FACULTY OF COMPUTER SCIENCE
AND INFORMATION TECHNOLOGY
UNIVERSITY MALAYA

KU CHIN SOON
WEK030084

Automatic Timetabling for Malaysian Secondary School Using Genetic Algorithm

Supervisor : Mrs. Siti Soraya binti Abdul Rahman
Moderators : Prof. Madya Dr. Roziati Zainuddin
Prof. Madya Dr. Mohd. Sapiyan Baba
Major : Artificial Intelligence

Acknowledgements

First of all, we will like to thank Pn. Siti Soraya for giving us a chance to accomplish this project. She has given me a lot of guidance and advice during the project. When I facing some problems, she patient to help me solve the problem together with me.

Thanks to my course mates and friends, they always had given me some help while I am facing some difficulty problem. Without their help, this project wouldn't be finish on time. I am very appreciate their help and given me new knowledge which I haven't heard before.

ABSTRACT

Solving timetable problem using genetic algorithms was the recent approach applies by the researchers. This project is schedule the Malaysian secondary school timetable using genetic algorithm method. The system developed manages to schedule school timetable in all forms.

This document contains eight chapters which is introduction, literature review, methodology, system analysis, system design, system implementation, and system testing and system evaluation. Each of the chapter was explain details of the development process for the Automatic Timetabling MSSUGA system.

Table of Content

List of Figure

List of Table

Chapter 1: Introduction

1.1 Introduction

This document is proposed to developing a new timetabling tool for Malaysian secondary school using genetic algorithm. The following section contain the overview of timetable and genetic algorithm, project objective and aim, problem statement, proposed solution, project scope, target user, expected outcome, and summary.

1.1.1 Timetable

Timetable vary use in daily life nowadays in educational (school timetable, university timetable, exam timetable), public transport services (train travel timetable, airlines timetable), organization (staff shifting timetable) and other areas. Each of those timetable have own pattern or structure to scheduling. A timetable is an organized list or schedule that performs in tabular form. It does arrange detail about activity, time to having the activity, and where having the activity.

1.1.2 School Timetable

For school timetable, it consists of three set of timetables which is student timetable, teacher timetable, and classroom (include labs) timetable. Three of those timetables have same structure as show as figure 1.1 which is sample of structure school timetable. The timetable is schedule a week activity in a specified time or period for specified task in needed location.

Day \ Time	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
Monday					TB					
Tuesday					TB					
wednesday					TB					
Thusday					TB					
Friday					TB					

TB : Time Break : Subject & (Teacher / Classroom) P_i : Period

Figure 1.1: *Sample of structure school timetable*

1.1.3 Genetic Algorithm (GA)

Genetic Algorithm was developed by John Holland in 1970's [Holland, 1975] [Rooij, A.J.C.V., et al., 1998]. They are base on a Darwinian-type survival of the fittest strategy, whereby potential solutions to a problem compete and mute with each other in order to produce increasingly stronger individuals [Rooij, A.J.C.V., et al., 1998]. The idea of genetic algorithm is to create an initial a population of individual's. Each of those individual's representing a possible timetable. The population must fulfill the requirement of the constraints which is hard constraints, soft constraints, and fixed constraints. Fixed constraints and hard constraint is important to produce free conflict population.

There are two fitness function apply in genetic algorithm whish is hard fitness and soft fitness. Fitness function is a measurement of the weight or score for a chromosome. Hard fitness determines hard constraints rate and soft fitness determines soft constraints. The stronger individual will be select to do the evolution depend on the fitness rate given to the individuals.

The strong survive individuals will generate a new population using genetic operators. Those processes will repeat until a new free conflict population was generated. The genetic operators are crossover operator, mutation operator and,

selection operator. The flow chart of standard genetic algorithm is shown in figure 1.2.

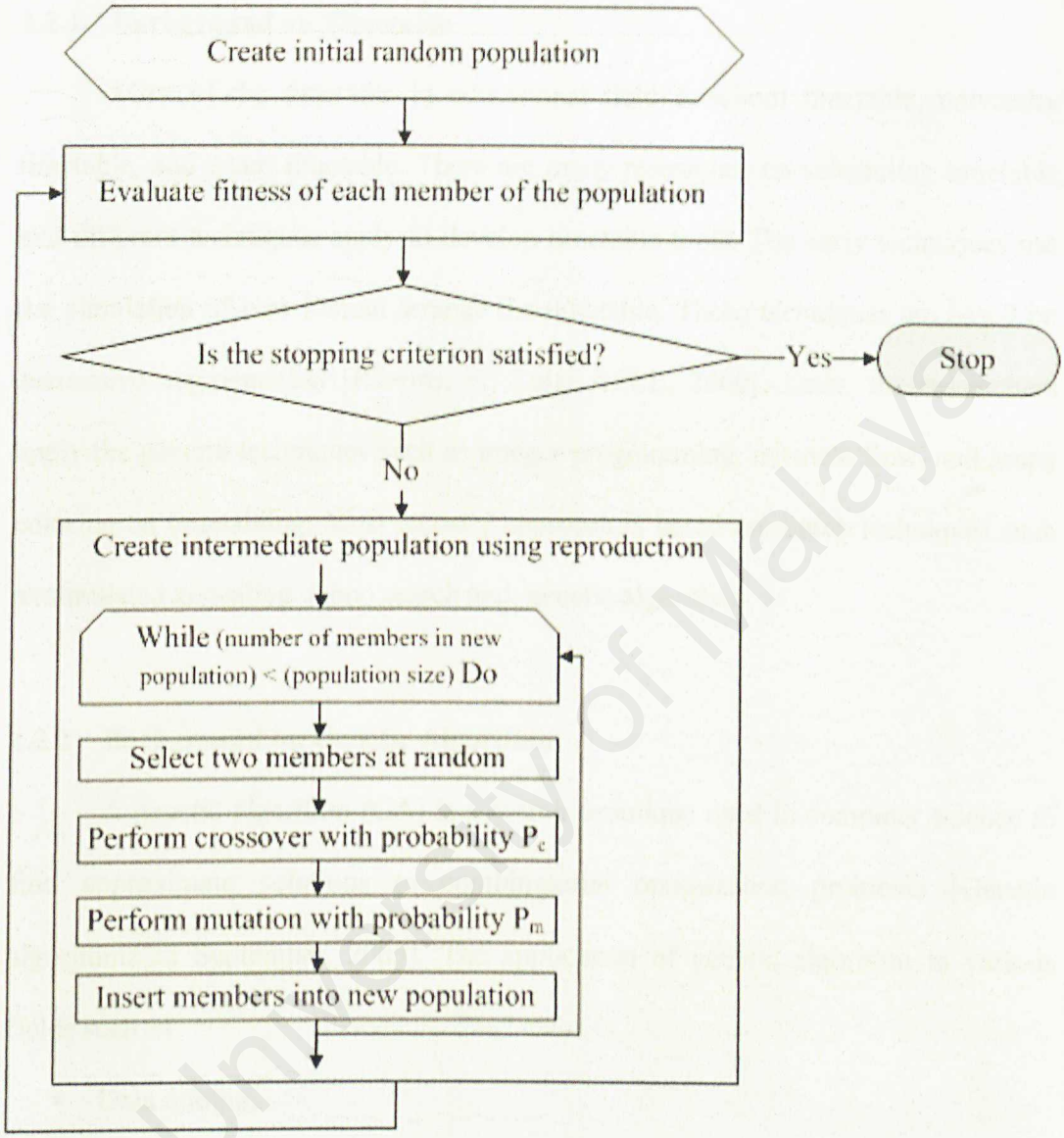


Figure 1.2: *Flow Chart of the standard genetic algorithm [Rooij, A.J.C.V., et al., 1998]*

1.2 Research Background

This section is written about the timetable research background and the genetic algorithms background.

1.2.1 Background on Timetable

Most of the timetable in educational field is school timetable, university timetable, and exam timetable. There are many researches on scheduling timetable and different techniques apply to develop timetable tools. The early techniques use the simulation of how human arrange the timetable. Those techniques are based on successive augmentation [Ribeiro, F., Luiz, A.N.L, 2000]. Later, the researchers apply the general techniques such as integer programming, network flow, and grapy coloring on timetabling. Most recently approach is based on search techniques such as simulated annealing, taboo search and, genetic algorithm.

1.2.2 Background on Genetic Algorithm

A genetic algorithm (GA) is a search technique used in computer science to find approximate solutions to combinatorial optimization problems [Genetic algorithm, 28 September, 2005]. The application of genetic algorithm in various fields such as

- Data mining,
- Training Neural Network,
- Scheduling / Timetabling,
- Traveling salesmen problem ,
- Learning fuzzy rules base, and others application.

1.3 Objectives

This project is develop an automatic timetabling tool to schedule the timetables for secondary school (Form 1 to Form 5) in Malaysia using genetic algorithm approach. This timetabling tool can use to obtain a quality and free conflict timetable which is satisfy all the hard constraints and as many soft constraints as possible. Applying genetic algorithm to timetable problem can optimize the problems.

1.4 School Timetable Problem

School timetable problem is a weekly scheduling for the entire teacher doesn't meet two group of student at the same time. This problem is assign group of student to rooms and timeslot, teacher to rooms and timeslots which will fulfill the requirement of hard constraints and soft constraints. Those of the basic constraints is stated as follow

- Same teacher cannot teach more than one classroom at same time,
- Two or more teacher cannot attend in one class at same time (Except some subject can combine the classroom, such as sports period),
- Assign suitable type of classroom for some subject (such as science period),
- Subject taught by same teacher should not be assigned to the same timeslot,
- Some period are reserved for specific actives such as time break, assembly, sport, and else.

1.5 Proposed Solution

This project is developing a timetabling tool use genetic algorithm. A complete student timetable is determined by teacher timetable and classroom timetable. There important here is teacher and classroom timetable. Whole scheduling process is depending on teacher timetable and follows with classroom timetable. There have three constraints must be determine which is fixed constraints, soft constraint, and hard constraints. Fixed constraints and hard constraints almost same, differentiate between them is fixed constraints have high priority to schedule first.

Fixed constraints must not fail since they are essential for valid solution and have a high priority to soft first. Fixed constraints such as

- Some subject only valid in specific period (such as sport class for morning school usually teaches before time break, while afternoon school usually teaches after time break.),
- Each of period (timeslots) is determine with week of day and range of time (Example, Mon 9.00am-9.45am or Fri 3.00pm-3.45pm), and
- Some period are reserved for specific activities (such as time break, assembly, sport, and else).

Although violation of the soft constraint does not imply an invalid timetable, but soft constraints should be minimized. Soft constraints such as

- Class must no gaps between period, and
- Limitation of period take by a teacher,

Hard constraints must not fail since they are essential for valid solution. Hard constraints such as

- Same teacher cannot teach more than one classroom at same time,

- Two or more teacher cannot attend in one class at same time (Except some subject can combine the classroom, such as sports period), and
- No student can assign more that one subject at same period.

Crossover operator is one of the genetic operators. The individual's of each child are formed as a mixture of the individual's of the parents. Each child is obtained as a random mixture of its parent using a crossover operation. Figure 1.3 shows simple crossover operators.

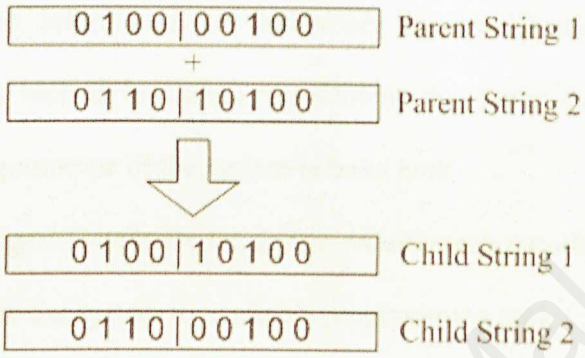


Figure 1.3 *Simple crossover operators*

Mutation operator is a greater scope for improvement. At times individual in the population undergoes a random mutation. Figure 1.4 shows simple mutation operators.

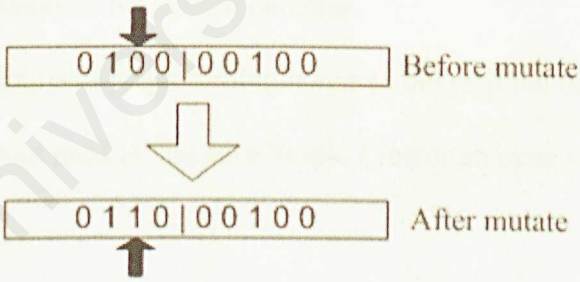


Figure 1.4 *Simple mutation operators*

Selection operator is to fitter individual mate. Selects two parents form the population for mating. The selection is biased towards fitter individuals.

This project is developing basic on part of the system development life cycle (SDLC) methodology. SDLC methodology involves the process as follow planning, gather information and analysis, system design, testing, and implementation.

- Planning – Define the basic need for the system, expected outcomes and else to produce a proposal.
- Gather information and Analysis – Gather information from journal, article, or other source. After gather the entire relevant information and detail, analysis the information to determine the specification of the system requirement such as functional requirement, non-functional requirement, and else. Some prototype of the system is build here.
- System design – Determine the design requirement, pseudo code and protocol system. Code the system use suitable programming tool.
- Testing – System is testing for logical error and physical error. The goal here is to determine system can provide output which is free error and get the respective output.
- Implementation – System can be implementing for some new feature or function to improve the system outcome.

In this project, java programming is use to develop the timetabling tool. The java programming tool used is Sun Java Studio Creator an open source programming tool.

1.6 Project Scope

In this project, the timetabling tool is developing to schedule the timetables for secondary school from form 1 to form 5. This project will focus on the object teachers, classrooms, subjects, and periods.

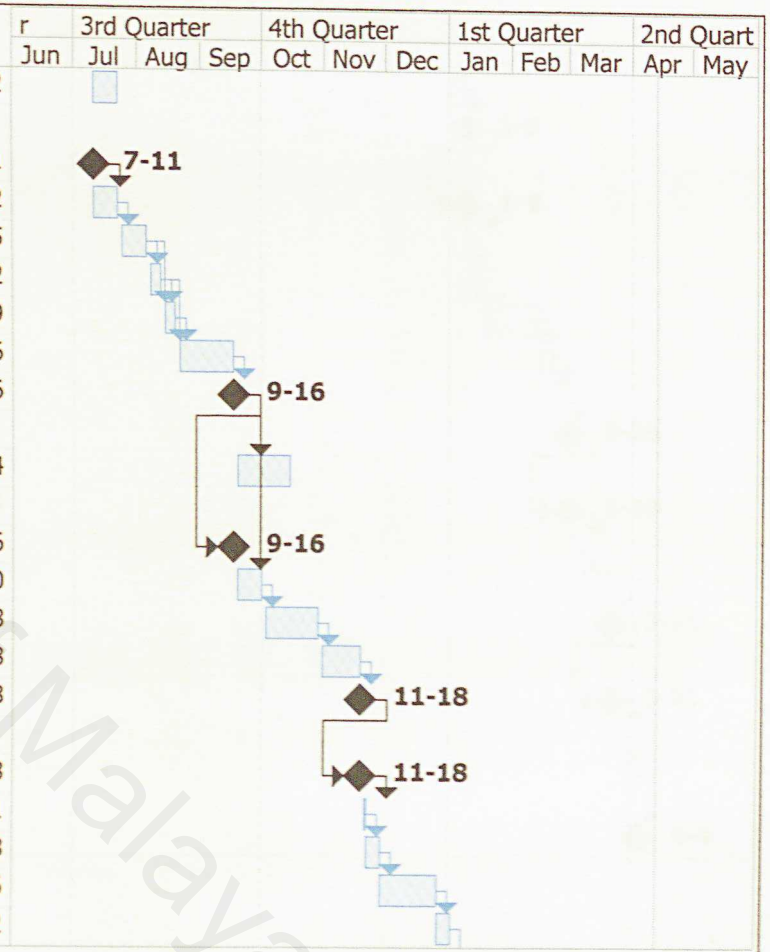
1.7 Target User

School staff will use this timetable tool to schedule the timetable.

1.8 Expected Outcome

Timetabling tool can provide an optimal solution for the problem. It does also generate a set of timetable in non conflicts conditions which is satisfying all of the fixed constraints, hard constraints and soft constraints. Those of the timetable are absolute fit the entire period with the teacher, subject, and classroom.

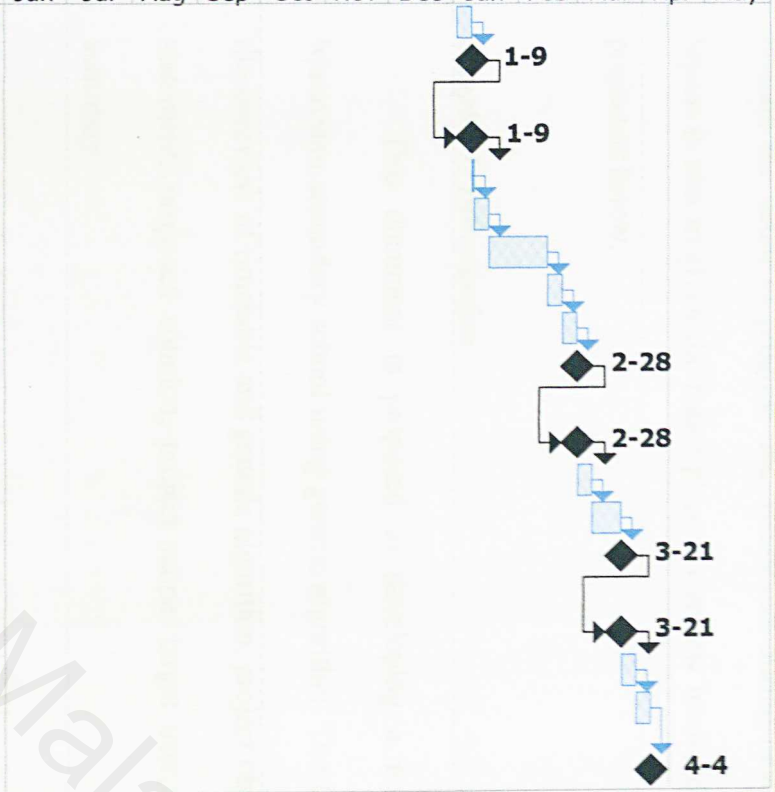
ID	Task Name	Duration	Start	Finish
1	Select suitable project	2 wks	Mon 05-7-11	Fri 05-7-22
2				
3	Start planning stage	0 days	Mon 05-7-11	Mon 05-7-11
4	Search relevant information	2 wks	Mon 05-7-11	Fri 05-7-22
5	Analysis article or journal and review it	2 wks	Mon 05-7-25	Fri 05-8-5
6	Determine problem statement	1 wk	Mon 05-8-8	Fri 05-8-12
7	Determine proposed solution	1 wk	Mon 05-8-15	Fri 05-8-19
8	Proposal	4 wks	Mon 05-8-22	Fri 05-9-16
9	Finish planning stage	0 days	Fri 05-9-16	Fri 05-9-16
10				
11	Learning java programming tool	4 wks	Mon 05-9-19	Fri 05-10-14
12				
13	Start stage Gather information and analysis	0 days	Fri 05-9-16	Fri 05-9-16
14	Gather relevent information	2 wks	Mon 05-9-19	Fri 05-9-30
15	Analysis information	4 wks	Mon 05-10-3	Fri 05-10-28
16	Determine entire system requirement	3 wks	Mon 05-10-31	Fri 05-11-18
17	Finish stage Gather information and analysi	0 days	Fri 05-11-18	Fri 05-11-18
18				
19	Start System Design stage	0 days	Fri 05-11-18	Fri 05-11-18
20	System Flow Chart	1 day	Mon 05-11-21	Mon 05-11-21
21	User Input module	1 wk	Tue 05-11-22	Mon 05-11-28
22	Application Genetic algorithm module	4 wks	Tue 05-11-29	Mon 05-12-26
23	Display analysis and results	1 wk	Tue 05-12-27	Mon 06-1-2



Project: timelineproject
Date: Thu 06-4-13

Task
Split
Progress
Milestone
Summary
Project Summary
External Tasks
External Milestone
Deadline

ID	Task Name	Duration	Start	Finish
24	User Interfaces design	1 wk	Tue 06-1-3	Mon 06-1-9
25	Finish System Design stage	0 days	Mon 06-1-9	Mon 06-1-9
26				
27	Start System Implementation stage	0 days	Mon 06-1-9	Mon 06-1-9
28	Login module	1 day	Tue 06-1-10	Tue 06-1-10
29	User Input module	1 wk	Wed 06-1-11	Tue 06-1-17
30	Application genetic algorithm module	4 wks	Wed 06-1-18	Tue 06-2-14
31	Display analysis and result module	1 wk	Wed 06-2-15	Tue 06-2-21
32	System Integration	1 wk	Wed 06-2-22	Tue 06-2-28
33	Finish System Implementation stage	0 days	Tue 06-2-28	Tue 06-2-28
34				
35	Start System Testing stage	0 days	Tue 06-2-28	Tue 06-2-28
36	System Integration testing	1 wk	Wed 06-3-1	Tue 06-3-7
37	Full System testing	2 wks	Wed 06-3-8	Tue 06-3-21
38	Finish System Testing stage	0 days	Tue 06-3-21	Tue 06-3-21
39				
40	Start Compile Documentation stage	0 days	Tue 06-3-21	Tue 06-3-21
41	Document the System evaluation	1 wk	Wed 06-3-22	Tue 06-3-28
42	Compile all document from previous stage in to complete documentation	1 wk	Wed 06-3-29	Tue 06-4-4
43	Finish Compile Documentation stage	0 days	Tue 06-4-4	Tue 06-4-4



Project: timelineproject
Date: Thu 06-4-13

Task
Split
Progress
Milestone
Summary
Project Summary
External Tasks
External Milestone
Deadline

1.10 Report Layout

This project document has divided to eight chapters. Each of those chapter contain the detail of progress and assessment during develop the project. Project layout is aim to given the major phases overview involved for whole development project as below.

Chapter 1: Introduction

This document is proposed to developing a new timetabling tool for Malaysian secondary school using genetic algorithm. The following section contain the overview of timetable and genetic algorithm, project objective and aim, problem statement, proposed solution, project scope, target user, expected outcome, and summary.

Chapter 2: Literature Review

This chapter is cover about the information from some books, journals, articles and else to assist the project. Timetable method, schedule and timetable approaches, timetable problem, genetic algorithm approaches, timetable problem solute using genetic algorithm method, and comparison between article was detailed discuss and review in this chapter.

Chapter 3: Methodology

This chapter is emphasis on the methodology used to develop the project. It also discuss the whole phases with be used in the project briefly.

Chapter 4: System Analysis

This chapter is describing the way of gathering the information needed. Then, make some analysis the information which gathered and produces a possible system requirement base on the analysis.

Chapter 5: System Design

This chapter is describing the conceptual and technical design of the whole system. It covers process flow chart, user interfaces and structure charts.

Chapter 6: System Implementation

This chapter is describing the detail explanation of the implementation phases and the process involves code transforming of the design into a programming language.

Chapter 7: System Testing

This chapter is written about the testing phases. In this phase will test the system output, and check weather the output were satisfy. The objective of system testing is to find system accuracy and any error occurs.

Chapter 8: System Evaluation

This chapter will written about the discussion of strength and limitation system, what problem occur during the development process and else.

1.11 Chapter Summary

This project is developing a timetabling tool using genetic algorithms. The purpose for develop this tool is to generate a set of timetables without any conflict between timetable. Fixed, hard, and soft constraints are defined here. Furthermore, genetic algorithm operator is important as the process generate a new population. Teacher, classroom, subject and period are the basic object in the program. This timetabling tool is providing to school staff that will monitoring this tool. The whole project is using system development life cycle (SDLC) methodology to develop the timetabling tool. Java programming tool is use to program this timetabling tool.

Chapter 2: Literature Review

2.1 Introduction

This chapter is cover about the information from some books, journals, articles and else to assist the project. Three articles were used as main review. The first article was entitled "A Constructive Evolutionary Approach to school Timetabling" and written by Geraldo Ribeiro Filho and Luiz Antonio Nogueira Lorena.

The second article was entitled "An Evolutionary Approach for School Timetabling" and written by Liviu Lalescu, and Costin Badica from University of Craiova, Faculty of Control, Computers and Electronics Software Engineering Department.

The third article was entitled "Genetic Algorithm Approach for Finding Good Course Schedule" and written by Knut-Edvart Ellingsen and Manuel Penaloza from Department of Mathematics and Computer Science South Dakota School of Mines and Technology. Other article and books also used to do the review of this project.

2.2 Scheduling and Timetabling Approach

For this school timetabling problem was research by many of researcher using several of techniques. Early technique is based on simulation of the human array of solving the problem. All such technique based on successive augmentation [Ribeiro, F., Luiz, A.N.L, 2000]. Successive augmentation related to a partial timetable is extended, lecture by lecture, until all lectures has been schedule. Many other general technique and search technique is implementing and apply to this problem. General techniques such as

integer programming [Tripathy, A., 1984], network flow, graph colorings. Search technique such as simulated annealing, taboo search, and genetic algorithms. It also can apply in representative of the class of multi-constrained, NP-hard, combinatorial optimization problem with real-world application [colorni93genetic.pdf].

2.3 Timetable Problem

Timetable problem is form of scheduling problem. It always arrange in tabular format. Timetable can involve planning daily, weekly, or even monthly schedule. Each of the timetabling must involve a lot of constraints to fulfill the timetable. Most of the timetable arrange event, time of an event and location hold to have an even.

While schedule a timetable usually will facing the problem such as

- Some event cannot be place at some location because of the location not suitable, like not enough space to fit a number of people,
- Some event is conduct by same people cannot place the event run at same time,
- Same location cannot place more the one event at same time, and else.

