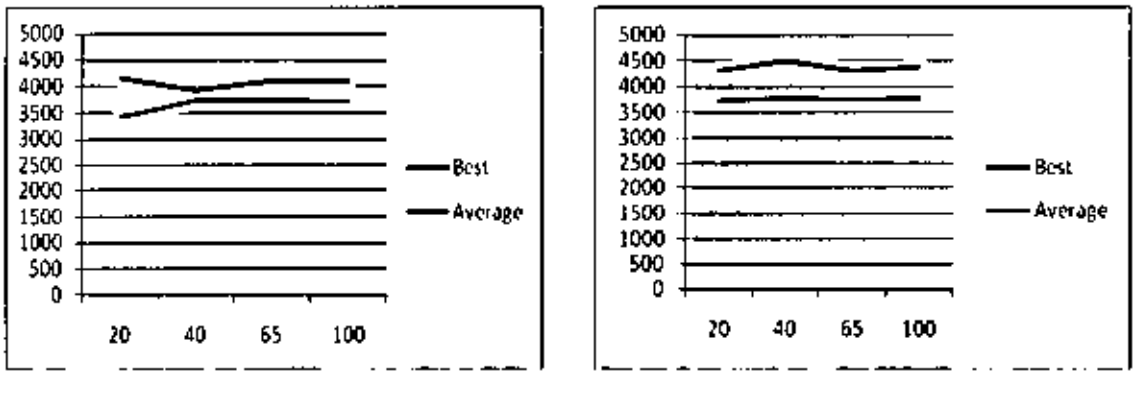
Figure (6.5) chromosome implementation (100 generation, population 100)

A sample result is shown in table 6.6 of generation 100 and population size 100.

Mutation Scheme	Generations No	multi points Crossover		one point Crossover	
		Best	Average	best	Average
Random Mutation	90	4340	3787	4100	3640
	91	4340	3791	4100	3678
	92	4340	3787	4100	3660
	93	4250	3756	4100	3666
	94	4290	3775	4100	3664
	95	4290	3755	4100	3657
	96	4370	3756	4100	3678
	97	4370	3770	4100	3636
	98	4370	3790	4100	3661
	99	4370	3792	4100	3660
	100	4370	3792	4100	3644

Table (6.6) the result of one & multi point crossover with generation 100 and population 100

Figure 6.6 shows the result of comparative between one & multi point crossover with (20, 40, 65 and 100) of population size 100.



(a) One point crossover (b) Multi point crossover

Figure (6.6) result of comparative between one & multi point crossover with (20, 40, 65 and 100) of population size 100

Table 6.7 shows a sample result of comparative between one & multi point crossover with (20, 40, 65 and 100) of population size 100.

Mutation Scheme	Generations No	multi points Crossover		one point Crossover	
		Best	Average	best	Average
Random Mutation	20	4295	3723	4160	3410
	40	4500	3754	3930	3740
	65	4300	3738	4100	3742
	100	4370	3763	4100	3713

Table (6.7) the result of competition crossover schemes

The weekly school timetable obtained from different generations (20, 40, 65 and 100) of population size 100

