TEACHER INFORMATION SYSTEM USING

GENETIC ALGORITHM

Master Thesis Dissertation

SANJU VARGHESE  (HW00175012)

MSc IT – Software Systems

Heriot Watt University, Dubai

Supervisor:

Mrs Smitha Kumar

Second reader:

Dr. Rob J. Pooley
DECLARATION

I, Sanju Varghese, confirm that this work submitted for assessment is my own and is expressed in my own words. Any uses made within it of the works of other authors in any form (e.g.: ideas, equations, figures, texts, tables, programs) are properly acknowledged at any point of their use. A list of the references employed is included.

Dubai, August 2015
Abstract

One of the most valued resources in an educational institution is teachers. A structured environment is essential for the effective working of all resources in an institution. The most critical factor behind the success of an educational institute is a well-planned and error free timetable. In the earlier days it was manually created by the administrative staff which could take several weeks of repeated trials and consultations until a final timetable evolved. Timetabling is a hard to solve problem as it has many possible solutions and numerous constraints to satisfy.

A teacher information system is being developed which will contain information about teachers, a timetable generator, and cover lesson and tests scheduler. A genetic algorithm has been applied to this scheduling problem in order to create a feasible solution with fewer clashes in minimal time.
Acknowledgement

I would like to thank my supervisor Smitha Kumar for her guidelines and suggestions throughout the entire project.

I would also like to thank my colleagues and friends for their time. The interview conducted with them (Appendix A) was a major influence regarding the system evaluation.

I would also like to thank my family for their encouragement and support.
# CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Declaration</td>
<td>2</td>
</tr>
<tr>
<td>Abstract</td>
<td>3</td>
</tr>
<tr>
<td>Acknowledgement</td>
<td>4</td>
</tr>
<tr>
<td>List of figures</td>
<td>7</td>
</tr>
<tr>
<td>List of tables</td>
<td>9</td>
</tr>
<tr>
<td>1. Introduction</td>
<td>10</td>
</tr>
<tr>
<td>1.1 Motivation</td>
<td>10</td>
</tr>
<tr>
<td>1.2 Aims and objectives</td>
<td>11</td>
</tr>
<tr>
<td>1.3 Significance of study</td>
<td>11</td>
</tr>
<tr>
<td>1.4 Limitations</td>
<td>12</td>
</tr>
<tr>
<td>1.5 Report outline</td>
<td>12</td>
</tr>
<tr>
<td>2. Literature review</td>
<td>13</td>
</tr>
<tr>
<td>2.1 Background study of existing approaches</td>
<td>13</td>
</tr>
<tr>
<td>2.2 Study of Genetic algorithm</td>
<td>16</td>
</tr>
<tr>
<td>2.2.1 Biological background</td>
<td>16</td>
</tr>
<tr>
<td>2.2.2 Outline of Genetic algorithm</td>
<td>17</td>
</tr>
<tr>
<td>2.2.3 Process of Genetic algorithm</td>
<td>19</td>
</tr>
<tr>
<td>2.2.3.1 Representation techniques or encoding</td>
<td>19</td>
</tr>
<tr>
<td>2.2.3.2 Selection methods</td>
<td>20</td>
</tr>
<tr>
<td>2.2.3.3 Operators of genetic algorithm</td>
<td>22</td>
</tr>
<tr>
<td>2.3 Flowchart of Genetic algorithm</td>
<td>25</td>
</tr>
<tr>
<td>2.4 Strengths of Genetic algorithm</td>
<td>26</td>
</tr>
<tr>
<td>2.5 Limitations of Genetic algorithm</td>
<td>22</td>
</tr>
<tr>
<td>3. System Study</td>
<td>27</td>
</tr>
<tr>
<td>3.1 Study of existing system</td>
<td>27</td>
</tr>
<tr>
<td>3.1.1 Advantages of existing system</td>
<td>27</td>
</tr>
<tr>
<td>3.1.2 Limitations of existing system</td>
<td>27</td>
</tr>
<tr>
<td>3.2 Proposed system</td>
<td>28</td>
</tr>
<tr>
<td>3.2.1 Advantages of proposed system</td>
<td>29</td>
</tr>
<tr>
<td>3.2.2 Limitations of proposed system</td>
<td>29</td>
</tr>
<tr>
<td>3.3 Functional Requirements</td>
<td>30</td>
</tr>
<tr>
<td>3.4 Non-Functional Requirements</td>
<td>31</td>
</tr>
<tr>
<td>3.5 Evaluation Requirements</td>
<td>31</td>
</tr>
<tr>
<td>4. System design and Project plan</td>
<td>32</td>
</tr>
<tr>
<td>4.1 Tools and technologies</td>
<td>32</td>
</tr>
<tr>
<td>4.2 Database design and implementation</td>
<td>33</td>
</tr>
<tr>
<td>4.2.1 Entity relationship diagram</td>
<td>35</td>
</tr>
<tr>
<td>4.3 UML representation of proposed system</td>
<td>36</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>4.4 Project plan</td>
<td>37</td>
</tr>
<tr>
<td>4.5 Professional, Legal and Ethical issues</td>
<td>38</td>
</tr>
<tr>
<td>5. Development and Testing</td>
<td>39</td>
</tr>
<tr>
<td>5.1 Stages of development</td>
<td>39</td>
</tr>
<tr>
<td>5.1.1 Module 1</td>
<td>39</td>
</tr>
<tr>
<td>5.1.2 Module 2</td>
<td>40</td>
</tr>
<tr>
<td>5.1.3 Module 3</td>
<td>42</td>
</tr>
<tr>
<td>5.2 Pseudo code of genetic algorithm</td>
<td>43</td>
</tr>
<tr>
<td>5.3 UML class diagram</td>
<td>44</td>
</tr>
<tr>
<td>5.4 Code extracts of genetic algorithm</td>
<td>45</td>
</tr>
<tr>
<td>5.5 Testing of web pages and database</td>
<td>50</td>
</tr>
<tr>
<td>5.5.1 Testing objectives</td>
<td>50</td>
</tr>
<tr>
<td>5.5.2 Software requirements for testing</td>
<td>50</td>
</tr>
<tr>
<td>5.5.3 Individual modules test scenarios and results</td>
<td>51</td>
</tr>
<tr>
<td>5.5.3.1 Module 1 testing</td>
<td>51</td>
</tr>
<tr>
<td>5.5.3.2 Module 2 testing</td>
<td>57</td>
</tr>
<tr>
<td>5.5.3.3 Module 3 testing</td>
<td>62</td>
</tr>
<tr>
<td>6. Evaluation</td>
<td>65</td>
</tr>
<tr>
<td>6.1 Evaluation plan</td>
<td>65</td>
</tr>
<tr>
<td>6.1.1 Interviews with non-teaching staff</td>
<td>66</td>
</tr>
<tr>
<td>6.1.2 Interviews with teaching staff</td>
<td>67</td>
</tr>
<tr>
<td>6.2 Evaluation discussion</td>
<td>68</td>
</tr>
<tr>
<td>7. Summary</td>
<td>70</td>
</tr>
<tr>
<td>7.1 Achievements</td>
<td>70</td>
</tr>
<tr>
<td>7.2 Limitations</td>
<td>71</td>
</tr>
<tr>
<td>7.3 Future recommendations</td>
<td>71</td>
</tr>
<tr>
<td>7.4 Conclusion</td>
<td>72</td>
</tr>
<tr>
<td>References and bibliography</td>
<td>73</td>
</tr>
</tbody>
</table>
**LIST OF FIGURES**