Timetable involve in many fields such as education, transportation, business, and else. Educational timetable involve school, exam, university and others timetables. Transportation timetable involve airline services, train services, and others timetables. Business timetable involve planning staff shifting schedule, business planning and others timetables. This project is concentrate in secondary school educational timetable.

2.3.1 Educational Timetabling

Most of the researchers are always research on exam, school and university timetable. Each type of the timetable problem has different way to schedule, different characteristic, different rules and different constraints. Later have more detail describe about school, exam, and university timetable.

2.3.2 School timetabling

School timetable usually is a weekly schedule for the entire activity (subject) running includes teachers, location (classroom), and student in each period.

Characteristic of school timetable:-

- Full use of available rooms
- Closed timetabling – at any timeslot all rooms are occupied
- Usual timeslot conflicts of classes and teachers, and
- Soft constraints for teachers – performance to some determine timeslots and in general avoiding the waiting timeslot (windows) [Ribeiro, F., Luiz, A.N.L, 2000]

There have a few assumptions:-

- Assign teacher to class on specific timeslots at one time.
- Pair of teacher and classes and used to form conflict free cluster for each timeslot.
- Binary string representing pairs are grouped based on dissimilarity waiting times between classes are explicitly considered as additional objectives.
- Avoid teach meet two class in same time, and vice versa.

2.3.3 University / Course Timetabling

University timetable same as school timetable is a weekly timetable activity (subject) running includes teachers, location (classroom), and student in each period. There have a few assumptions (It's consist of hard constraints):-

- No teacher who is teaching more than a single lesson at a time.
- No student who is attending more than a single lesson at a time.
- Two teachers cannot exist in a classroom at one time.
- Certain activity must take place in suitable type of class room.

2.4 Timetabling Constraints

A set of timetable problem rules with break into three group which is very important rules (fixed constraints), important rules (hard constraint), and intermediate important rules (soft constraints). Fixed constraints and hard constraints are important to determine the timetable without conflict. If those two constraints in not completely fulfill the requirement during timetable arrangement, the conflict will occurrence in timetable. Soft constraints is some rules that have to satisfy for the condition, but if not satisfy, the timetable still don't have any conflict. Those three constraints is mention on the comparison table which is Table 2.1 shows comparison between 3 review and proposed solution.

2.4.1 Hard Constraints

Hard constraints - must not fail since they are essential for valid solution

Hard constraints must fulfill the conflict free while the final result generated.

Hard constraints

- Faculty member only take one course at a time or period.
- Check course was assign in unwanted timeslot.
- Assign the same classroom and at the same time in timetable.
- Check weather a course is scheduled during faculty meetings (Usually not apply in school timetable).

2.4.2 Soft Constraints

Soft constraints consist of preference constraints which use to identify some undesirable timeslot or else. It may or may not fulfill the conflict free in the final generated result. Although violation of the soft constraint does not imply an invalid timetable, but soft constraints should be minimized.

Soft constraints for teachers are considered implicitly on the representation. Set of teacher partitioned on three levels, according the number of class and overall time dedicated to the school. Teachers are asked to identify undesirable timeslots (preference constraints) conformable with their number of class per week [Ribeiro, F., Luiz, A.N.L, 2000].

Soft constraints

- Two courses in a row are assigned to the same faculty member (Usually not apply in school timetable).
- Faculty member do not want to teach in specific classrooms.

Assign course into a required equipment classroom (Usually not apply in school timetable).

2.4.3 Fixed Constraints

Fixed constraints must not fail since they are essential for valid solution and have a high priority to soft first. Fixed constraint actually is part of the hard constraints.

Fixed constraints

- Occurrences of lab courses, and check each lab course in taught in a lab room.
- Check capacity of classroom. (Usually not apply in school timetable).
- Course taught two hour put in a row, and both hours take lace in same classroom.
- Course to available classroom (Usually not apply in school timetable).

2.5 Overview Genetic Algorithms

The Genetic algorithm (GA) is use to solve school timetable problem [Liviu Lalescu, and Costin Badica.]. This approach is applied for university timetabling. The genetic algorithm is use to simulate the suitable scheduling for university timetable. New evolution method is implement which is experimental method to replace the classical method. It focuses on the new evolution method for genetic algorithms to solving problem and optimal solution is performing in schedule timetable free conflict.

The Genetic algorithm (GA) use search method to schedule the university timetable. This approach is applied for university timetabling. It can focus on how to design university timetable use genetic algorithm approach. Fitness function is discussed in this paper. Genetic algorithms provide robust search in complex spaces [Goldberg, David, 1984].

2.5.1 Genetic Algorithms

University has a set of teacher, a set of student (organized hierarchically in years of study, groups, subgroups) and a set of activities (lessons). Each activity represents a "teacher and subject - set of students" relation. A time and computational saving idea is to split the complete timetabling problem into two phases: time (day and hour) allocation and place (classroom) allocation.

School timetable need not allocate the class room because the class room is fixed. This mean don't have to allocate place. There is one exception if science lab in not enough in school or teacher is not enough. The only possible perfect algorithm is to perform an exhaustive search through the solution space and choose the optimal one. The idea is to create a population of individuals, each individual representing a possible timetable.

Fitness Function gives a measurement of the degree of optimality of a chromosome. Hard fitness is counting number for hard constraints conflicts. This approach is work well in simulation. Soft fitness is weight counting for soft constraints conflicts. This approach assigns weight to give its relevance. Allele is representing using set of {0, 1, *}.

There two implemented evolution methods which is classical method, and experimental method. Classical method related to three tournament selection [Lee, Ho Sung C., 2000]. Selecting a candidate chromosome for crossover, mutation, or a simple propagation is done. It will choose either one to generate next new generation randomly. The experimental method consists of double up the number for current individuals. The doubling of population is done use crossover and mutation, but no propagation. The experimental method is to implement an extensive elitism, because weak individual

cannot survive for further generation. Figure 2.1 shows the design of experimental program.

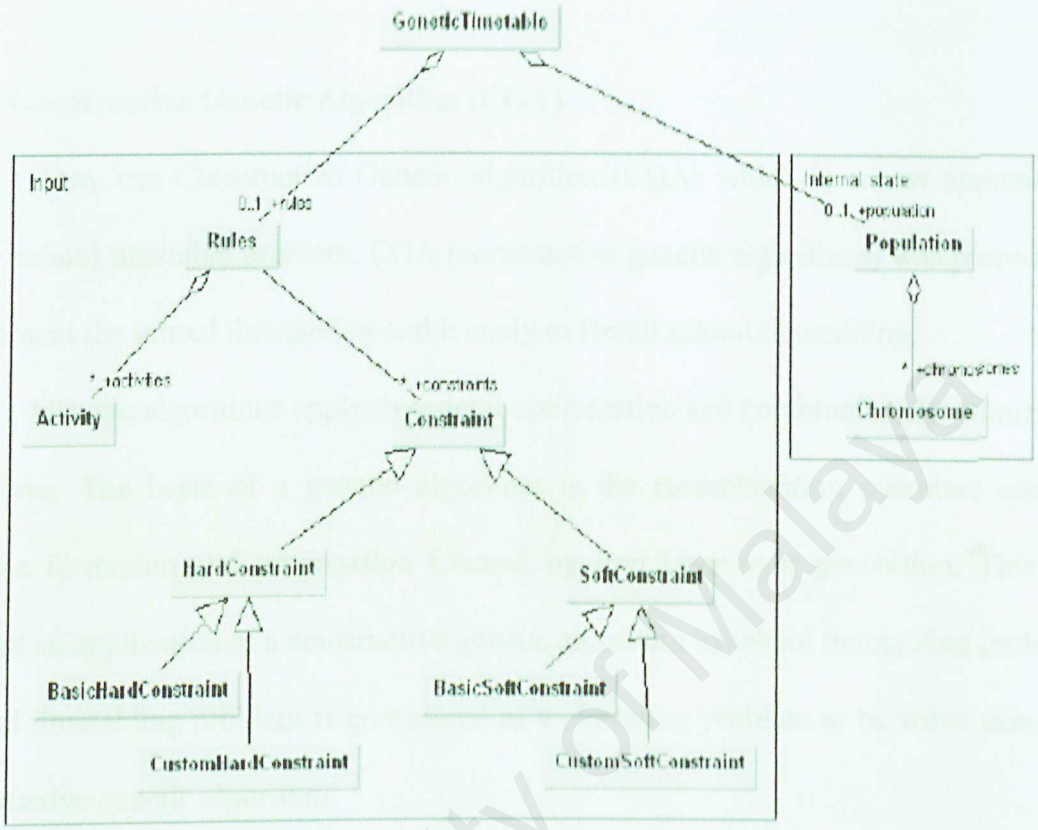


Figure 2.1: *Design of experimental program*

The program has few classes of function such as below:

- Genetic Timetable – encapsulating a set of rules and it has all necessary interfaces function to read/write/save rules, start/stop simulation and save/view result.
- Rules – the set of rules to make up timetable. Include set of teacher, subject taught, classroom, and student.
- Population – represents a set of rules of different member function for implement the current evolution method.

- Chromosome – represent candidate solution. Include interface crossover, mutation, evaluate fitness, and stored information genes [Liviu Lalescu, and Costin Badica.].

2.5.2 Constructive Genetic Algorithm (CGA)

They use Constructive Genetic algorithm (CGA) which is a new approach to solve school timetable problem. CGA (constructive genetic algorithms) was proposes to implement the school timetabling and it apply to Brazil school timetabling.

Genetic algorithms apply to general optimization and combinational optimization problems. The basis of a genetic algorithm is the recombination operators and the schema formation and propagation Created by End User over generation. This will present an application of a constructive genetic algorithm to school timetabling problems. School timetabling problem is considered as a clustering problem to be solve using the constructive genetic algorithm.

CGA proposed to address the problem pf evaluating schemata and structure in a common basis. CGA relies on two functions, mapping the space of structure and schemata. Let *p* contain timeslots in a week, and respecting the lecture requirements of each class. Form possible pair of (teacher, class) which implemented in the p timeslot. *n* contain the total of possible pair [Ribeiro, F., Luiz, A.N.L, 2000]. Soft constraints for teachers are considered implicitly on the representation. Pair (teacher, class) represent by binary columns.

In CGA, a population of schemata (string) formed by n symbols, one for each column. Corresponding symbol is:-

- 1 – the corresponding column is a seed to form a cluster

- 0 – the corresponding column is assigned to a cluster
- # – the column is consider temporarily out of the problem [Ribeiro, F., Luiz, A.N.L, 2000]

To find out the cluster to which a non-seed column will be assigned. Columns are ordered according teacher level and number of performance constraints. We take the seed column that is most dissimilar. The process then continues until all non-seed columns are assigned to a cluster [Ribeiro, F., Luiz, A.N.L, 2000].

Evolution selection, recombination, and mutation process will reach a solution which structure free of conflict, but the soft constraints are not directly considered. Selective will consider explicitly the soft constraints. Population is kept in a non-decreasing order. First schema is complemented to generate a structure representing feasible solution (all #'s are replaced by 0's) before recombination process [Ribeiro, F., Luiz, A.N.L, 2000]. Recombination merges information from both selected schemata, but preserves the number of labels 1 (number of colors) in new generate schema. Mutation process has three processes. First two parts is repair infeasible solutions eventually produce by recombination. The third parts maximum soft constraints satisfactions.

2.5.3 Initialize Population

The idea is to create a population of individuals, each individual representing a possible timetable. Genetic Algorithm that applies to build timetable scheduling is: Population >> Chromosome >> Gene >> Allele. Population contains a number of individuals or chromosome which duplicate not allow. Chromosome is made up by a set

of genes. Chromosomes are an array of genes. Each chromosome has hard and soft fitness factor. Genes represent schedule time of an activity. Allele is representing using set of {0, 1, *}, {0, 1, #} or else.

A population contains a set of chromosome. Population = $\{C_1, C_2, C_3, \ldots, C_i\}$, where i =1,2,3,..., C represent chromosome. Population size are determined by the total chromosome exist in a population.

2.5.4 Encoding of Chromosomes

Chromosomes encode using a set of parameter, where parameter represent use symbol 'P', and it also can call gene. Parameter is important entity or attribute to solve a problem. Different problem may use different parameter to solute the tasks. $C_i = \{P_1, P_2, P_3, \ldots, P_j\}$, i =1,2,3,..., j =1,2,3,.... Chromosomes might be code in form of string (e.g.: $C_i = P_1\ P_2 P_3 \ldots P_j$). Parameter or gene is code using a set of number or alphabet which is call allele.

2.5.5 Genetic Operator

Selection, mutation, crossover is perform in the program. There are sometime mutations operations perform as crossover operations, and act reversely. There have four kind of mutations used in the program. There are three regular crossovers in multiple populations and the last one is mutation function.

2.5.5.1 Selection Operator

The selection of individuals for mating was done in two phrases: pre-selection by tournament, and by random selection [Edvart Ellingsen, Manuel Penaloza., 2003].

2.5.5.2 Crossover Operator

Crossover takes two schedule and swap the time of two random classes, one from each schedule. . There are four simple crossover function is shown as figure 2.2.

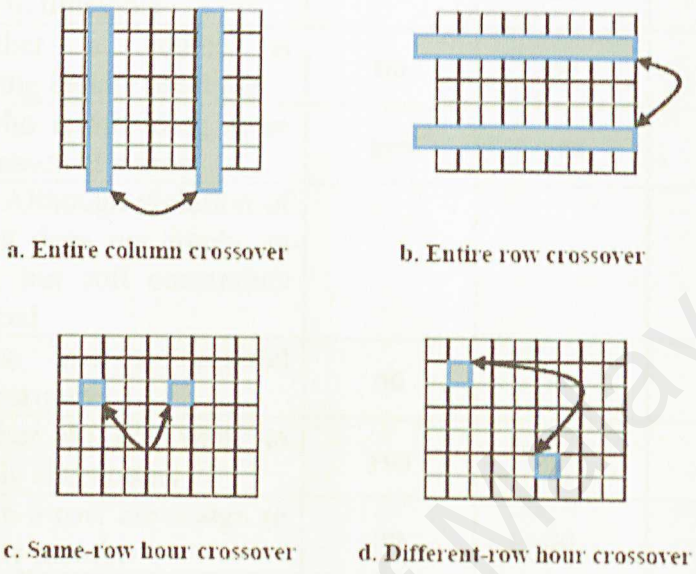


Figure 2.2: *Four simple crossover function*

2.6 Comparison between 3 review and proposed solution

Table 2.1: *Comparison between 3 review and proposed solution*

Description	Review 1	Review 2	Review 3	Project proposed
1. Specific domain area	school	school	university	School
2. Genetic Algorithms	yes	yes	yes	yes
3. Fixed constraints - must not fail since they are essential for valid solution and have a high priority to soft first				
i. Certain activity must take place in suitable type of class room	yes	yes	yes	yes
ii. Capacity of classroom	fixed	fixed	variable	fixed
iii. Course taught two hour put in a row, and both hours take lace in same classroom	no	no	yes	no
iv. Course to available classroom	full use	full use	partially	full use

			use	
4. Hard constraints - must not fail since they are essential for valid solution				
i. Faculty member or teacher only take one course at a time or period	yes	yes	yes	yes
ii. Check course was assign in unwanted timeslot	yes	yes	yes	yes
iii. Assign the same classroom and at the same time in timetable	no	no	yes	no
iv. Check weather a course is scheduled during faculty meetings	no	no	yes	no
v. No student who is attending more than a single lesson at a time	yes	yes	yes	yes
5. Soft constraints - Although violation of the soft constraint does not imply an invalid timetable, but soft constraints should be minimized				
i. Assign course into a required equipment classroom	no	no	yes	no
ii. Faculty member do not want to teach in specific classrooms	yes	yes	yes	yes
iii. Two courses in a row are assign to the same faculty member	no	no	yes	no
iv. Avoiding the waiting timeslot	yes	yes	yes	yes
6. Fitness function - is a measurement of the weight or score for a chromosome	yes	yes	yes	no
7. Chromosome representation	{0, 1, #}	{0, 1, *}	No mention	{0, 1, #}
8. Evolution Method				
i. Selection - Selects two parents form the population for mating	yes	yes	yes	yes
ii. Crossover - individual's of each child are formed as a mixture of the individual's of the parents	yes	yes	yes	yes
iii. Mutation - individual in the population undergoes a random mutation	yes	yes	yes	no
iv. Recombination	yes	yes	yes	no
9. Type of problem solving	Searching	-	Searching	Searching
10. Tools use to develop program	No mention	C++	C++	Java
11. Others features				
i. Class must no gaps between period	-	-	-	yes
ii. Two or more teacher cannot attend in one class at same time (Except some subject can combine the	-	-	-	yes

	classroom, such as sports period)				
iii.	Some subject only valid in specific period (such as sport class for morning school usually teaches before time break, while afternoon school usually teaches after time break.)	-	-	-	yes

2.7 Chapter Summary

These three articles are written about solving timetable problem using genetic algorithm. Two of the article is focus on school and one is focus on university. Schedule the school timetable is slightly different with schedule the university timetable. Each of both timetables has different elements, events or rules to produce a timetable. Those set of rules are important to organize a set of timetable in school or university without conflict between teacher and classroom.

Those set of rules with break into three group which is very important rules (fixed constraints), important rules (hard constraint), and intermediate important rules (soft constraints). Fixed constraints and hard constraints are important to determine the timetable without conflict. If those two constraints in not completely fulfill the requirement during timetable arrangement, the conflict will occurrence in timetable. Soft constraints is some rules that have to satisfy for the condition, but if not satisfy, the timetable still don't have any conflict. Those three constraints is mention on the comparison table which is Table 1.1 shows comparison between 3 review and proposed solution.

Below shown some of the proposed constraints which are written in review 1 and review 2.

- Certain activity must take place in suitable type of class room,
- Capacity of classroom is fixed,
- Full use available classroom to course,
- Faculty member or teacher only takes one course at a time or period,
- Check course was assign in unwanted timeslot,
- No student who is attending more than a single lesson at a time,
- Faculty member do not want to teach in specific classrooms, and
- Avoiding the waiting timeslot.

Other feature which shows in comparison table session 1.4, and not mention for those three reviews is important to determine for the secondary school timetabling tool. The actual situation of those three constraints will finalize in Chapter 3.

Below shown some of the proposed constraints which are not mention in review 1 and review 2.

- Class must no gaps between period,
- Two or more teacher cannot attend in one class at same time (Except some subject can combine the classroom, such as sports period), and
- Some subject only valid in specific period (such as sport class for morning school usually teaches before time break, while afternoon school usually teaches after time break.).

Population, chromosome, gene, allele is hierarchy sequence to form a population. In population contain many of chromosomes and each of chromosomes has a set of gene. Allele (example: {0, 1, #} or {0, 1}) is a set of value assign in to string to form gene. Review 1 and review 2 stated that its use a set of values which consist three elements

like {0, 1, # or *}. Those three will be available, not available, and fixed condition which can be change. Those three conditions are proposed to school timetabling tool.

Genetic Algorithm has three type of basic genetic operator which is crossover, mutation, and selection. Three of those articles have mention using those three operators. Each of those operators has different way to perform their operation. The simple application of the genetic operator is stated in Chapter 1 session 1.4 which is proposed solution. The implementation of the genetic operator will be stated in next Chapter 3.

Genetic algorithm is one of the searching methods. Three of those articles are use the searching techniques to implementing the timetable. Searching techniques usually use to optimize the solution. Java programming is use to develop the timetabling tool compare to others two use programming C++, and one not mention. Java is free programming tool while C++ is cost money. Java is one of the tool that can develop program other than windows platform such as Linux, Solaris, and else.

Chapter 3: Methodology

3.1 Systems Development Life Cycle (SDLC)

This project used system development life system methodology to manage every stage of development process phase. This project used five phrases system development life cycle as below:

- Identifying problems, opportunity, and objective (Planning)
- Determining information requirement and analyzing system needs (Information gathering and analysis requirement)
- Designing the recommended system and developing and documenting (System design and coding)
- Testing and maintaining the system (System testing)
- Implementing and evaluating the system (System implementation and System Evaluation)

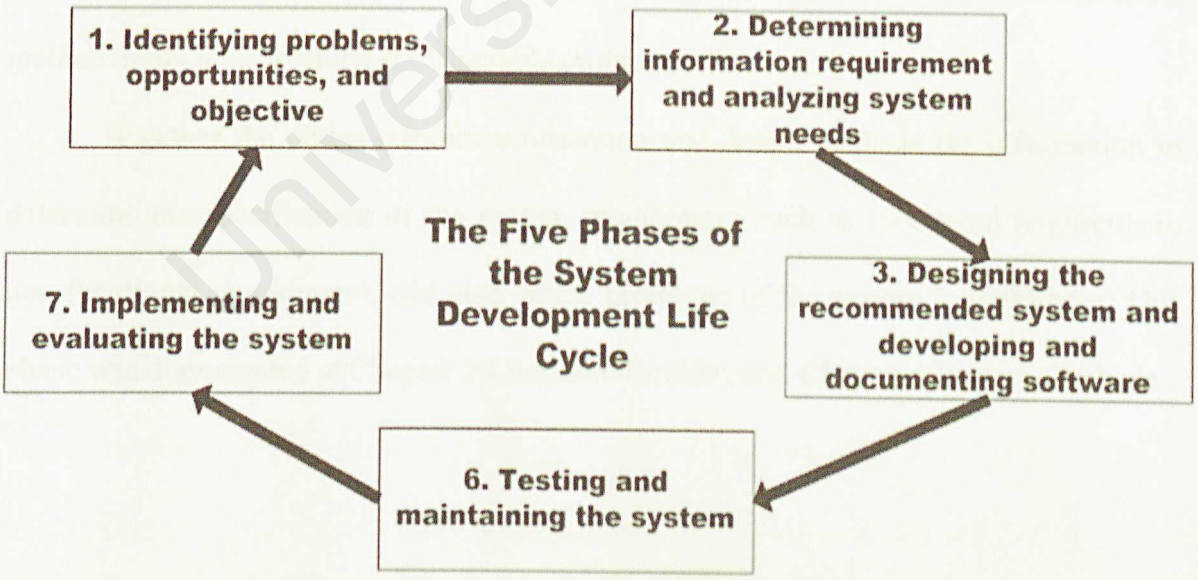


Figure 3.1: *Five phrases of the system development life cycle (SDLC)*

3.1.1 Identifying Problem, Opportunities, and Objectives

This phase is to define problem statement, basic need for the system, scope, expected outcomes and else to produce a proposal. This phase is very important to determine the future development of whole system development. It acts as a guard line to achieve the final goal of the project. Project timeline planning is available in this phase. This phase was documented at Chapter 1 Introduction, Chapter 2 Literature Review, and Chapter 3 Methodology.

3.1.2 Determining Information Requirements and analyzing System Needs

This phase is to gather information from journal, article, or other source. Most of the information gathers according to the project needs. During gathering information, data collected and doing analysis the requirement needs for the system.

Study the information about the genetic algorithm applies in education timetable such as college timetable, exam timetable, school timetable and else. Determine the suitable constraints will be used in the project. Study the way how the genetic algorithm method apply in scheduling the timetable problems.

It gather the entire relevant information and detail, analysis the information to determine the specification of the system requirement such as functional requirement, non-functional requirement, and else. Some prototype of the system is build here. This phase was documented at Chapter 2 Literature Review, and Chapter 4 System Analysis.

3.1.3 Designing the recommended system and developing and documenting

This phase is to determine the design requirement, pseudo code and protocol system, design for input and output graphic user interface (GUI), flow chart for whole system is designed, process genetic algorithm module. Code the system use suitable programming tool. This phase was documented at Chapter 5 System Design.

3.1.4 Testing and maintaining the system

System is testing for logical error and physical error. The goal here is to determine system can provide output which is free error and get the respective output. This phase was documented at Chapter 7 System Testing.

3.1.5 Implementing and evaluating the system

System can be implementing for some new feature or function to improve the system outcome. The limitation and strength of the timetabling tool was discussed in this phase. This phase was documented at Chapter 6 System Implementation, and Chapter 8 Evaluation.

3.2 Chapter Summary

This chapter is written about the five stages of the System Development Life Cycle (SDLC) will used to develop this project. Each phase of the SDLC was important. Those phases are (1) identifying problems, opportunity, and objective, (2) determining information requirement and analyzing system, (3) designing the recommended system and developing and documenting, (4) testing and maintaining the system, and (5) implementing and evaluating the system.

Chapter 4: System Analysis

4.1 System Requirement Analysis

System requirement includes functional and non functional requirement. These projects have the functional requirement and non functional as below:

Functional requirement

- Simple login module
- User data input module
- Process of generate timetable
- Display of analysis process
- Display of result

Non functional requirement

- User friendly Interface requirement
- Performance requirement
- Software and hardware requirement

4.1.1 Simple login module

This module is avoiding the unauthorized person using this program. This module only set a static username and password. If access granted, it will straightly enter to program. Otherwise, user cannot use the program.

4.1.2 User data input module

This module is let user key in all the data needed to scheduling the timetable. Those input include teacher, classroom, subject, timetable information are very important to generate a timetable. Below shows the data that will input by user:-

Teacher information

- Teacher first name,
- Teacher last name, and
- Teacher middle name.

Subject information

- Subject for each form level,
- Subject name,
- Subject Package Number, and
- Subject Credit Hour,

Classroom information

- Total of Classroom use in each form level and package

Timetable information

- Decide how many periods needed for each form level,
- Number of period might be 5 X N, where N=9, 10, 11, …,
- First lesson start and Last lesson finish each day, and
- Eliminate unused timeslot.

4.1.3 Process of generate timetable

This stage is process the data input and schedule a set of optimum timetable. Genetic algorithm was used in this process. First, it will generate a population of individual chromosome use the data input by user. Next, check the criteria to make sure that the population is archive the optimum solution. Those criteria also call constraints which are hard, soft, and fixed constraints. If not satisfy, crossover will performed to produce a new population. The new population will loop back to check criteria until it satisfy all constraints. Then will get optimum timetable result.

