Exam Scheduler Application

Hamid Shayestehmanesh

December 10, 2020

Contents

1 Introduction **2**

2 Goal **3**

3 Overview **4**
- 3.1 Input . . . 4
- 3.2 Output . . . 4
- 3.3 Functionality . . . 4
  - 3.3.1 Constraints . . . 4
  - 3.3.2 Database . . . 5

4 Optimization Linear Model **7**
- 4.1 Tables . . . 7
- 4.2 Variables . . . 7
- 4.3 Model . . . 7
  - 4.3.1 Objective . . . 7
  - 4.3.2 Constraints and Explanations . . . 8
- 4.4 Explanation . . . 8

5 Manual **9**

Chapter 1

Introduction

Examinations are essential because they compel students to learn. Without them, most students would not learn much. They would learn only subjects they are interested in and ignore the other topics, though they are significant in the modern age.

The way examinations are held critical. If they are taken too hard or too easy, the results cannot be trusted for judging and ranking students. Institutions should do their best to prepare a proper condition during exams for students to do their bests. However, they may have limitations in resources.

Chapter 2

Goal

Our goal in this project is to design and implement an application for scheduling exams. Many institutions introduce some time slots during a few days or a week to place each exam in these time slots. For example, they will take all the exams in one week, and exams must start at 8 A.M, 11 A.M or 2 in the afternoon. The apparent consideration for scheduling the exams are 1) All the students must be able to attend all of their exams 2) instructors also have to be available on their exams. Thus, an exam should not be held in its instructor's absence, nor two courses with the same instructor should be held at the same time.

Possibly, a student may have two exams in one day or even in consecutive time slots, which feels unfair. This should be considered that universities have different limitation based on their policies. Our goal is to improve students' conditions while considering the policies of an institution. Our soft and hard constraints in this project will be explained entirely later.

Chapter 3

Overview

In this chapter, we will review the input, the output, and the functionality of the application.

3.1 Input

Information given to the system contains all the possible time slots, courses' names, students, professors, and constraints. For the input format, please see the user manual.

3.2 Output

The output is an exam's schedule on the predefined time slots, where it satisfies all the constraints given to the system. The system not only meets the hard constraints but also optimizes soft conditions.

3.3 Functionality

The most important functionalities are explained in the following.

Optimization Our optimization engine is the primary significance of the application. Our engine schedule exams so that all the constraints are completely satisfied, and the soft constraints are minimized.

3.3.1 Constraints

Hard Constrains

Hard Constraints must be met, and if they cannot be satisfied, then the problem has no answer.

- No two courses of an instructor overlap.
- Students can attend all of their exams (No student has two exams at the same time).
- All exams will be held.
- No student has two exams back to back. For example, no one will have an exam on Sunday 8 A.M and Sunday 11 A.M however, one can have exams on 8 A.M and 2 P.M.
- No two major courses of current semester will be in same day, major courses are given to the system as input.

Soft Constrains

Soft constraints are constraints that we are trying to minimize.

- We minimize the number of instances that a student has two exams in one day. If a student has two exams in a day, it is counted as one conflict, but if a student has two exams in a day and two exams in another day, it's counted as two conflicts.
- We minimize the number of instances that a student has two exams in consecutive days.

In this application, we consider the first soft constraint far more critical than the second one; therefore, we minimize the first one and then focus on the second one (This is not how the model is implemented, but it behaves like that).

Conflict Engine There is a function which counts the three types of conflicts in a schedule.

Schedule Manually Rather than scheduling automatically user can schedule the exams manually.

3.3.2 Database

The database contains all the information we need. See the following diagram.

