Progress Report 6

Date: 2020-12-10
Members: Edwin Aldrich, Logan Drescher, Bettina King-Smith

Overview

This week has been exclusively focused on building out the algorithm. We have conducted research, spoken to Professor Armen, and started coding proof of concept tests.

Accomplished

- **Edwin, Logan, and Bettina spoke with Professor Armen.** We met with Professor Armen to discuss our project and received feedback from him on a research direction for our algorithm. Particularly, Professor Armen recommended looking into linear programming (the mathematical concept) and the Simplex Algorithm.
- **Edwin, Logan, and Bettina worked on a course-first based implementation algorithm as a proof of concept.** Before speaking with Professor Armen, we were feeling uncertain of our direction, so we programmed a small proof of concept algorithm for a course registration system that is course-first. The process works as follows:
    - All classes designated as “conflict free”, where the number of students who want the class is less than or equal to the number of seats available, are enrolled in the class.
    - All classes with more students than seats are handled. All students with a higher student priority are enrolled, the remainder are waitlisted. Those that are waitlisted are then enrolled in their next-choice class.
- **Logan conducted introductory research on Linear Programming and the Simplex Algorithm.** Logan learned about linear programming and the Simplex Algorithm and conveyed this information to Edwin and Bettina. While definitely a worthy option, Logan does note that the linear requirement does limit what kind of rules and relationships can be applied.
- **Bettina developed the presentation and abstract. Edwin, Logan, and Bettina met to practice this presentation together.**
- **Bettina conducted research on alternative algorithms.** She learned about Monte Carlo simulations and Tabu search, which she proposes as an alternative to the Simplex algorithm.

The complete potential algorithm list includes:

1. **A student-first system**: the student with the highest priority enrolls first.
2. **A course-first system**: sections where the number of students is less than or equal to the number of seats process enrollment first.

3. **A course-first system (alternate)**: sections where the number of students is greater than to the number of seats process enrollment first.
4. **Simplex algorithm system**: convert requirements (class capacity, student min / max number of credits, etc.) to a linear system of equations and maximize a weighted student happiness.
5. **Tabu table system**: write a simple program - likely some combination of the student-first and course-first system - that will likely produce a relatively good, though not perfect, matchup of students and courses. Calculate the “weighted happiness” score of this system. Calculate the “weighted happiness” score of the related “neighboring” system by randomly changing a couple of student enrollments around. Set the enrollment system to the one with the maximum score. Repeat until the current system is the one with the maximum score.
6. **Some/All of the above:** Realizing the limitations of all the above algorithms, we could potentially try and run some or all of the algorithms above in order to ensure a maximal result. Of course there is no free lunch, and we’ll be sacrificing more time to do so.

Next Tasks

Our top priorities before the next meeting will include:

- **Figure out how to convert our course registration requirements and limitations into standard LP form**. Once we have figured out how to translate our design into the necessary system of equations, we can pass all the relevant info to our algorithm in the proper form.
- **Find and integrate (or implement from scratch) a Simplex Algorithm script**. This will take a bit of time either due to either building wrappers + library testing or implementing code from scratch.
- **(Blocked) Start designing + developing interface b/t Algorithm and backend.** This will allow us to develop our algorithm without having to worry about compatibility with our previous and future work. However, before we can do this part, we need to identify what our algorithm will consist of.