POLITECNICO
DI TORINO

Exam Time Table Report

Group 3 [MA-ZZ]

Jacopo Maggio
Stefano Munna
Jacopo Nasi
Andrea Santu
Marco Torlaschi

8 gennaio 2018

1 Report

This document provides some explenation about our solution to the problem proposed. The project is developed with C with support of CLion (JetBrains IDE) and GitHub.

1.1 Data Structure

The main structure is a graph, build on a adjacency matrix. This matrix have value:

$$adjM_{e1,e2} = \begin{cases} 1 \rightarrow \textit{If } \boldsymbol{e1} \textit{ and } \boldsymbol{e2} \text{ can't be sustained in the same ts} \\ -1 \rightarrow Otherwise \end{cases} \tag{1}$$

The choice over this structure is related to its ease, it allows a fast reading of the instances, also the benchmark and feasability become easy to check.

1.2 Workflow

The first problem to be solved is to finding a **feasible** solution to be used for the future improvement.
Our solution, is divided in two parts, a greedy data preparation and a tabu search. Using only the tabu, that will work anyway, will require too much time for finding the first feasible solution.

Greedy The first phase try do generate a feasible solution adding one at the time the exam. The idea is to reduce "the complexity" of the problem. It start from a sorted data, from the exam with more collision to the fewer ones. The flow is the following:

1. Use the first available timeslots that not generate conflicts for each exams.

2. If there aren't enough timeslots it will add a new timeslot.

Of course adding more timeslots not fit the problem constraints. These adding will be solved with the next step. The advantages of these implementations are the reduction of computational time and the easiness to implement it.

Tabu Search The goal of the second step is to reduce the number of added, by the previous step, timeslots to the correct number, generating a feasible solution. The workflow is:

1. Decrease the number of timeslots.

2. Try to resolve the conflict.

- If not: Backtrack.
- If yes: Restart from point 1.

Using the tabu search is a really reliable, the main problem is its slowness that is partially solved by the greedy step.

The next step try to improve our founded solution by using some heuristic/meta-heuristic algorithms.
These step implement four different procedures: *Local Search, Local Swap, Simulated Annealing and Greedy Slot Shuffle.*

The **Local Search** is divided in three different algorithms:

- **Move Exam**: Change the timeslot of en exam.
- **Swap Exam**: Swap the timeslot of two exams.
- **Bounded Shift**: Shift a portion of timeslots.

The **Local Swap** implements a steepest descent strategy and it evaluate the Neighborhood *N(x)* of the current optimal solution *x*. The flow:

1. Switches exams scheduled in a timeslot with those contained in any another to find a new feasible *x'*.
2. If *x'* has a better benchmark than that of x, then *x=x'*.
3. Loop until no improvement.

We have choose the ST strategy respect the first improvement due to its efficiency, despite its slower speed. All the local search and the local swap strategies are applied in loop until the solution is improving.

Our implementation of the **Simulated Annealing** is not different from the studied one. After some tuning we have used a starting temperature of 1000, that decrease of alfa = 0.9 every L = 10 iterations. After the temperature reduction the feasability is restored using the tabu search and the minimum is searched using the local search.
Applying these algorithm, due to its randomness, could give different (even worst between different running) results.

The last implemented function is the **Greedy Slot Shuffle**, its a really simple algorithm that, starting from a feasible, add, one at a time a new timeslot evaluating the benchmark. Of course applying these flow not require to restore the feasability because is always guarantee by starting from and already feasible solution.

1.3 Other Solution

Another tried, but not implemented algorithm is the **genetic**. We have choose to not used it because it seems that not fit very well our problem. Due to its randomness the majority of generated solutions are unfeasbile.