Case Study – Timetable Generation

Sujit Kumar Chakrabarti

March 3, 2023

The course coordinators of all programmes prepare the academic timetable for each semester based on the registration details of all courses. The generated timetable is valid only if it doesn't have any schedule conflicts. A conflict arises when two interfering courses occupy the same or overlapping slot. Two courses are said to interfere if they have at least one common participant.

At the top level, the application context looks like this:

Input and Output Data

To get started, we may want to define the structures for the input and output data.

- **Course list:** Represented by a list of course code
- **Participant list:** Represented by a list of roll number/employee ID
- **Registration Data:** Participant $\longrightarrow$ Course list

Timetable Scheduling and Graph Colouring

Graph Colouring Problem

The problem of assigning a colour to each node in the graph such that no two neighbours get the same colour.

Interference Graph

Each course is represented by a node. If two courses interfere with each other (i.e. have at least one common participant), draw an edge between them. Now, if each available slot is presented by a colour, then generating a valid time-table is nothing but k-colouring the graph, where k is the number of available slots.

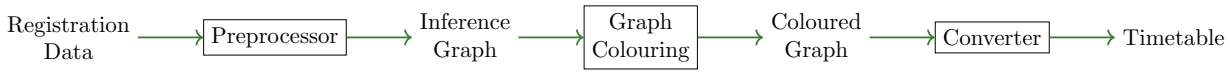
Hardness of Graph Colouring

Graph colouring is an NP-complete problem. So, we have to go with a heuristic.

Heuristics

1. Chaitin
2. Genetic algorithms
3. Ant colony
4. ML

Architecture



In the beginning, interference graph can be implemented as an edge list, where each edge is a pair (2-tuple) of nodes.

Coloured graph can be a list of pairs: $(node \times colour)list$

Development Approach

- Teams of 2.
- Develop the initial set of test cases together
- APIs (if any) for data-structures
- Split as follows:
  - Preprocessor
  - Graph colouring
  - Integrate and test
  - Optimise (faster implementation of data structures)