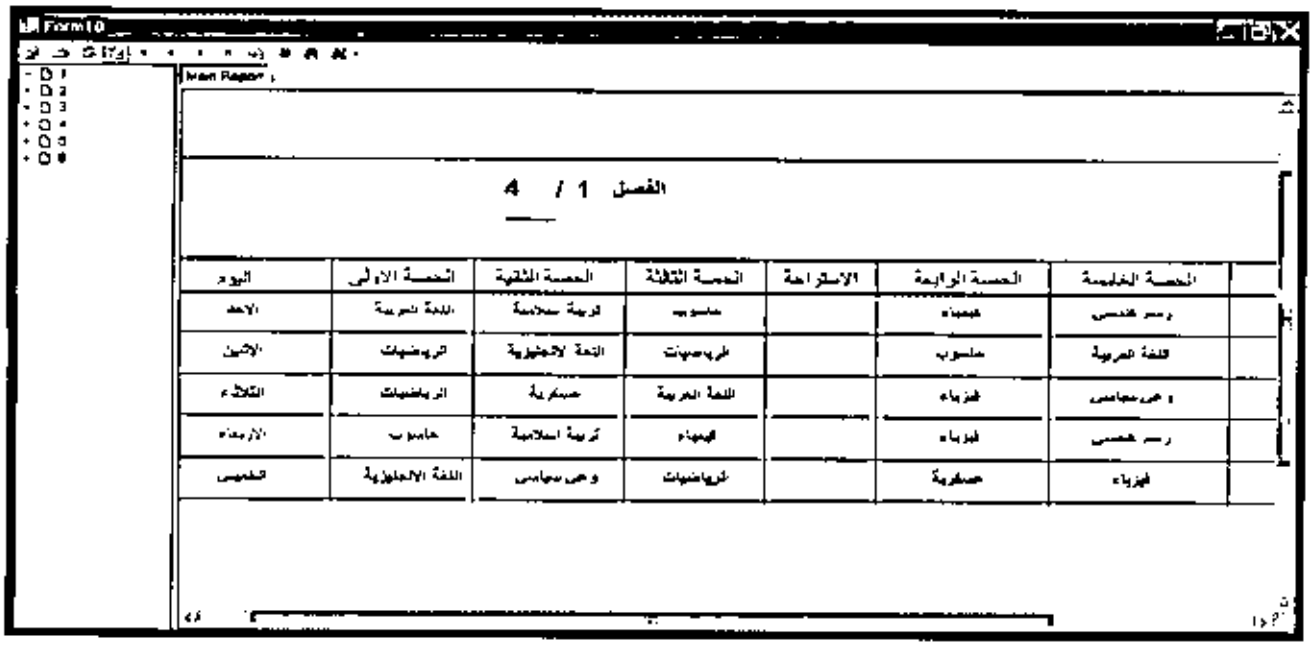


Figure (6.7) the weekly school timetable obtained from different generations (20, 40, 65 and 100) of population size 100

6.10 Conclusion

This thesis developed a school timetable system using GA to fulfill the main objective that is defined in chapter one (objective study) this thesis investigates several references related to GA and there use in real life to develop a system which can be applied In the Libyan secondary school timetable taking in consideration the hard and soft constraints to solve the weekly school timetable problem. Through the developed system the user can enter the initial population of school timetable solutions that are feasible (satisfies hard constraints) all the time. The feasible solutions produced by system are passed to simple GA and evaluate each solution individually to make them satisfy to the soft constraints. The results shows that using multi point crossover is better than one point crossover and the size of the population doesn't have a large effect on the optimal or near optimal solution .

6.11 Future Work

In this study an adaptive GA is designed, developed and applied to the timetabling problem of Libyan secondary schools in order to create feasible and efficient timetables. Simulation results showed that the algorithm is able to construct a valid and very efficient timetable quickly and easily satisfying the hard and soft constraints. The used algorithm allows for criteria adaptation, thus producing different timetables for different constraints priorities. The used algorithm is designed in order to face the timetabling problem of Libyan secondary schools for each level alone. We propose developing this system to produce a complete timetable where the teacher teaches different subjects for different levels. Also a recommendation to compare different GA operators to be carried out.

References

[1] A. Colorni, M. Dorigo, and V. Maniezzo, "Genetic algorithms – A new approach to the timetable problem," In Lecture Notes in Computer Science - NATO ASI Series, Vol. F 82, Combinatorial Optimization, (Akgul et al eds), Springer-Verlag, 1990, pp. 235-239.

[2]Alexander Brownlee,"An Application of Genetic Algorithms to University Timetabling", Honours Project. 2005

[3]Andrea Schaerf (1995) "A Survey of Automated Timetabling" Report CS-R9567 ISSN 0169-118X.

[4]Burhaneddin sandikci ,Genetic Algorithms ,Bilkent University URL: http://www.bilkent.edu.tr/~burhan Spring 2000.

