Use of Genetic Algorithm in timetabling system

Sofiia Sarana

Table Of Contents

Abstract

This document provides an overview about the timetabling system based on the genetic algorithm that takes the excel file containing data about groups, modules, and rooms as input and creates a new excel file with the timetable for each group. This optimises the way of creating timetables and provides ways for developers to integrate it in the websites and apps used for students. Because this simple code is written by a student, the program tries to avoid scheduling classes for Friday to make both students and lecturers a little bit happier, at least some of them.

Summary workflow of the algorithm:

1. Initialise a population of random timetables
2. Evaluate each timetable using a fitness function that considers penalties for conflicts and use of Fridays
3. Select a best half of the individuals based on the score
4. Crossover the selected timetables to create offsprings
5. Mutate some of the offspring timetables for the sake of diversity
6. Replace the individuals that are “weak” with the new offspring to create new population
7. Repeat the process for 100 generations
8. Select the best individual as the final solution

Algorithm Summary

The Genetic Algorithm is an algorithm inspired by natural selection, that is used for combinatorial and optimisation problems that are too complex to be solved by traditional methods. In simple words, “in each generation, only the strongest from the population will survive”. Genetic algorithm is based on an analogy with the genetic structure and behaviour of chromosomes of the population. Following this, the main approach of the genetic algorithm will be:

Individuals within a population engage in competition for resources and opportunities to mate. Those who succeed the most, often considered the “strongest" have more offspring compared to others. The genes of these more successful parents spread more widely in the population, sometimes resulting in offspring that are even better adapted than either parent. Consequently, each new generation becomes increasingly better suited to thrive in their environment.

The advantages of using this algorithm for the timetabling problem are:

1. Handling complex constraints
   The use of probabilistic rules instead of deterministic rules helps to solve conflicting goals of the schedule.
2. Exploration and exploitation
   Random initialisation and mutation is good for exploring the solution space efficiently. Selection and crossover helps with exploiting a better solution.

The disadvantage include:

1. Expensive computation
   If the problem is large, the use of computational resources increases significantly, so does the execution time.
2. Not exactly optimal
   The quality of the output depends on the size of population, number of generations and mutation rate. It can be handy since it is adjustable, but it can miss some of the most optimal solutions.

Applying the algorithm to the problem

The individuals in the population are represented as dictionaries. Each key in the dictionary is a session variable, each value represents the time, room, and lecturer assigned to this session. The domains for each variable include all possible combinations of time slots, rooms, and lecturers.

1. Initialisation

   Randomly generates a population of individuals - complete timetables. Each individual is represented as a dictionary of VARIABLES, where each variable is assigned a value from its DOMAIN.

2. Fitness Function

   Evaluation of how good the timetable is. The goal here is to minimise penalties that include:

   Group conflicts - each group can't have more than one session (either lab or lecture) at the same time.

   Lecturer conflicts - each lecturer can't give more than one lecture at a time.

   Room conflicts - only one group can be in the room at a given time.

   Friday penalty - avoid using Fridays unless it's necessary.

3. Selection

   Select best individuals based on the fitness scores to form a mating pool.Best half is selected for reproduction.

4. Crossover

   Combine pairs of individuals to produce an offspring. Takes half of the schedule from one parent and half from the other. This will perform until we can partially replace the original population.

5. Mutation

   Introduces random small changes to the population to maintain diversity. Mutation rate is set to 1%, which helps to avoid being stuck in local optima.

6. New population

   After step 4 and 5, the new population is formed with the best half of the previous population and generated offspring.

7. Termination

   Iterated through 100 generations, and after the final generation the best available solution is selected (again, based on the fitness score).

How to use it

1. Create an Excel file with 3 sheets - groups, modules, and rooms (example of this is provided on github).

   Groups sheet should contain 1 column, where the first element is “GroupName” heading, under this cell provide the name of each group one by one.

   Modules sheet should contain 4 columns: Group, Module, ModuleName, Lecturers. First row contains corresponding headings that should be provided the same as mentioned here. All other rows contain the name of the group (preferably contains at least one alphabetic character), module code, module name and lecturers. Lecturers for the module should be separated by comma and be in one cell.

   Rooms sheet should contain 2 columns: LectureRooms, LabRooms with corresponding headings. All rows under the heading row should contain names of the lecture rooms and lab rooms.
2. Install requirements

   Run this in terminal:

   pip3 install -r requirements.txt
3. Organise folder

   Put the created excel file in the folder with the program.
4. Run program

   python3 timetable.py

   The program will ask you to provide the name of the file, you should enter it with extension, for example:

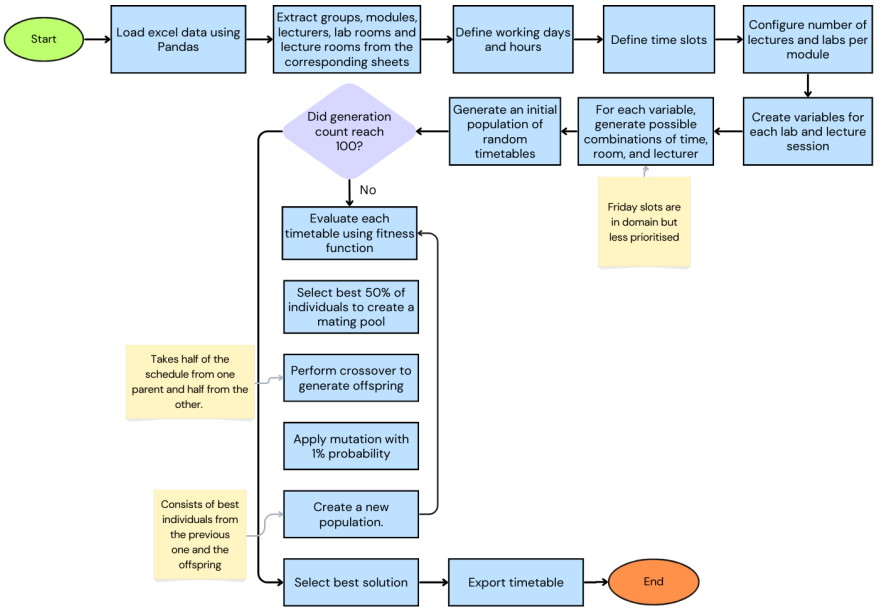
   ```
   Enter the name of the file: timetable_data_xlsx
   File timetable_data_xlsx doesn't exist here, try again
   Enter the name of the file: timetable_data.xlsx
   Timetable has been successfully exported to 'timetable_output_matrix.xlsx'
   ```

   As you can see, if the filename is not valid you will be asked to provide it again and again until it is valid. If you don’t wish to proceed, press Ctrl+C.
5. View output

   The generated file is located in the same directory as the program file. Note that if you run the program again without changing the name of the output file, it will be overwritten.

Appendix

The diagram below shows the workflow as described in previous sections:



Integration

The code provided can be advanced and integrated into websites for timetable creation, or directly integrated into universities websites to fetch the data from the database, and create timetables that can be displayed using designing tools.
The code on its own makes the process of timetable creation easier and can be done by individual students, or even a very small group of people to create all timetables at once for the year.

Further development

The possibility of further development include:

- Reducing computational load
- Research to find the best value for parameters such as mutation and number of generations.
- Interface!!!

Note: this code wasn't tested on big amounts of data or real systems.