A Framework for Design of Automated Lecture Timetabling using Memetic Algorithm

Zarina Zainul Rashid
Roslan Sadjirin
Norhafizah Hashim

ABSTRACT

Lecture timetabling is a complex real-world problem with many interdependencies. This study describes the framework design of an automated lecture timetabling in educational institutions using soft computing techniques. In this paper, the writerspresent the use of memetic algorithm in the framework design with affective explorative ability to model huge search space. The writers analyse the process, problem and the constraints in timetabling development and study the disciplines that relate to memetic algorithm approach. These requirement specifications are used to design the data structure and the algorithms to produce system architecture for a framework of the automated timetabling system.

Keywords: *framework design, automated lecture timetabling, soft computing, memetic algorithm*

Introduction

Lecture timetabling refers to the allocation of lecture sessions and lab sessions to time slot and to classroom. Lecture timetabling problem occurs in many educational institutions. This problem is a complex real-world problem with many interdependencies. Wren (as cited in Chand, 2002) defines timetabling as 'the allocation, subject to constraint of given resources to objects being placed in space time, in such a way to satisfy as nearly as possible as set of desirable objectives'.

The size and complexity of lecture timetabling problems has encouraged research in soft computing techniques. Studies have shown that metaheuristic approach for the lecture timetabling can give encouraging results. Memetic Algorithm (MA) is a metaheuristic approach that combines Genetic Algorithm (GA) and Local Search heuristic to produce required solution for a particular optimatisation problem. This hybrid technique has a great potential to be an efficient algorithm to automate the development of timetables or schedules. Therefore, this paper presents the framework design for the automated timetabling system using memetic algorithm.

Timetabling

The task in timetabling is to accommodate a set of entities, for example, classroom, lecturer and student-group into a time slot so that the available resources are utilised in the best possible way and the existing constraints are satisfied. The problem of searching for solutions in timetabling is subjected to the constraints. These constraints will give a fitness evaluation to the data. According to the Burke & Petrovic (2002), these constraints can be divided into two categories, namely hard constraints and soft constraints.

Hard constraints

Hard constraints are rigidly enforced. Deris et al. (1999) provide several examples of such constraints:

a. Courses taught by the same lecturer should not be assigned to the same timeslot:

$T(C_i) \neq T(C_j)$ if $L(C_i) = L(C_j)$.
where $T(C_i)$ and $T(C_j)$ are timeslot for courses
C_i and C_j, $i, j = 1, 2, 3, 4\ldots, n$
$L(C_i)$ and $L(C_j)$ are lecturers.

Timeslot for courses C_i, $T(C_i)$ must not be the same as the timeslots for courses $T(C_j)$ if the lecturers are the same.

b. Courses from the same student-group should not be assigned to the same timeslot:

$T(C_i) \neq T(C_j)$ if $SG(C_i) = SG(C_j)$.
where $T(C_i)$ and $T(C_j)$ are timeslot for courses
C_i and C_j, $i, j = 1, 2, 3, 4\ldots, n$
$SG(C_i)$ and $SG(C_j)$ are student group.

Timeslot for courses C_i, $T(C_i)$ must not be the same as the timeslots for courses $T(C_j)$ if the student groups are the same.

c. One classroom should no be assigned to more than one course for the same timeslot:

$T(C_i) \neq T(C_j)$ if $CL(C_i) = CL(C_j)$.
where $T(C_i)$ and $T(C_j)$ are timeslot for courses
C_i and C_j, $i, j = 1, 2, 3, 4\ldots, n$
$CL(C_i)$ and $CL(C_j)$ is classroom or lab.

Timeslot for courses C_i, $T(C_i)$ must not be the same as the timeslots for courses $T(C_j)$ if the classroom or labs are the same.

d. Some timeslot are not available for lectures because they are reserved for specific activities such as co-curriculum.