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Model of a chromosome</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>Basic steps of a genetic algorithm</td>
<td>18</td>
</tr>
<tr>
<td>3</td>
<td>Binary encoding technique</td>
<td>19</td>
</tr>
<tr>
<td>4</td>
<td>Value encoding technique</td>
<td>19</td>
</tr>
<tr>
<td>5</td>
<td>Tree encoding technique</td>
<td>20</td>
</tr>
<tr>
<td>6</td>
<td>Roulette wheel selection</td>
<td>21</td>
</tr>
<tr>
<td>7</td>
<td>Single point crossover technique</td>
<td>22</td>
</tr>
<tr>
<td>8</td>
<td>Uniform crossover technique</td>
<td>23</td>
</tr>
<tr>
<td>9a</td>
<td>Before tree encoding crossover technique</td>
<td>23</td>
</tr>
<tr>
<td>9b</td>
<td>After tree encoding crossover technique</td>
<td>24</td>
</tr>
<tr>
<td>10</td>
<td>Binary encoding mutation</td>
<td>24</td>
</tr>
<tr>
<td>11</td>
<td>Flowchart of Genetic algorithm</td>
<td>25</td>
</tr>
<tr>
<td>12</td>
<td>ER diagram of the proposed system</td>
<td>35</td>
</tr>
<tr>
<td>13</td>
<td>Use Case diagram of the proposed system</td>
<td>36</td>
</tr>
<tr>
<td>14</td>
<td>Use case diagram of module 1</td>
<td>40</td>
</tr>
<tr>
<td>15</td>
<td>Use case diagram of module 2</td>
<td>41</td>
</tr>
<tr>
<td>16</td>
<td>Use case diagram of module 3</td>
<td>42</td>
</tr>
<tr>
<td>17</td>
<td>UML class diagram of genetic algorithm implementation</td>
<td>44</td>
</tr>
<tr>
<td>18</td>
<td>Creating a population for genetic algorithm</td>
<td>45</td>
</tr>
<tr>
<td>19</td>
<td>Fitness Calculation for genetic algorithm</td>
<td>45</td>
</tr>
<tr>
<td>20</td>
<td>Fitness Calculation comparison for genetic algorithm</td>
<td>46</td>
</tr>
<tr>
<td>21</td>
<td>Tournament selection for crossover</td>
<td>46</td>
</tr>
<tr>
<td>22</td>
<td>Crossover and mutation functions for genetic algorithm</td>
<td>47</td>
</tr>
<tr>
<td>23</td>
<td>Evolving new population after genetic operations</td>
<td>48</td>
</tr>
<tr>
<td>24</td>
<td>Main function for genetic algorithm</td>
<td>49</td>
</tr>
<tr>
<td>25</td>
<td>Application build</td>
<td>51</td>
</tr>
<tr>
<td>26</td>
<td>Web page loading</td>
<td>52</td>
</tr>
<tr>
<td>27</td>
<td>User link enabling the registration of a new user</td>
<td>52</td>
</tr>
<tr>
<td>28</td>
<td>database results verified for new user registration</td>
<td>52</td>
</tr>
</tbody>
</table>
Figure 29: Login enabled for registered user

Figure 30: Access denied - Incorrect username or password

Figure 31: Teacher profile

Figure 32: Creating a new Teacher profile

Figure 33: Database results verified for the new teacher profile created

Figure 34: New profile and message displayed

Figure 35: Existing profile deleted and message displayed

Figure 36: Database results verified for profile deletion

Figure 37: Timetable generation module

Figure 38: Success message displayed

Figure 39: Display results of genetic algorithm

Figure 40: Database results verified for new teacher timetable generated

Figure 41: Webpage for viewing the generated timetables

Figure 42: Webpage for viewing the school timetables

Figure 43: School Timetable Download option

Figure 44: Folder containing the downloaded timetables

Figure 45: Individual teacher timetable display

Figure 46: Teacher timetable download option

Figure 47: Folder containing the downloaded teacher timetables

Figure 48: Subject lessons covered by available subject teachers

Figure 49: Teacher timetable to check availability

Figure 50: Lessons covered by non-subject teachers

Figure 51: Tests scheduler results
LIST OF TABLES

Table 1 Gantt chart for project plan ......................................................... 37
Table 2 Test conditions of module 1 ............................................................. 51
Table 3 Test conditions of module 2............................................................... 57
Table 4 Test conditions of module 3............................................................... 62
Chapter 1

Introduction:

1.1 Motivation

A timetable indicates the time slots at which certain tasks or actions are expected to take place. Timetable scheduling is a common problem within every educational institution which tries to coordinate teachers, students, classrooms and time periods based on the policies of the institution. It is important to schedule resources effectively for smooth functioning of the institution. The scheduling is usually done by a person who might take several days or weeks to complete due to the conflicts that may arise in trying to effectively utilise the resources.

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 [1]. It can be described as the allocation of resources against a set of requirements so as to maximise the possibility of allocation and minimise the violation of constraints. There are broadly two types of requirements that are considered during the scheduling of a timetable. Hard constraints are the conditions to which a timetable must strictly adhere to and soft constraints are the requirements that would be preferred to be satisfied. Each institution has a different set of hard and soft constraints that must be addressed in the timetable.

Hard constraints have been identified as:

- A teacher can take only one class during a given time slot
- A group of students can attend only one subject during a given time slot.
- There can be only one subject in a classroom at a given time slot.
- No classes can be scheduled during the two allotted break periods.
- Classes cannot be allotted during the activities lesson.
Soft constraints have been identified as:

- A teacher should not be assigned for more than 2 lessons continuously.
- A group of students should not have two timeslots of a subject on the same day.
- No additional duties for teachers on full time table load.
- Number of students must be less than or equal to the room's capacity.

The scheduling of a school's timetable revolves around a set of time slots and a set of limited resources (teachers, students and class rooms).

1.2 Aims and Objectives

Different institutions have different policies upon which a timetable gets created. Lesson timings, number of students in a student group, teachers' allocation, room allocation may all vary for different institutions. Timetable scheduling is a multi-dimensional problem and each dimension may add on to its own complexity. Therefore a common solution may not be developed for use by all the institutions.

The aim of this project is to create a website that contains teachers' information, a timetable generator and a cover lesson & tests scheduler. Optimal timetables can be generated automatically using a computer with little to no clashes or errors through the design and development of timetabling software.

The objective is to use genetic algorithms in the development of the timetable generator and to construct the most optimal timetable with little or no redundancy.

1.3 Significance of the study

This project will take into consideration the scheduling of classes from primary level to high school level classes. Input data to the system must be provided appropriately in order to derive the most optimal solutions at all times.

- The proposed system will add flexibility to the task of timetable generation.
- The proposed system will enable time saving.
- Timetables generated may be revised by altering the database data.
- Less people involved in the tedious task and more productivity.
1.4 Limitations

Generating an optimal timetable depends entirely on the data input to the system. Major discrepancies in the data may result in clashes or errors.

- Incorrect constraint specification
- Incorrect input data
- Incomplete data
- System performance

1.5 Report outline

The report will contain five sections.

Chapter 2 contains the review of various research works in the field of scheduling and the background study of genetic algorithms.

Chapter 3 contains the study of existing and proposed systems.

Chapter 4 contains the system design, testing plan, project plan and PLE issues.

Chapter 5 contains the conclusion of the project.

References and bibliography used for the study and research of this project have been listed after chapter 5.
Chapter 2

Literature review

2.1 Background study of existing approaches

The scheduling of a school timetable is an important and continuing activity of every educational organisation. It involves the effective allocation of teachers, students and class rooms against pre-determined time slots in order to maximise performance of the institution and minimise conflicts.

The main goal of scheduling is to assign time slots to the resources while satisfying all the hard constraints and minimizing soft constraints. When all the hard constraints are satisfied a feasible timetable is created but the quality of the timetable depends upon the extent to which the soft constraints are additionally satisfied. Therefore the basic aim is to create a feasible and quality ensured timetable.

Various methods have been developed to solve the complex optimization problem of timetable scheduling. There are broadly four categories of methods developed:

a) Sequential method such as the Graph colouring techniques [2]

b) Clustering methods [3]

c) Constraint based methods like Integer programming [4]

d) Heuristic methods like Genetic algorithms [5], Tabu search [6], Ant colony systems[7], Sequential Annealing[8]

The timetabling problem is being continuously studied using different approaches and different methods. Initial studies were performed for primary schools and as the number of constraints were limited, linear or integer programming techniques were designed and implemented [9][10]. In case of higher secondary schools and universities the timetabling problems becomes more complex as the number of constraints increased relatively and its applications caused the linear programming techniques inadequate to solve it.
Further researches developed the heuristic approaches to solving the timetabling problems. A heuristic approach is one which aids complex problem solving, guaranteeing solutions that may not be optimal or perfect but will achieve all the goals in minimal time. Meta heuristic approaches are broadly divided into local search based and population based techniques.

Local area search based techniques perform single directional search and the quality of the solution is relatively high. Examples of this local search based technique are Simulated Annealing (SA) and Tabu search. Population based also called global-area based techniques begins with a number of solutions and refines the solutions further to get an optimal solution. Examples of population based search techniques are Evolutionary Algorithms, Particle Swarm optimization, ant colony optimization.

Ant colony optimization algorithm [11] is used in the framework of swarm intelligence where ants are used as agents to construct events to time slots assignments using heuristics and pheromone information. These assignments are then improved using local search methods and the pheromone matrix is updated accordingly for the next loop. Honey bee mating optimization [12] is also another approach towards solving the exam and course scheduling.

Another approach is the Simulated Annealing (SA) [13] where a two phased algorithm is used with a new neighbourhood structure which swaps between pairs of timeslots instead of assignments.

In particle swarm optimization [14] approach, a population of particles (solutions) are initially created and then improved over a pre-defined number of generations (iterations). In this approach, timetables are evolved by swapping the tuples or columns.