4.1.3.1 Fixed Constraints

- Some subject only valid in specific period (such as sport class for morning school usually teaches before time break, while afternoon school usually teaches after time break.),
- Each of period (timeslots) is determine with week of day and range of time (Example, Mon 9.00am-9.45am or Fri 3.00pm-3.45pm), and
- Some period are reserved for specific activities (such as time break, assembly, sport, and else).

4.1.3.2 Hard Constraints

- Same teacher cannot teach more than one classroom at same time,
- Two or more teacher cannot attend in one class at same time (Except some subject can combine the classroom, such as sports period),
- No student can assign more that one subject at same period.

4.1.3.3 Soft Constraints

- Some subjects maybe want to combine the classroom, such as sports, and
- Class must no gaps between periods

4.1.4 Analysis Module

This module is let user able to view the process for how the timetable was scheduled using genetic algorithm. It allow user to view teacher and student timetable according to each population.

4.1.5 Result Module

This module is let user able to view the final result of the timetable. It is able to view teacher and student timetable.

4.1.6 User friendly Interface requirement

This graphic user interface will be user friendly. It may let user easy to manage the program. Avoid the complexity of the interface.

4.1.7 Performance requirement

This program must generate an output in a respected time. The response time is important. If too long, that means the program is not efficiency.

4.1.8 Software and hardware requirement

The program is able running in Microsoft Windows XP platform in the desktop or laptop. Java Runtime Environment is needed.

4.2 Samples of timetable

School timetable contain 5 working days and each day is cluster in many period. It may be 10 period per day (Table 4.1: Sample timetable 10 period per day in a week) or 11 period (Table 4.2: Sample timetable 11 period per day in a week) per day. School has teacher timetable and classroom or student timetable. Both of those timetables have the same format.

Table 4.1: *Sample timetable 10 period per day in a week*

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
MON	-	BC	BM	BI	SC	T	PM	MM	KH	KH
TUE	BM	BC	BC	SC	SC	T	MM	MM	SL	SL
WED	MM	GEO	GEO	BM	BM	T	MM	BI	BI	PJ
THU	BI	GEO	SC	SC	SJ	T	MM	KH	BM	PJ
FRI	MM	BI	BM	BC	ASB	T	PM	PM	SJ	SJ

Table 4.2: *Sample timetable 11 period per day in a week*

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11
MON	PJ	BM	KM	KM	T	BIO	BIO	BI	BI	MT	-
TUE	FZ	FZ	SEJ	SEJ	T	ST	ST	MM	BC	BI	BI
WED	MM	PJ	BIO	BIO	T	KM	KM	BC	BC	ST	BM
THU	MM	MM	BM	BM	T	MT	MT	FZ	FZ	PM	PM
FRI	BM	BM	MT	MM	T	BC	BC	BI	PM	SEJ	-

4.3 Tools and Technology Proposed

Java programming was proposed in this project. Java in an object oriented programming and suitable use to this project. No database will used. Save file was saving the data into *.dat file format.

4.4 Chapter Summary

This chapter was written about the entire functional requirement and non-functional requirement of the system. The functional requirements were simple login module, user data input modules, process of generate timetable, display of analysis process, display of result, and printing selected timetable. The non-functional requirements were user friendly Interface requirement, performance requirement, and software and hardware requirement.

Chapter 5: System Design

5.1 System objectives

This project is developing to automatic schedule the timetables for secondary school from form1 to form 5. Designing this timetable scheduling tool is use to produce an optimum solution and expected result. Genetic algorithm is use to solve the school timetable problem.

5.2 System Major Features

System feature involve user data input, application of genetic algorithm, display result, and print result. User data input is important because all of those data will be use to generate a set of timetables using genetic algorithm method. Users can also view and analysis the result.

5.2.1 Simple login module

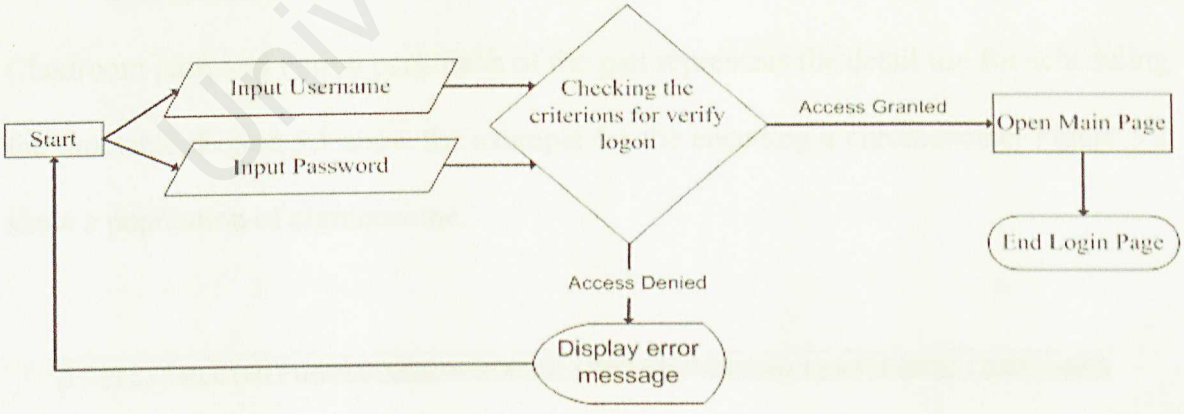


Figure 5.1: *Flow chart of login module*

The program will start with a login page and request user input username and password. If those to input is matched, access will be granted and enter the main program, else access denied and continue let user login or exit program.

5.2.2 User Data Input

User needed to input the teacher, classroom, subject, timetable information detail for predefine section. Predefine section will let user less key in the data in the future use. For schedule input and special feature input, it depend on the predefine data. Schedule input let user to select who is teaching, taught what subject in which form and packages, and how many class teacher will handle. The special input let user choose which subject will limit in which periods were available. Those input detail was written in Chapter 4 section 4.1.2.

After that, all those input will be encoded to the chromosome in a manner of string format. A population of individual's chromosomes represent in an array.

5.2.2.1 Encoding Chromosome

Chromosome is code into four parts which is Teacher part, Subject part, Classroom part, and Period part. Each of the part represents the detail use for scheduling the timetable. Figure 5.1 show the example for the encoding a chromosome. Figure 5.2 show a population of chromosome.

```
T001S1601C101P000000000000000011000000000000110000000011000000000
```

Figure 5.2: *Example encode chromosome in string*

Figure 5.3: *Example for a set of chromosome in a population*

Teacher part

"T001" from Figure 5.1 is representing the detail for a teacher of each chromosome. The first element is representing use alphabet T, it means the header of the teacher part. The second to fourth element in Figure5.1 is representing a teacher code number and the range of the code is in between 001 to 999.

Subject part

"S11601" from Figure 5.1 is representing the detail for a subject of each chromosome. The first element is representing use alphabet S, it means the header of the subject part. The second element is representing form level in school for each subject and the range of the number is in between 1 to 5. The third element is representing

package number for a subject and the range of the number is in between 1 to 9. The fourth element is representing number of credit hour for a subject and the range of the number is in between 1 to 9. The last two elements are representing subject code number and the range of the code is in between 01 to 99.

Classroom part

"C101" from Figure 5.1 is representing the detail for a classroom of each chromosome. The first element is representing use alphabet C, it means the header of the classroom part. The second element is representing form level in school for each classroom and the range of the number is in between 1 to 5. The third and fourth element is representing classroom code number and the range of the code is in between 01 to 99.

Period part

"P00000000..." from Figure 5.1 is representing the detail for a period of each chromosome. The first element is representing use alphabet P, it means the header of the period part. The following element is representing total period for a week and it determined by user. Number of period might be 5 x N, where N = 10, 11, 12 ... and N determine by user. Each of the elements is representing period status and code with {0, 1, and 2}. 0 represent status period is available, 1 represent status period is full, and 2 represent status periods is lock or cannot use.

5.2.3 Application Genetic Algorithms

Genetic algorithms apply to timetable problem to generate an optimum solution. After defining a new population, all the individual's' in population will assign fitness value to 0. Then start to check the population satisfies the criterion to build up an optimum chromosome. It also breaks the individual's in population into good population or bad population depends on the checking result. If 4 criteria are satisfy for individual chromosome, it will assign a fitness value to individual and add it into good population. If not satisfy either one of 4 criteria, individual will assign to bad chromosome.

After check those criteria, if all individual's in population satisfy for those criteria, then the optimum result will be provided. If not satisfy the criteria, the population will use the genetic operator to produce a new population. Good population was the strong survivor and not need to do genetic operation. Bad population was the weak survivor and need to do genetic operation. It will select one of individual in current population randomly to doing the crossover operation to produce new individuals.

For the use of fitness value, it use to determine weather each individual's in current population can continue survive in new population or not.

5.2.3.1 Checking Criterion

There were 4 criteria used to check the timetables were scheduled in the manner of optimum. This 4 criteria was use to solve all the constraints which was defined in chapter 4 section 4.1.3.

4 criteria

- Checking the subject weather can locate in that period or not (Subject was define only available in some period only),
- Checking the teacher table to avoid subject crashing,
- Checking the subject table to avoid subject crashing, and
- Checking the student/classroom table to avoid subject crashing.

5.2.3.2 Genetic Operator

Genetic operator use in this project is selection, crossover, and mutation. Each of those operator is use to produce a new population.

5.2.3.2.1 Selection Operator

Two of the chromosome is being chosen randomly in the process. Later, those two chromosome is use to do crossover operation.

Pseudo code for selecting two chromosomes to do crossover:

- Step1: Get the bad chromosome, c1
- Step2: bFlagT = true;
- Step3: Generate a random value to choose one of the chromosomes in population list, c2.
- Step4: IF c1.Form == c2.Form AND c1.classroomNo == c2.classroomNo AND c1.packageNo == c2.packageNo THEN bFlagT = false
- Step5: IF bFlagT==true GOTO Step3, ELSE continue.

5.2.3.2.2 Crossover Operator

The crossover process in this project is show in Figure 5.3. Crossover process is performing in period part. In period part partition in to 5 sub parts which represent a day. The sub part will selected to do the crossover.

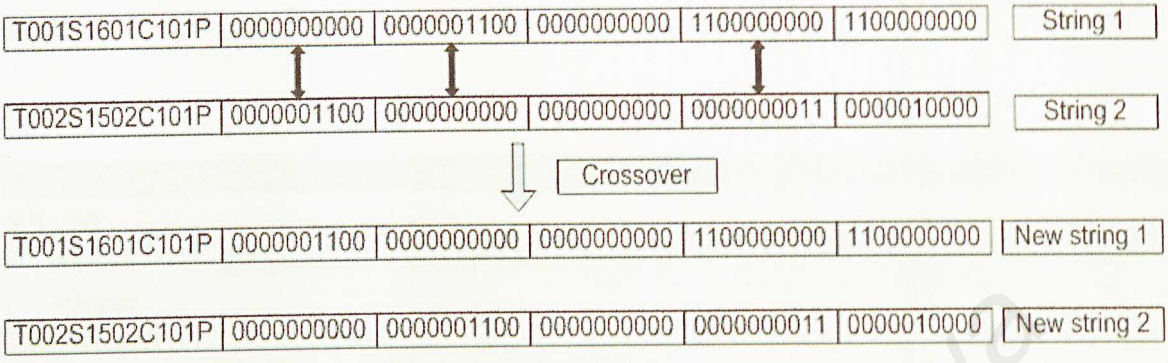


Figure 5.4: *Sample crossover process*

Pseudo code for crossover between two chromosomes:

- Step1: IF subject credit Chromosome 1, c1 equal to Chromosome, c2 THEN crossover in period parts between two chromosomes. ELSE GOTO Step2.
- Step2: Break the period part into 5 pieces per chromosome. Those 10 pieces will reallocate random to two new strings.
- Step3: Checking the bit string of each new string. IF satisfy GOTO Step4, ELSE GOTO Step2.
- Step4: Combine the new string wills the respected chromosomes.

5.2.4 Display of Analysis Process

Each stage of the population generate in the process until achieve the result can be analysis here. Each population data will keep in an array. It can show the result of each population by choosing the population value and analysis it.

5.2.5 Display Result

After finish the process genetic algorithm, the population will be decoded all full set of timetables include student and teacher timetables. User can choose to view the timetable needed, and it will display in manner of tabular which predefined in the early stage.

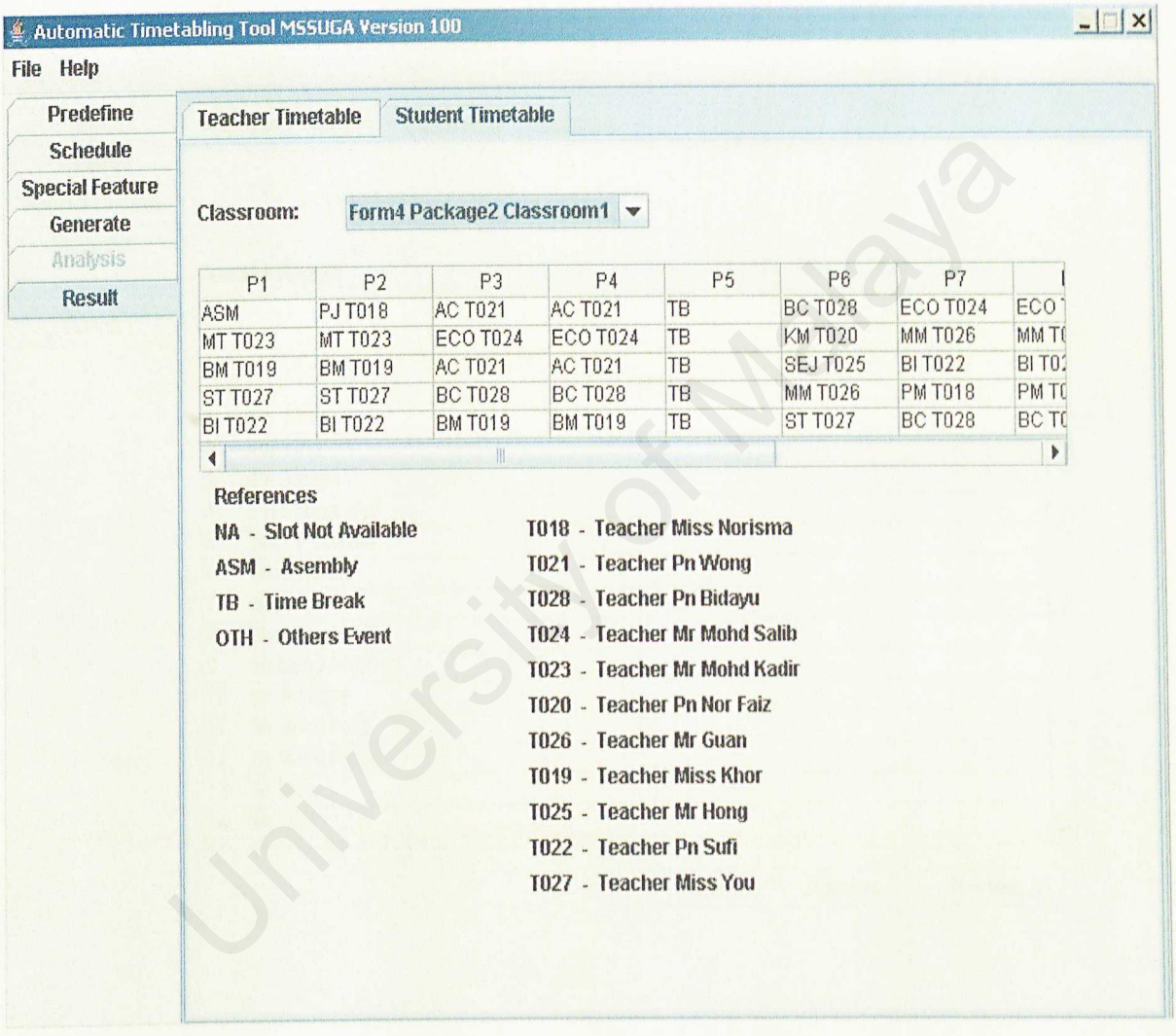


Figure 5.5: *Sample interface for user data input*

5.2.6 Graphic User Interface

The data input screens are almost same view show in Figure 5.4. For the output view timetable screen is same view show in Table 5.1 and Table 5.2. This maybe the suitable display or look to the user. This project is also include Main Menu, few data input screens, choose which timetable to display screen, timetable view screen, and else.

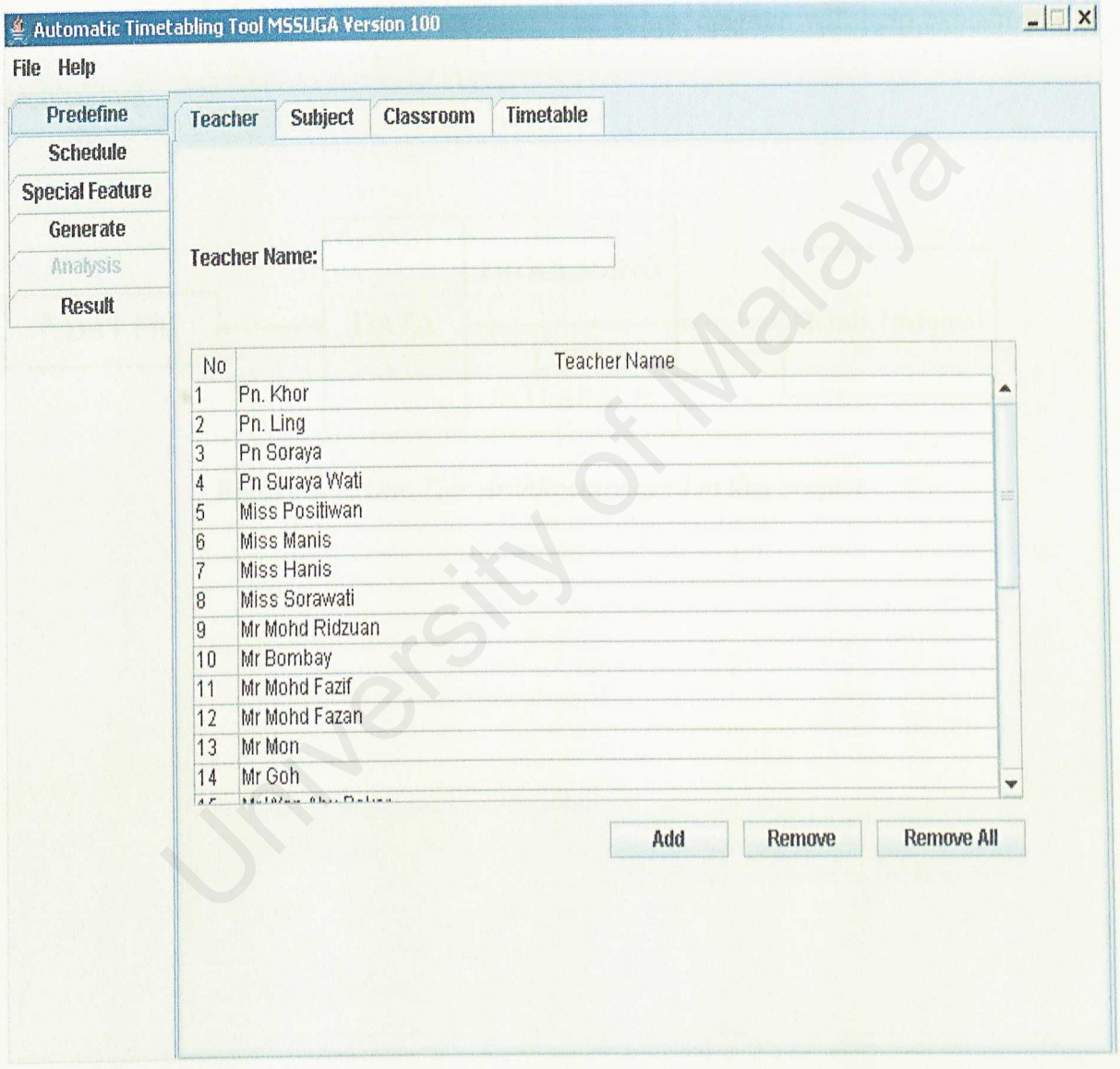


Figure 5.6: *Sample of friendly user interface for teacher input*

5.3 System Architecture

This project is developing using one-tier system architecture. One-tier system architecture is chosen because of no interacting client with the timetabling tool. The timetabling tool is a stand alone tool that only involve user in one computer.

This type of architecture can be explained best, as single computer that performs and deals with all the data, processing and presentation components of the system and outputs are passed to single dumb terminal. The below diagram will help explain this architecture.

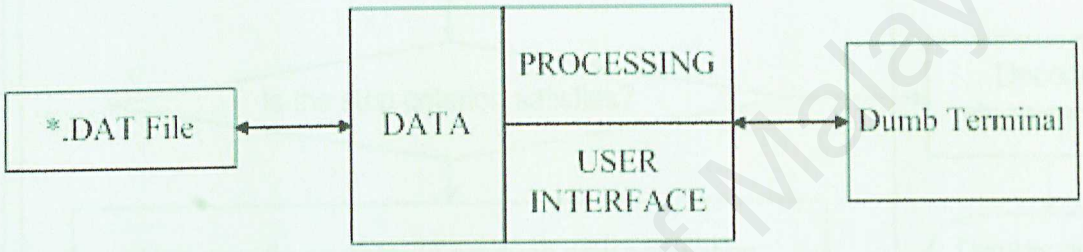


Figure 5.7: *One Tier Architecture used in this project*

5.4 System Flowchart

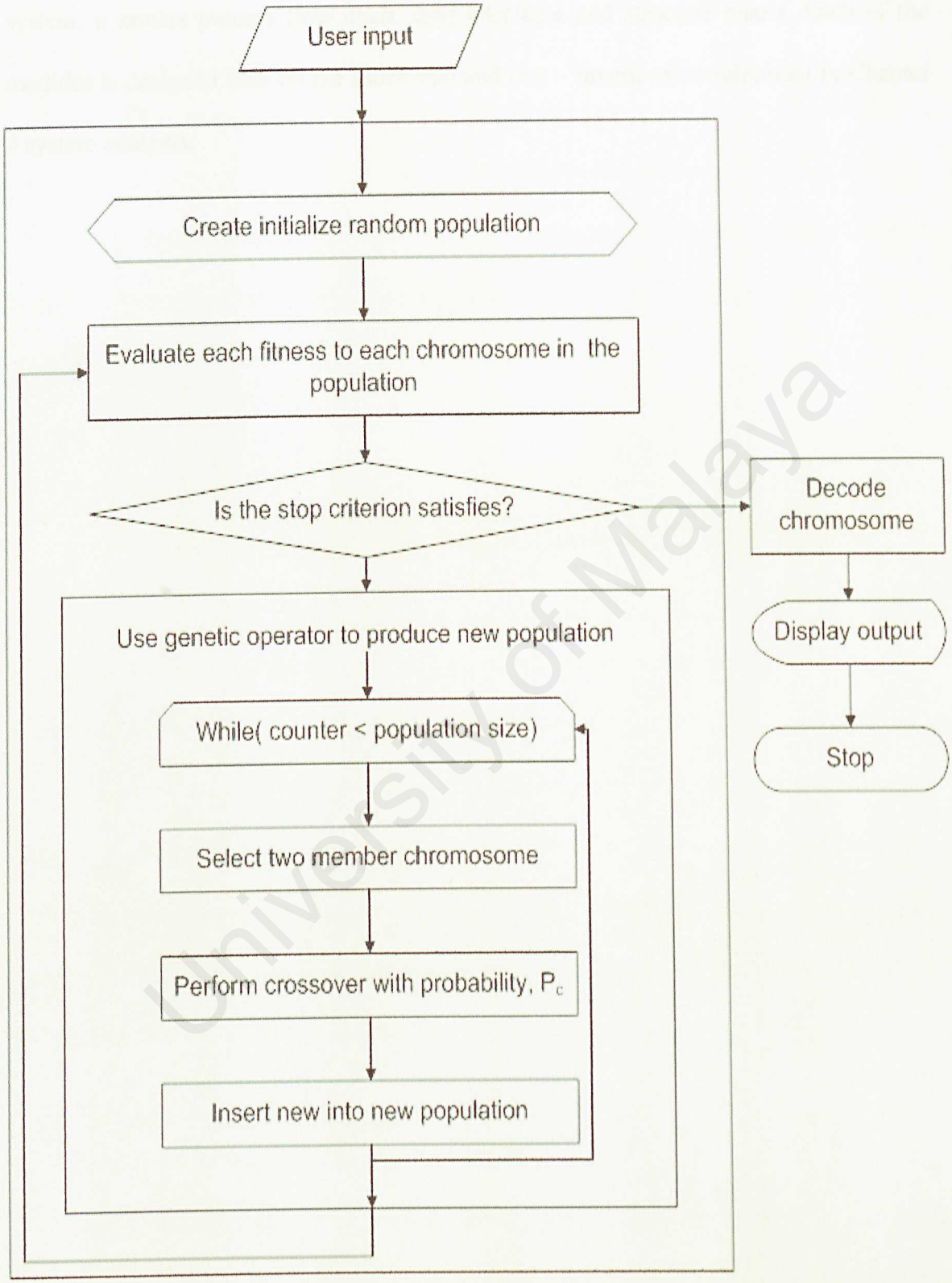


Figure 5.8: *Flowchart for system design*

5.5 Chapter Summary

This chapter is describing the conceptual and technical design of the whole system. It covers process flow chart, user interfaces and structure charts. Each of the modules is designed base on the functional and non – functional requirement in Chapter 4 system analysis.

Chapter 6: System Implementation

6.1 System Implementation

System implementation in software development is a process to convert system requirements into program codes. It also contains further implement information for the system. All requirements determine in Chapter 4 system analysis and Chapter 5 system design will convert it to the programming language. The programming language used to develop the system is JAVA.