Soft constraints

Soft constraints are those are desirable but not absolutely essential (Burke & Petrovic, 2002). Examples of soft constraints are:

a. The number of students of a course assigned to a classroom should be less than or equal to the capacity of the classroom. Classroom capacity constraints are represented as:

$Z(CL(C_i)) \geq N(C_i)$

where $Z(CL(C_i))$ is the capacity of the classroom allocated to course C_i, i = 1,2,3,4..... p. $N(C_i)$ is the number of students of the course C_i.

b. A lecturer who holds a position in the management group may prefer to have all their lectures on a certain number of days and to have a number of lecture-free days.

c. A lecturer may prefer to conduct a lecture in a particular classroom or lab.

Memetic Algorithms

Memetic Algorithm (MA) is a metaheuristics technique that combines Genetic Algorithm (GA) and Local Search technique. According to Burke and Petrovic (2002), the main idea of the MA is to explore the neighbourhood of the solution obtained by a GA and to navigate the search toward the local optima, which is Local Search heuristic, for each solution before passing back to the GA and continuing the process. The purpose of using GA and Local Search heuristic is simply because of their characteristics. GA can deal successfully with a wide range of problems area and can produce many solutions to the particular problem. Meanwhile, Local Search heuristic is best at finding the best solution in its neighbourhood and only performed if the resulting solution is better than the current solution (Blum & Roli, 2003), and will only terminate when it produces the best solution. Thus, the combination of these techniques might produce an optimal solution in a short time.

Furthermore, according to Burke & Silva (2005), the use of Local Search heuristic in MA serves as an effective intensification mechanis that is very useful when using sophisticated representations schemes and time consuming fitness evaluation functions. They also argue that by studying the problem domain in detail, there is always a way or alternative to create a new technique in MA approach. They also state that MA is a good approach in solving timetabling problems as follows:

1. MA approaches have a good explorative ability in huge size of search space.
2. MA incorporates specialised encodings and operators for self improvement of solutions which are based on the knowledge of the problem domain.
3. MA is more robust in population of new solutions compare with the single solution methods. It can reduce the effect of the error in the fitness estimation that will improve the time consuming.

MA was used by Boughaci, Benhamou & Drias (2009) in the optimal winner determination problem. The objective is to achieve a good compromise between the intensification and diversification and the search process.

Since MA is a strategy used by many successful global optimisation approaches, the researchers feel that a priority should be given to MA technique. The writers studied the problem domain and propose a new approach to develop a framework design on the automated lecture timetabling.

Framework Architecture for Automated Lecture Timetabling

The combination of Genetic Algorithm (GA) and Local Search heuristic is appropriate because the searching for reasonable good solution would be much effective since the mechanism is based on the existing constraints and guarantees terminated whenever the possible and reasonable good solution were obtained. GA techniques will be used to implement the development of automated timetabling system as follows:

Generate
Data (chromosomes) are evaluated randomly upon the existing constraint and objective of the problem. Next, the best matches of data (chromosome) are searched using Local Search technique. Data that are successfully matched will be inserted into table 'Lecturer Timetable', 'Student Group Timetable', and 'Classroom Timetable'. While data that are failed to be matched with those constraints will be inserted into temporary table called 'Non-Matches Data'.

Crossover
Data from tables 'Lecturer Timetable', 'Student Group Timetable' and 'Classroom Timetable' as well as 'Non-Matches Data' are matched in order to improve the failure rate of data that are failed to be matched. This matching process will be done using Local Search techniques.

Mutation
Any modification to the timetable will cause mutation process to happen. At this stage, two levels of operator are used to decrease the modification to these timetables. The first level is used to modify one timetable only and followed by second level that combines the timetables which have been modified by first level of operator with those non-matches data. These two levels of operation will go through the matching process using Local Search techniques.

Mechanisms to reach the possible optimal solution for all phases will use Local Search techniques. Figure 1 illustrates a framework for Memetic Algorithm technique.

