Automatic Roster Generator

Shashwat Sparsh

20 March 2021

—

MAE 199

—

Dr. Copp

Approach:

General:

The goal of the algorithm is to generate rosters for Capstone Design Projects (MAE 189) based on their preferences and their skillset in relation to their preferred projects. The relationship between their self-reported skillset and the project's preferred skillset is described by a singular affinity score which is calculated for each preference for each student. The sorting algorithm can also be used to generate trends in conjunction with post class surveys; for example, based on peer evaluations, it can be determined whether students on particular projects tend to overstate their skills or if sponsors tend to list requirements highlighted by their preferred skillsets that aren't as impactful on the project itself.

Process:

Step 1: Preprocessing

The excel/csv file containing student information first must be preprocessed, the columns must be renamed and reduced in length and must be rearranged. Care should be taken to ensure that there aren't any missing or empty cells as that edge case has not been implemented yet.

Step 2: Project and Student Construction & Primary Sort

The excel/csv file is imported into the script which then converts it into a Pandas Data-frame for easier and faster parsing. Student Objects are constructed by going row by row through the frame and their preferences are set via the use of a dictionary function. The students are then stored in a list. Projects are then constructed along with their respective max roster sizes as attributes and stored into another list. The student list is then parsed, and students are sorted into their primary preferred projects ignoring the max roster sizes at first.

Step 3: Secondary Sort & Looping

The Project list is then parsed for overflow. When a project is overflowed, two functions can be used. A random sort takes a random student from the roster and moves them to their second preferred project. This is done repeatedly until the project is no longer in overflow. The rest of the projects are subject to the same process. The alternative function looks at the rosters preferences and determines the secondary preferred project that is the emptiest. Then students are moved to that second project instead. If not possible, the process would be repeated with their third preference instead.

Step 4: Affinity Score

After the primary sort, this step would take place instead of Step 3 after implementation. An affinity score is a numerical value that is associated with every student and each preference (if a student has 5 preferences, a score is generated for each of the 5 projects). After primary sort, the Project list is parsed for overflow. When overflow is detected, the roster is then analyzed for student preferences and their respective affinity scores. The student with the highest affinity score for an alternative project within their preferences is then moved to it.

Spring 2021 Steps:

During the spring quarter, the entire algorithm would need to be overhauled with legacy code removed and the process compartmentalized. The primary sort and data frame construction should be turned into functions to avoid confusion and a system should be implemented that allows user input to determine the max size for each project. The old random sort and secondary sort functions need to be re-written as they are not currently implemented and do not function. The algorithm would also need to utilize new functions such as the affinity score generation and affinity score sort as outlined in step 4. Ideally, it would also be able to create graphs and trendlines based on the final rosters to illustrate patterns in students if any.