6.2 Changes on System

The few changes during the system implementation which in user input and encodes an individual are in population.

6.2.1 User Input

In order let user to key in data one by one every time, the predefine mode is created. Predefine mode data will hold in a temporary data file and load back every time start up or run the system. It let user save time to re-enter the detail will not change rapidly. User only needed to choose the data select from combo box in schedule mode and special mode without key in any data.

6.2.2 Encode an individual's in population

First, the individual's in population was encoding in the manner of an object which will create an object array for a population. Then, the encode individual were

implemented to encode it in the manner of a string which will create a string array for a population. The string is encoded used integer value for each substring. Size of saving the array object data in data file is larger if compare to save an array string. During the system process, the string only needed to compare integer value between individual's and else. Comparing integer value is faster than comparing the words or alphabet.

6.3 Coding

The coding stage is right after system design. All of the design specification will be converting to code to program a system.

6.3.1 Coding Login Module

The code of simple login module was located in Login.java file.

```
public void actionPerformed(ActionEvent e) {
    if (e.getSource() == cmdOk){
        if(txtUsername.getText().compareTo("")==0 || txtPassword.getPassword().length==0){
            //Invalid input
            msgbox.showMessageDialog(null, "Please enter the username and password again."
                                    , "alert", JOptionPane.ERROR_MESSAGE);
        }
        else if(txtUsername.getText().compareTo("Admin")==0 && CheckPassword(
                txtPassword.getPassword())){
            //Access granted
            MainPage TF = new MainPage();
            setVisible(false);
        }
        else{
            //Wrong Username and Password
            msgbox.showMessageDialog(null, "Invalid Username and Password. Please try
                                    again", "alert", JOptionPane.ERROR_MESSAGE);
        }
    }

    if (e.getSource() == cmdCancel){
        System.exit(0);
    }
```

Figure 6.1 *Login module java source codes*

6.3.2 Saving and Open Data File

This features is use to read and write a data file. Below is the code to read and write the data file. The java involve saving and open data are clsSaveFileData.java, and clsCollectionPopulation.java.

```
public void savaFileMode(File fName) throws IOException{
    ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(fName));
    out.writeObject(clsSFD);
    out.close();
}

public void loadFileMode(File fName) throws IOException, ClassNotFoundException
{
    ObjectInputStream inp = new ObjectInputStream(new FileInputStream(fName));
    clsSFD = (clsSaveFileData)inp.readObject();
    inp.close();
}
```

Figure 6.2 *Save and open a data file*

6.3.3 Minimize Usage of Memory Block

Each time create a new array will build a memory block. Expand the memory block while the memory spaces are needed. It also can avoid waste memory. First, define an array with 1 element. While needed to add new data to array, the array will be added one memory space into array. This can save more memory for the system during the process.

```
public void redimArraysListChromo(){
    String [] sTempListArray;
    int i;
    sTempListArray = new String [sListChromo.length];
    sTempListArray = sListChromo;

    sListChromo = new String [sListChromo.length+1];
    for(i=0;i<sTempListArray.length;i++){
        sListChromo[i] = sTempListArray[i];
    }
}
```

Figure 6.3 *Redefine an array element*

6.3.4 User Input

There have few input screen which let user to key in and control the data. The java file involve user input module as Table 6.1 below. The entire user input interface implemented with user friendly interfaces.

Table 6.1: *Description of user input java file*

File Name	Description
ClassroomForm.java	This file is let user key in the classroom information.
clsSaveFileData.java	This is the class object use to save all the detail.
EntryPredefine.java	Create JTabbedPane for Predefine form.
EntryPredefineSubject.java	Create JTabbedPane for Subject form.
EntryPredefineTimetable.java	Create JTabbedPane for Timetable form.
ScheduleForm.java	This file is let user choose relevant information to create new population.
SpecialFeature1.java	This file is let user choose relevant information to define which period unused for a subject.
SubjectForm.java	This file is let user key in the Subject information.
TeacherForm.java	This file is let user key in the Teacher information.
TimetableForm.java	This file is let user key in the Timetable information.

User Interface for Input Screen

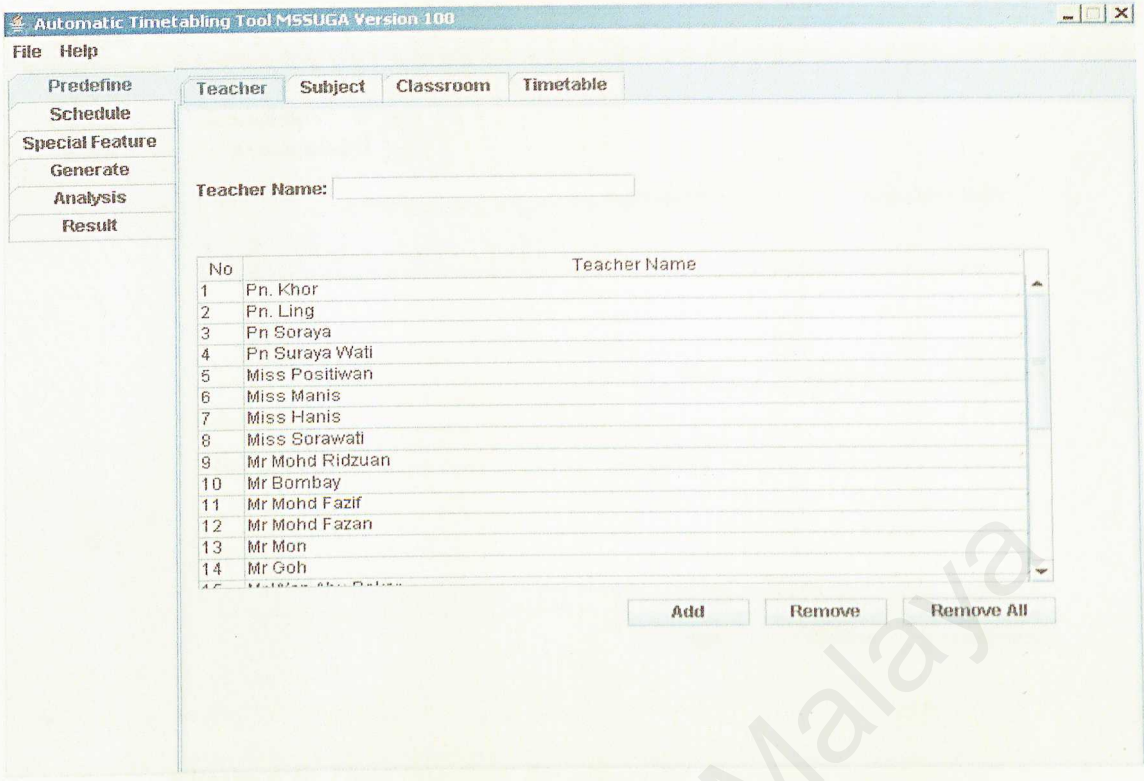


Figure 6.4: *User interface for predefine teacher information*

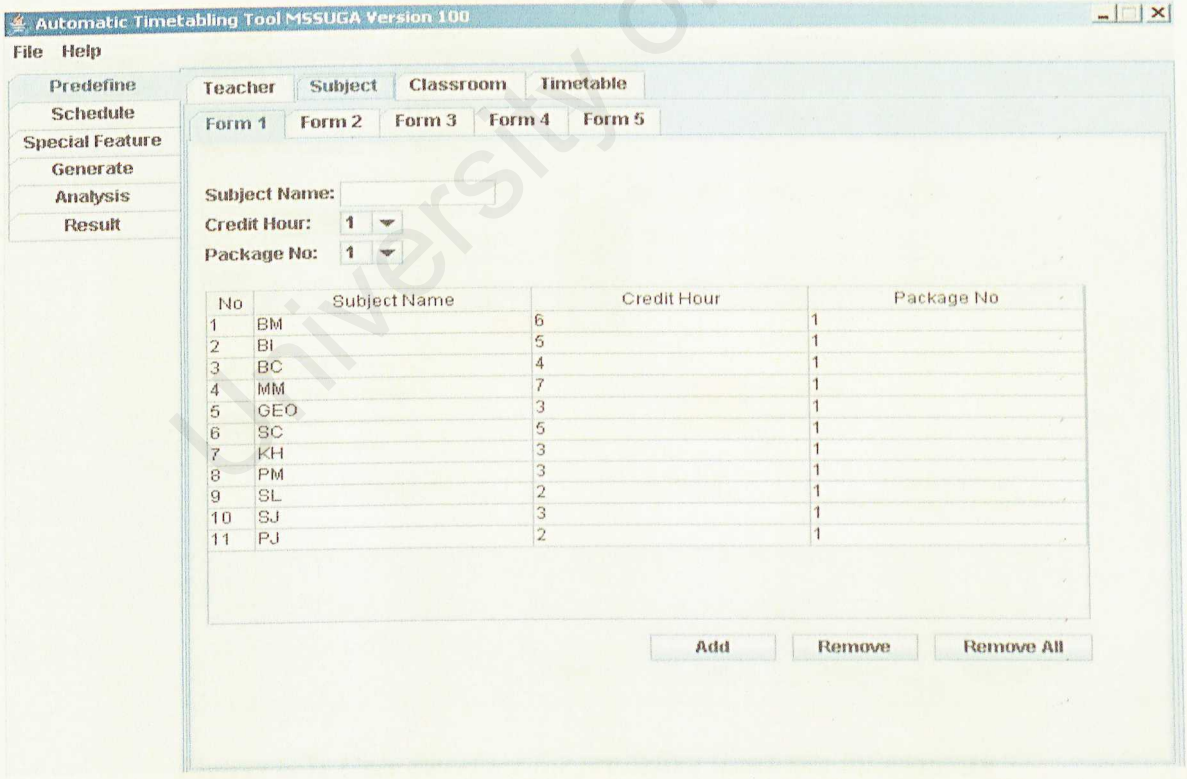


Figure 6.5: *User interface for predefine subject information*

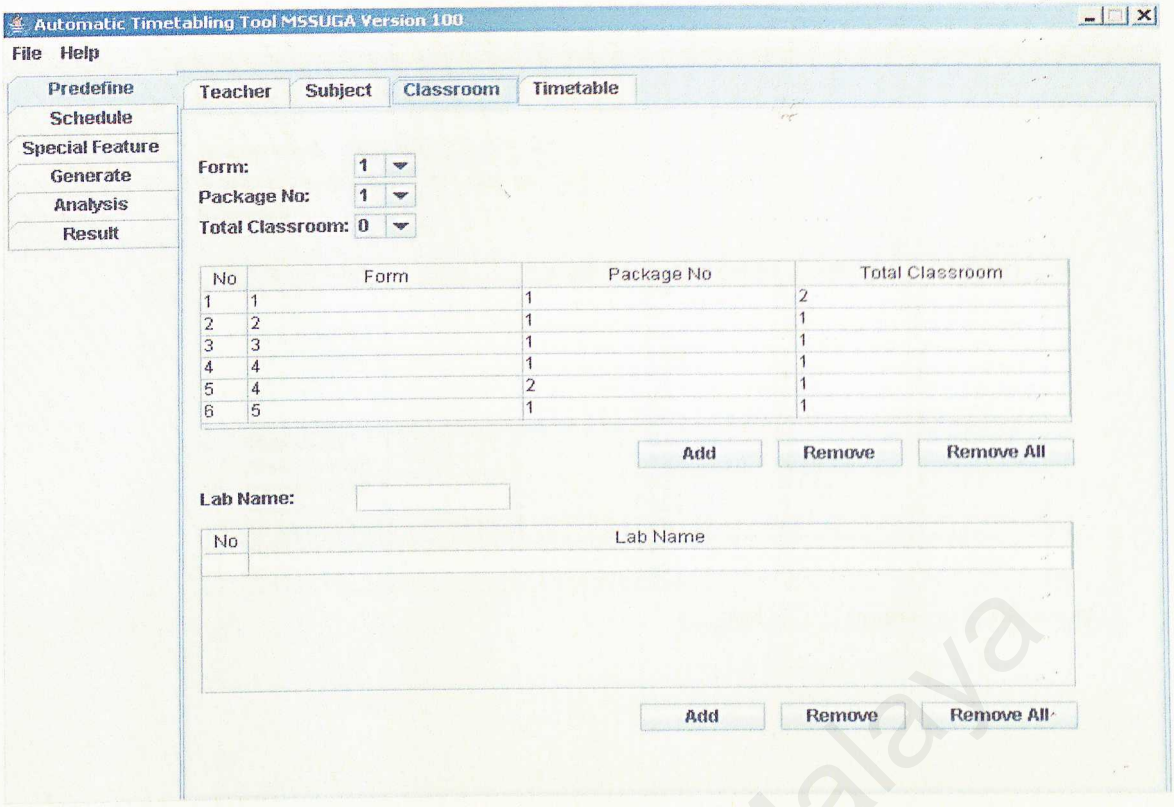


Figure 6.6: *User interface for predefine classroom information*

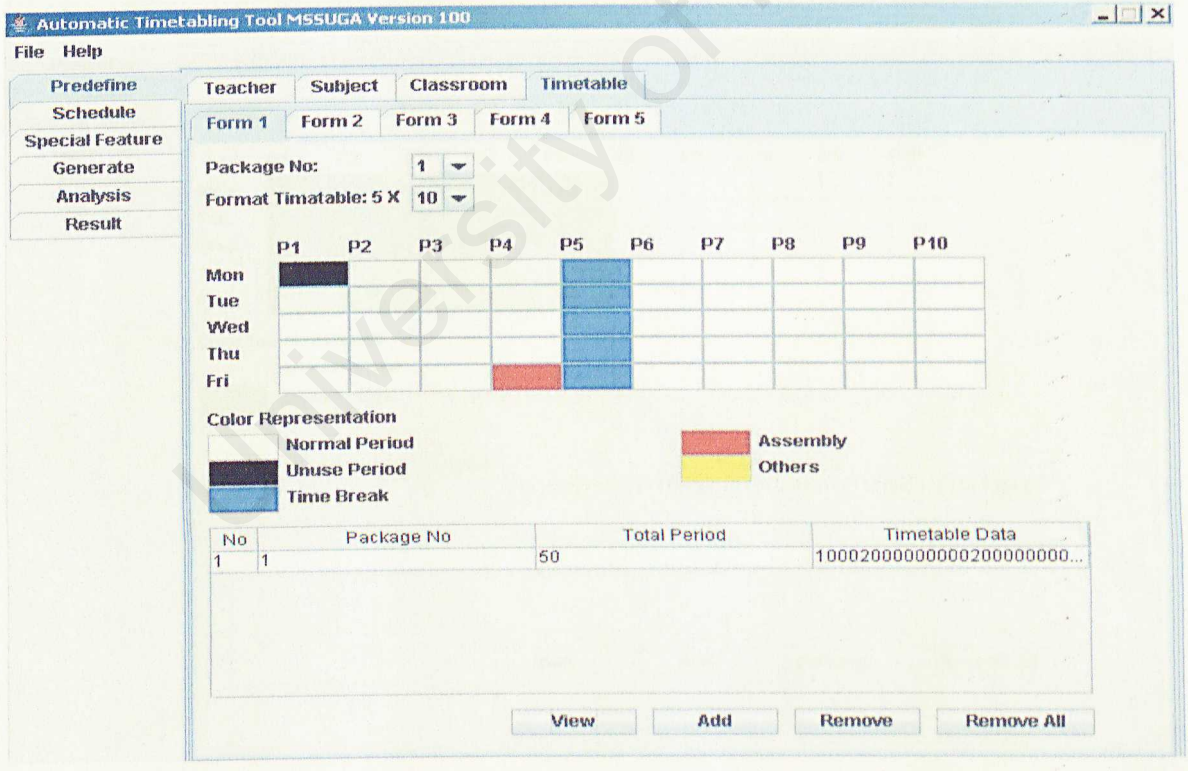


Figure 6.7: *User interface for predefine timetable information*

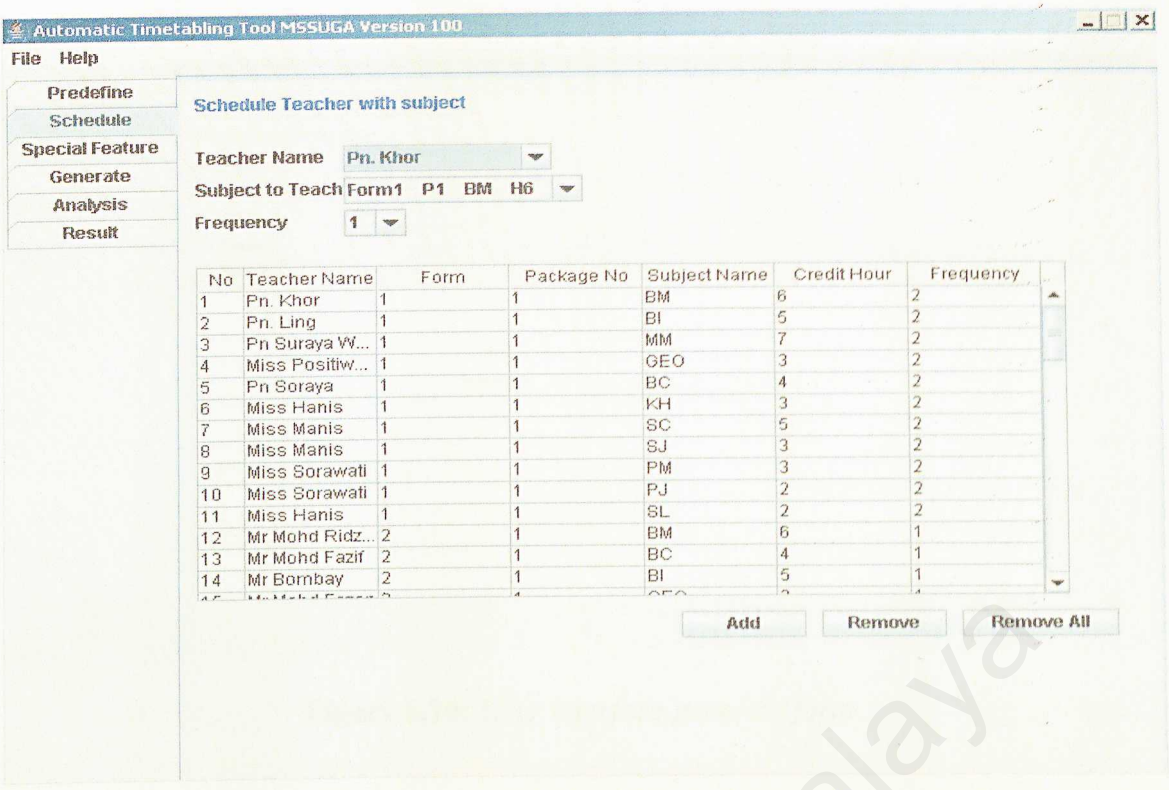


Figure 6.8: *User interface for schedule information*

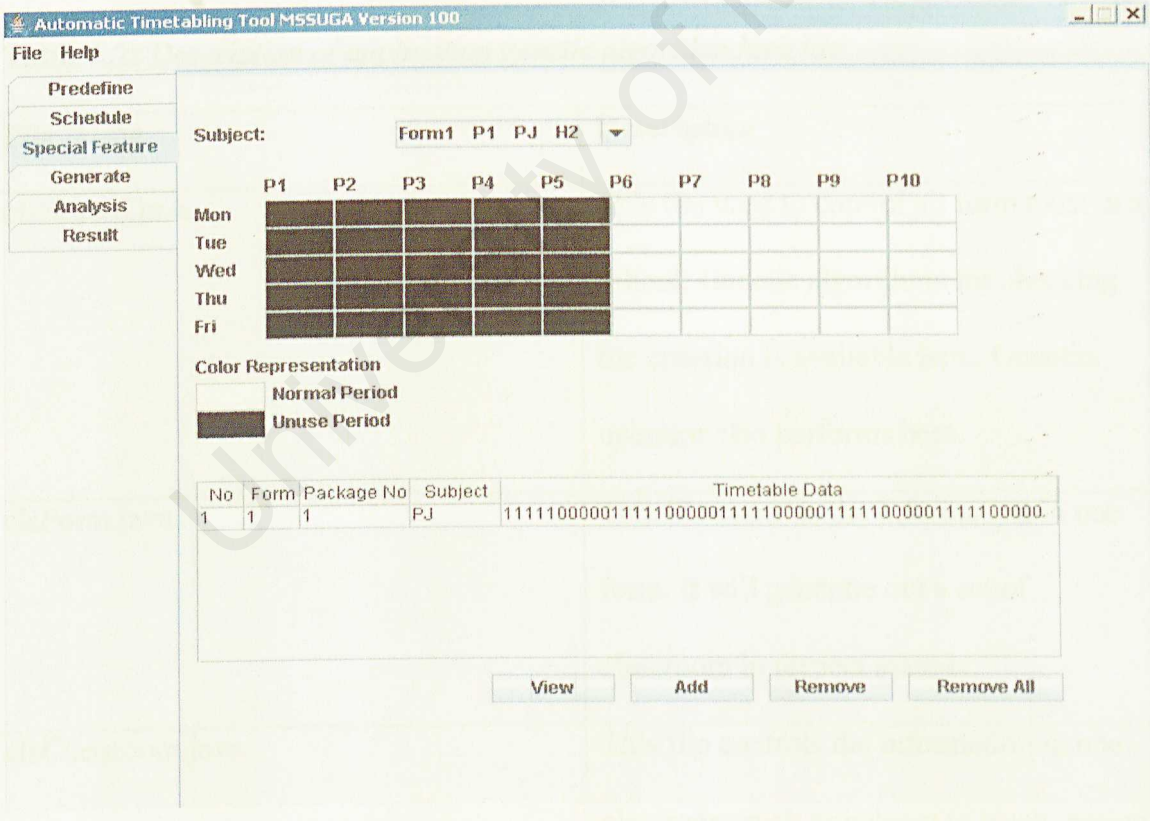


Figure 6.9: *User interface for special feature information*

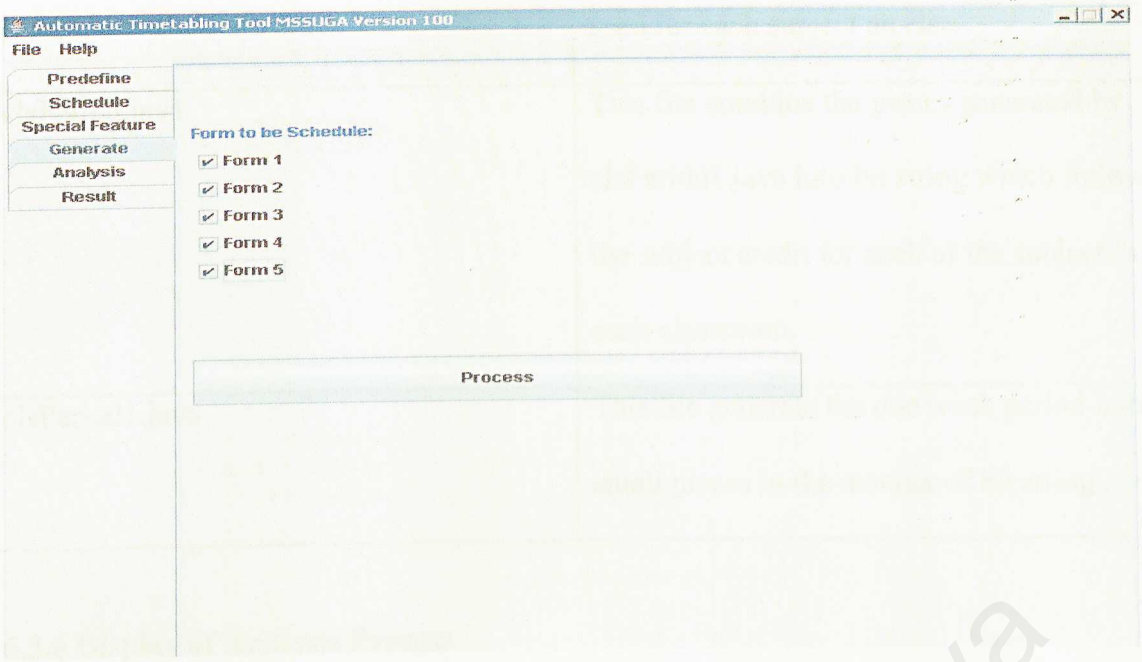


Figure 6.10: *User interface generate form*

6.3.5 Application Genetic Algorithm

Table 6.2: *Description of application genetic algorithm java file*

File Name	Description
clsSchool.java	This file used to control all form exist in a school. Genetic algorithms for checking the criterion is available here. Genetic operator also performs here.
clsForm.java	This file controls the information in one form. It will generate out a set of classroom in respect manner.
clsClassroom.java	This file controls the information in one classroom such as subject to teach, period

	used for each subject an else.
clsPeriod.java	This file combine the pieces generated by clsPeriod1.java into bit string which follow the subject credit for each of the subject in each classroom.
clsPeriod1.java	This file generate the one week period into small pieces in the manner of bit string .

6.3.6 Display of Analysis Process

The codes for the display of analysis process are included in AnalysisViewStudent.java, and AnalysisViewTeacher.java. User can view the process according to teacher or classroom. The interfaces design for display result show in figure 6.11 for analysis teacher timetables and figure 6.12 for analysis student timetables.