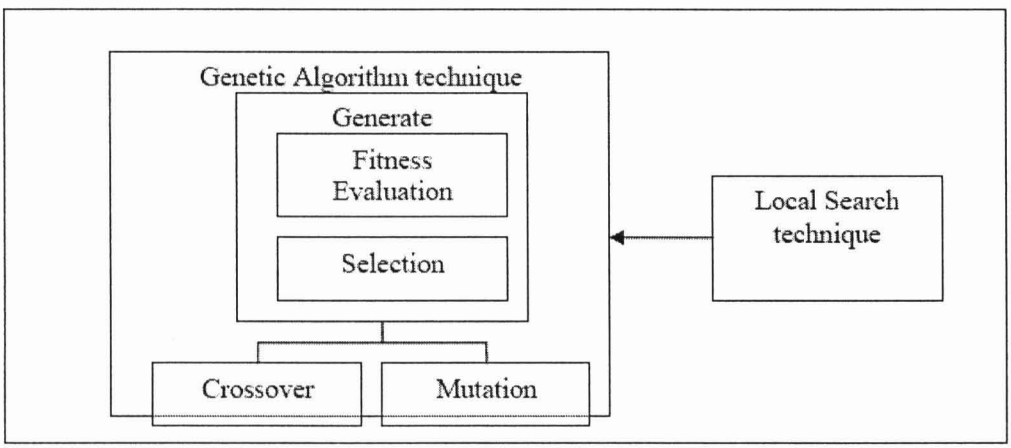

Figure 1. A framework of Memetic Algorithm (MA)

Architecture Design

There are three components involved in the architecture design as shown in Figure 2. The first component is the timetable interface that allows the users to interact with the system. The second component is a data maintenance that manages and stores data in the database. The third component is the timetable generator which generates timetables by using Memetic Algorithm technique.

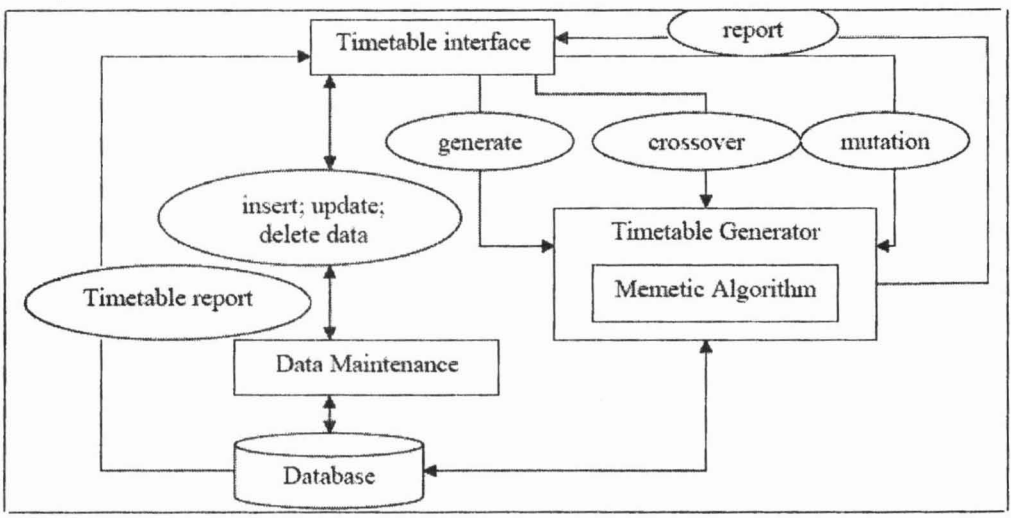

Figure 2. Architecture design of the system

Software Hierarchy Design

This system is divided into three modules which are data maintenance module, generator module and report module as shown in Figure 3. Data maintenance module handles operations on data of lecturers, courses and student groups. Generator module is the main part of the system which uses the MA technique in the generator, crossover and mutation modules. While, Report Module is used to generate report for the 'generated timetables' from the system.

Architecture of Flow Process

The architecture of flow process is focused into three main operations. These three operators are Generate operator, Crossover operator and Mutation operator.