Tabu search is a local area based technique [15] in which a two dimensional matrix is used to represent the school timetable. An initial solution was generated randomly and the tabu search used to determine a feasible timetable. Swapping of tuples are performed to move from one neighbourhood to the next.
Local area and global area (population search) based algorithms have been combined together to solve the timetabling problems. Genetic algorithms have been modified to improve its performance by using modified genetic operators, heuristic operators and local search techniques. Genetic algorithms are stochastic search techniques based on the mechanism of natural selection and genetics [16].

The use of genetic algorithm to solve timetabling problem was investigated by Fang [17] in his doctoral thesis. He presented a framework with the following features: constraints specification and a specialised function for the evaluation of solutions.

Fernandes [18] grouped constraints of timetabling problems into strong and weak constraints. Any violations made to the strong constraints would result in a non-feasible solution to the problem, whereas violations to the weak constraints would result in a feasible solution but with limited quality.

Various other approaches have also been studied in the course of this project work, some of which include neural networks [19], knowledge based and constraint logic programming [20]. The models generated by some of these techniques cannot be customized to handle any changes.

Genetic algorithms are used popularly for solving the complex timetabling problems mainly because it does not include tough mathematical calculations; efficiency of the genetic operators makes the algorithm effective in global searches; flexibility in allowing for hybridization. Hence genetic algorithms have been chosen for the implementation of this project work.
2.2 Study of Genetic algorithm

2.2.1 Biological Background

The basic building block of a living organism is the cell. The cell contains nucleus and the nucleus holds genetic information in the form of chromosomes. Chromosomes are strands of DNA containing genes. The fundamental unit of information in a living system is a gene [21]. The science that deals with the mechanisms responsible for similarities and differences in a species is called Genetics [22].

The figure below shows a chromosome and the presentation of a gene in a DNA strand [22, p17]

![Figure 1: Model of a chromosome](image)

A gene pool can contain all the possible variations of characteristics for the future generations. The size of this pool can also determine the diversity of the individuals.

Chromosomes can be arranged as paired (diploid) or unpaired (haploid). All sexually reproducing species have diploid chromosomes consisting of 23 pairs in each cell. During reproduction, a full set of chromosomes are generated from the parents' genes by exchanging and recombining. From the new set of chromosomes, elementary bits of DNA get changed often resulting from copying errors. The fitness of an organism can be determined by its ability of the organism to reproduce.

Organisms having haploid chromosome structure will reproduce to generate offsprings by exchanging the parents' single stranded genes.
2.2.2 Outline of Genetic Algorithm

Inspired by Darwin’s theory, Genetic algorithm is an adaptive search technique based on the idea of natural selection and genetics, highly successful in solving complex constraint specific problems [23]. John Holland invented Genetic Algorithm in the early 1970's. In his book “Adaptation in natural and artificial systems” he described the application of the genetic algorithms. The algorithm has further been developed and it has now evolved as a very powerful tool for solving optimization problems. The basic technique of the algorithm is designed to mimic the processes of natural systems which follow the principles of Charles Darwin’s “survival of the fittest”.

John Holland's original idea was to classify items and then selectively rear those items to produce new ones that can be further classified [24]. Therefore genetic algorithms are capable of finding optimal solutions for most of the complex problems.

The process begins with a population (group) of randomly generated individuals (solutions) and this happens in generations (iterations). Within each generation individuals are tested for fitness using a method, multiple fit individuals are then selected and modified to form a new generation. Process is repeated until an optimal solution has been generated or the required fitness levels have not been achieved.

The term chromosome refers to the solution of a problem in genetic algorithms. It is most often encoded as a bit string; it can also be encoded as integer or real numbers or some permutations. A gene refers to either a single bit or blocks of adjacent bits in a chromosome.

The figure (Fig 2) given below indicates the basic steps in genetic algorithms as shown in Kim-Fung, Kit and Sam’s Genetic Algorithms: Concepts and design book.

a) Create and maintain a population set (search space) consisting of random solutions for the problem.

b) Selection process to identify parents for the next generation by evaluating fitness.

c) The selected fit parents will then create a child using methods such as crossover and mutation.
d) Finally add the new child into the population by removing the old population and repeat the processes until the best solution has been derived.

The above figure [21, p11] clearly shows that for a specific problem the input is a set of solutions encoded in a predetermined fashion, and a method named fitness function which evaluates each solution quantitatively to select the fittest solution. Most of the solutions in the randomly created set may not work and gets deleted from the set after undergoing the fitness function.

However, fit or promising solutions are retained and made to reproduce and create new offspring. Several copies of the fit solutions are made and new changes are injected during the process of copying. These new solutions are then added on to the set of solutions and subjected to a new round of fitness function evaluation. The new round of fitness function
evaluates all the solutions in the pool and eliminates those that are weak after the injected changes and retains those that have improved in providing complete and efficient solutions to the problem. The improved solutions are again subjected to selection and copying with random changes into a new generation and the above process repeats itself until an optimal solution has been derived.

2.2.3 Process of genetic algorithm

The processes explained in detail here are the encoding of chromosomes, fitness function for selection and the injection of random changes.

2.2.3.1 Representation techniques or Encoding

Encoding of chromosomes is a method of representing the genetic information in chromosome. It may be done in the form of bit strings, strings of letters, integer or decimal numbers, or even as data structures called trees.

Potential solutions to a problem must first be represented in a computer understandable format and this method of representation is called encoding. The most common approach is Binary encoding which is to represent or encode as bit strings, a sequence of 1’s and 0’s [25]. The figure below indicates how binary encoding is done in a chromosome:

| Chromosome 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| Chromosome 2 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |

**Figure 3:** Binary encoding technique

Another approach used towards representing or encoding information in a chromosome is in the form of numbers or strings of letters. This is a technique used in complicated problems and is mostly referred to as Value encoding. The figure given below indicates value encoding of a chromosome:

| Chromosome 1 | 6 | 4.1 | 3.2 | 5 | 7 |
| Chromosome 2 | Sam | Alex | Matt | Wayne |

**Figure 4:** Value encoding technique
John Koza, Stanford University developed a new representation method in which programs can be represented as data structures called trees [26]. The best feature of this technique is the injection of changes which can be done by changing an operator, value at a particular node of the tree or even by replacing one entire sub tree with another one. This is the best method for evolving expressions or programs and is mostly referred to as the Tree encoding. The figure below indicates tree encoding [28, p9]:

![Tree Encoding Diagram]

Figure 5: Tree encoding technique

The tree structure shown above in the figure can be expressed as

\[
\text{OR (AND (NOT D0) (NOT D1)) (AND D0 D1)}
\]

2.2.3.2 Selection methods

One of the important functions in the process of genetic algorithms is the selection of the fittest solutions from the pool of solutions. This method plays a very important role in determining the most eligible solutions for the next generation. There are numerous selection methods, a few of which are: Elitist selection, Fitness proportionate selection, Roulette wheel selection, Scaling selection, Tournament selection, Rank selection, Generational selection, Steady state selection, Hierarchical selection.

The most commonly used methods discussed here are Roulette wheel, Steady-state selection and Rank selection.
Roulette Wheel selection technique: This is a widely used technique in which parents are selected according to their fitness. Steps involved in this selection technique are:

- **Sum** – Calculate total fitness $T$ of all the chromosomes in the population pool.
- **Select** – Select a random number $n$ in the range $0$ to $T$.
- **Loop** – For each chromosome in the pool sum up the fitness value and when it more than a specific fitness criteria, the chromosome is selected.

The figure given below shows the roulette wheel representation for 5 chromosomes [21, p9]:

![Roulette Wheel Selection](image)

**Figure 6:** Roulette Wheel selection

The circumference of the roulette wheel is the total fitness $T$ of all the chromosomes. In the figure given above chromosome 4 owns the largest area on the roulette wheel and so stands a greater chance of getting selected when then wheel is spun whereas the chromosome 1 has the smallest slice on the wheel and hence stands the lowest chances of getting selected. The fitness criteria for selection can be number of spins of the wheel, amount of variations in solutions, or even a pre-determined random fitness value.

Steady-state selection technique: This technique is mainly involved in replacing individuals in each generation instead of selecting parents from the pool. More fit individuals replace
the least fit ones in the population pool and also retains a bigger part of the chromosome for the future generations.

Rank Selection technique: Unlike the roulette wheel selection technique where importance is given to the fitness factor of a chromosome, this technique assigns a numerical rank to each chromosome based on the fitness and selection is performed based on the rank instead of differences in fitness. In this technique the least fit individuals are also considered important in selection whereas in the roulette wheel selection technique the least fit individuals hardly stand a chance of selection which could affect the diversity and variation of solutions in the new generation.