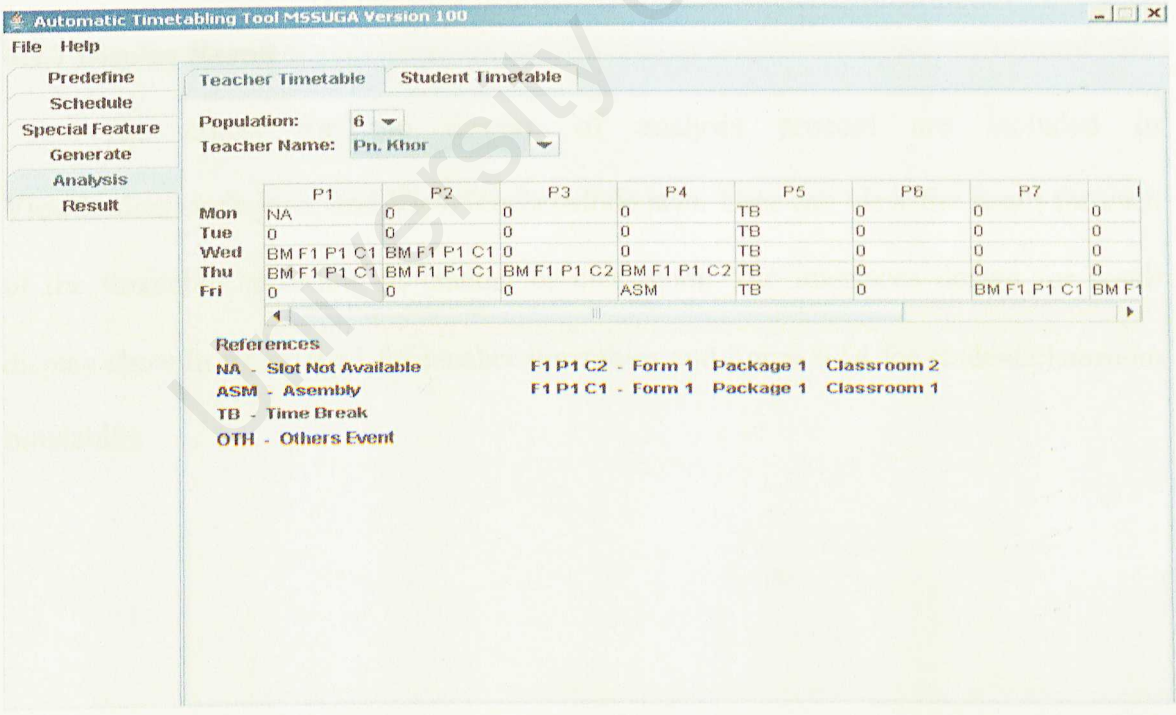


Figure 6.11: *User interface for analysis process of teacher timetable*

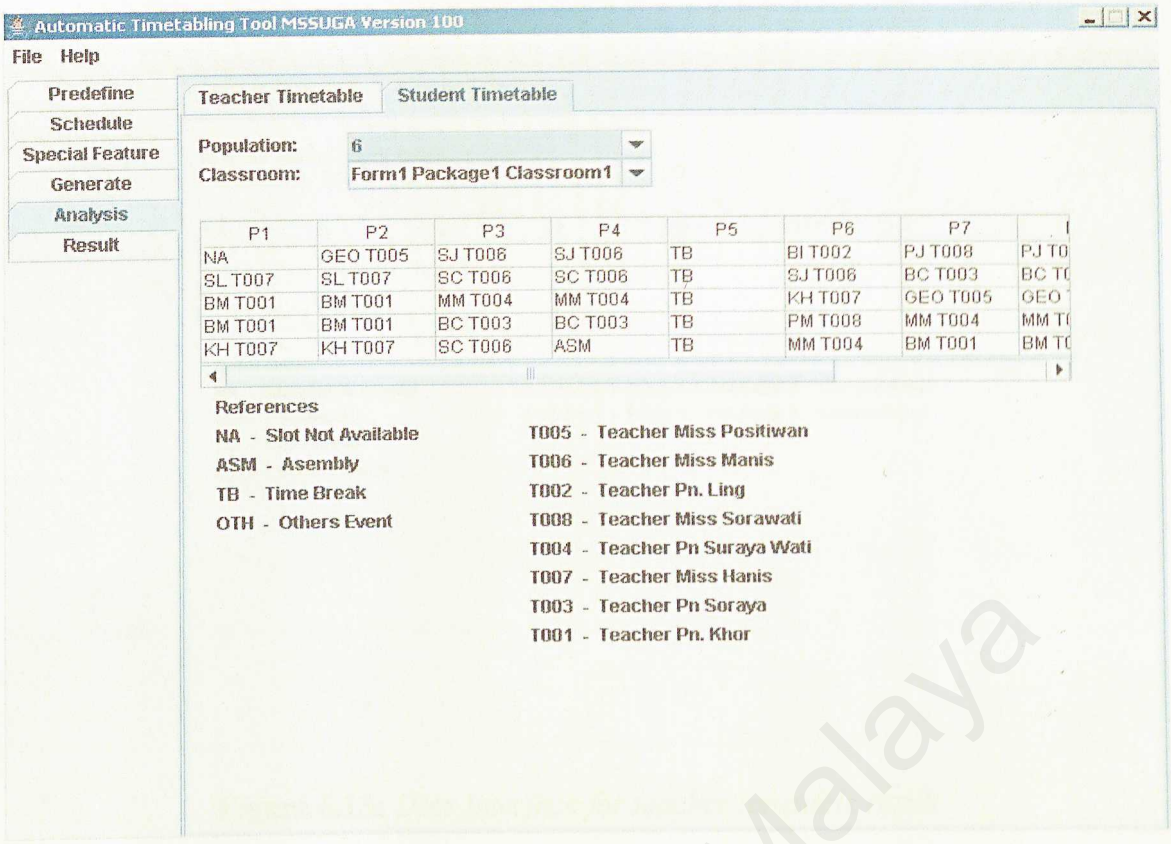


Figure 6.12: *User interface for analysis process of student timetable*

6.3.7 Display Result

The codes for the display of analysis process are included in ViewStudentMode.java, and ViewTeacherMode.java. User can view the result for each of the timetable according to teacher or classroom. The interfaces design for result display show in figure 6.13 for teacher timetables and figure 6.14 for student/classroom timetables.

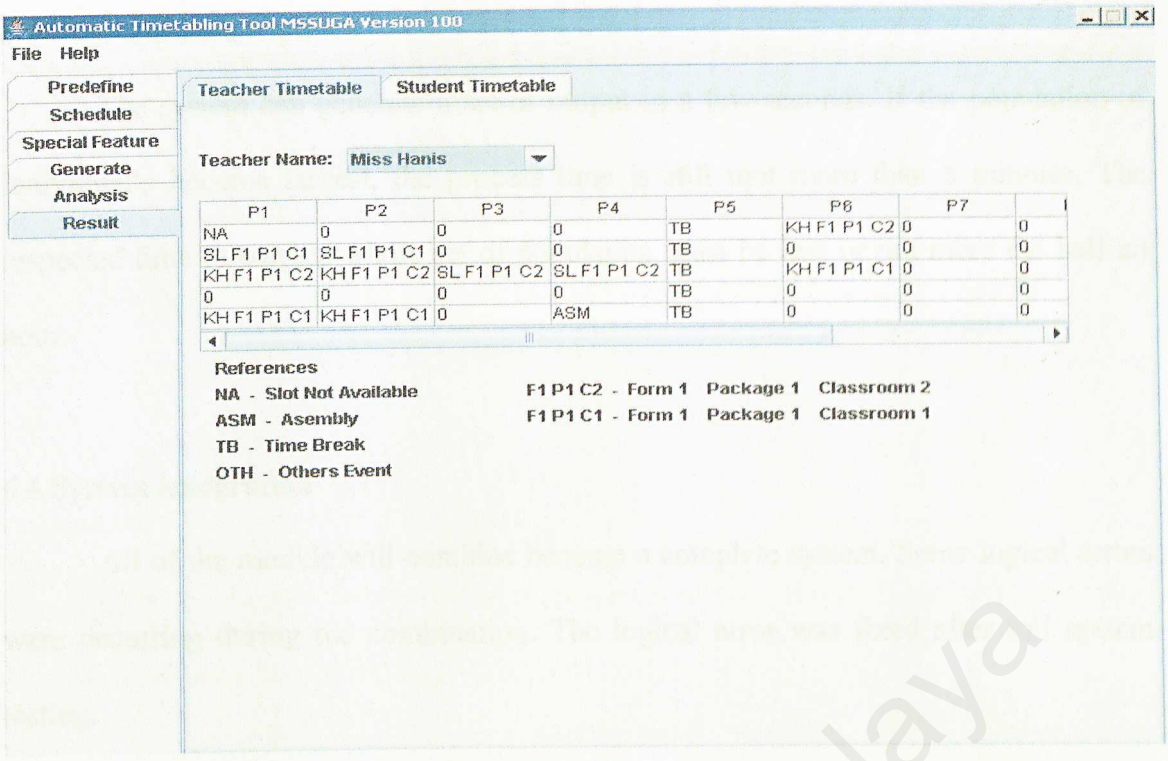


Figure 6.13: *User interface for teacher timetable result*

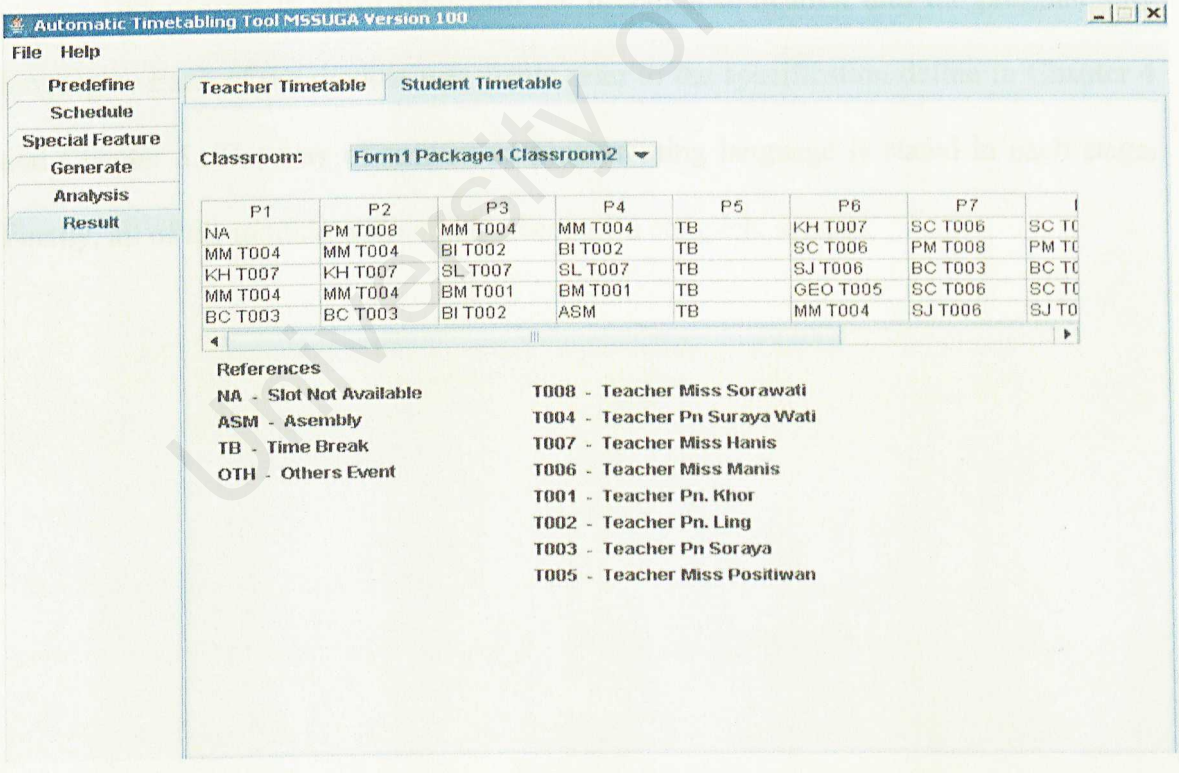


Figure 6.14: *User interface for student/classroom timetable result*

6.3.8 System Performance

The system can generate a set of output in a few seconds. If the population of individual's become largest, the process time is still mot more than 5 minutes. The respected time to solve a whole set of population must be less or not more the half an hour.

6.4 System Integration

All of the module will combine become a complete system. Some logical errors were occurring during the combination. The logical error was fixed after full system testing.

6.5 Chapter Summary

This chapter is about transform the design specification into the programming languages. Each of the module was create refer to the specification define in Chapter 4 and Chapter 5. The way of convert to programming language is stated in each stage. Finally, all of the module will combine become a complete system.

Chapter 7: System Testing

7.1 System Testing

System testing in this project divided to two parts which is component testing during the developing process and full system testing for the production. Full system testing was done in complete system to ensure that no logical error or bugs existing in the system.

System testing also determines weather the system was archive goal and fulfills all functional requirement and non-functional requirement.

7.2 Testing Phase

There are four type of testing performed during this project system development which is Automatic Timetabling MSSUGA version 1.00.

4 Type of testing

- Acceptable testing result,
- Unit testing,
- Integration testing, and
- Full system testing

7.2.1 Acceptable Testing Result

The result is acceptable only if the system satisfying all the constraints which has being defined and the functional and non-functional requirement. The constraints must fully satisfy are fixed constraints and hard constraints, while soft constraints were

optional. The output result will generate output the optimum timetables, and not the best timetables. There was several of optimum output for the same set of testing result.

7.2.2 Unit Testing

Unit testing also called component testing. Unit testing perform during implementation the functional and non-functional module. Each of the module will be tested to get expect output. This type of testing can surely test the module haven't logical error or not. It was very useful in this project.

7.2.3 Integration Testing

Integration testing always performed during combine the entire functional module to be a complete system. During the integration process might be occur some logical error. Integrating testing is used to eliminate all those errors occurred.

7.2.4 Full System Testing

Full system testing is use to test overall of the performance and the expected outcome for the system. This was important to do full system testing before publish the system to the market. Full system testing will test the expected output for the system and looking for logical error and bugs. Few set of testing data will be prepared to test the system and check the expected output.

7.3 Testing Result

The system was test using the information below:

- Teacher used: 28 people
- Subject used: Form1 Package1 11 subjects, Form2 Package1 11 subjects, Form3 Package1 11 subjects, Form4 Package1 12 subjects, Form4 Package1 12 subjects, and Form 5 Package1 12 subjects.
- Classroom used: Form1 2 classrooms and Form2 to Form5 each form 1 classroom
- Timetable used: Define expect timetable format and content.
- Schedule: There were around 80 individual's create in a population.
- Special feature: There were defining Form1 Package1 PJ only available for period 6 to 10 each day.

The result of each population for student timetable has shown as below:

P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11
ASM	0	BIO T020	BIO T020	TB	0	KM T020	KM T020	SEJ T024	SEJ T024	ST T025
0	0	0	0	TB	SEJ T024	KM T020	KM T020	0	0	0
FZ T023	FZ T023	0	0	TB	MM T026	0	0	BIO T020	BIO T020	MM T026
0	0	0	0	TB	0	FZ T023	FZ T023	MM T026	MM T026	0
0	0	0	0	TB	MM T026	ST T025	ST T025	0	0	NA

Figure 7.1: *Testing result of population 1 for student timetable*

P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11
ASM	0	SEJ T024	SEJ T024	TB	0	KM T020	KM T020	0	0	ST T025
BM T019	BM T019	BI T021	BI T021	TB	SEJ T024	KM T020	KM T020	PM T028	PM T028	BC T027
FZ T023	FZ T023	0	0	TB	MM T026	BC T027	BC T027	0	0	MM T026
FZ T023	FZ T023	BC T027	BC T027	TB	PM T028	BM T019	BM T019	MM T026	MM T026	BI T021
BI T021	BI T021	BM T019	BM T019	TB	MM T026	ST T025	ST T025	0	0	NA

Figure 7.2: *Testing result of population 2 for student timetable*

P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11
ASM	BIO T020	SEJ T024	SEJ T024	TB	BIO T020	KM T020	KM T020	PJ T018	PJ T018	ST T025
BI T021	BI T021	BM T019	BM T019	TB	SEJ T024	KM T020	KM T020	PM T028	PM T028	BC T027
FZ T023	FZ T023	0	0	TB	MM T026	BC T027	BC T027	BIO T020	BIO T020	MM T026
FZ T023	FZ T023	BC T027	BC T027	TB	PM T028	BM T019	BM T019	MM T026	MM T026	BI T021
BM T019	BM T019	BI T021	BI T021	TB	MM T026	ST T025	ST T025	0	0	NA

Figure 7.3: *Testing result of population 3 for student timetable*

P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11
ASM	BIO T020	SEJ T024	SEJ T024	TB	BIO T020	KM T020	KM T020	PJ T018	PJ T018	ST T025
BI T021	BI T021	MT T022	MT T022	TB	SEJ T024	KM T020	KM T020	PM T028	PM T028	BC T027
FZ T023	FZ T023	0	0	TB	MM T026	BC T027	BC T027	BIO T020	BIO T020	MM T026
FZ T023	FZ T023	BC T027	BC T027	TB	PM T028	0	0	MM T026	MM T026	BI T021
MT T022	MT T022	BI T021	BI T021	TB	MM T026	ST T025	ST T025	0	0	NA

Figure 7.4: *Testing result of population 4 for student timetable*

P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11
ASM	BIO T020	MT T022	MT T022	TB	BIO T020	KM T020	KM T020	PJ T018	PJ T018	ST T025
BI T021	BI T021	0	0	TB	SEJ T024	KM T020	KM T020	PM T028	PM T028	BC T027
FZ T023	FZ T023	0	0	TB	MM T026	BC T027	BC T027	BIO T020	BIO T020	MM T026
FZ T023	FZ T023	BC T027	BC T027	TB	PM T028	MT T022	MT T022	MM T026	MM T026	BI T021
0	0	BI T021	BI T021	TB	MM T026	ST T025	ST T025	SEJ T024	SEJ T024	NA

Figure 7.5: *Testing result of population 5 for student timetable*

P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11
ASM	FZ T023	MT T022	MT T022	TB	FZ T023	KM T020	KM T020	PJ T018	PJ T018	ST T025
BI T021	BI T021	0	0	TB	SEJ T024	KM T020	KM T020	PM T028	PM T028	BC T027
0	0	0	0	TB	MM T026	BC T027	BC T027	FZ T023	FZ T023	MM T026
0	0	BC T027	BC T027	TB	PM T028	MT T022	MT T022	MM T026	MM T026	BI T021
0	0	BI T021	BI T021	TB	MM T026	ST T025	ST T025	SEJ T024	SEJ T024	NA

Figure 7.6: *Testing result of population 6 for student timetable*

P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11
ASM	FZ T023	MT T022	MT T022	TB	FZ T023	KM T020	KM T020	PJ T018	PJ T018	ST T025
BI T021	BI T021	0	0	TB	SEJ T024	KM T020	KM T020	PM T028	PM T028	BC T027
BIO T020	BIO T020	0	0	TB	MM T026	BC T027	BC T027	FZ T023	FZ T023	MM T026
BIO T020	BIO T020	BC T027	BC T027	TB	PM T028	MT T022	MT T022	MM T026	MM T026	BI T021
0	0	BI T021	BI T021	TB	MM T026	ST T025	ST T025	SEJ T024	SEJ T024	NA

Figure 7.7: *Testing result of population 7 for student timetable*

P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11
ASM	FZ T023	MT T022	MT T022	TB	FZ T023	KM T020	KM T020	PJ T018	PJ T018	ST T025
BI T021	BI T021	BM T019	BM T019	TB	SEJ T024	KM T020	KM T020	PM T028	PM T028	BC T027
BIO T020	BIO T020	BM T019	BM T019	TB	MM T026	BC T027	BC T027	FZ T023	FZ T023	MM T026
BIO T020	BIO T020	BC T027	BC T027	TB	PM T028	MT T022	MT T022	MM T026	MM T026	BI T021
BM T019	BM T019	BI T021	BI T021	TB	MM T026	ST T025	ST T025	SEJ T024	SEJ T024	NA

Figure 7.8: *Testing result of population 8 for student timetable*

The result from figure 7.1 to figure 7.8 was Form5 Package1 Classroom1 timetable. In the result, it shows that each of the subjects is in group of couple or in single subject. There are no gaps between subjects each day. This was satisfying the one of the soft constraints which "class must no gaps between periods".

Furthermore, the result show that the assembly, time break, and NA (unused period) is fixed. This might archive one of the fixed constraints which is defined as "some period are reserved for specific activities". Subsequently, the period also fixed in the timetable which means "each of period (timeslots) is determining with week of day and range of time" is used.

Actually, the students will allocate to a classroom and will be fixed in that classroom in the whole year study. So that, the students wouldn't be assign in to two classrooms in school. The classroom timetables only have all relevant subjects without repeating the subject twice. As the result in figure 7.8, the final timetable only has 12 unique subjects in a classroom and in will satisfy the hard constraints "No student can assign more that one subject at same period".

P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
NA	GEO T005	MM T004	MM T004	TB	SC T006	BM T001	BM T001	KH T007	KH T007
BM T001	BM T001	BI T002	BI T002	TB	MM T004	BC T003	BC T003	PJ T008	PJ T008
PM T008	PM T008	SL T007	SL T007	TB	BI T002	BM T001	BM T001	GEO T005	GEO T005
MM T004	MM T004	SJ T006	SJ T006	TB	PM T008	SC T006	SC T006	BC T003	BC T003
MM T004	MM T004	SJ T006	ASM	TB	KH T007	SC T006	SC T006	BI T002	BI T002

Figure 7.9: *Final result for Form1 Package1 Classroom1*

P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
NA	SC T006	KH T007	KH T007	TB	PM T008	BI T002	BI T002	BM T001	BM T001
MM T004	MM T004	SJ T006	SJ T006	TB	GEO T005	PJ T008	PJ T008	SL T007	SL T007
SC T006	SC T006	BM T001	BM T001	TB	MM T004	GEO T005	GEO T005	BI T002	BI T002
SC T006	SC T006	BC T003	BC T003	TB	SJ T006	PM T008	PM T008	MM T004	MM T004
BC T003	BC T003	KH T007	ASM	TB	BI T002	BM T001	BM T001	MM T004	MM T004

Figure 7.10: *Final result for Form1 Package1 Classroom2*

In figure 7.9 and figure 7.10 was shown that the PJ subject is in the range of period 6 to period 10 which is define in the special feature mode. This prove that the system manage to allocation the subject to the period that were satisfy. In means the system can perform the fixed constraints which are "Some subject only valid in specific period".

Further, only one teacher will assign to each period in classroom timetable. This can see in figure 7.10. There have not more the one teacher assign to each period. In means the system is satisfying the hard constraints "Two or more teacher cannot attend in one class at same time".

P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11
ASM	BM F4 P1 C1	0	0	TB	BM F4 P1 C1	BM F4 P2 C1	BM F4 P2 C1	0	0	0
BM F4 P1 C1	BM F4 P1 C1	BM F5 P1 C1	BM F5 P1 C1	TB	0	0	0	BM F4 P2 C1	BM F4 P2 C1	0
0	0	BM F4 P1 C1	BM F4 P1 C1	TB	0	0	0	BM F4 P2 C1	BM F4 P2 C1	0
0	0	0	0	TB	0	0	0	0	0	0
BM F5 P1 C1	BM F5 P1 C1	0	0	TB	0	0	0	0	0	NA

Figure 7.11: *Final result for Teacher Miss Khor (T019)*

In figure 7.11 shows that the teacher timetable result. The teacher timetable was assign with the subject without crash. On of the checking criteria of application genetic algorithm in the program was used this timetable to check weather the subject can allocate into the teacher timetable. So that, the timetable was no conflict and teacher can assign two classrooms at the same period. This was satisfying the hard constraints "Same teacher cannot teach more than one classroom at same time".

7.4 Chapter Summary

System testing was performed on the Automatic Timetabling MSSUGA version 1.00 tool. The system was satisfying all the constraints which define in chapter 4 system analysis. It also performs will in all functional and non-functional requirements. The Automatic Timetabling MSSUGA has archive the expected outcome which stated in Chapter 1 introduction.

Chapter 8: System Evaluation

8.1 Problems Encountered and Solutions

Some problem was encounter during the process of developing system. Those occurring problems such as analysis information, system implementation, system testing and else. Each of those has been solved assist by lecturers, friends and searching information through books or internet services. Those problem will be detail explain in later section of this chapter.

8.1.1 Analysis Information

At the beginning, the author was lack of knowledge about scheduling school timetable using genetic algorithm methods. It is very hard to analysis the information gather from journals, articles, books, and internet. The author doesn't know which information needed or important and which information is not because genetic algorithm method is something new in author knowledge. After getting the help from supervisor and discuss with friend, the author have some brief idea to emphasis and analysis the information.

8.1.2 Lack of Knowledge in Using the Programming Language

The system is developing using Java programming language. Because lack of knowledge in using java, the author facing the problem during system implementation. The author was study many java programming books and searching examples through

the internet and it assist author to create the user interface using coding without assisting drag and drop workspace provide by the java programming software.

8.1.3 Inexperience in Scheduling School Timetable

Since the author doesn't have the experience in scheduling the school timetable. It let the author hardly design the output result which the user wanted. After analysis the timetables collect from students, the author have get the idea to showing the data output result.

8.2 System Strengths

The Automatic timetabling MSSUGA version 1.00 strengths were discussed in the later section.

8.2.1 User Friendly Interface

The system have friendly interface which let user easy to use. Those interface is quiet simple and provide predefine mode to let user define the data for future use. The predefine mode only have to define one times if and only if not data information wanted to change. User only needs to manage the schedule mode and special feature mode only for process the output result.

8.2.2 Fast Respond Time

The system produces the result within a short period. If the population is less than 10000 individuals, the respond time is just few seconds only. The respond time will be increased while individual's in population is increasing. Increasing large amount of individual's in population make respond time increased little only.

8.2.3 Data Saving

The system provide services for saving data to *.dat file format. This will let user access back the result generates from the previous time. User also can use back the data to generate new optimum timetable.

8.2.4 Message Prompt

The system provides the message prompt up to let user know the information and error message. When user wants to add and remove the data, the system will prompt up the message to let user confirm the action. When user have key in wrong information, the alert message will prompt up to let user know.

