## INTRODUCTION

Practically all educational institutions have an issue concerning the planning of the time table, notably University. Numerous things must be considered to mastermind the schedule. One of them is the accessibility of instructors. Not all professors are accessible at every time. Some of them are Accessible at some time. Hence, when a Time Table is created, this thing must be considered. Different components are the number of classes and courses advertised.

The working hours of a professor is also a parameter to consider while creating a Time Table. This element is to fixed or to be regarded as a constant except for an academic calamity like insufficiency of lecturers at a particular time.

Availability of classrooms is also a significant issue while designing a schedule for professors.

Considering all the factors above mentioned demonstrates the fact that making an accurate time table in the first go is a very time-consuming task with no guarantee of a maximized economy.

Therefore it is only better to improve the quality and maximize the time table economically but periodically considering and hoping for better mutation at every generated time table.

This project uses a genetic algorithm to design and improve class scheduling for professors. This will ensure that all the time slots and rooms are used such that waiting time or over exceeding the working hours for a professor is not there.

With the fixed number of lecturers, class-rooms, subjects and considering the fact every lecturer maps to only a few given subjects in the whole range, it opens up a possibility of perfection or to attain a global minimum in the graph with the factors as mentioned earlier, slope depicting the usage of resources economically.

## MOTIVATION

While designing a time-table, a significant problem that occurs is not using the resources optimally. There might be a circumstance where all the classrooms are full at a single time or every class empty at the same time, thus wastage of resources. Another major disadvantage is exceeding or not entirely using the working hours of professors. If other algorithms like that similar to round-robin etcetera are used, the structured schedule will prove to be procrustean to the real-time situations. The only solution is to work with an algorithm which works dynamically or which might change the time table for better in the future with the currently collected data. Therefore working with the genetic algorithm will be dynamic to the real-time change, not working at the exact time with new conditions but metamorphically creating an antigen for the future.

## OBJECTIVE

This project is one among many solutions for the Time Table problem. The problem to be noted is to create a schedule which minimizes the overuse of resources by evenly distributing the working hours of professors throughout the single span of time-table, thus reducing stress and avoiding excess relaxation. Another component of the problem is to ensure a classroom is empty at a given point in the time-table.The given parameter is the dataset of professors mapped to their respective subjects and availability of classes at any given instant and total no. of classrooms. The boundation of the solution is not to exceed the working hours of the professors and to ensure that no. of empty or usable classrooms $\geq$ no. of classes to be conducted. Thus with noted parameters and boundaries, the project will be carried out in the C-Language.

## METHODOLOGY

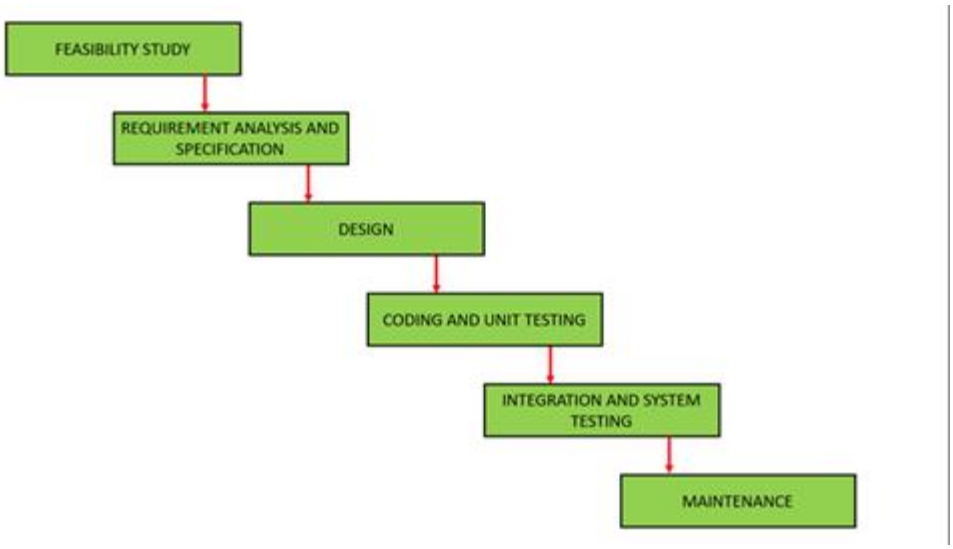
The project development is based on the Water Fall Model. Classical waterfall model divides the life cycle into a set of phases. This model considers that one phase can be started after completion of the previous phase. That is the output of one phase will be the input to the next phase. Thus the development process can be considered as a sequential flow in the waterfall. Here the phases do not overlap with each other. The different sequential phases of the classical waterfall model are shown in the below figure:

### Phase1: Feasibility Study:

The main goal of this phase is to determine whether it would be technically feasible to develop the software.
The feasibility study involves understanding the problem and then determine the various possible strategies to solve the problem. These different identified solutions are analysed based on their benefits and drawbacks, the best solution is chosen and all the other phases are carried out as per this solution strategy.

### Phase 2: Requirement Analysis:

Following points are taken into consideration.
1) Faculty Designation, workload and Preferences
2) Subject Credits
3) Available time slots and rooms
4) Different courses of different semesters




## Phase 3: Designing and Development:

- Genetic algorithm will process vast number of solutions simultaneously.

  The genetic algorithm uses three main types of rules at each step to create the next generation from the current population:

  Selection rules select the individuals, called parents, that contribute to the population at the next generation.

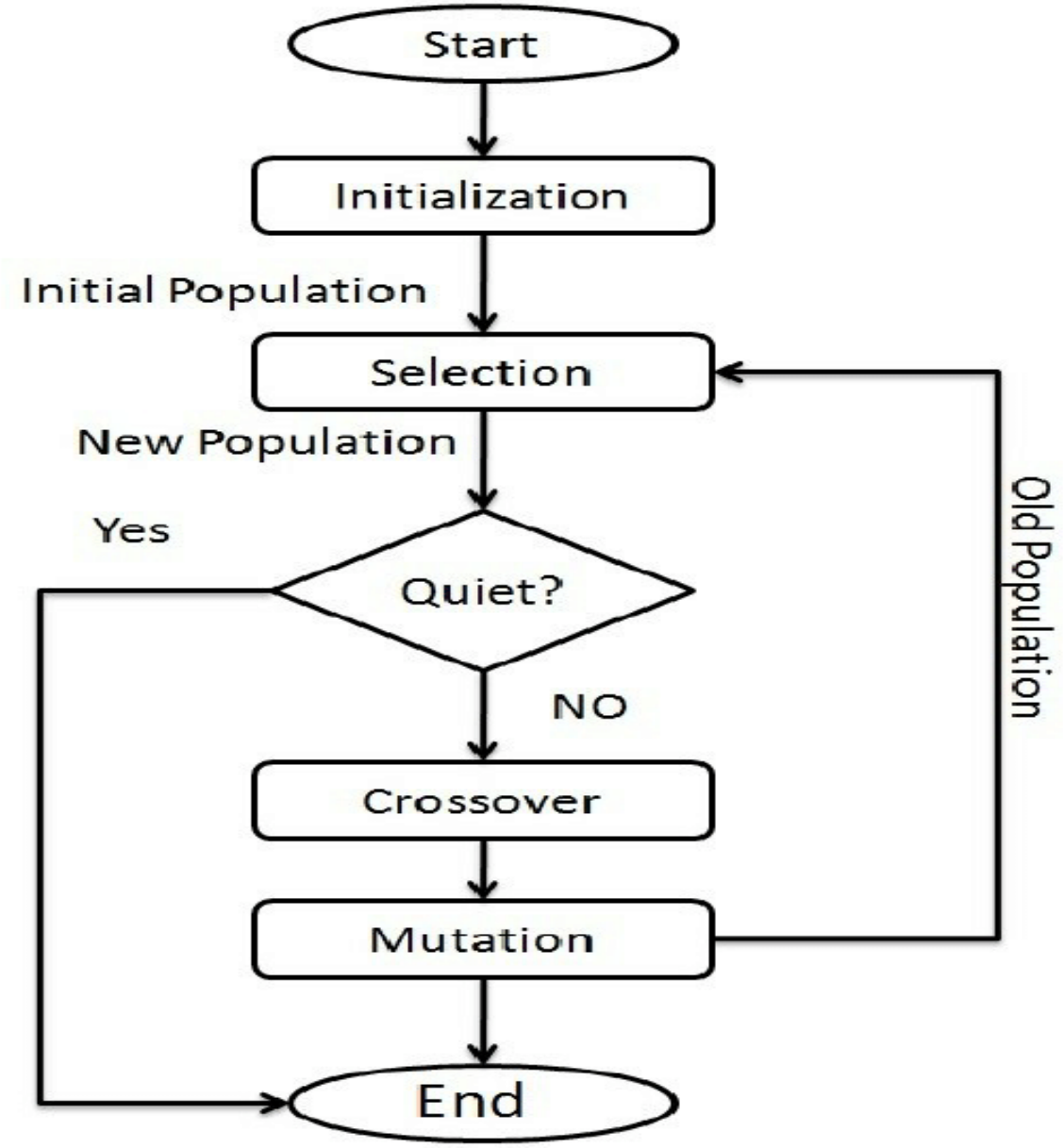
  Crossover rules combine two parents to form children for the next generation.