[5]Caldeira JP, Rosa AC, School Timetabling Using Genetic Search, In Burke and Carter, pages 115-122. 19. Patata, 1997.

[6]D. Abramson Ê J. Abela. "A Parallel Genetic Algorithm for Solving the School Timetabling Problem", 15 Australian Computer Science Conference, Hobart, Feb 1992.

[7]D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Pub. Co.1989. ISBN: 0201157675.

[8]Edin Mulalić, Leonid Stoimenov, Branislav Randelović Genetic algorithms with optimization methods school timetabling problems www.branislavrandjelovic.com/RanYuInfo2007.pdf.

[9] Fang H-L.,"Investigating genetic algorithms for scheduling", MSc Thesis, University of

Edinburgh, 1992.

[10]Filev, A.,Loton,T., McNeish,K., Schoellmann, B.,Slater,J.,and Wu,C. (2003) Professional UML with Visual Studio .NET: Unmasking visio for Enterprise Architets. Wrox press Ltd, Birmingham.

[11]G N. Beligiannisa,b et, Applying evolutionary computation to the school timetabling problem: 2006 Elsevier Ltd. All rights reserved. doi:10.1016/j.cor.2006.08.010

[12] GN Beligiannis1 , C Moschopoulos2 and SD Likothanassis , A genetic algorithm approach to school timetabling , University of Ioannina, Agrinio, Greece; and University of Patras, Rio, Patras, Greece puplished2007

[13]H. Fang, PhD, Thesis. Genetic algorithms in Timetabling and Scheduling, Department of Artificial Intelligence, University of Edinburgh, Scotland, 1994.

[14]Halvorson, M. (2005) Microsoft Visual Basic 2005 Step by Step Microsoft press. Redmond,Washington.

[15]Hamilton, K., and Miles, R. (2006) learning UML 2.0. O .Reilly.

[16]Henri Jan Pierrot., An investigation of Multi-chromosome Genetic Algorithms 1997

[17]Ho Sung C. Lee., Timetabling Highly Constrained System via Genetic Algorithms. 2000, University of the Philippines

[18]J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan, 1975.

[19]Leon Bambrick, Lecture Timetabling Using Genetic Algorithms. Undergraduate Thesis Bachelor of Computer Systems Engineering Department of Electrical and Computer

Engineering, University of Queensland, 1997.

[20]Luís Paulo Reis and Eugénio Oliveira A Language for Specifying Complete Timetabling Problems.

[21]M. Mitchell. An Introduction To Genetic Algorithms. Cambridge, MA: MIT Press 1996.

[22]Melanie Mitchell, Genetic Algorithms: An Overview, Santa Fe Institute,1399 Hyde Park Road, Santa Fe, NM 87501, 1995.

[23]N. A. AL-Madi, A. T. Khader, "A SOCIAL-BASED MODEL FOR GENETIC ALGORITHMS". Proceedings of the third International Conference on Information Technology (ICIT), AL-Zaytoonah Univ., Amman, Jordan, 2007.

[24Nagham Azmi AL-Madi, Ahamad Tajudin Khader De Jong's Sphere Model Test for A Social-Based Genetic Algorithm (SBGA) IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.3, March 2008

[25]Petres Z., Gy"ori S. and Várkonyi-Kóczy A.R, "Genetics Algorithms in Timetabling. A New Approach", MFT Periodika, 7, 2001

[26]Raja N Ainon, & Salihin F. 'Development of a Scheduling Tool for Constucting a Malaysian School Timetable ' University of Malaya

[27]Robertus Johannes Willemen, "School timetable construction Algorithms and complexity" Technische Universiteit Eindhoven, 2002. Proefschrift, ISBN 90-386-1011-4 NUGI 855.

[28]Roger L .Wainwright., Introduction to Genetic Algorithms, Theory and Applications, the Seventh Oklahoma Symposium on Artificial Intelligence, November 19, 1993.

[29]Sachi, Timetabling using Cellular Genetic Algorithms with Adapted Mutation Operator. Final Year Project Report, Departments of Computer Science, University of York 2001.