8.2.5 Reliable

The system was reliable because it will produce an output result that meets all the constraints of the timetable and free conflict timetable. Output produced in the respect time.

8.3 System Limitation

The Automatic timetabling MSSUGA version 1.00 strengths were discussed in the later section.

8.3.1 Natural Language Used

Only English version is provide in this Automatic timetabling MSSUGA version 1.00. User might be not familiar with English, so don't know how to manage the system.

8.3.2 Secondary School

Not all the school was suitable to use this system because some of the school timetable problem was difference with the system timetabling problem defined. Only suitable used in the school that fulfill the functional and non-functional requirement same as the system defined. There might be some special feature is not perform in this system. Furthermore, the school must provide enough teachers and, classrooms to let the system generate the timetables. If not, the process wouldn't be run.

8.4 Future Enhancement

This section was discussed the enhancement for the system in future time. Later section will discuss what can be enhancing to the system in the future.

8.4.1 Increase the constraints

There might be a lot of constraints haven't implemented in the system. For future, the system will enhanced with more constraints. This will let more secondary school can

use this system to scheduling timetables. This was an idea to group all the secondary school in Malaysia can use the same system to schedule the timetables.

8.4.2 Adding More Feature

Automatic timetabling MSSUGA tool needs a printing feature to print out the timetables. It might need to implement a report module to print the relevant timetable. Other needed feature have to enhance the system and let the system functionality increased.

8.5 Conclusion

This chapter were discussed the problem encounter during the development process. It also discussed about the system strength and system limitation for Automatic timetabling MSSUGA version 1.00 tools. In the last section was discussed the future enhancement for this system.

References

Edvart Ellingsen, and Manuel Penaloza. A genetic algorithm approach for finding a good course schedule. http://www.css.edu/depts/cis/mics_2003/MICS2003_Papers/**Ellingsen**.PDF

Genetic algorithm. *http://en.wikipedia.org/wiki/Genetic_algorithm.* Last updated: 28 September, 2005. Wikipedia. (Accessed Date: 6 October, 2005)

Goldberg, David (1989). Genetic Algorithms in Search, Optimization and Machine Learning. Addison Wesley Longman, Inc.

Home News . *http://www.utusan.com.my/utusan/content.asp?y=2004&dt=0709&pub=Utusan_Express&sec=Home_News&pg=hn_02.htm.* Last updated: 8 July, 2005. Utusan Online. (Accessed Date: 2 September, 2005).

Lee, Ho Sung C. (2000). Timetabling Highly Constrained System via Genetic Algorithms. Masters Thesis. University of Philippines, Diliman, Quezon City

Liviu Lalescu, and Costin Badica. An evolutionary approach for school timetabling. http://www.software.ucv.ro/~badica_costin/cercetare/papers/cscs14.pdf

Ribeiro, F., Luiz, A.N.L. (2000). *A Constructive Evolutionary Approach to School Timetabling.* 1-10. http://www.lac.inpe.br/~lorena/geraldo/CGA-timet.pdf

Rooij, A.J.C.V., Jain, L.C., Johnson, R.P. (1998). *Neural Network Training using Genetic Algorithms*. 1st ed. World Scientific Publishers.

Tripathy, A.: School timetabling : A case in large binary integer linear programming. Management Science, 30 (12) (1984) 1473-1489

Bibliography

Deitel, H.M., Deitel, P.J. (1999). *Java: How to Program*. 3rd ed. Prentice Hall

Edvart Ellingsen, and Manuel Penaloza. A genetic algorithm approach for finding a good course schedule. http://www.css.edu/depts/cis/mics_2003/MICS2003_Papers/**Ellingsen**.PDF

Genetic algorithm. *http://en.wikipedia.org/wiki/Genetic_algorithm.* Last updated: 28 September, 2005. Wikipedia. (Accessed Date: 6 October, 2005)

Goldberg, David (1989). Genetic Algorithms in Search, Optimization and Machine Learning. Addison Wesley Longman, Inc.

Home News. *http://www.utusan.com.my/utusan/content.asp?y=2004&dt=0709&pub=Utusan_Express&sec=Home_News&pg=hn_02.htm.* Last updated: 8 July, 2005. Utusan Online. (Accessed Date: 2 September, 2005).

Hubbard, J.R. (1999). *Programming with Java*. International ed. New York, McGraw-Hill Publishers.

Kenneth, E.K. (1994). *Advance in Genetic Programming*. 1st ed. MIT Press Cambridge

Kenneth, E.K., Julie, E.K (1999). System Analysis and Design. 4th ed. Prentice Hall

Lee, Ho Sung C. (2000). Timetabling Highly Constrained System via Genetic Algorithms. Masters Thesis. University of Philippines, Diliman, Quezon City

Liviu Lalescu, and Costin Badica. An evolutionary approach for school timetabling. http://www.software.ucv.ro/~badica_costin/cercetare/papers/cscs14.pdf

Ribeiro, F., Luiz, A.N.L. (2000). *A Constructive Evolutionary Approach to School Timetabling*. 1-10.

Rooij, A.J.C.V., Jain, L.C., Johnson, R.P. (1998). *Neural Network Training using Genetic Algorithms*. 1st ed. World Scientific Publishers.

Tripathy, A.: School timetabling : A case in large binary integer linear programming. Management Science, 30 (12) (1984) 1473-1489

Watt, D.A., Brown D.F. (2001). *Java Collection: An Introduction to Abstract Data Type, Data Structures and Algorithms*. John Wiley & Son Publisher.

http://www.java2s.com/Code/Java/CatalogJava.htm

http://www.planetsourcecode.com/

Appendix I - User Manual

Introduction

Automatic Timetabling tool MSSUGA is a tool used to schedule timetable in Malaysian secondary school. This program was applying genetic algorithm method to generate optimum solution. This manual was creating to assist user to manage the program easily.

How to start a new project / existing project

Figure A1.1: *Main page interface*

File > New Project

This option was used to create a new project.

File > Open Existing Project

This option was used to open an existing project which previous created.

File > Close

This option was used to terminate or close a program.

Help > About Program

This option was used to show brief detail about this program.

How to use the predefine teacher mode

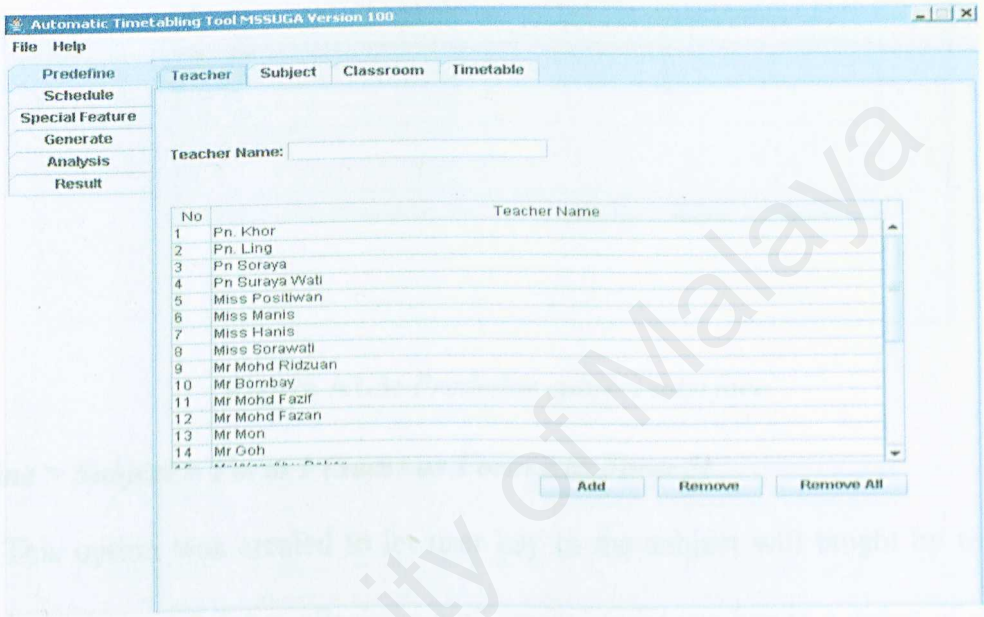


Figure A1.2: *Predefine teacher interface*

Predefine > Teacher

This option was created to let user key in the teacher name who involve in teaching.

- "Teacher Name" - Enter full teacher name in the box.
- "Add" button - Add new item which filled in Teacher Name box into table.
- "Remove" button - Select an item in table first and delete the item after clicking Remove" button.
- "Remove All" button - Remove all item in table after clicking this button.

How to use the predefine subject mode

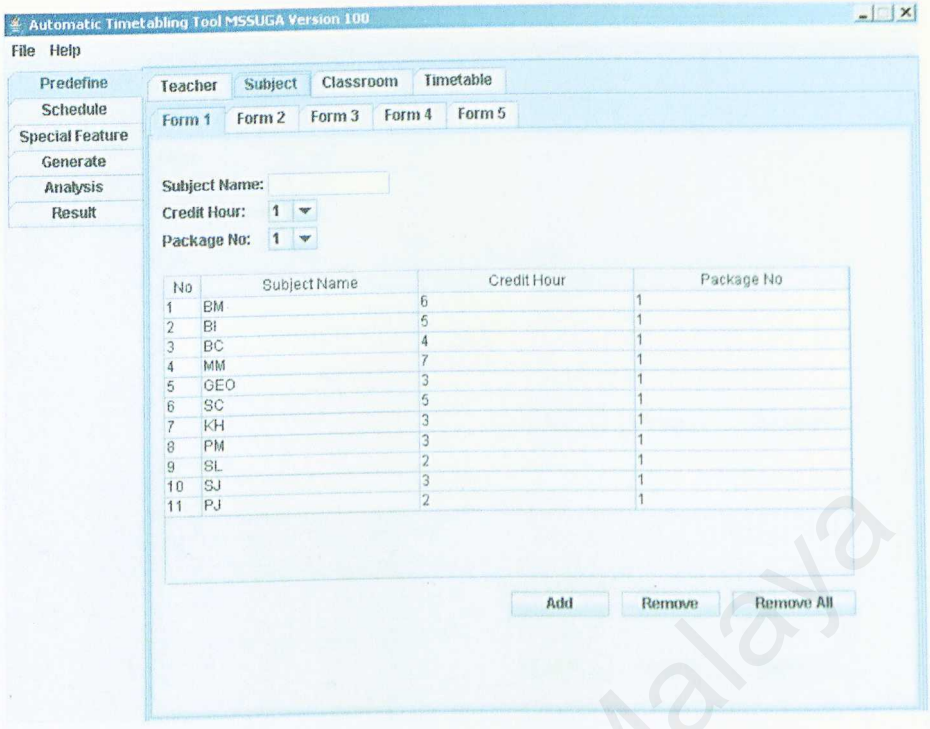


Figure A1.3: *Predefine subject interface*

Predefine > Subject > Form 1 (Same as Form 2 to Form 5)

This option was created to let user key in the subject will taught by teacher in school depend on each form. "Package No" created to let use key in a group of subject if one form exist more the one package.

- "Subject Name" – Enter subject name in the box.
- "Credit Hour" – Choose how many periods will teach for each subject. (1-9)
- "Package No" – Choose one of the value to represent package of subject. (1-9)
- "Add" button - Add new item which filled in Teacher Name box into table.
- "Remove" button - Select an item in table first and delete the item after clicking Remove" button.
- "Remove All" button - Remove all item in table after clicking this button.

How to use the predefine classroom mode

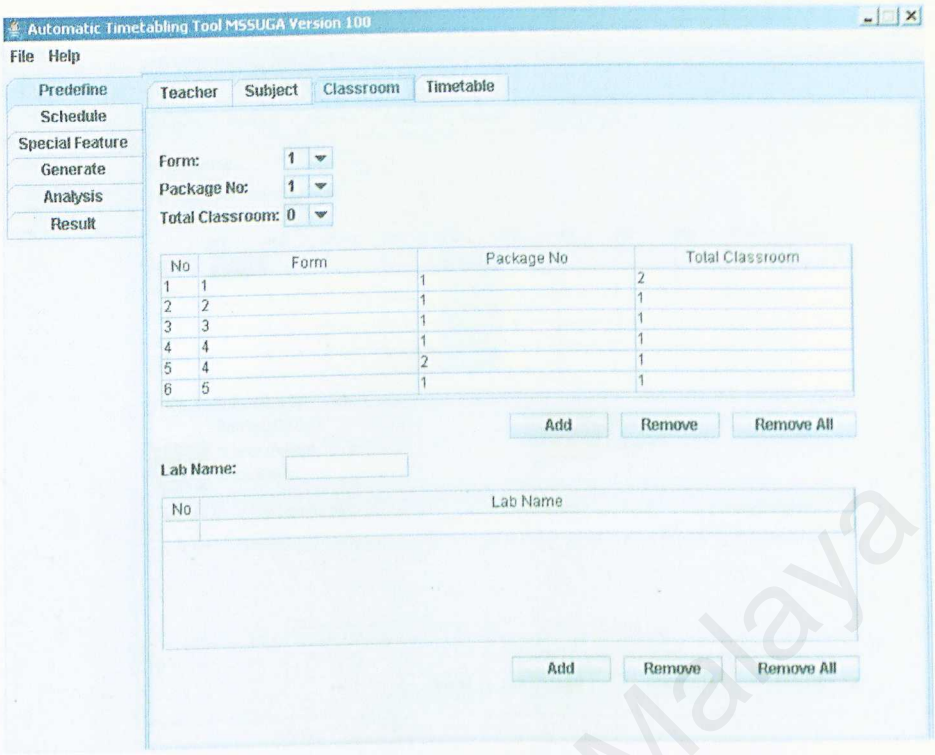


Figure A1.4: *Predefine classroom interface*

Predefine > Classroom

This option was created to let used to key in the classroom and lab will use for the school.

- "Form" – Choose which form classroom you want. (1-5)
- "Package No: - Choose one of the values to represent package of subject. (1-9)
- "Total Classroom" – Choose how many classrooms used for each form and each package. (0-20)
- "Lab Name" - Enter lab name in the box.
- "Add" button - Add new item which filled in Teacher Name box into table.
- "Remove" button - Select an item in table first and delete the item after clicking Remove" button.
- "Remove All" button - Remove all item in table after clicking this button.

How to use the predefine timetable mode

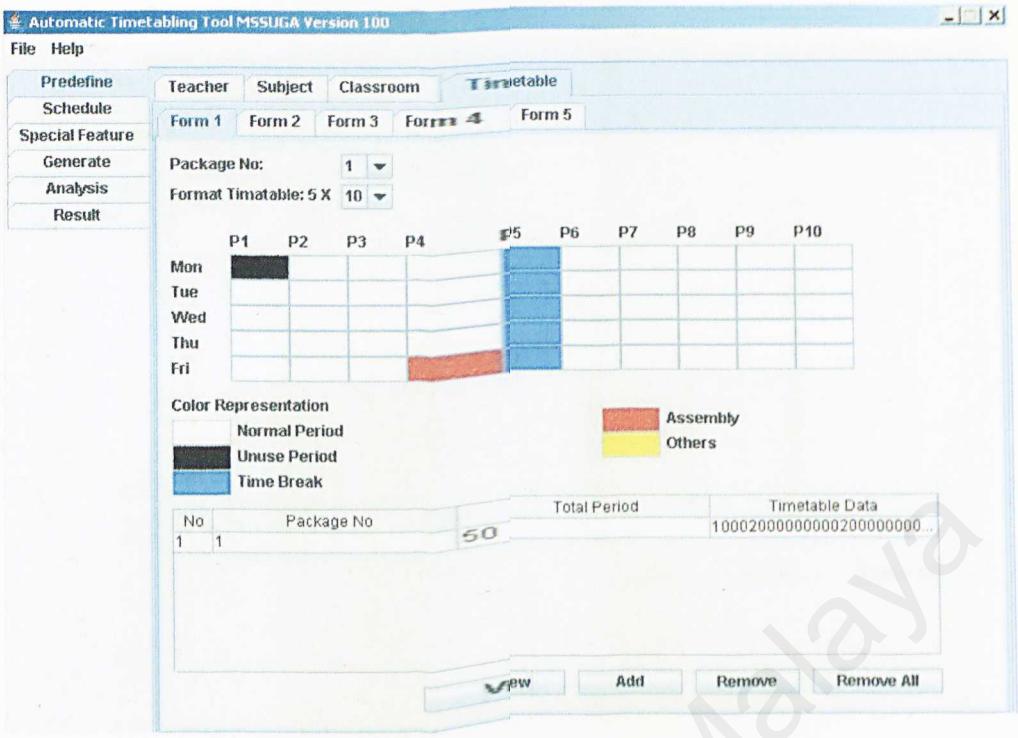


Figure A1.5: *Predefine timetable interface*

Predefine > Timetable > Form 1 (Same as Form 2 to Form 5)

This option was created to let user key the timetable format for each form and each format.

- "Package No" - Choose one of the values to represent package of subject. (1-9)
- Format Timetable – Choose the format of timetable. The value chosen depend on maximum period in each day. (10 – 13)
- "Add" button - Add new item which filled in Teacher Name box into table.
- "Remove" button - Select an item in table first and delete the item after clicking Remove" button.
- "Remove All" button - Remove all item in table after clicking this button.

How to use the predefine timetable mode

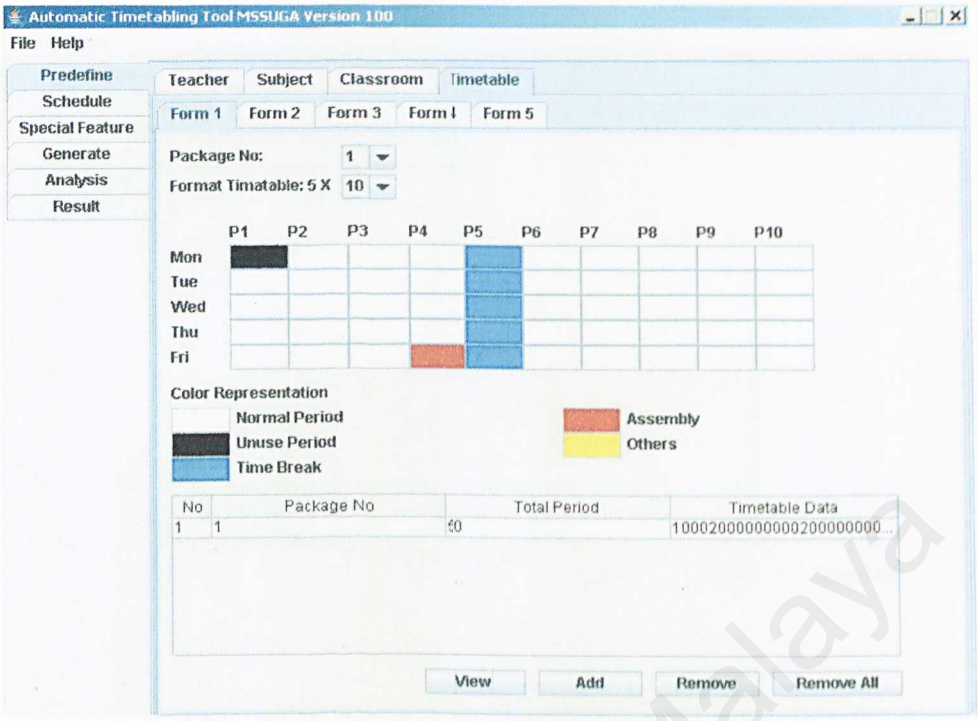


Figure A1.5: *Predefine timetable interface*

Predefine > Timetable > Form 1 (Same as Form 2 to Form 5)

This option was created to let user key the timetable format for each form and each format.

- "Package No" - Choose one of the values to represent package of subject. (1-9)
- Format Timetable – Choose the format of timetable. The value chosen depend on maximum period in each day. (10 – 13)
- "Add" button - Add new item which filled in Teacher Name box into table.
- "Remove" button - Select an item in table first and delete the item after clicking Remove" button.
- "Remove All" button - Remove all item in table after clicking this button.

How to use the schedule mode

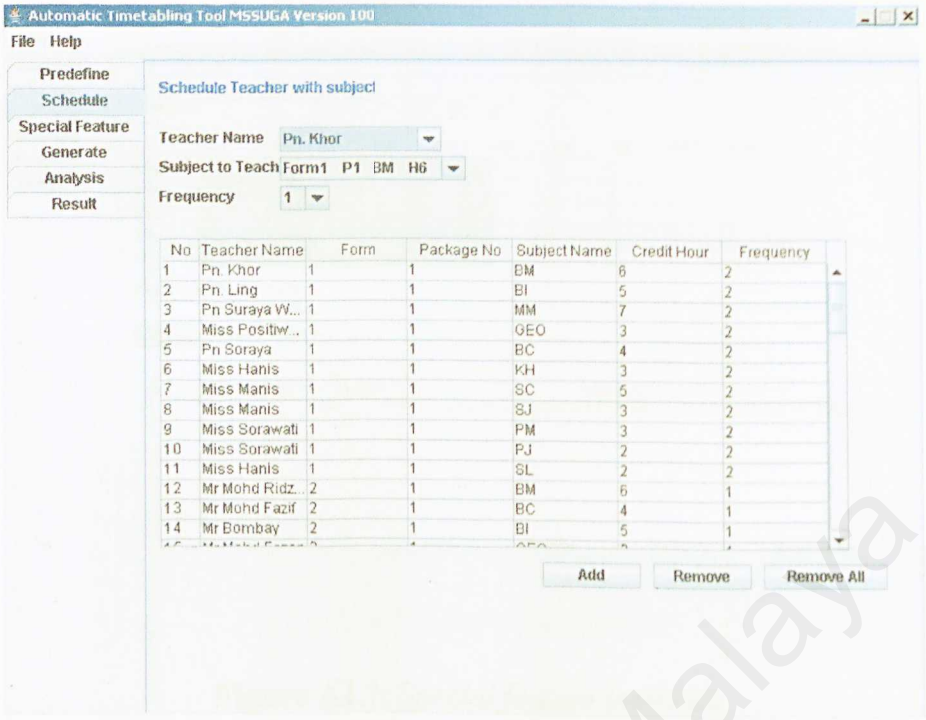


Figure A1.6: *Schedule interface*

Schedule

This option was created to let user key in which teacher teach in which package of subject and how many class he/she will teach.

- "Teacher Name" – Choose one of the teacher name where define in (Predefine > Teacher)
- "Subject to teach" – Choose one subject the teacher will teach.
- "Frequency" – Choose how many class the teacher will teach for this subject.
- "Add" button - Add new item which filled in Teacher Name box into table.
- "Remove" button - Select an item in table first and delete the item after clicking Remove" button.
- "Remove All" button - Remove all item in table after clicking this button.

How to use the special feature mode

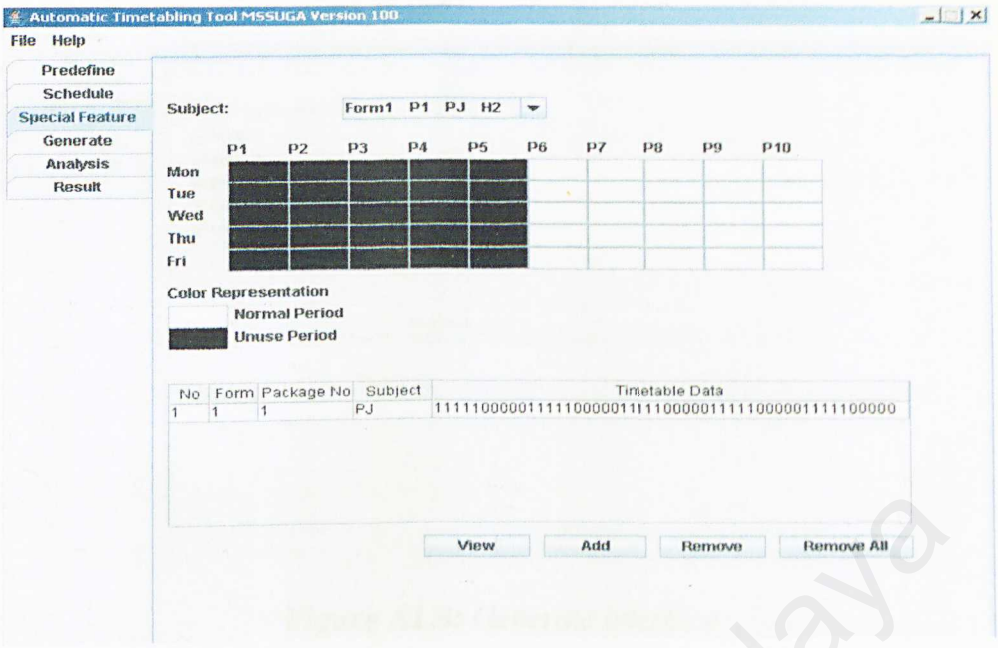


Figure A1.7: *Special feature interface*

Special Feature

This option was created to let user key in which subject, form and package will be limit the period used in the timetable.

- "Subject" - Choose one subject will limit its period.
- "Editing Table" – Click to the slot you want to modify.
- "Add" button - Add new item which filled in Teacher Name box into table.
- "Remove" button - Select an item in table first and delete the item after clicking Remove" button.
- "Remove All" button - Remove all item in table after clicking this button.