2.2.3.3 Operators of genetic algorithm

An important third step in the process of genetic algorithm is to apply suitable operators to the encoded and selected solutions so as to produce new offspring. Crossover and Mutation are the two operators used in order to accomplish the creation of next generation.

Crossover or recombination operation: In this operation off springs are generated by combining features of both the parents’ solutions. Crossover can be performed by choosing a random crossover point and then swap corresponding segments of the parent’s chromosomes to create off springs. Based on the scope of the problem different crossover techniques can be used. Commonly used crossover operations are single point crossover and uniform crossover. In the single point crossover, a random crossover point is determined and then chromosomes swapped between the parents.

The figure below shows single point crossover: [21, p10]
In the uniform crossover method, a particular point is initially chosen and then either of the parents chromosomes are copied onto the off springs chromosome. The figure below shows uniform crossover [27]:

![Uniform crossover diagram](image)

**Figure 8: Uniform crossover technique**

Crossover point is chosen as the fifth chromosome, starting from the sixth chromosome the offspring's last three bits are changed to the parents last three bits in order to create variations.

In the tree encoding crossover technique, a crossover point is selected in both the parent tree chromosomes and parts of the tree from the crossover points are exchanged between each other in order to create new offspring's.

The figures below shows tree encoding crossover [28, p12, 13]:

![Tree encoding crossover diagram](image)

**Figure 9(a): Before tree encoding crossover technique**

Shown above are two parenting expressions; the left sub tree of the first parent and the right sub tree of the second parent are crossed over in order to create new off spring's.
Mutation Operation: Mutation in living organisms is the changes that happen to particular genes in DNA strands. In genetic algorithms, mutation is performed on particular bits of the solutions in order to create variations in the population pool. In this process the parents’ bits are copied and mutations performed on only the pre-determined bit locations. Crossover is termed as an exploitation operator whereas mutation is termed as exploration operator. In this operation new search areas can be created unlike with crossover. Mutation can also be performed on all sorts of encoding techniques.

![Figure 9(b): After tree encoding crossover technique](image)

The two offspring resulting from crossover are shown below.

![Figure 10: Binary encoding mutation](image)

The figure above clearly shows the binary encoding mutation operation on the fifth bit of the offspring's chromosome.
2.3 Flow chart of Genetic algorithm:

Start

Create search set or population

Assign fitness to each individual in the set

Select two fit individuals as parents

Crossover to produce children

Assign fitness to children

Crossover complete

Stop

Select a child

Apply mutation

Assign fitness to children

Mutation complete

Replace children with parents in the set

Done?

Yes

No

Figure 11: Flowchart of genetic algorithm
2.4 Strengths of GA

- Multidirectional approach – Most algorithms explore solutions in single directional approach and therefore if the solution is not optimal then all the work gets wasted. However, genetic algorithms have multiple solutions and explore in multi directional approach, therefore if any solution turns out to be non-optimal it can be eliminated from the pool of possible solutions.
- Genetic algorithms work on the fittest solutions from the pool of solutions and therefore find the most optimal one from the group of fittest solutions.
- In single directional approach any updates made to a portion of the system requires updates applied to the entire system whereas in genetic algorithm each change leads to greater improvements in fitness of solutions.
- Fast processing time in finding an optimal solution from the vast pool of solutions makes the genetic algorithm much stronger.
- Crossover feature of the genetic algorithm enables the transfer of genetics from parents to off springs in order to include only the strengths of both the parents. This is a very important feature as it helps in creating variations and diversity in the new generation.

2.5 Limitations of GA

- Languages used to represent the genetic algorithm’s solution set must be robust enough to withstand the changes in genetic information caused due to crossover and mutation.
- Fitness functions must be created and used appropriately in order to attain the most optimal solution in the least time.
- All other factors (size of the solution set, crossover points, and selection criteria) involved in the process of genetic algorithm must also be chosen appropriately in order to avoid any disruption in the process of finding the most optimal solution.
Chapter 3

System Study

3.1 Study of Existing system

The process of arranging resources effectively by satisfying necessary constraints is called timetabling. Every institution or organization must have timetabling implemented in one form or the other in order to schedule daily activities for effective resource utilization. The process of timetabling is time consuming and subjected to several revisions before an acceptable solution has been made.

A general school timetable consists of six study periods (each 55 minutes long) and two break periods (25 minutes each), starting from 8 am to 2:30 pm Sunday through Thursday. For grades 1 to 6 main subjects (English, maths, science, and social studies) are taught by the class teachers whereas activity subjects (ICT, PE, ART, Arabic) are taught by specialist teachers. Grades 7 -12 have specialist teachers for each subject. Timetable generation is currently done manually by the admin staffs that are provided with all the details of the teachers and the subjects they will handle for the entire academic year.

Several revisions of the school timetable are made throughout the academic year in order to accommodate any minor changes.

3.1.1 Advantages of existing system

- Timetable is generated after collaborating with several staff members before an optimal one is finalised.

3.1.2 Limitations of existing system

- Time consuming process requiring lots of paperwork.
- Reduced flexibility as making changes may not always be easy to implement.
- Several revisions of timetables required due to clashes between teachers allocations.
3.2 Proposed system

More than once in a year every academic institution deals with the most time and resource consuming process of timetabling. This process can be done automatically using the proposed system. In the proposed system timetables can be generated with the input data that is provided at the front end.

For the scheduling of a timetable the resources are identified as teachers, students and class rooms. The resources must be mapped effectively onto pre-determined time slots. The school timings, lesson duration, number of lessons, break durations, number of break periods, teachers’ details and associated subjects, student groups and their sizes, and all necessary constraints required for developing a timetable must be provided accurately to the system.

While creating a timetable there are some constraints that must be surely satisfied (hard constraints) and some that can be done if possible (soft constraints) in order to construct an optimal one.

Hard constraints

- A teacher can take only one class during a given time slot
- A group of students can attend only one subject during a given time slot
- There can be only one subject in a classroom at a given time slot
- All student groups must be occupied during all time slots
- No two consecutive timeslots must have the same subject
- A teacher must teach at least one subject in a day.
- No classes must be scheduled during the break periods.
- Teaching lessons must not be scheduled during the activities periods (Grades 1 – 6 on Tuesday lesson 4 and grades 7-12 on Tuesday lesson 6)
- Form time (class teachers’ time) must be allocated during assigned time slots.
Soft constraints

- A teacher should not be assigned for more than 2 lessons continuously.
- A group of students should not have two timeslots of a subject on the same day.
- No additional pastoral duties for teachers on full time table load.

The system will use genetic algorithm on the input and constraints list provided to construct an optimal timetable within minimal time thereby reducing the cost and turnaround involved in manually creating a timetable.

3.2.1 Advantages of proposed system

- Flexibility in implementing new changes to the timetable.
- Reduced time and staff effort in generating an optimal timetable.
- Avoids paperwork and enables easy updating of information even after a timetable is constructed.
- Overall timetabling process made lighter and easier to perform several revisions.

3.2.2 Limitations of proposed system

- Timetables generated must still be subjected to approval by the admin staff.
- Incorrect or inappropriate information may not provide complete timetables.
3.3 Functional requirements of the proposed system

The functional requirements of the proposed system are listed and described below:

- Login functionality with account management: The proposed system must provide functionalities to register a new user and to login with their accounts. The system must also verify the users' identities.
- Teacher profile creation and maintenance: The system must provide functionality to create a new teacher profile, view, modify and delete any profiles from the database.
- Generate school and teacher timetable: A School timetable must be generated for the whole school based on the different subjects, grades, sections and number of teaching hours for each subject stored in the database. Individual teacher timetables will be generated for teachers whose profiles are created.
- View and download the timetables generated: The timetables can be viewed for a particular grade and section or for a particular teacher. The timetables must be downloadable in excel sheet format and stored on to a computer drive.
- Cover lessons Generator: The system must be able to generate cover lessons for teachers who are absent on a particular day of the week. It must initially check for availability of teachers teaching the same subject, if unavailable then allocate a free teacher.
- Tests Scheduler: The system must be able to display the available dates for a class test within a specified from and to date range.
3.4 Non-functional requirements of the proposed system

The non-functional technical requirements of the system are listed below:

- SQL connections – too many database requests or actions performed may return errors. Therefore the connection variable value must be incremented to hold more connections.

3.5 Evaluation requirements of the proposed system

The proposed system has user interfaces and registered users can directly interact with the system and database.