Generate Operator

Population is generated during the Generate operation. Parents are selected from the individuals that are chosen at sequence. The individuals are selected according to the fitness evaluation of the chromosomes to become a parent. The basic local search using the matching algorithm is applied. The flow process Generate operator is shown below in Figure 4.

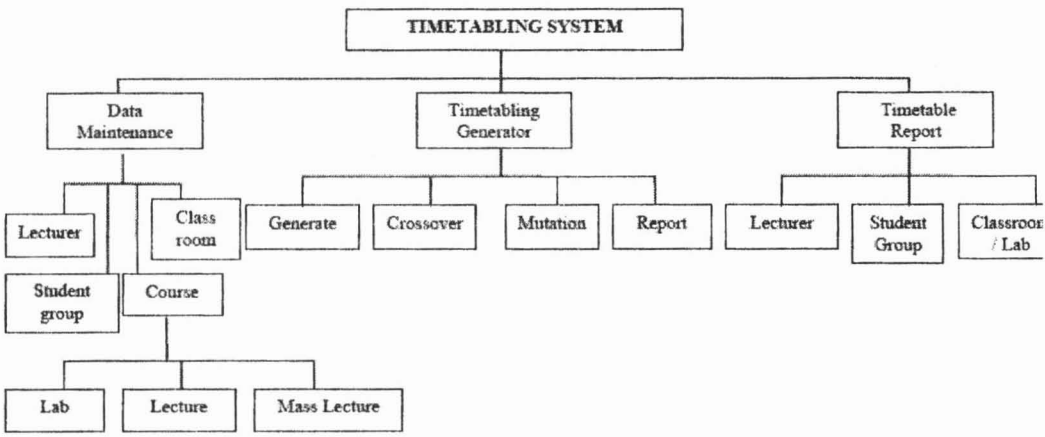

Figure 3: Software Hierarchy Design

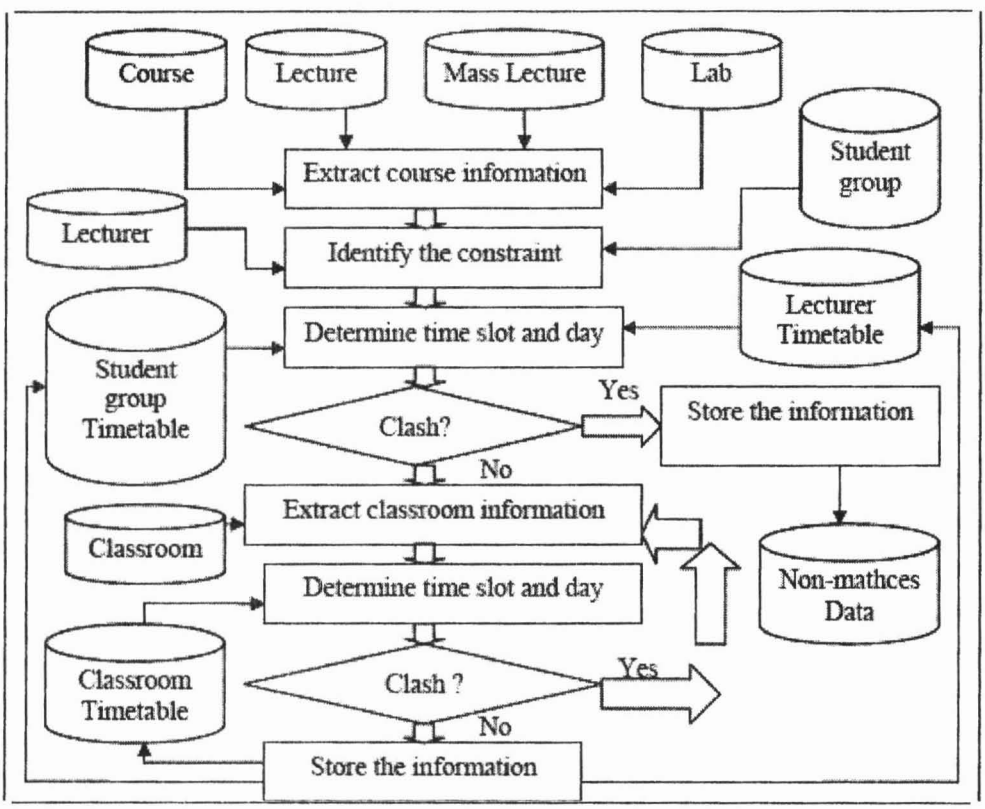

Figure 4. Flow process of generate operation

Crossover Operator

The two selected parents from the population are recombined. The basic local search is used similarly as the Generate operator. The Crossover operator (Figure 5) will reduce the information in the Non-matches Data table.