How to use the generate mode

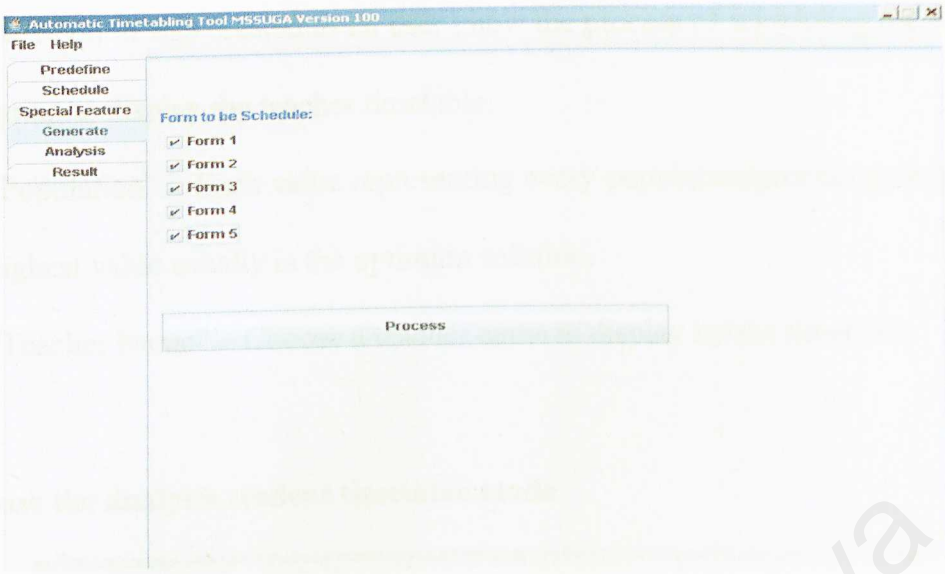


Figure A1.8: *Generate interface*

Generate

This option was created to let user choose which form want to process and schedule the school timetable. Choose which form you wanted.

"Process" button - Process the input data and schedule the timetable.

How to use the analysis teacher timetable mode

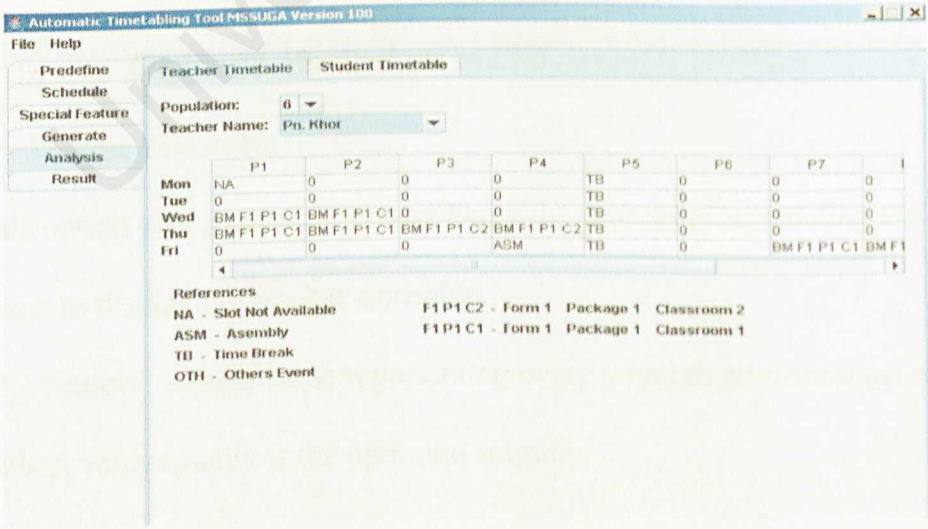


Figure A1.9: *Analysis teacher timetable interface*

Analysis > Teacher Timetable

This option was created to let user know the process of scheduling the timetable. Here is going to display the teacher timetable.

- "Population" – Each value representing every population/process generated. The highest value usually is the optimum solution.
- "Teacher Name" – Choose a teacher name to display he/she timetable.

How to use the analysis student timetable mode

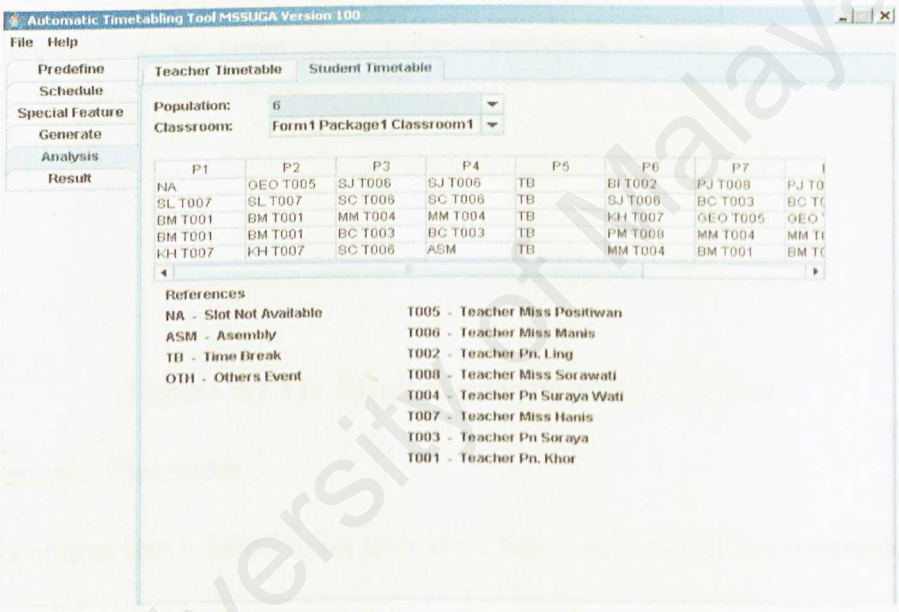


Figure A1.10: *Analysis student timetable interface*

Analysis > Student Timetable

This option was created to let user know the process of scheduling the timetable. Here is going to display the teacher timetable.

- "Population" – Each value representing every population/process generated. The highest value usually is the optimum solution.

- "Classroom" – Choose a classroom where student located and show the display of timetable.

How to use the result teacher timetable mode

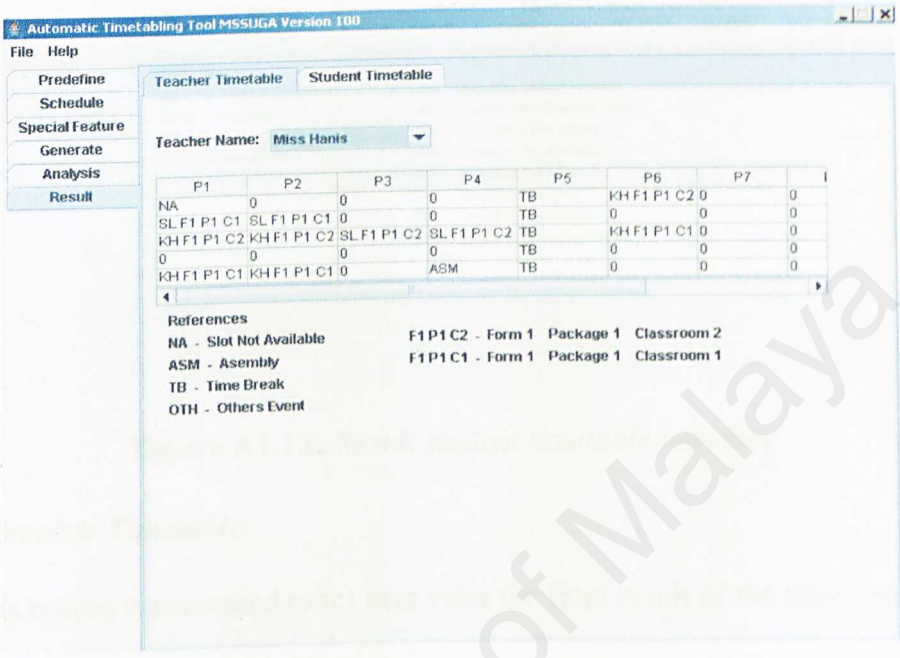


Figure A1.11: *Result teacher timetable interface*

Result > Teacher Timetable

This option was created to let user view the final result of the timetable.

- Teacher Name" – Choose a teacher name to display he/she timetable.

How to use the result student timetable mode

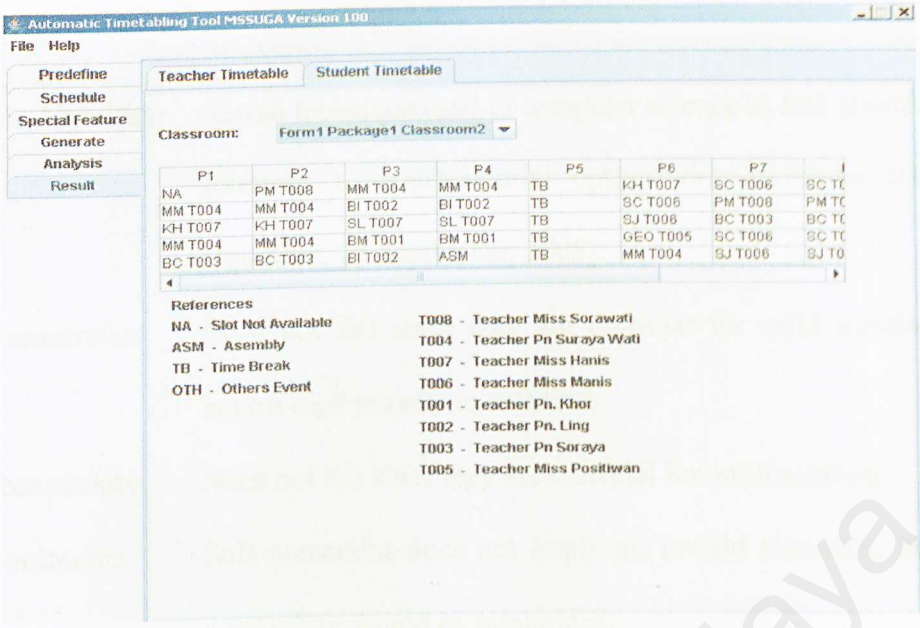


Figure A1.12: *Result student timetable interface*

Result > Teacher Timetable

This option was created to let user view the final result of the timetable.

- "Classroom" – Choose a classroom where student located and show the display of timetable.

Appendix II – Glossary

Genetic Algorithm	Search technique used in computer science to find approximate solutions to combinatorial optimization problems [Genetic algorithm, 28 September, 2005].
Fixed constraints	Must not fail since they are essential for valid solution and have a high priority to soft first.
Hard constraints	Must not fail since they are essential for valid solution.
Soft constraints	Soft constraint does not imply an invalid timetable, but soft constraints should be minimized.
Selection	Selects two parents form the population for mating.
Crossover	Individuals of each child are formed as a mixture of the individual's of the parents.
Mutation	Individual in the population undergoes a random mutation.