The evaluation requirements of an end user can be summed up as:

- The testing of the functionalities described must be evaluated by the end user who will be teaching or non-teaching staff.
- The timetables generated must be downloadable separately for different grades and individual teachers. New timetables must override the previous ones.
- Databases must be evaluated by the nonteaching staff to verify the data stored.
- Cover lessons must be generated appropriately.
- Tests schedules must be displayed appropriately.
Chapter 4

System design and project plan

4.1 Tools and technologies

During the research report, the choice of languages made was PHP, MySQL and Matlab. An important aspect in the implementation of this project is that the tools and technologies used must be platform independent. A technical problem which arose while coding was the integration between Matlab and PHP files. It was then decided to use Java as the programming language as it was powerful enough to support the implementation of Genetic algorithm as well as can be used for developing efficient and user friendly forms.

The tools and technologies used in this project are listed and described below:

Database Management:

MySQL has been used for this project as it can be installed on all platforms such as Windows, Mac OS and Linux. Additionally, it also has a free license and therefore MySQL has been chosen for the database management.

Web services programming languages:

Different languages provide web services. Java is platform independent and most of the IDEs used to implement it has free licenses. It can also perform genetic algorithm and therefore it has been chosen as the programming language for web service. As Java is the programming language chosen, JavaServer Pages (JSP) is used to create dynamically generated web pages. Web browsers used for the testing purposes are Mozilla Firefox and Google Chrome.

Web Container:

There are a number of web containers with free licenses available as the programming language chosen is Java. The available web containers are GlassFish and Apache. Extra java libraries and configuration are required for Apache therefore GlassFish has been chosen as the web container.
Java IDEs:

In order to implement Java applications, a number of IDEs are available namely Eclipse and NetBeans which has free license. NetBeans has been familiar to me and therefore NetBeans has been chosen as the Java IDE for the project.

4.2 Database design and implementation

An ER (Entity relationship) diagram is created based on the functional requirements gathered. Attributes and data types of each entity is identified and used to implement the database. As part of database management each functional requirement and its entities has been further described below:

- Login functionality and account management: The system must provide functionality to allow a new user to register and verify the users login details. An entity Userinfo is used to store the users’ information and also verifies the users login details. There will be only one record for each user in Userinfo.
- Teacher profile creation and maintenance: The system must provide functionality to create a new teacher profile, view, modify and delete any profiles from the database. An entity teacherinfo is used to store the details of the teachers in the database. The details can be entered by the verified user. So a user may create, modify or delete one or more teachers’ details at a time. Therefore the relationship between the userinfo and teacherinfo is one-to-many.
- Generate school and teacher timetable: A School timetable must be generated for the whole school based on the different subjects, grades, sections and number of teaching hours for each subject stored in the database. Individual teacher timetables will be generated for teachers whose profiles are created in the teacherinfo entity. Entities such as subjects, grades, sections and subject_grade contain details about the different subjects, grades, sections and number of teaching hours required for each subject in a school.

When teacher information is added on to teacherinfo, it must also populate values into teacher_grades, teacher_subjects and teacher_sections which contain information about
the different grades, subjects and sections respectively for a specific teacher. Therefore userinfo has one-to-many relationship with teacher_grades, teacher_subjects and teacher_sections.

When a school timetable is generated, based on the values in teacher_grades, teacher_subjects and teacher_sections it creates entities timetable_(each section name) for each section mentioned in the sections entity. For example: Timetable_W entity contains the timetable generated for all the grades of the school but of section W.

The project is developed for grades 1 to 12 and five sections for each grade. Subjects used as part of this project are Literature, Language, Maths, Science, Social Studies, Humanities, Arabic, Islamic, ICT, ART, PE, English.

When a teacher timetable is generated, based on the values in teacher_grades, teacher_subjects, teacher_sections and teacherinfo it creates entities timetable_(each teacher name) for each teacher in the teacherinfo entity. For example: Timetable_Sarah entity contains the timetable generated for a teaching staff named Sarah. Therefore Timetable_Sarah entity has a many-to-many relationship with teacher_grades, teacher_subjects, teacher_sections and teacherinfo.

- Cover lessons generator – When a teacher is absent the system must automatically cover the lessons with available teachers. If a maths teacher is absent the system must first check for an available maths teacher, if no maths teachers are available then the system must check for any other free non subject teachers. Teacherinfo, timetable_(teacher name), subject entities are used for the implementation of this module.
- Test scheduler – when a test has to be scheduled the system must display the dates available for the test. Entities used in this functionality are teacherinfo, subjects, timetable_(teacher name) for displaying the data.
4.2.1 Entity relationship diagram

Figure 12: ER Diagram of the proposed system
4.3 UML representation of the proposed system

Unified Modelling Language (UML) helps to describe the proposed system in terms of objects and their relationships. A use case diagram has been used to represent the interaction between the admin staff and the proposed system.

![Use case diagram of the proposed system](image)

Figure 13: Use case diagram of the proposed system
4.4 Project plan

A Gantt chart is used to indicate the plan of project implementation which includes the start date, duration in weeks and the tasks involved in the project implementation. Microsoft Excel is used to create the Gantt chart.

Table 1: Gantt chart indicating the project plans.
4.5 Professional, Legal and Ethical issues

4.5.1 Professional Issues

Professional competence has been taken into consideration as part of professional issues. Professional competence could be a possible issue in this section as I know the programming languages PHP and MySQL. However MATLAB which is to be used for implementing the genetic algorithm was not taught as part of this course. The development of the timetable generator requires the knowledge of MATLAB and it will be learned for the project.

4.5.2 Legal Issues

Data protection law is a very important aspect to be discussed under this section. Utmost care has to be taken against infringement of any schools personal data which might lead to plagiarism. Details about teachers, student groups and subjects must be kept generic in order to avoid this issue.

4.5.3 Ethical issues

The only ethical issue that can be discussed under this section is whether this system will replace a staff member. This system works only with appropriate data and is not intended at replacing any member of staff. The timetables generated by the module will still require revisions and approvals by authorised staff members. Therefore it will still remain as a collaborated work instead of replacing any staff members.

4.5.4 Summary

The professional, legal and ethical issues can be summarised as shown below:

- On the professional front, gaining knowledge of the required software for implementing genetic algorithm.
- On the legal front, care must be taken against infringement of data.
- On the ethical front, this system is not intended to replace any member but only simplify the process of timetabling.
Chapter 5

Development and Testing

5.1 Stages of development

This chapter details about the different stages of development involved in the project.

5.1.1 Module 1: Teachers information module

The development of this module includes the implementation of new user registration, Login functionality and Teacher profile creation and maintenance. The new user registration functionality requires new users to register themselves so that the data gets stored on the database. The login data from the database allows further login and working on the system. The user must login with appropriate login details which will be verified against the data in the database. After successful login the username must be displayed on top along with the welcome message. The user must be able to enter teachers' information into the system. The details entered must be viewable, able to modify and delete from the system.

For new user registration: Details of the user to be entered on the form is Name, Phone number, Email id, Address, User Name and password. These details must be stored on to Userinfo entity and then verified each time a username and password is entered.

For teacher information: Details of the teacher to be entered on the form is Name, Age, Nationality, Contact No:, Address, Passport No:, Passport Expiry date, Visa Type, School accommodation, Subjects taught, Grades, Sections, Form Tutor, Grade. These details must get stored on to teacherinfo, teacher_subjects, teacher_grades, teacher_sections entities. The system also provides additional options to view, modify or delete a teacher profile.
The use case diagram for the module 1 can be seen in figure 14 which indicates the different actions that can be performed by a user.

![Use case diagram of module 1](image)

**Figure 14: Use case diagram of module 1**

JavaServer pages have been used to create forms in the front end and accept data entered by the user. The data is stored on to appropriate mySql tables.

### 5.1.2 Module 2: Timetable generator

The developments of this module include the implementation of school and teacher timetable generation and download the timetables generated. For generating school timetable, the system must obtain the data stored on the entities *subjects, grades, sections* and *subject_grades*. These entities will contain details about the subjects taught in the school, different grades in the school (grade 1 to grade 12), different sections of each grade (12A, 12W, etc.) The system must delete existing timetables and create new ones each time the generate timetable option is selected. The database must have entities named as *timetable_(each section)* when new timetables are generated for the school.

The system must generate timetables for each teacher in the database. In order to generate a teacher timetable the system must use the data stored on the entities *teacher_subjects*,...
teacher_grades, teacher_sections and teacherinfo. The timetables generated for each teacher must match with the school timetable. When a timetable is generated for each teacher the system must create an entity named timetable_(teacher name) which makes it easier to display on the web page.

The use case diagrams for the required functionalities can be seen in figure 15.

