CS 509 Advanced C++ Programming
Term Project Report

Teaching Assistant Scheduling Framework

By

YAKUP GÖRÜR – S017327

Project Definition

In every school, there are some teaching assistants and offered courses. In the beginning of the term these assistants must be assigned for offered courses which they have some requirements about teaching assistantance. Due to huge numbers of assistants and courses, this assignment task could be so painful. In this project the task is the writing a framework about this assignments with using modern C++ (C++17) features.

Project Description

Teaching assistants framework is a framework that reads data from two files that are in the same folder with the application: COURSES.csv and ASSISTANTS.csv. Application will then write the solution to a SOLUTION.csv file. Expected file format can be illustared like below:

COURSES.csv:

```
CourseCode1,InstructorName,MinTAHours,MaxTAHours,MinTACount\n
CourseCode2,InstructorName,MinTAHours,MaxTAHours,MinTACount\n
…
```

For Instance:

```
CS101, Kubra Kalkan Cakmakci, 60, 100, 3
CS321, Furkan Kirac, 10, 20, 1
CS409, Furkan Kirac, 10, 20, 2
```

ASSISTANTS.csv: this file will contain the name of the assistant and his/her required hours to assist per week. HoursToAssists can either be 5, 10, 15, 20, or 30.

```
NameOfAssistant1, HoursToAssist, MaxCoursesToAssist, OldAssistedCourses, …\n
NameOfAssistant2, HoursToAssist, MaxCoursesToAssist, OldAssistedCourses, …\n
…
```

For Instance:

```
Ahmet, 20, 3           // 20h/week req., at most 3 courses, no old experience
Aytekin, 10, 2, CS101  // 10h/week req., at most 2 courses, assisted CS101 once
Ebru, 20, 3, CS321     // assisted CS321 once
Baris, 30, 3, CS321, CS409, CS409, CS409  // assisted CS321 once and CS409 for 3 times
Zeynep, 30, 3          // …
Hasan, 10, 2
```

Application must write its solution to a file in the same folder whose name is SOLUTION.csv.

SOLUTION.csv:

```
CourseCode1-SectionCode, NameOfAssistant, HoursToAssist, NameOfAssistant, HoursToAssist, …\n
CourseCode2-SectionCode, NameOfAssistant, HoursToAssist, NameOfAssistant, HoursToAssist, …\n
…
NameOfAssistantX, HoursFreeX,
NameOfAssistantX, HoursFreeY,
…
```

For Instance:

```
CS101, Ahmet, 20, Ebru, 10, Hasan, 10, Zeynep, 20
CS321, Baris, 20
CS409, Ebru, 10, Aytekin, 10
Baris, 10
Zeynep, 10
```

Code Structure

The framework consists of 1 main cpp file and 7 header files which are:

1. main.cpp
2. csvio.h
3. data.h
4. assistant.h
5. lecturer.h
6. course.h
7. greedysolver.h
8. optimalsolver.h

main.cpp

"main.cpp" controls operation of the program. It creates corresponding objects from header files to read data, solve problem, save output and writing necessary information about process on console. It also includes "INPUT" folder which consist of COURSES.csv and ASSISTANTS.csv as hardcoded.

csvio.h

It includes a class for reading csv files row by row.

data.h
It reads the assistants.csv and courses.csv. Keeps the corresponding type vector as

assistances_vec, courses_vec. All data are keeping in this class.

assistant.h
It keeps an assistant information from asistants.csv

lecturer.h
Create Lecturer with name and given courses. In this project it is not necessary. Because we can

keep information about lecturer in “courses.h” header file as just string. But to think further

improvements of this framework it could be meaningfull.

course.h
It keeps a course information from courses.csv

greedysolver.h
it's includes greedy algorithms to solve scheduing problem.

optimalsolver.h
It’s derived from greedysolver class and it also inludes calculation of cost function. In this

project optimal solver is not completely optimizated solving algorithm.

Solver Algorithm

Solver algorithm has been implemented considering the conditions which are:

From “ASSISTANTS.csv”:

HoursToAssist: hours of serving as a assistance for a specific teaching assistance (TA)

MaxCoursesToAssist: maximum course number which can assigned for a specific TA.

From “COURSES.csv”:

MinTAHours: minumum requriement TA hours to serve.

MaxTAHours: maximum requriement TA hours to serve.

MinTACount: minumum requriement TA number to serve.

The solver algotithm basically consist of these steps:

Iterate over courses:

Iterate over assistances:

- If there is no remaining “course.Course TA hours” [MinTAHours, MaxTAHours](min or max corresponding to solution type) and there is **no** remaining **“course.MinTACount” pass this course.**
- If there is no remaining “course.Course TA hours” but there is remainig **“course.MinTACount”** behave like there is remaining 5 hours of “Course TA hours”.
- If there is **no** available number of TA serves for this assistant (assistance.**MaxCoursesToAssist**) **Pass this assistant.**
- If there is **no** remaining serve hours for this assistant (assistance.**HoursToAssist**) **Pass this assistant.**
- If the algorith still in this loops, assign the TA on this course and save this assignment on a map.

Cost Function (from OptimalSolver.h)

The cost function calculated on solving output as these conditions:

- **“Cost = penalty – bonus” as below:**

  Assistant with old course experience assigned to a new course -> 5 pts penalty per unexperienced course

- Assistant assigned to 2 courses at the same time -> 5 pts penalty
- Assistant assigned to 3 or more courses at the same time -> 20 pts penalty
- Course requirement not met -> 100 pts penalty
- For each X h/week unassigned assistant time -> X pts bonus

Test Environment

The code runned on main.cpp on 3.generation of intel core i5 with 8 gb RAM on a single core (very old computer) and 2 of the solver algorithms running below 1 second for 50 assistances and 20 courses.

Design choises

All parts of this project designed as object-oriented programming. Assistances and courses are different class and they keep their datas in “data” class as vectors. This implementation make easy to calculation on datas. There is no unnecessary reading data to make calculation. So it enable to gain speed with losing RAM memory.

Also the solver algorithm designed and optimized with minumum conditions to speed up framework.

The another optimization is that when running the solver algorithm, some part of the caluclation of the cost of the solver also being runnig to save memory and to save IO calculations.