  Mutation rules apply random changes to individual parents to form children.

- A time table is essentially a schedule which must suit a number of constraints.
- Heuristic search is applied in this project. It uses a 2d matrix called target matrix. This matrix is used to find suitable time slots for scheduling courses.
- There are 6 sets applied in target matrix. This target matrix

They are course code set M={m1,m2,…}, type course class

set T={t0,t1,t2,…}, lecturer code set L={l1,l2,…}, class name set C={c1,c2,…}, day set D={d1,d2,…} and hour set H={h1,h2,…}. All index set start with one. Only type course class set T has member index zero, t0. It indicates only for the case that for a certain course of some parallel classes are merged into one class.


Start
Initialization
Initial Population
Selection
New Population
Yes
Quiet?
NO
Crossover
Mutation
End
Old Population

| | $m_1, t_0, l_1$ | $m_2, t_1, l_1$ | $m_3, t_2, l_2$ | ... |
|---|---|---|---|---|
| $c_1, d_1, h_1$ | 1 | -1 | -1 ... | |
| $c_1, d_1, h_2$ | 1 | -1 | -1 ... | |
| $c_1, d_2, h_1$ | -1 | 1 | | -1 ... |
| $c_1, d_2, h_2$ | -1 | 1 | | -1 ... |
| $c_2, d_1, h_1$ | 1 | -1 | -1 ... | |
| $c_2, d_1, h_2$ | 1 | -1 | -1 ... | |
| $c_2, d_2, h_1$ | 0 | 0 | | 0 ... |
| $c_2, d_2, h_2$ | -1 | -1 | | 1 ... |
| ... | ... | ... | | |
| # of units scheduled | 4 2 1 | ... | | |

## Phase 4: Coding and Unit Testing

In coding phase software design is translated into source code using any suitable programming language. Thus each designed module is coded. The aim of the unit testing phase is to check whether each module is working properly or not.

## Phase 5: Integration and System testing

Integration of different modules are undertaken soon after they have been coded and unit tested. Integration of various modules is carried out incrementally over a number of steps. During each integration step, previously planned modules are added to the partially integrated system and the resultant system is tested. Finally, after all the modules have been successfully integrated and tested, the full working system is obtained and system testing is carried out on this.