![Use case diagram of module 2](image)

**Figure 15: Use case diagram of module 2**

The system must be able to provide an option to download the timetables generated for the school and teacher. The timetables must be downloadable in excel sheet format with the name of the grade and the section or the name of the teacher. If the user wishes to download the timetable generated for grade 9 section W, the option for download must be selected and it will provide the user choices for opening or saving the excel file. If the user chooses to save the file then it must be named Timetable_gradessection and downloaded on to Drive D in a folder named Timetables which will contain all the timetables.
5.1.3 **Module 3: Cover Lesson and Tests Scheduler**

The developments of this module include the implementation of cover lesson generator and tests scheduler. The system aims at covering an absent teacher’s classes to another teacher who is free during that lesson. The system must first check for a free subject teacher, if unavailable then the system must check for other subject teachers and allot them to cover the lesson.

The use case diagrams for the required functionalities can be seen in figure 16.

*Figure 16: Use case diagram of module 3*

Details entered on the form for the implementation of the cover lessons generator will be name of the absent teacher and the day of the week. Entities used will be `timetable_(name of the absent teacher)`, `teacherinfo`, `subjects and teacher_subjects` to fetch details from the database. The system must display the name of the subject along with the grade and name of the cover teacher. There must not be any database update in this module.

For the implementation of the Tests Scheduler the system must obtain the following details form the user namely teacher name, term, test no.; section, grade, subject, From date and To date. The entities used
5.2 Pseudo code of the Genetic Algorithm in Java

```
// start with an initial time
    t := 0;

// initialize a usually random population of individuals
    initpopulation P (t);

// evaluate fitness of all initial individuals of population
    evaluate P (t);

// test for termination criterion (time, fitness, etc.)
    while not done do
        // increase the time counter
            t := t + 1;

        // select a sub-population for offspring production
            P' := selectparents P (t);

        // recombine the "genes" of selected parents
            recombine P' (t);

        // perturb the mated population stochastically
            mutate P' (t);

        // evaluate its new fitness
            evaluate P' (t);

        // select the survivors from actual fitness
            P := survive P,P' (t);
    od
end GA.
```

The working of Genetic algorithm is involved while creating timetables for the school and individual teachers. Five java classes are created for the implementation of Genetic algorithm namely Population.java, FitnessCalc.java, Algorithm.java, Individual.java and GA.java.
5.3 UML Class diagram for the genetic algorithm implementation in Java

Figure 17: UML class diagram of genetic algorithm implementation
5.4 Code extracts from the classes for the genetic algorithm implementation

The figure 18 given below shows the code extracts from Population.java in creating a population.

```java
// Create a population
public Population(int populationSize, boolean initialise) {
    individuals = new Individual[populationSize];
    // Initialise population
    if (initialise) {
        // Loop and create individuals
        for (int i = 0; i < size(); i++) {
            Individual newIndividual = new Individual();
            newIndividual.generateIndividual();
            saveIndividual(i, newIndividual);
        }
    }
}
```

**Figure 18**: Creating a population for genetic algorithm

The figure 19 given below shows the code extracts from FitnessCalc.java for setting a candidate solution against which the fitness is calculated.

```java
// This method sets the candidate solution with string of 0s and 1s
public static void setSolution(String newSolution) {
    solution = new byte[newSolution.length()];
    // Loop through each character of the string and save it into byte array
    for (int i = 0; i < newSolution.length(); i++) {
        String character = newSolution.substring(i, i + 1);
        if (character.contains("0") || character.contains("1")) {
            solution[i] = Byte.parseByte(character);
        } else {
            solution[i] = 0;
        }
    }
}
```

**Figure 19**: Fitness Calculation for genetic algorithm
The figure 20 given below shows the code extracts from FitnessCalc.java for calculating fitness of an individual by comparing it to the candidate solutions

```java
// Calculate individuals fitness by comparing it to the candidate solution
static int getFitness(Individual individual) {
    int fitness = 0;
    // Loop through the individual genes and compare them to the candidates
    for (int i = 0; i < individual.size() && i < solution.length; i++) {
        if (individual.getGene(i) == solution[i]) {
            fitness++;
        }
    }
    return fitness;
}
```

**Figure 20: Fitness Calculation comparison for genetic algorithm**

The figure 21 given below shows the code extracts from Algorithm.java for selecting individuals for crossover using tournament selection

```java
// Select individuals for crossover
private static Individual tournamentSelection(Population pop) {
    // Create a tournament population
    Population tournament = new Population(tournamentSize, false);
    // For each place in the tournament get a random individual
    for (int i = 0; i < tournamentSize; i++) {
        int randomId = (int) (Math.random() * pop.size());
        tournament.saveIndividual(i, pop.getIndividual(randomId));
    }
    // Get the fittest
    Individual fittest = tournament.getFittest();
    return fittest;
}
```

**Figure 21: Tournament selection for crossover**
The figure 22 given below shows the Code extracts from Algorithm.java for performing crossover and mutation of individuals

```java
// Crossover individuals
private static Individual crossover(Individual indiv1, Individual indiv2) {
    Individual newSol = new Individual();
    // Loop through genes
    for (int i = 0; i < indiv1.size(); i++) {
        // Crossover
        if (Math.random() <= uniformRate) {
            newSol.setGene(i, indiv1.getGene(i));
        } else {
            newSol.setGene(i, indiv2.getGene(i));
        }
    }
    return newSol;
}

// Mutate an individual
private static void mutate(Individual indiv) {
    // Loop through genes
    for (int i = 0; i < indiv.size(); i++) {
        if (Math.random() <= mutationRate) {
            // Create random gene
            byte gene = (byte) Math.round(Math.random());
            indiv.setGene(i, gene);
        }
    }
}
```

**Figure 22:** Crossover and mutation functions for genetic algorithm
The figure 23 given below shows the Code extracts from Algorithm.java for evolving a new population with crossover and mutation operations performed.

```java
// Evolve a population
public static Population evolvePopulation(Population pop) {
    Population newPopulation = new Population(pop.size(), false);
    // Keep our best individual
    if (elitism) {
        newPopulation.saveIndividual(0, pop.getFittest());
    }
    // Crossover population
    int elitismOffset;
    if (elitism) {
        elitismOffset = 1;
    } else {
        elitismOffset = 0;
    }
    // Loop over the population size and create new individuals with crossover
    for (int i = elitismOffset; i < pop.size(); i++) {
        Individual indiv1 = tournamentSelection(pop);
        Individual indiv2 = tournamentSelection(pop);
        Individual newIndiv = crossover(indiv1, indiv2);
        newPopulation.saveIndividual(i, newIndiv);
    }
    // Mutate population
    for (int i = elitismOffset; i < newPopulation.size(); i++) {
        mutate(newPopulation.getIndividual(i));
    }
    return newPopulation;
}
```

**Figure 23:** Evolving new population after genetic operations
public class GA {
  public static void main(String[] args) {
    // Set a candidate solution
    FitnessCalc.setSolution("111100000000000000000000000000000000000000000000000000000000111");
    // Create an initial population
    Population myPop = new Population(35, true);
    // Evolve our population until we reach an optimum solution
    int generationCount = 0;
    while (myPop.getFittest().getFitness() < FitnessCalc.getMaxFitness())
    {
      generationCount++;
      System.out.println("Generation: " + generationCount + " Fittest: " +
      myPop.getFittest().getFitness());
      myPop = Algorithm.evolvePopulation(myPop);
    }
    System.out.println("Solution found!");
    System.out.println("Generation: " + generationCount);
    System.out.println("Genes:");
    System.out.println(myPop.getFittest());
  }
}

Figure 24: Main function for genetic algorithm
5.5 Testing of Web Pages and databases

Testing is one of the most important steps in any project implementation as it helps to identify design errors. During the development of this system all of the implemented functionalities have been tested. The testing methodology focuses on testing the functionalities specified in each module.

The testing process comprises of the following steps:

- Objectives to be achieved in the testing of each module
- Software requirements to conduct the testing process
- Different test scenarios for each module and its results

5.5.1 Testing objectives

Objectives to be achieved in the testing process are described as given below:

- Verification of the system functionality in each module
- Working of the genetic algorithm
- Validation of results against the data stored in the database.

5.5.2 Software requirements for testing

While testing the modules the software requirements used were:

- Java IDE NetBeans
- Xampp web server
- Web browser (Google Chrome, Mozilla FireFox or Internet Explorer)
5.5.3 Individual modules test scenarios and results

5.5.3.1 Module 1: Teachers information module

The forms on the webpage must take information from the admin staff and insert into the assigned database tables.