Appendix III – Current Program Genetic Algorithms Part Source Code

```
//Class Name: clsSchool.java
//Written by: Ku Chin Soon
import java.io.*;
import javax.swing.*;
import javax.swing.event.*;
import java.util.Random;

class clsSchool{
        clsForm [] clsF;
        private String [] sListSchool; // T001S11601C101 string
        private String [] sListSchool1;
        private String [] sGoodListSchool;
        private String [] sBadListSchool;
        private int [][] CheckTP;
        private int iTotalTeacher;
        private int [] iTotalClassroom;
        private int [] iTotalPeriod;
        private int [] iTotalSubject;
        private String [] iDePeriod;
        private int iForm;
        private int [] iValueFit;
        private Random rndNumber;
        private int irndValue,irndValue1;
        private int [] chkForm, chkPackage;
        private String [][] sListNumberSubject;
        private String [][] sListSpecialFeature1;
        private JOptionPane msgbox;
        clsCollectionPopulation [] clsColPop, clsColPopCopy;

        public clsSchool(){}

        public void InitializeClass(){
                sListSchool1 = new String [1];
                sGoodListSchool= new String [1];
                sBadListSchool= new String [1];
                CheckTP = new int [iTotalTeacher][60];
                setCheckTP();
        }

        public void InitializeForm(int iIndex){
                int i;
                clsF = new clsForm [iIndex];
                for(i=0;i<iIndex;i++){
                        clsF[i] = new clsForm();
                }
        }

        public void InitializeIvalueFit(){
                int i;
                iValueFit = new int [sListSchool1.length];
                for(i=0;i<sListSchool1.length;i++){
                        iValueFit[i]=0;
                }
        }
```

```
	public void chooseFormX(int iIndex, int Form, int PackageNo, int Classroom, int Period, int
Subject,String iDePeriod){
		int i, iPackageNo;
		clsF[iIndex].InitializeClass(Classroom,Period,Subject, iDePeriod);
		for(i=0;i<sListSchool.length;i++){
			iForm = Integer.parseInt(sListSchool[i].substring(5,6));
			iPackageNo = Integer.parseInt(sListSchool[i].substring(6,7));
			if(iForm==Form && iPackageNo==PackageNo){
				clsF[iIndex].getItem(sListSchool[i]);
			}
		}
		clsF[iIndex].setClassroomtoList();
		clsF[iIndex].setPeriodtoList();
		clsF[iIndex].getListItem();
		clsF[iIndex].resetCheckCS();
	}

	private void setCheckTP(){
		int i,j;
		for(i=0;i<iTotalTeacher;i++){
			for(j=0;j<60;j++){
				CheckTP[i][j]=0;
			}
		}
	}

	public void getListItem(){
		int i,j;
		for(j=0;j<clsF.length;j++){
			for(i=0;i<clsF[j].passListItem().length;i++){
				if(clsF[j].passListItem()[i]!=null){
					getItem(clsF[j].passListItem()[i]);
				}
			}
		}
	}

	public void getItem(String sItem){
		if(sListSchool1[0]!=null){
			redimArray();
			sListSchool1[sListSchool1.length-1] = sItem;
		}
		else{
			sListSchool1[0]=sItem;
		}
	}

	public void redimArray(){
		String [] sTempListArray;
		int i;
		sTempListArray = new String [sListSchool1.length];
		sTempListArray = sListSchool1;

		sListSchool1 = new String [sListSchool1.length+1];
		for(i=0;i<sTempListArray.length;i++){
			sListSchool1[i] = sTempListArray[i];
		}
	}

	public void redimArray1(){
```

```
		String [] sTempListArray;
		int i;
		sTempListArray = new String [sGoodListSchool.length];
		sTempListArray = sGoodListSchool;

		sGoodListSchool = new String [sGoodListSchool.length+1];
		for(i=0;i<sTempListArray.length;i++){
			sGoodListSchool[i] = sTempListArray[i];
		}
	}

	public void redimArray2(){
		String [] sTempListArray;
		int i;
		sTempListArray = new String [sBadListSchool.length];
		sTempListArray = sBadListSchool;

		sBadListSchool = new String [sBadListSchool.length+1];
		for(i=0;i<sTempListArray.length;i++){
			sBadListSchool[i] = sTempListArray[i];
		}
	}

	public void getGoodList(String sItem){
		if(sGoodListSchool[0]!=null){
			redimArray1();
			sGoodListSchool[sGoodListSchool.length-1] = sItem;
		}
		else{
			sGoodListSchool[0]=sItem;
		}
	}

	public void getBadList(String sItem){
		if(sBadListSchool[0]!=null){
			redimArray2();
			sBadListSchool[sBadListSchool.length-1] = sItem;
		}
		else{
			sBadListSchool[0]=sItem;
		}
	}

	public void CheckingProcess(){
		int i,j;
		boolean bTempFlagT1,bTempFlagT2,bTempFlagT3, bTempFlagT4;
		bTempFlagT1=true;
		bTempFlagT2=true;
		bTempFlagT3=true;
		bTempFlagT4=true;
		int icounterpopulation=0;
		clsColPop = new clsCollectionPopulation [1];
		do{
			sGoodListSchool= new String [1];
			sBadListSchool= new String [1];
			for(i=0;i<sListSchool1.length;i++){
				if(iValueFit[i]<500){
					bTempFlagT1 =
CheckingSpecialFeature1(sListSchool1[i]);
					if(bTempFlagT1){
```

```
bTempFlagT2=CheckingCheckTP(sListSchool1[i]);
                                        }
                                        if(bTempFlagT1 && bTempFlagT2){
bTempFlagT3=CheckingCheckCP(sListSchool1[i]);
        }
        if(bTempFlagT1 && bTempFlagT2 && bTempFlagT3){
                bTempFlagT4=CheckingCheckCS(sListSchool1[i]);
        }
        if(bTempFlagT1 && bTempFlagT2 && bTempFlagT3 &&bTempFlagT4){
                setCheckTP021(sListSchool1[i]);
                setCheckCS021(sListSchool1[i]);
                setCheckCP021(sListSchool1[i]);
                iValueFit[i] = 800;
                getGoodList(sListSchool1[i]);
                }
        }
        else{
                getGoodList(sListSchool1[i]);
        }
}
        if(clsColPop[0]!=null){
                clsColPopCopy = new clsCollectionPopulation [clsColPop.length];
                        clsColPopCopy = clsColPop;
                        clsColPop = new clsCollectionPopulation [clsColPop.length+1];

                        for(j=0;j<clsColPopCopy.length;j++){
                                clsColPop[j] = clsColPopCopy[j];
                        }
                        clsColPop[j] = new clsCollectionPopulation();
                        clsColPop[j].setData(sGoodListSchool);
                }
                else{
                        clsColPop[0] = new clsCollectionPopulation();
                        clsColPop[0].setData(sGoodListSchool);
                }
                CheckingClassFull();
                for(i=0;i<sListSchool1.length;i++){
                        if(iValueFit[i]>0 && iValueFit[i]<1000)
                        iValueFit[i] -= 1;

                if(iValueFit[i]==499){
                                setCheckTP120(sListSchool1[i]);
                                setCheckCS120(sListSchool1[i]);
                                setCheckCP120(sListSchool1[i]);
                }

                if(iValueFit[i]<500){
                        getBadList(sListSchool1[i]);
                }
           }

           for(i=0;i<sListSchool1.length;i++){
                if(iValueFit[i]<500){
                        performCrossover(i);
                }
                }
                icounterpopulation++;
          }while(sListSchool1.length!=sGoodListSchool.length);
```

```
                try
                        {
                                savaFileMode();
                        }
                        catch (java.io.IOException exp)
                        {
                                System.out.println("Failed");
                        }
                msgbox.showMessageDialog(null, "Process Completed", "alert",
JOptionPane.INFORMATION_MESSAGE);
        }

        private void performCrossover(int badIndexChromo){
                int i,j,k;
                int iTempForm,iTempForm1;
                int iTempClassroom,iTempClassroom1;
                int iTempPackage,iTempPackage1;
                int iTempSubjectCH,iTempSubjectCH1;
                int iSub,ilen,iCounter=0;
                String sTemp;
                String [] partition1;
                String [] partition2;
                String iTempPeriod,iTempPeriod1;
                partition1 = new String [5];
                partition2 = new String [5];
                boolean bFlagT,bFlagT1;
                rndNumber =new Random();
                        bFlagT=true;
                        do{
        irndValue=Math.round(150*rndNumber.nextFloat())%sListSchool1.length;
        iTempForm = Integer.parseInt(sListSchool1[irndValue].substring(5,6));
        iTempClassroom = Integer.parseInt(sListSchool1[irndValue].substring(12,14));
        iTempPackage = Integer.parseInt(sListSchool1[irndValue].substring(6,7));
        iTempForm1 = Integer.parseInt(sListSchool1[badIndexChromo].substring(5,6));
        iTempClassroom1 = Integer.parseInt(sListSchool1[badIndexChromo].substring(12,14));
        iTempPackage1 = Integer.parseInt(sListSchool1[badIndexChromo].substring(6,7));
        if(iTempForm==iTempForm1 && iTempClassroom==iTempClassroom1 &&
iTempPackage==iTempPackage1){
                bFlagT=false;
                }
                        }while(bFlagT);
                        iTempSubjectCH =
Integer.parseInt(sListSchool1[irndValue].substring(7,8));
                        iTempSubjectCH1 =
Integer.parseInt(sListSchool1[badIndexChromo].substring(7,8));
                        iTempPeriod =sListSchool1[irndValue].substring(15);
                        iTempPeriod1 =sListSchool1[badIndexChromo].substring(15);
                        ilen=iTempPeriod.length()/5;
                        for(k=0;k<5;k++){
                                partition1[k]=iTempPeriod.substring(k*ilen,((k+1)*ilen));
                                partition2[k]=iTempPeriod1.substring(k*ilen,((k+1)*ilen));
                        }
                        iValueFit[irndValue]=499;
                        setCheckTP120(sListSchool1[irndValue]);
                        setCheckCS120(sListSchool1[irndValue]);
                        setCheckCP120(sListSchool1[irndValue]);
                        if(iTempSubjectCH==iTempSubjectCH1){
sListSchool1[irndValue]=sListSchool1[irndValue].substring(0,15)+iTempPeriod1;
sListSchool1[badIndexChromo]=sListSchool1[badIndexChromo].substring(0,15)+iTempPeriod;
                        }
```

```
                else{
                        bFlagT1=true;
                        iCounter=0;
                        do{
                                iTempPeriod="";
                                iTempPeriod1="";
                                for(k=0;k<5;k++){
        irndValue1=Math.round(150*rndNumber.nextFloat())%2;
                                        if(irndValue1==0){
                                                iTempPeriod+=partition1[k];
                                                iTempPeriod1+=partition2[k];
                                        }
                                        else if(irndValue1==1){
                                                iTempPeriod+=partition2[k];
                                                iTempPeriod1+=partition1[k];
                                        }
                                }
                                if(countBitString(iTempPeriod)==iTempSubjectCH &&
countBitString(iTempPeriod1)==iTempSubjectCH1){
                                        bFlagT1=false;
                                }
                        }while(bFlagT1);
sListSchool1[irndValue]=sListSchool1[irndValue].substring(0,15)+iTempPeriod;sListSchool1[badInd
exChromo]=sListSchool1[badIndexChromo].substring(0,15)+iTempPeriod1;
                }
        }

        private int countBitString(String s){
                int i;
                int iSum=0;
                for(i=0;i<s.length();i++){
                        iSum +=Integer.parseInt(s.substring(i,i+1));
                }
                return iSum;
        }

        private int substract(int x, int y){
                if(x<y)
                        return y-x;
                else
                        return x-y;
        }

        private boolean CheckingSpecialFeature1(String Chromo){
                boolean bTempFlagT, bTempFlagT1;
                String sTempSubjectName ="", sTempPeriod;
                int i, j;
                for(i=0;i<sListNumberSubject.length;i++){
                        if(sListNumberSubject[i][0].compareTo(Chromo.substring(5,6))==0 &&
                                sListNumberSubject[i][3].compareTo(Chromo.substring(6,7))==0
&&
        sListNumberSubject[i][4].compareTo(Integer.toString(Integer.parseInt(Chromo.substring(8,
10))))==0 ){
                                sTempSubjectName = sListNumberSubject[i][1];
                        }
                }
                bTempFlagT=true;
                sTempPeriod=Chromo.substring(15);
                if(sListSpecialFeature1[0][0]!=null){
                        for(i=0;i<sListSpecialFeature1.length;i++){
```

```
if(sListSpecialFeature1[i][1].compareTo(Chromo.substring(5,6))==0 &&
sListSpecialFeature1[i][2].compareTo(Chromo.substring(6,7))==0 &&
sListSpecialFeature1[i][3].compareTo(sTempSubjectName)==0){
        for(j=0;j<sTempPeriod.length();j++){

if(Integer.parseInt(sTempPeriod.substring(j,j+1))==1){
if(Integer.parseInt(sListSpecialFeature1[i][4].substring(j,j+1))==1){
                                                        bTempFlagT=false;
                                                }
                                        }
                                }
                        }
                }
        }
        return bTempFlagT;
}

private boolean CheckingCheckTP(String Chromo){
        int i;
        int iTempTeacher;
        String sTempPeriod;
        boolean bTempFlagT;
        bTempFlagT=true;
        iTempTeacher=Integer.parseInt(Chromo.substring(1,4));
        sTempPeriod=Chromo.substring(15);
        for(i=0;i<sTempPeriod.length();i++){
                if(Integer.parseInt(sTempPeriod.substring(i,i+1))==1){
                        if(CheckTP[iTempTeacher-1][i]==1){
                                bTempFlagT=false;
                        }
                }
        }
        return bTempFlagT;
}

private void setCheckTP021(String Chromo){
        int i;
        int iTempTeacher;
        iTempTeacher=Integer.parseInt(Chromo.substring(1,4));
        String sTempPeriod;
        sTempPeriod = Chromo.substring(15);
        for(i=0;i<sTempPeriod.length();i++){
                if(Integer.parseInt(sTempPeriod.substring(i,i+1))==1){
                        CheckTP[iTempTeacher-1][i]=1;
                }
        }
}

private void setCheckTP120(String Chromo){
        int i;
        int iTempTeacher;
        iTempTeacher=Integer.parseInt(Chromo.substring(1,4));
        String sTempPeriod;
        sTempPeriod = Chromo.substring(15);
        for(i=0;i<sTempPeriod.length();i++){
                if(Integer.parseInt(sTempPeriod.substring(i,i+1))==1){
                        CheckTP[iTempTeacher-1][i]=0;
                }
        }
```

```
}

private boolean CheckingCheckCP(String Chromo){
        int iTempForm, iTempPackage, i;
        boolean bTempFlagT=true;
        iTempForm=Integer.parseInt(Chromo.substring(5,6));
        iTempPackage = Integer.parseInt(Chromo.substring(6,7));
        for(i=0;i<clsF.length;i++){
                if(chkForm[i] == iTempForm && chkPackage[i]==iTempPackage){
                        bTempFlagT = clsF[i].CheckingCheckCP(Chromo);
                }
        }
        return bTempFlagT;
}

private boolean CheckingCheckCS(String Chromo){
        int iTempForm, iTempPackage, i;
        boolean bTempFlagT=true;
        iTempForm=Integer.parseInt(Chromo.substring(5,6));
        iTempPackage = Integer.parseInt(Chromo.substring(6,7));
        for(i=0;i<clsF.length;i++){
                if(chkForm[i] == iTempForm && chkPackage[i]==iTempPackage){
                        bTempFlagT = clsF[i].CheckingCheckCS(Chromo);
                }
        }
        return bTempFlagT;
}

private void setCheckCP021(String Chromo){
        int iTempForm, iTempPackage, i;
        iTempForm=Integer.parseInt(Chromo.substring(5,6));
        iTempPackage = Integer.parseInt(Chromo.substring(6,7));
        for(i=0;i<clsF.length;i++){
                if(chkForm[i] == iTempForm && chkPackage[i]==iTempPackage){
                        clsF[i].setCheckCP021(Chromo);
                }
        }
}

private void setCheckCP120(String Chromo){
        int iTempForm, iTempPackage, i;
        iTempForm=Integer.parseInt(Chromo.substring(5,6));
        iTempPackage = Integer.parseInt(Chromo.substring(6,7));
        for(i=0;i<clsF.length;i++){
                if(chkForm[i] == iTempForm && chkPackage[i]==iTempPackage){
                        clsF[i].setCheckCP120(Chromo);
                }
        }
}

private void setCheckCS021(String Chromo){
        int iTempForm, iTempPackage, i;
        iTempForm=Integer.parseInt(Chromo.substring(5,6));
        iTempPackage = Integer.parseInt(Chromo.substring(6,7));
        for(i=0;i<clsF.length;i++){
                if(chkForm[i] == iTempForm && chkPackage[i]==iTempPackage){
                        clsF[i].setCheckCS021(Chromo);
                }
        }
```

```
}

private void setCheckCS120(String Chromo){
        int iTempForm, iTempPackage, i;
        iTempForm=Integer.parseInt(Chromo.substring(5,6));
        iTempPackage = Integer.parseInt(Chromo.substring(6,7));
        for(i=0;i<clsF.length;i++){
                if(chkForm[i] == iTempForm && chkPackage[i]==iTempPackage){
                        clsF[i].setCheckCS120(Chromo);
                }
        }

}

public void CheckingClassFull(){
        int i;
        for(i=0;i<clsF.length;i++){
                iValueFit=clsF[i].CheckingClassFull(sListSchool1,iValueFit,i);
        }
}

public void getsListSchool(String [] sList){
        sListSchool = new String [sList.length];
        sListSchool = sList;
}

public void getiDePeriod(String [] sList){
        iDePeriod = new String [sList.length];
        iDePeriod = sList;
}

public String [] getsListSchool1(){
        return sListSchool1;
}


public void getiTotalClassroom(int [] iNum){
        iTotalClassroom = new int [iNum.length];
        iTotalClassroom = iNum;
}

public void getiTotalSubject(int [] iNum){
        iTotalSubject = new int [iNum.length];
        iTotalSubject = iNum;
}

public void getiTotalPeriod(int [] iNum){
        iTotalPeriod = new int [iNum.length];
        iTotalPeriod = iNum;
}

public void getiTotalTeacher(int iNum){
        iTotalTeacher = iNum;
}

public void receiveValue(int [] x, int [] y){
        chkForm = new int [x.length];
        chkPackage = new int [y.length];
        chkForm = x;
        chkPackage = y;
```

```
        }

        public void savaFileMode() throws IOException{
        ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream("C:/Program
  Files/TimetableTool MSSUGA/temp/clsColPop.dat"));
                out.writeObject(clsColPop);
                out.close();
    }

    public void setsListNumberSubject(String [][] sList){
        sListNumberSubject = new String [sList.length][4];
        sListNumberSubject = sList;
    }

    public void setsListSpecialFeature1(String [][] sList){
        if(sList!=null){
                sListSpecialFeature1 = new String [sList.length][4];
                sListSpecialFeature1 = sList;
          }
          else
                sListSpecialFeature1 = new String [1][4];
    }
}


//Class Name: clsForm.java
//Written by: Ku Chin Soon
import java.util.Random;


class clsForm{

        private String [] sFormList;
        private int sFormList1;
        private int iTotalClassroom;
        private int iTotalPeriod;
        private int iTotalSubject;
        private int iSubject;
        private int iClasroom;
        private Random rndNumber;
        private int irndvalue;
        String iDePeriod;
        clsClassroom [] clsClass;

        public clsForm(){}

        public void InitializeClass(int iSumClassroom, int iSumPeriod, int iSumSubject, String
dePeriod){
                int i;
                sFormList = new String [1];

                iTotalClassroom = iSumClassroom;
                iTotalPeriod = iSumPeriod;
                iTotalSubject = iSumSubject;
                iDePeriod=dePeriod;
                setClsClassroom();

        }

        public void setClsClassroom(){
```

```
        int i;
        clsClass = new clsClassroom [iTotalClassroom];
        for(i=0;i<iTotalClassroom;i++){
            clsClass[i] = new clsClassroom();
        }

        for(i=0;i<iTotalClassroom;i++){
            clsClass[i].InitializeClass(iTotalSubject,iTotalPeriod,iDePeriod);
        }
    }

    /*Get chromo string one by one to sformlist*/
    public void getItem(String sItem){

        if(sFormList[0]!=null){
            redimArray();
            sFormList[sFormList.length-1] = sItem;
        }
        else{
            sFormList[0]=sItem;
        }

    }

    /*Redim element for array form list*/
    public void redimArray(){
        String [] sTempListArray;
        int i;
        sTempListArray = new String [sFormList.length];
        sTempListArray = sFormList;

        sFormList = new String [sFormList.length+1];
        for(i=0;i<sTempListArray.length;i++){
            sFormList[i] = sTempListArray[i];
        }
    }

    public void displayArray(){
        int i;
        for(i=0;i<sFormList.length;i++){
            System.out.println(sFormList[i]);
        }
    }

    public void setClassroomtoList(){
        int i;
        boolean bFlagT;
        rndNumber = new Random();
        for(i=0;i<sFormList.length;i++){
            bFlagT = true;
            do{
                iSubject = Integer.parseInt(sFormList[i].substring(8,10));

    irndvalue=Math.round(150*rndNumber.nextFloat())%iTotalClassroom;

                if(clsClass[irndvalue].booleanCheckCS(sFormList[i])){
                    if(irndvalue<10)
                        sFormList[i] = sFormList[i].substring(0,13) +
(irndvalue+1);
                    else
```

```
                                        sFormList[i] = sFormList[i].substring(0,12) +
(irndvalue+1);
                                        clsClass[irndvalue].setCheckCS021(sFormList[i]);
                                        clsClass[irndvalue].getItem(sFormList[i]);
                                        bFlagT = false;
                                }
                        }while(bFlagT);
                }

        }

        public void setPeriodtoList(){
                int i;
                for(i=0;i<iTotalClassroom;i++){
                        clsClass[i].setPeriodtoList();
                }
        }

        public void getListItem(){
                int i,j;
                sFormList = new String [1];

                for(i=0;i<iTotalClassroom;i++){
                        for(j=0;j<clsClass[i].passListItem().length;j++){
                                getItem(clsClass[i].passListItem()[j]);
                        }
                }

        }

        public String [] passListItem(){
                return sFormList;
        }

        public boolean CheckingCheckCP(String Chromo){
                int iTempClassroom;
                iTempClassroom = Integer.parseInt(Chromo.substring(12,14));
                return clsClass[iTempClassroom-1].CheckingCheckCP(Chromo);
        }

        public boolean CheckingCheckCS(String Chromo){
                int iTempClassroom;
                iTempClassroom = Integer.parseInt(Chromo.substring(12,14));
                return clsClass[iTempClassroom-1].booleanCheckCS(Chromo);
        }

        public void setCheckCP021(String Chromo){
                int iTempClassroom;
                iTempClassroom = Integer.parseInt(Chromo.substring(12,14));
                clsClass[iTempClassroom-1].setCheckCP021(Chromo);
        }

        public void setCheckCP120(String Chromo){
                int iTempClassroom;
                iTempClassroom = Integer.parseInt(Chromo.substring(12,14));
                clsClass[iTempClassroom-1].setCheckCP120(Chromo);
        }

        public void setCheckCS021(String Chromo){
                int iTempClassroom;
```

```
                iTempClassroom = Integer.parseInt(Chromo.substring(12,14));
                clsClass[iTempClassroom-1].setCheckCS021(Chromo);
        }

        public void setCheckCS120(String Chromo){
                int iTempClassroom;
                iTempClassroom = Integer.parseInt(Chromo.substring(12,14));
                clsClass[iTempClassroom-1].setCheckCS120(Chromo);
        }

        public void resetCheckCS(){
                int i;
                for(i=0;i<iTotalClassroom;i++){
                        clsClass[i].resetCheckCS();
                }
        }

        public int [] CheckingClassFull(String [] sList, int [] iValue,int iForm){
                int i,j;
                int iTempForm;
                int iTempClassroom;

                for(i=0;i<iTotalClassroom;i++){
                        if(clsClass[i].checkclassfull()){
                                for(j=0;j<sList.length;j++){
                                        iTempForm = Integer.parseInt(sList[j].substring(5,6));
                                        iTempClassroom =
Integer.parseInt(sList[j].substring(12,14));
                                        if(iTempForm-1==iForm && i==iTempClassroom-1){
                                                iValue[j] = 1000;
                                        }
                                }
                        }
                }
                return iValue;
        }
}


//Class Name: clsClassroom.java
//Written by: Ku Chin Soon
class clsClassroom{

        private String [] sClassRoomList;
        private int CheckCS[];
        private int CheckCP[];
        private int iTotalSubject;
        private int iTotalPeriod;
        private int [] iSubjectCH;
        String iDePeriod;
        clsPeriod clsPe;
        public clsClassroom(){}

        public void InitializeClass(int iSumSubject, int iSumPeriod,String dePeriod){
                sClassRoomList = new String [1];
                CheckCS = new int [iSumSubject];
                CheckCP = new int [iSumPeriod];
                iSubjectCH = new int [iSumSubject];

                iTotalSubject = iSumSubject;
```

```
        iTotalPeriod = iSumPeriod;

        iDePeriod=dePeriod;

        resetCheckCS();
        resetCheckCP();

}

/*Get chromo string one by one to sclassroomlist*/
public void getItem(String sItem){
        if(sClassRoomList[0]!=null){
                redimArray();
                sClassRoomList[sClassRoomList.length-1] = sItem;
        }
        else{
                sClassRoomList[0]=sItem;
        }
}

/*Redim element for array classroom list*/
public void redimArray(){
        String [] sTempListArray;
        int i;
        sTempListArray = new String [sClassRoomList.length];
        sTempListArray = sClassRoomList;

        sClassRoomList = new String [sClassRoomList.length+1];
        for(i=0;i<sTempListArray.length;i++){
                sClassRoomList[i] = sTempListArray[i];
        }
}

/*Reset CheckCS to 0 */
public void resetCheckCS(){
        int i;
        for(i=0;i<CheckCS.length;i++){
                CheckCS[i]=0;
        }
}

/*Reset CheckCP to 0 */
public void resetCheckCP(){
        int i;
        for(i=0;i<CheckCP.length;i++){
                CheckCP[i]=0;
        }
}

public boolean booleanCheckCS(String Chromo){
        int iTempSubject;
        iTempSubject = Integer.parseInt(Chromo.substring(8,10));
        if(CheckCS[iTempSubject-1]==0)
                return true;
        else
                return false;
}

public void setCheckCS021(String Chromo){
        int iTempSubject;
```

```
        iTempSubject = Integer.parseInt(Chromo.substring(8,10));
        CheckCS[iTempSubject-1]=1;
}

public void setCheckCS120(String Chromo){
        int iTempSubject;
        iTempSubject = Integer.parseInt(Chromo.substring(8,10));
        CheckCS[iTempSubject-1]=0;
}

public void display1(){
        int i;
        int x=0;
        for(i=0;i<CheckCS.length;i++){
                x+=CheckCS[i];
        }
}

public void setPeriodtoList(){
        int i;
        InitializePeriod();
        for(i=0;i<sClassRoomList.length;i++){
                sClassRoomList[i] = clsPe.pairPeriod(sClassRoomList[i]);
        }
}

private void collectCHperiod(){
        int i;
        int iTempSubjectCH;
        for(i=0;i<sClassRoomList.length;i++){
                iTempSubjectCH = Integer.parseInt(sClassRoomList[i].substring(7,8));
                iSubjectCH[i]=iTempSubjectCH;
        }

}

private void InitializePeriod(){
        int iTempSubjectCH;
        collectCHperiod();
        clsPe = new clsPeriod();
        clsPe.initialize(iTotalSubject,iSubjectCH,iTotalPeriod,iDePeriod);
}

public void displayArray(){
        int i;
        for(i=0;i<sClassRoomList.length;i++){
                System.out.println(sClassRoomList[i]);
        }
}

public String [] passListItem(){
        return sClassRoomList;
}

public boolean CheckingCheckCP(String Chromo){
        String sTempPeriod;
        int i;
        boolean bTempFlagT;
        bTempFlagT=true;
        sTempPeriod = Chromo.substring(15);
```

```
                for(i=0;i<sTempPeriod.length();i++){
                        if(Integer.parseInt(sTempPeriod.substring(i,i+1))==1){
                                if(CheckCP[i]==1){
                                        bTempFlagT=false;
                                }
                        }
                }
                return bTempFlagT;
        }

        public void setCheckCP021(String Chromo){
                String sTempPeriod;
                int i;
                sTempPeriod = Chromo.substring(15);
                for(i=0;i<sTempPeriod.length();i++){
                        if(Integer.parseInt(sTempPeriod.substring(i,i+1))==1){
                                CheckCP[i]=1;
                        }
                }
        }

        public void setCheckCP120(String Chromo){
                String sTempPeriod;
                int i;
                sTempPeriod = Chromo.substring(15);
                for(i=0;i<sTempPeriod.length();i++){
                        if(Integer.parseInt(sTempPeriod.substring(i,i+1))==1){
                                CheckCP[i]=0;
                        }
                }
        }

        public boolean checkclassfull(){
                int i;
                int sum=0;
                for(i=0;i<CheckCS.length;i++){
                        sum += CheckCS[i];
                }
                if(sum == iTotalSubject)
                        return true;
                else
                        return false;
        }
}


//Class Name: clsPeriod.java
//Written by: Ku Chin Soon
import java.awt.*;
import java.util.Random;

class clsPeriod {
        private int [] subjectCH;
        private int [] CopysubjectCH;
        private String [] sValue;
        private String [] chk1;
        private int [] bValueChk;
        private int [] bDay;
        private Random r;
        private int r1;
```

```
	private int iTotalPeriod;
	String iDePeriod;
	private String [] sListPeriod;

	public clsPeriod(){}

	public void initialize(int iTotalSubject, int [] iSubjectCH, int iSumPeriod, String
dePeriod){//int iTotalClassroom,
		int x,y;
		boolean bFlagT;
		sListPeriod = new String [iTotalSubject];
		subjectCH = iSubjectCH;
		iTotalPeriod=iSumPeriod;
		iDePeriod=dePeriod;
		clsPeriod1 t1= new clsPeriod1();
		t1.InitializeClass(iSubjectCH, iTotalPeriod, iDePeriod);
		t1.generateSubPeriod();
		chk1 = t1.getValue();
		bFlagT=false;
		do{
			bFlagT=generate();
		}while(!bFlagT);

		for(y=0;y<iTotalSubject;y++){
			sListPeriod[y]=sValue[y];
		}
	}

	public String pairPeriod(String Chromo){
		String sTemp="";
		int x;
		boolean bFlagT=true;
		int iSubjectCredit=Integer.parseInt(Chromo.substring(7,8));
		for(x=0;x<sListPeriod.length;x++){
			if(sListPeriod[x]!="" && bFlagT ){
		if(Integer.parseInt(sListPeriod[x].substring(1,2))==iSubjectCredit){
					sTemp=sListPeriod[x];
					sListPeriod[x]="";
					bFlagT=false;
				}
			}
		}
		sTemp =Chromo.substring(0,14) + "P" +sTemp.substring(2);
		return sTemp;
	}

	private boolean generate(){
		int i,j,c1=0;
		int iDay,iCh;
		boolean bFlagT=true;
		bDay = new int [5];
		r = new Random();
		bValueChk = new int [chk1.length];
		sValue = new String [subjectCH.length];
		CopysubjectCH = new int [subjectCH.length];
		for(i=0;i<subjectCH.length;i++)
			CopysubjectCH[i]=subjectCH[i];
		for(i=0;i<bValueChk.length;i++)
			bValueChk[i]=0;
		initsValue();
```

```
            for(i=0;i<CopysubjectCH.length;i++){
                        resetbDay();
                        do{
                                    r1 = Math.round(150*r.nextFloat())%chk1.length;
                                    iDay=Integer.parseInt(chk1[r1].substring(0,1));
                                    iCh=Integer.parseInt(chk1[r1].substring(1,2));
                                    if(CopysubjectCH[i]==1){
                                                if(iCh==1)
                                                            if(bDay[iDay]==0)
                                                                        if(bValueChk[r1]==0){
                                                                                    bValueChk[r1]=1;
                                                                                    bDay[iDay]=1;
                                                                                    CopysubjectCH[i]-=1;

        sValue[i]=addString(chk1[r1],sValue[i]);
                                                                        }
                                    }
                                    else if(CopysubjectCH[i]>=2){
                                                if(iCh==2)
                                                            if(bDay[iDay]==0)
                                                                        if(bValueChk[r1]==0){
                                                                                    bValueChk[r1]=1;
                                                                                    bDay[iDay]=1;
                                                                                    CopysubjectCH[i]-=2;

        sValue[i]=addString(chk1[r1],sValue[i]);
                                                                        }
                                    }
                                    c1++;
                                    if(c1==1000){
                                                c1=0;
                                                bFlagT=false;

                                                i=CopysubjectCH.length-1;
                                                CopysubjectCH[i]=0;
                                    }
                        }while(CopysubjectCH[i]>0);
            }
            return bFlagT;
}

private void resetbDay(){
            int i;
            for(i=0;i<bDay.length;i++)
                        bDay[i]=0;

}

private String addString(String s1, String s2){
            int i;
            String sTemp;
            sTemp="";
            for(i=0;i<s1.length();i++)
                        if(i==0)
                                    sTemp+=s1.substring(i,i+1);
                        else

        sTemp+=Integer.toString(Integer.parseInt(s1.substring(i,i+1))+Integer.parseInt(s2.substring(i
,i+1)));
            return sTemp;
```

```
	}

	private void initsValue(){
		int i,j;
		for(i=0;i<sValue.length;i++){
			sValue[i]="";
			for(j=0;j<chk1[0].length();j++)
				sValue[i]+="0";
		}
	}

	private void changeHead(){
		int i;
		for(i=0;i<sValue.length;i++)
			sValue[i]="P" + sValue[i].substring(2);
	}

	public String [] getValue(){
		return sValue;
	}
}


//Class Name: clsPeriod1.java
//Written by: Ku Chin Soon
import java.awt.*;
import java.util.*;
import java.util.Random;
class clsPeriod1{
	private int [][] chkTb;
	private String  str;
	private String [] str1;
	private String [] Copystr1;
	private int iPeriodPerDay;
	private int [] iCH;
	private int cntOrgP1, cntOrgP2, cntConP1, cntConP2;
	private int subP2, rnd1;
	private Random r;
	public clsPeriod1(){}

	public void InitializeClass(int [] iSubjectCH, int iTotalPeriod,String definePeriod){
		int i,j;
		iPeriodPerDay=iTotalPeriod/5;
		iCH = new int [iSubjectCH.length];
		iCH = iSubjectCH;
		chkTb = new int [5][iPeriodPerDay];
		for(i=0;i<5;i++){
			for(j=0;j<iPeriodPerDay;j++){
	chkTb[i][j]=Integer.parseInt(definePeriod.substring(i*iPeriodPerDay+j,i*iPeriodPerDay+j+1
));
			}
		}
	}

	public void generateSubPeriod(){
		int c1, c2, x, y, i, j,l ,ihold=0,ilen=0,ilen1=0, count=0;
		boolean bFlag;
		String sTemp,sHead="";
		int iIndex1=0, iIndex2=0;
		str1 = new String [100];
```

```
int counter1=0;
r = new Random();
c1=0;
c2=0;
for (i=0;i<5;i++){
        for (j=0;j<iPeriodPerDay;j++){
                str="";
                str+=Long.toString(i);
                if(j%2==0){
                        if (chkTb[i][j]==0 ){
                                if (j+1<iPeriodPerDay){
                                        if(chkTb[i][j+1]==0){
                                                str+="2";
                                                for( x =0; x<5; x++)

for( y=0;y<iPeriodPerDay;y++){
if(x==i){
        if(y==j ||y==j+1)
                str+="1";
        else
                str+="0";
        }
else
str+="0";
}
c1++;
j++;
}
else{
        str+="1";
        for(x =0; x<5; x++)
                for(y=0;y<iPeriodPerDay;y++){
                        if(x==i){
                                if(y==j)
                                        str+="1";
                                else
                                        str+="0";
                        }
                        Else
                                str+="0";
                }c2++;
}
}
        else{
                str+="1";
                for(x =0; x<5; x++)
                        for(y=0;y<iPeriodPerDay;y++){
                                if(x==i){
                                        if(y==j)
                                                str+="1";
                                        else
                                                str+="0";
                                }
                                else
                                        str+="0";
        }
        c2++;
}
}
}
```

```
else if(j%2==1){
        str+="1";
        if(chkTb[i][j]==0)
                for(x=0; x<5; x++)
                        for(y=0;y<iPeriodPerDay;y++){
                                if(x==i){
                                        if(y==j)
                                                str+="1";
                                        else
                                                str+="0";
                                        }
                                else
                                        str+="0";
                        }
                        c2++;
}

if(str.length()>2){
        str1[counter1] = str;
        counter1++;
}

}
}
        Copystr1 = new String [counter1];
        Copystr1=str1;
        str1 = new String [counter1];
        for (i=0;i<counter1;i++){
                str1[i]=Copystr1[i];
        }
        calOrgCH();
        if(cntOrgP1==cntConP1 && cntOrgP2==cntConP2){
                //continue next
        }
        else{
                if(cntOrgP1>cntConP1){
                        subP2 = cntOrgP1-cntConP1;
                        i=0;
                        do{
                                ilen1 = str1.length;
                                rnd1 = Math.round(150*r.nextFloat())%ilen1;
                                if(rnd1<ilen1){
                                        if(countBit(str1[rnd1].substring(2))!=1){
                                                bFlag =true;
                                                for(j=2;j<str1[rnd1].length() &&
bFlag;j++){
        if(Integer.parseInt(str1[rnd1].substring(j,j+1))==1){
                ihold = j;
                =false;
        }
}
        ilen = str1[rnd1].length();
        sTemp ="";
        sHead = str1[rnd1].substring(0,1);
        sTemp += sHead;
        sTemp += "1";
        for(j=2;j<ilen;j++){
        if(j==ihold){
                sTemp +="1";
        }
```

```
                else
                        sTemp +="0";
        }
        str1[rnd1] = sTemp;
        Copystr1 = new String [str1.length];
        Copystr1 = str1;
        str1 = new String [Copystr1.length+1];
        for(j=0;j<Copystr1.length;j++){
        str1[j] = Copystr1[j];
        }
                str1[j] ="";
                str1[j] += sHead;
                str1[j] += "1";
                for(l=2;l<ilen;l++){
                        if(l==ihold+1){
                                str1[j] +="1";
                        }
                        else
                                str1[j] +="0";
                        }
                        i++;
                }
        }

}while(subP2/2!=i);
}
else{
        subP2 = cntConP1-cntOrgP1;
        i=0;
        do{
                ilen = str1.length;
                bFlag = true;
                do{
                rnd1 = Math.round(150*r.nextFloat())%ilen;
                if(rnd1<ilen){
                        if(countBit(str1[rnd1].substring(2))==1){
                                iIndex1 = rnd1;
                                bFlag = false;
                        }
                                                        }
                }while(bFlag);
                bFlag = true;
        do{
                rnd1 = Math.round(150*r.nextFloat())%ilen;
                if(rnd1<ilen){
                        if(countBit(str1[rnd1].substring(2))==1){
                                iIndex2 = rnd1;
                                bFlag = false;
                        }

        }while(bFlag);
        if(str1[iIndex1].substring(0,1).compareTo(str1[iIndex2].substring(0,1))==0 &&
        str1[iIndex1].substring(1,2).compareTo(str1[iIndex2].substring(1,2))==0 &&
str1[iIndex1].compareTo(str1[iIndex2])!=0){
        sTemp ="";
        sTemp +=str1[iIndex1].substring(0,1);
        sTemp +="2";
        for(l=2;l<str1[iIndex1].length();l++){
```

```
                    sTemp +=
Integer.toString(Integer.parseInt(str1[iIndex1].substring(l,l+1))+Integer.parseInt(str1[iIndex2].substri
ng(l,l+1)));
            }
            i++;
            str1[iIndex1]="null";
            str1[iIndex2]="null";
            Copystr1 = new String [str1.length-1];
            count=0;
            for(l=0;l<str1.length;l++){
                    if(str1[l]!="null"){
                            Copystr1[count] = str1[l];
                            count++;
                    }
            }
            Copystr1[count] = sTemp;
            str1 = new String [Copystr1.length];
            str1 = Copystr1;
}

}while(subP2/2!=i);
}
}
}

public void calOrgCH(){
                    int i, countP1=0, countP2=0;
                    for(i=0;i<iCH.length;i++){
                            countP1 += iCH[i]%2;
                            countP2 += iCH[i]/2;
                    }
                    cntOrgP1=countP1;
                    cntOrgP2=countP2;
                    countP1=0;
                    countP2=0;
                    for(i=0;i<str1.length;i++){
                            if(countBit(str1[i].substring(2))==1){
                                    countP1++;
                            }
                            else if(countBit(str1[i].substring(2))==2){
                                    countP2++;
                            }
                    }
                    cntConP1=countP1;
                    cntConP2=countP2;
            }

            private int countBit(String str){
                    int i,cnt=0;
                    for(i=0;i<str.length();i++){
                            if(Integer.parseInt(str.substring(i,i+1))==1){
                                    cnt++;
                            }
                    }
                    return cnt;
            }

            public String [] getValue(){
                    return str1;
            }}
```