## SYSTEM REQUIREMENTS

### •Hardware Interface:

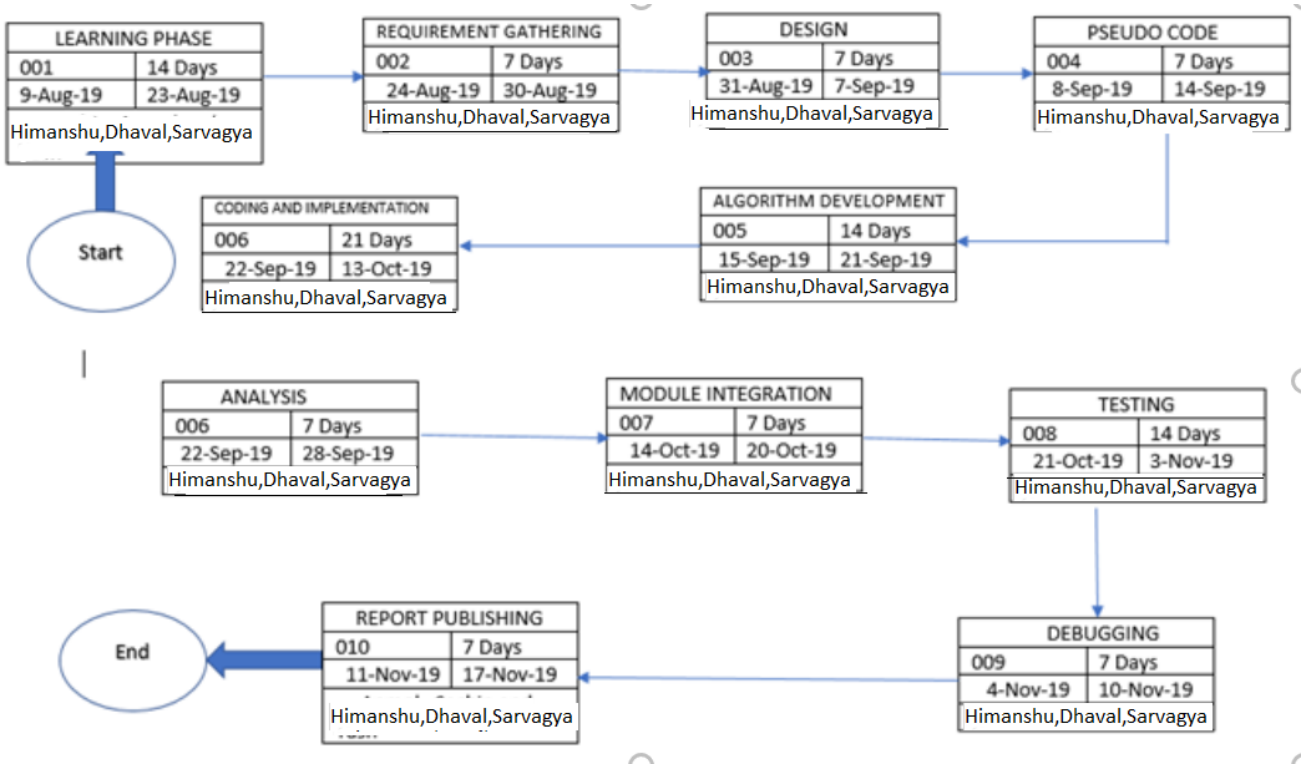
–Required 64-bit processor architecture supported by windows.

### • Software Interface:

–This system has to be developed in C programming language.

–It requires C compiler.

## SCHEDULE (PERT Chart)



## REFERENCES


https://www.researchgate.net/figure/Flow-chart-of-genetic-algorithm_fig2_265208874

https://www.researchgate.net/publication/221927228_Solving_Timetable_Problem_by_Genetic_Algorithm_and_Heuristic_Search_Case_Study_Universitas_Pelita_Harapan_Timetable

Burke E.K., Elliman D.G. and Weare R.F. (1994) "A Genetic Algorithm for University Timetabling", AISB Workshop on Evolutionary Computing, Leeds.

Burke E and Ross P (Eds) (1996): Lecture Notes in Computer Science 1153 Practice and Theory of Automated Timetabling First International Conference, Edinburgh, U.K., August/September 1995, Selected Papers. New York: Springer-Verlag Berlin Heidelberg.

Davis L. (1991) "Handbook of Genetic Algorithms" Van Nostrand Reinhold

Erben W and Keppler J (1995): A Genetic Algorithm Solving a Weekly Course- Timetabling Problem. In Burke E and Ross P (Eds): Lecture Notes in Computer Science 1153 Practice and Theory of Automated Timetabling First International Conference, Edinburgh, U.K., August/September 1995, Selected Papers. New York: Springer-Verlag Berlin Heidelberg. pp 198-211.

Enmund Burke, David Ellimand and Rupert Weare. (1994). "A Genetic Algorithm Based University Timetabling System", August 2011, http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.2.2659