**Table 2**: Test conditions for Module 1:

<table>
<thead>
<tr>
<th>Test criteria</th>
<th>Pass</th>
</tr>
</thead>
<tbody>
<tr>
<td>Webpage loading properly</td>
<td>Pass (Figure 25)</td>
</tr>
<tr>
<td>Links working correctly</td>
<td>Pass (Figure 26)</td>
</tr>
<tr>
<td>New user registration</td>
<td>Pass (Figure 27)</td>
</tr>
<tr>
<td>Data inserted into the appropriate database tables for user</td>
<td>Pass (Figure 28)</td>
</tr>
<tr>
<td>Login for registered users</td>
<td>Pass (Figure 29)</td>
</tr>
<tr>
<td>Access denied for incorrect user name password combinations</td>
<td>Pass (Figure 30)</td>
</tr>
<tr>
<td>Create a new teacher profile</td>
<td>Pass (Figure 32)</td>
</tr>
<tr>
<td>Forms validations and messages display</td>
<td>Pass (Figure 34)</td>
</tr>
<tr>
<td>Data inserted into the appropriate database tables</td>
<td>Pass (Figure 33)</td>
</tr>
<tr>
<td>View/modify/delete a teacher profile</td>
<td>Pass (Figure 35)</td>
</tr>
<tr>
<td>Data can be retrieved appropriately from the database</td>
<td>Pass (Figure 36)</td>
</tr>
</tbody>
</table>

This module contains three functionalities, new user registration, user login and teacher profile creation & modification. Figures 25 and 26 indicate the properly loading of webpages.

*Figure 25: Application build*
Figures 27 and 28 demonstrates a new user registration and its database results.
Figures 29 and 30 demonstrate the login page messages for correct and incorrect login operations.

**Figure 29:** Login enabled for registered user

**Figure 30:** Access denied - Incorrect username or password
Figures 31 and 32 demonstrates the creation of a teacher profile.

**Figure 31: Teacher profile**

**Figure 32: Creating a new Teacher profile**
Figure 33 displays the database results for the new profile created.

![Database results verified for the new teacher profile created](image)

**Figure 33: Database results verified for the new teacher profile created**

Figure 34 displays the new teachers' details on the webpage along with success message.

![New profile and message displayed](image)

**Figure 34: New profile and message displayed**
Figure 35 display the message for the deletion of a record. Teacher profile Mathew deleted from the web page and MySQL table. The database results are displayed on figure 36.

Figure 35: Existing profile deleted and message displayed

Figure 36: database results verified for profile deletion
5.5.3.2 Module 2: Timetable generator

Timetables generated can be either for individual teachers or for the whole school.

**Table 3:** Test conditions of module 2:

<table>
<thead>
<tr>
<th>Test criteria</th>
<th>Pass</th>
<th>Fail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Webpage loading properly</td>
<td>Pass (Figure 37)</td>
<td></td>
</tr>
<tr>
<td>Timetables generated appropriately for school and teachers</td>
<td>Pass (Figure 38)</td>
<td></td>
</tr>
<tr>
<td>Results of genetic algorithm displayed</td>
<td>Pass (Figure 39)</td>
<td></td>
</tr>
<tr>
<td>Database result for new teacher timetable generation</td>
<td>Pass (Figure 40)</td>
<td></td>
</tr>
<tr>
<td>Viewing of generated timetables</td>
<td>Pass (Figure 41)</td>
<td></td>
</tr>
<tr>
<td>Timetable download option working properly with correct naming</td>
<td>Pass (Figure 46)</td>
<td></td>
</tr>
<tr>
<td>A teacher must be assigned only one subject at a time slot</td>
<td>Pass (Figure 45)</td>
<td></td>
</tr>
<tr>
<td>A student group is assigned only one subject at a time slot</td>
<td>Pass (Figure 45)</td>
<td></td>
</tr>
<tr>
<td>A classroom must occupy only one student group at a time slot</td>
<td>Pass (Figure 45)</td>
<td></td>
</tr>
<tr>
<td>All student groups are occupied during all time slots</td>
<td>Pass (Figure 45)</td>
<td></td>
</tr>
<tr>
<td>No classes must be scheduled during break periods</td>
<td>Pass (Figure 45)</td>
<td></td>
</tr>
<tr>
<td>No teaching periods must be scheduled during activities/PE periods</td>
<td>Pass (Figure 45)</td>
<td></td>
</tr>
<tr>
<td>Appropriate messages displayed</td>
<td>Pass (Figure 38)</td>
<td></td>
</tr>
</tbody>
</table>

This module contains three functionalities, new timetables generated for the school, individual teacher timetable generation, display and download of the timetables.

Appropriate messages are displayed at the bottom of each web page to indicate the status of each action performed. Break periods have been assigned at the timeslot of 12 to 1 PM for all grades. The figures given below indicates the testing of module 2.
Figures 37 demonstrates the web page loading properly for generating timetables.

**Figure 37: timetable generation module**

**Figure 38: Success message displayed**

Figure 39 displays the results of a successful genetic algorithm implementation.

**Figure 39: Display results of genetic algorithm**
Figure 40 shows a new table timetable_dijo created for the new profile

![Database results verified for new teacher timetable generated](image)

Figure 40: Database results verified for new teacher timetable generated

Figure 41 shows the web page enabling the viewing of generated school or individual timetables

![Webpage for viewing the generated timetables](image)

Figure 41: Webpage for viewing the generated timetables
Figure 42 shows the timetable of grade 3 section I displayed

![Webpage for viewing the school timetables](image)

**Figure 42: Webpage for viewing the school timetables**

Figures 43 and 44 shows the download option for the timetable of grade 3I

![School Timetable Download option](image)

**Figure 43: School Timetable Download option**

![Folder containing the downloaded timetables](image)

**Figure 44: Folder containing the downloaded timetables**
Figure 45 shows the download option for the timetable of a teacher.

Figure 45: individual teacher timetable display

Figure 46 shows the Download option for the timetable of teaching staff Sarah.

Figure 46: Teacher timetable download option

Figure 47: Folder containing the downloaded teacher timetables
5.5.3.3 Module 3: Cover lessons and Tests Scheduler

This module contains two functionalities, covering the lessons of an absent teacher and indicates available test dates and days for a teacher within the specified date range. Appropriate messages are displayed at the bottom of each web page to indicate the status of each action performed.

Table 4: Test conditions of module 3:

<table>
<thead>
<tr>
<th>Test criteria</th>
<th>Pass</th>
<th>Fail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Webpage loading properly</td>
<td>Pass (figure 48)</td>
<td></td>
</tr>
<tr>
<td>Allocate subject teacher if available</td>
<td>Pass (figure 48)</td>
<td></td>
</tr>
<tr>
<td>If subject teacher is unavailable allocate non-subject teacher</td>
<td>Pass (figure 50)</td>
<td></td>
</tr>
<tr>
<td>Allocated teacher must be free for the cover lesson</td>
<td>Pass (figure 50)</td>
<td></td>
</tr>
<tr>
<td>Tests scheduler page loads properly</td>
<td>Pass (figure 51)</td>
<td></td>
</tr>
<tr>
<td>Displays the available days of the week for a test</td>
<td>Pass (figure 51)</td>
<td></td>
</tr>
</tbody>
</table>

In the figures 48 and 49 given below the system checks to see if another Maths teacher is available to cover the lesson, if nobody is available then it checks for any other subject teacher.

Figure 48 displays the loading of the cover lesson page properly.
The timetable given below indicates the availability of a maths subject teacher and being utilised for a cover lesson.

![Teacher Timetable](image)

*Figure 49: Teacher timetable to check availability*

Whereas in the figure 50 given below, no other Literature teachers are available to cover the lessons, so the lessons are covered by other subject teachers. Literature teachers are unavailable to cover the lessons and therefore other subject teachers have been utilised for the cover lessons by the generator.
Figure 50: Lessons covered by non-subject teachers

Figure 51 shows the tests scheduler webpage loads properly and displays the days of a week during which a test could be conducted based on the school timetables.

Figure 51: Tests scheduler results
Chapter 6

Evaluation

This chapter details about the evaluation plan of the project. The proposed system has user interfaces and registered users can directly interact with the system and database. The evaluation requirements state that an end user must evaluate the system based on the following points:

- The functionalities provided must be evaluated by the end user who will be teaching or non-teaching staff.
- The timetables generated must be downloadable separately for different grades and individual teachers. New timetables must override the previous ones.
- Databases must be evaluated by the end users to verify the results stored.

Therefore this chapter describes the evaluation and its results in detail.

6.1 Evaluation plan

The evaluation plan is to conduct interviews with school teaching and non-teaching staff. In these interviews the functionalities of the system must be described in detail. The timetables generated by the system must be evaluated in detail by verifying the data stored by the system in the database. During the interview a general overview of the system must be provided based on the functionalities that can be evaluated by the target groups. School teaching staff must be described about the timetables generation and its usability whereas a non-teaching staff must be described about the entire system functionality.