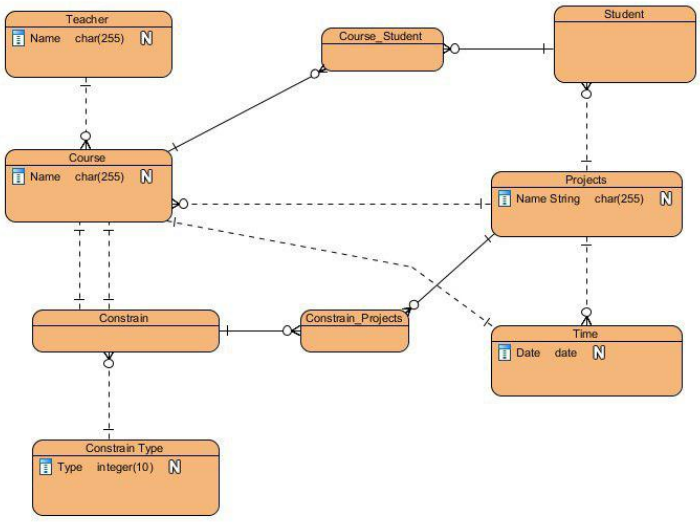

Figure 3.1: Database ERD

Chapter 4

Optimization Linear Model

In this chapter we take a look at the model designed to optimize our problem. C, T are defined as the number of courses and the number of time slots.

4.1 Tables

This table can be constructed from the database.

- **Conf** [**C**][**C**] $=$

number of common students between class i, j	if $i \neq j$
0	if $i = j$

4.2 Variables

Variables values' are assigned by the optimization solver.

- **SD** [**C**][**C**] $=$

1	course i and j being in same day for $i \neq j$
0	not being in same day

- **FS** [**C**][**C**] $=$

1	course i and j being in consecutive days for $i \neq j$
0	not being in consecutive day

- **CT** [**C**][**T**] $=$

1	if Course c_i is decided to be in time t_j
0	if Course c_i is decided not to be in time t_j

4.3 Model

4.3.1 Objective

Min: $\Sigma_{c_1}^{C}\Sigma_{c_2}^{C} SD[c_1][c_2] \times BIGM9 \times conf[c_1][c_2] + \Sigma_{c_1}^{C}\Sigma_{c_2}^{C} FD[c_1][c_2] \times BIGM6 \times conf[c_1][c_2]$

$BIGM9$ and $BIGM6$ are two big numbers which indicates the importance of each variable in our objective.

4.3.2 Constraints and Explanations

We add the following constraints to our linear model.

- $\forall C \quad \Sigma_t^T CT_{ct} = 1$
  Exactly one time is assigned to each exam.

- $\forall c_1, c_2, t$ if c_1, c_2 has conflicts $CT_{c_1 t} + CT_{c_2 t} \leq 1$
  It assures that if two courses have any form of conflicts they will not be on same time. Conflict could, having common students, same instructor, or any other type of conflicts.

- $\forall c_1, c_2, t_1, t_2$ if c_1, c_2 has conflicts and t_1, t_2 are consecutive times in a day, add the following constraints to the linear model.

$$CT_{c_1 t_1} + CT_{c_2 t_2} \leq 1$$
$$CT_{c_2 t_1} + CT_{c_1 t_2} \leq 1$$

  Assures no student has consecutive exams.

- $\forall c_1, c_2, t_1, t_2$ if c_1, c_2 has students in common and t_1, t_2 are in same day

$$CT_{c_1 t_1} + CT_{c_2 t_2} \leq 1 + SD[c_1][c_2]$$

$$CT_{c_1 t_2} + CT_{c_2 t_1} \leq 1 + SD[c_1][c_2].$$

  This helps us count the number of students who have more than one exam in one day.

- $\forall c_1, c_2, t_1, t_2$ if c_1, c_2 has conflicts and t_1, t_2 are consecutive days

$$CT_{c_1 t_1} + CT_{c_2 t_2} \leq 1 + FD[c_1][c_2]$$
$$CT_{c_2 t_1} + CT_{c_1 t_2} \leq 1 + FD[c_1][c_2].$$

  Finally, this helps us count the number of students who have exams in consecutive days.

4.4 Explanation

FD and SD tables help us to calculate our objective cost.

Chapter 5

Manual

Due to the fact that mother tongue of our costumer is Farsi, the user manual is written in a separate document in Farsi.