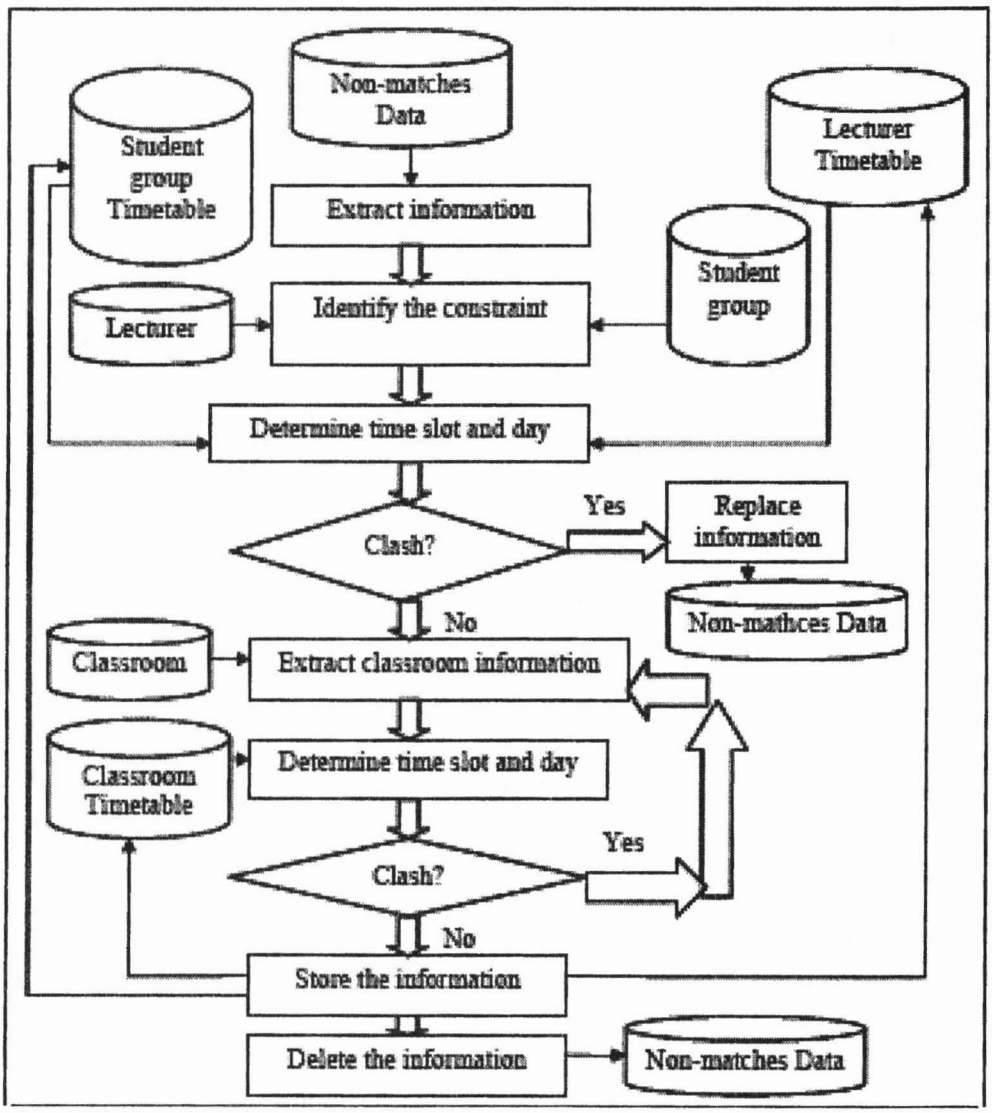

Figure 5. Flow Process of crossover operation

Mutation Operator

In mutation, information of a particular timeslot is replaced with selected information from Non-matches Data. The Mutation operator is restricted to two different levels. In the first level, operator will swap the selected time slot with the free time slot. The second level shows that information from Non-matches Data will be stored into the selected timeslot. The flow process Mutation operation is illustrated in Figures 6 and 7.