[30]Y.Zheng,S.Kiyooka,"GeneticAlgorithmApplications",www.me.uvic.ca/~zdong/course/mech 620/GA_App.PDF/, 1999.

الملخص

توضح هذه الدراسة استخدامات الخوارزمية الوراثية (المتسلسلة والمتوازية) في الجداول الزمنية المدرسية ولقد تم إجراء دراسة مقارنة علي تطبيقات الخوارزمية الوراثية بين الدراسات السابقة بهدف الحصول علي جدول زمني فعال وعملي يتناسب مع مراحل التعليم الثانوي في ليبيا . ويمكن الحصول علي جداول دراسية مختلفة وذلك من خلال تطبيق قيود مختلفة وباستخدام المعاملات (التقاطع في نقطة واحدة أو التقاطع في نقاط متعددة).

استخدمت الدراسات السابقة الخوارزمية الوراثية لحل مشكلة الجدولة الزمنية في المدارس بطرق مختلفة معتمدة علي معاملات الخوارزمية الوراثية (الاختيار والتقاطع والتبديل) بهدف الوصول إلي حل قريب من المثالي الفرق الرئيسي أو الأساسي بين هذه الدراسات هو تمثيل الكرموسوم والذي يعتبر من أهم العوامل المؤثرة للحصول علي الحل المناسب بشكل ملائم وبسرعة. مصطلح الترميز يستخدم لوصف طريقة تمثيل الجدول الأسبوعي . بمقارنة الدراسات السابقة مع نظام المدارس الثانوية الليبية من الملائم تصميم وتنفيذ الخوارزمية البسيطة(المتسلسلة) للحصول علي جدول زمني للمدارس الثانوية الليبية يكون فيه مجموعات الطلبة والفصول والفترات الزمنية ثابتة وجدولة المواد علي مجموعة من الفترات الزمنية مع الاحتفاظ بتأمين الشروط الصعبة واللينة .

في التصميم المقترح أكثر من مدرس يمكن إن يقوم بتدريس المادة بمجموعات مختلفة في فترات زمنية مختلفة مع تجنب حدوث تعارض (مشكلة تعارض المواد أو الفصول) . سيتم تطبيق معاملات الخوارزمية الوراثية للحصول علي كرموسوم (جدول زمني جديد) بهدف الحصول علي حل مثالي .

دراسة مقارنة للخوارزميات الوراثية وتطبيقاتها في المدارس الثانوية

مقدمة من الطالب :-

عثمان عمران بالقاسم محمد

إشراف:- الدكتور عبدالحميد عبدالكافي

رسالة مقدمة لقسم الحاسوب كمتطلب جزئي للحصول علي درجة الماجستير في علم الحاسوب

جامعة التحدي - كلية العلوم

قسم الحاسوب

العام الجامعي2009\2010

الجماهيرية العربية الليبية الشعبية الاشتراكية العظمى
اللجنة الشعبية العامة للتعليم والبحث العلمي
جامعة التحدي

إن الدراسة ليست غاية في حد ذاتها
وانما الغاية هي خلق الإنسان النموذجي المجتهد

الرقم الإشاري : ج ت س/024/ 2010//152 م

التاريخ

الموافق 2010/3/30 ميلادي

كلية العلوم

قسم الحاسوب

عنوان البحث

دراسة مقارنة للخوارزميات الوراثية و تطبيقاتها في المدارس الثانوية

مقدمة من الطالب

عثمان عمران بالقاسم

** لجنة المناقشة :

1 – د.عبد الحميد محمد عبدالكافي ..... (مشرفاً)

2 – د.زكريا سليمان زوبي ..... (ممتحناً داخلياً)

3 – د.ناجي امحمد بازينة ..... (ممتحناً خارجياً)

وعتمد

أمين اللجنة الشعبية لكلية العلوم

30/05/2010

WWW.alitahadi.edu.ly
info@alitahadi.edu.ly
+218 54 52 60363- 52 65704
+218 54 52 60 361 - 52 62 152
ص.ب 674