During the evaluation phase of the project seven interviews were conducted with two groups of teaching and non-teaching staff. The details of the interviews are described in detail.
6.1.1 Interviews with non-teaching staff

Four interviews were conducted with the administration staff of schools in India and UAE. They were provided with an oral description of the entire system's functionality and how they could evaluate its working and effectiveness. Members of administration staff were elaborated on the following points:

- New user registration
- Login functionality
- Teacher profile creation
- Timetable generation, viewing and downloading
- Cover lessons generator
- Tests scheduler

Additionally, they were also briefed about the types of data stored in the database. During the interview they were asked opinions about the system and its functionalities.

Positive feedback points from the evaluation by non-teaching staff:

- Majority of the interviewees evaluated timetable generation to be quick and efficient.
- They liked the usage of genetic algorithm as each time a new timetable was generated and it would be different from the previous one.
- They stressed on the user friendly forms, messages displayed and the downloadable timetables.
- Majority of them liked the absent teachers' cover lessons generated from the timetables for individual teachers.

Negative feedback points from the evaluation by non-teaching staff:

- Some of the interviewees suggested that the subjects taught, grades and sections of the school must be included as part of another form for the user to input instead of the current approach where those details are already stored into database tables.
• They preferred the test scheduler module to be able to automatically reserve available test dates instead of the current approach where the test days are only displayed.

6.1.2 Interviews with teaching staff

Three interviews were conducted with the teaching staff of schools in India and UAE. They were provided with an oral description of the systems’ functionality and how they could evaluate its working and effectiveness. Members of teaching staff were elaborated on the following points:

• Timetable generation, viewing and downloading
• Cover lessons generator
• Tests scheduler

Members of teaching staff would not be intended regular users of the system and therefore they were not briefed about the new user registration, login functionality and teacher profile creation. During the interview they were asked opinions about the system and its functionalities.

Positive feedback points from the evaluation by teaching staff:

• Majority of the interviewees were impressed with the timetables generated. They liked the option to download timetables according to grades and individual teachers.
• They also positively commented on different timetables created each time the generate button was clicked.
• Most of them were impressed with the cover lessons generator which would automatically suggest the names of available subject teachers to cover a lesson.

Negative feedback points from the evaluation by teaching staff:

• Some of them were not impressed with the test scheduler module to just display the days of the week for a test instead they suggested that the module must be able to block available test dates and to maintain a single excel sheet containing the test schedules of a particular grade or whole school.
• They suggested that in the timetables displayed the grades and sections must also be visible instead of only the subjects that a particular teacher teaches.

6.2 Evaluation discussion

Based on the results of the interviews conducted, it is seen that the system fulfils its aim and objectives in terms of the functionalities and the data stored in the database. Additionally the forms were evaluated as user friendly and messages displayed provided the user with the appropriate information after each button was clicked.

Some of the interviewees could not understand the concept and need for genetic algorithm in generating a timetable. Although it was observed by them that different timetables were getting generated each time without updating any data stored in the database. The interviewees from the administration staff agreed that there was no expert knowledge required in creating a manual timetable but it was done based on trial and error and required a lot of working hours. They also confessed that any time a change was required in the existing manually created timetable it normally took at least 1-2 working days for a new timetable to come into effect. It was also reported that extra resources would be utilised for creating individual teacher timetables. Therefore the generation of timetables by the developed system was evaluated positively by the administration staff. Downloading timetables according to grades and sections or individual teachers was also evaluated positively.

As my own evaluation of the developed system, there are positive and negative aspects. A negative aspect of this system is the tests scheduler which does not enable the actual reservation of certain dates for conducting tests. Another negative aspect is the database connectivity which raises an error if too many SQL connections are requested at the same time. This is due to all available connections being in use by clients.

One positive aspect is the generation of timetables within less than a minute for all the grades of a school. The objective of this project was to use genetic algorithm in the development of the timetable generator and to construct the most optimal timetable with little or no redundancy. Therefore the developed web application meets its aims and
objectives and can be used by schools to generate timetables with slight changes made to the subjects taught, grades and sections.

Another positive aspect is that the school timetable generation has no dependency on the data entered through any front-end forms. It only depends on the data stored in the MySql tables whereas the teacher timetable generation depends on the number of teacher profiles created. Teacher timetables will be generated only for the teachers’ whose data are stored on the tables. Cover lesson generator and tests scheduler also depends on the teacher profiles created. These modules take information from the teacher profiles and the teacher timetables generated.
Chapter 7

Summary

This chapter details about the achievements, limitations and future recommendation of this project.

7.1 Achievements

All of the requirements mentioned in the initial chapters of this report have been achieved. Therefore the system developed has the following functionalities:

- Enables a new user to register in to the system
- Enables login after verifying the user details from the database
- Denies login access to unregistered users
- Create, modify or delete teacher profiles
- Teacher profiles created or modified are stored in the database, deleted profiles are removed from the database.
- School timetables are generated independent of the data entered by the user
- Teacher timetables are generated based on the teacher profiles created
- Cover lessons generator which checks for the subject teachers' availability to cover an absent teachers' lessons. If subject teacher is unavailable then non subject teachers' availability is checked.
- Tests scheduler which provides the day of the week available to conduct the test for a specific grade

However during the testing of this project several school timetables were created and fifteen teacher profiles were created. Some of the teacher profiles were modified and some deleted. Teacher timetables were generated only for the profiles available in the database at the time of generation.
7.2 Limitations

There are a few limitations in this project. Initially a valuable amount of time was dedicated towards learning the technologies chosen. Matlab (Matrix laboratory) was chosen as the programming language to implement genetic algorithm. However the language had to be learnt as it was not part of any course. Secondly due to technical problems of integration between PHP and Matlab files, the choice of programming languages had to be reviewed again. Implementing genetic algorithm in Java was a challenge as it had to be completely programmed before testing whereas in Matlab it could be tested in parts using the genetic algorithm toolbox.

As a result of the time spent on learning and programming on these different languages the project has been affected negatively. Therefore the soft constraints which are of low priority have not been achieved. The unachieved requirements are the below mentioned soft constraints:

- A teacher should not be assigned for more than 2 lessons continuously.
- A group of students should not have two timeslots of a subject on the same day.
- No pastoral duties for teachers on full time table load.

Tests scheduler module also could not be completely implemented as it was initially intended to make the user able to schedule only two tests for a day and not more than three tests in a week.

7.3 Future recommendations

It has been noticed that a number of improvements could be made to this system. Instead of providing schools’ information such as subjects taught, grades and sections in the database directly it could be improved by developing a particular form which takes these values from a user. The system developed as part of the project can be used for schools which teach a specific set of subjects, grades from 1 to 12 and 5 sections each. Whenever there is any change made to the sections or subjects the database has to be edited directly. Instead of editing the database directly it is a better option to make this option available through a form.
Another aspect is the tests scheduler as it currently only displays the available days of a week for a section of a grade for a test. This could be improved by making it possible to display the dates and also giving the user an option to reserve certain dates for a test. The dates reserved must then get populated onto an excel sheet for each grade.

7.4 Conclusion

This report details the development of a web application for generating timetables aimed at reducing time and resources and producing optimum timetables with little or no redundancy. The idea of using genetic algorithm for generating timetables was derived from a review of relevant literature work where research studies prove that genetic algorithm is ideal for solving complex problems.

The development methodology used in implementing this project was iterative incremental development. Evaluation clearly indicates that the developed application meets the aims of this project which is to develop optimal school timetables within very less time. However, evaluation also indicates scope for improvement in the developed application.

To conclude, I have understood the concept of genetic algorithm and learned a number of tools and languages as part of this project implementation. I also faced and overcame a number of challenges such as the limitations of certain languages. This project has broadened my knowledge, understanding and programming skills in this subject.
References & Bibliography

References


[10] Tripathy A, School timetabling – A case in large binary integer linear programming, 1473 – 1489


[23] Lawrence Davis, Handbook of genetic algorithms


[27] Adam Marczyk, Genetic algorithms and Evolutionary computations, 2004

[28] John R Koza, Genetic programming: a paradigm for genetically breeding populations of computer programs to solve problems, 1990, Computer Science department, University of Stanford

Bibliography


[29] A genetic algorithm for a university weekly courses timetabling problem Enzhe Yu and Ki-Seok Sung, Received 10 July 1999; received in revised form 11 February 2002; accepted 4 April 2002


[31] The nature of code by Daniel Shiffman, Ch. 9 - the evolution of code


[37] Christopher Head, Sami Shaban, A heuristic approach to simultaneous course/student timetabling, Computers and Operations research 2007, p 919-933

[38] Danial Qaurooni, Mohammad-R. Akbarzadeh-T, Course Timetabling using evolutionary operators, Applied Soft Computing