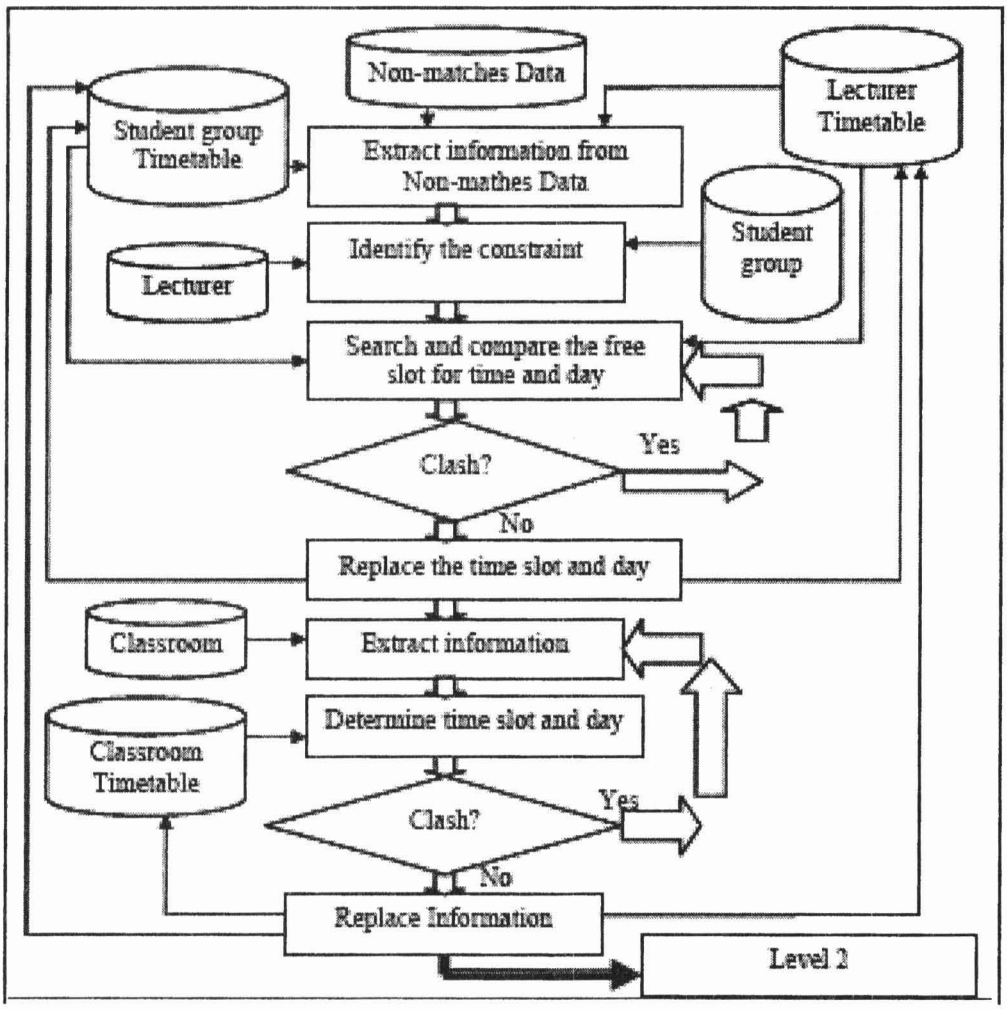

Figure 6. Flow process of mutation operation - Level 1

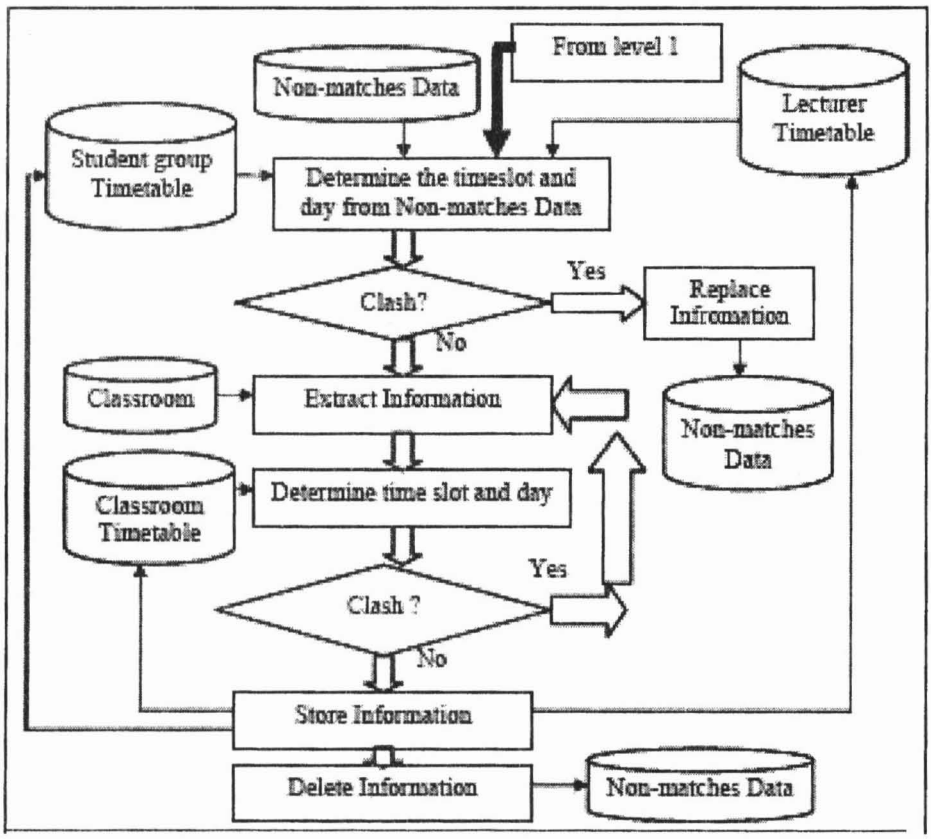

Figure 7. Flow process of mutation operation - Level 2

Database Design

There are eleven tables involved in this framework design. Figure 8 illusrates the ER diagram for the data structure of the framework design. Seven tables contain the data (chromosomes) that can be used as a fitness evaluation. Meanwhile, the other four tables store the selection that had been made during the Generate operator. These four tables are also recombined by using a Crossover operator.

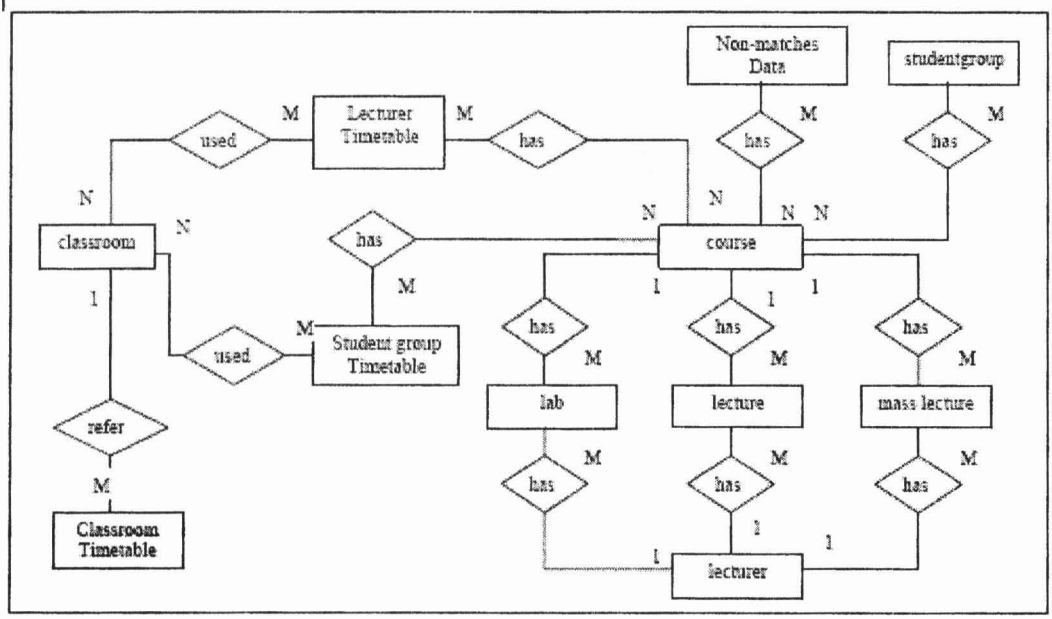

Figure 8. E-R diagram

Conclusion

Memetic Algorithm (MA) is metaheuristic method that applies synergy of evolutionary based approach to search for possible optimal or at least reasonable good solution for a given computational problem. Genetic Algorithm (GA) and Local Search heuristic have their own strength to solve the computational problem. GA is one of the main components of soft computing which is categorised as a global search technique that can find possible optimised solution for a searching problem. On the other hand, Local Search heuristic has an effective intensification mechanism that can search feasible solution quickly within its reachable and available solutions. Furthermore, the properties of this technique can promote the memory space and processing time efficiency because searching process for reasonable solution can be found quickly.

Therefore, by using the combination of GA and Local Search heuristic, the writers propose a 'Framework for Design of Automated Lecture Timetabling Using Memetic Algorithm (MA) for Academic Institution'. The design is an improved version of the MA that suits the problem domain. The innovation in this solution is that, MA technique can be used in the optimisation of the timetabling process. Future work will include the development of the system of automated lecture timetabling and study about the exploritive of search ability.

References

Chand, A. (2002). A Heuristic Approach to Constraint Optimazation in Timetabling. *South Pacific Journal National Scientific.* Vol. 20, 64-67.

Blum, C. & Roli, A. (2003). Metaheuristics In Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Computing Surveys (CSUR).* Vol. 35(3), 268-308.

Boughaci, D., Benhamou, B. & Drias, D. (2009). A Memetic algorithm for the optimal winner determination problem. *Soft Computing - A Fusion of Foundations, Methodologies and Applications.* Vol. 13(8, 9), 905-917.

Burke, E.K. & Petrovic, S. (2002). Recent research directions in automated timetabling. *Europian Journal of Operation Research.* Vol. 140(2), 266-280.

Burke, E.K. & Silva, J.D., (2005). The design of memetic algorithm for scheduling and timetabling problems. *Recent Advances in Memetic Algorithm. Springer.* Vol. 166: 289-311.

Deris, S., Omatu, S., Ohta, H. & Saad, P. (1999). Incorporating Constraint propagation in Genetic Algorithm for University Timetabling Planning. *Engineering Applications of Artificial Intelligent.* Vol. 12 (3): 241-253.

ZARINA ZAINUL RASHID, ROSLAN SADJIRIN & NORHAFIZAH HASHIM, Fakulti Sains Komputer dan Matematik UiTM Pahang, Universiti Teknologi MARA Malaysia. zzr@pahang.uitm.edu.my, roslan81@pahang.uitm.edu.my, fiza_hz@pahang.uitm.edu.my