Scheduling Meetings between Teachers and Students based on Availability

Keshane Gan

2016 February 11

Motivation

In one of my organizations, the members must be matched up with other students (candidates) who wish to join. The members give the candidates weekly lessons on one instrument. In order to match members with their candidates, each person both members and candidates submit a set of times during which they are free. Each year, there are between 15 to 20 members and 50 to 100 candidates, so each member should be matched with no more than 5 candidates but no fewer than 4.

There are certain constrains to the member-candidate assignments. Candidates must be taught by a member who is at least one class higher (e.g. a sophomore may be taught by a junior, but not by another sophomore). Graduate students must be taught by a senior undergraduate if there are no graduate students in the organization. Candidates must not be taught by a member whom he or she already knows. And if possible, members should be given candidates with a variey of musical experience to avoid having a member stuck with low-experience candidates.

There are also constraints on lesson times. There may only be one lesson at a time. Lessons taught by one member should be scheduled in blocks of two or three consecutive lessons at a time if possible. A candidate and a member may only be paired if they are both free at the same time.

With so many factors to consider, the members in charge of putting this schedule together often spend several hours doing so. I aim to automate this process and to answer this problem within minutes.

Background

This situation has many features of the assignment problem in combinatorial optimization [1].

In the assignment problem, there are n people to be assigned to n jobs. Only one person may be assigned to a job, and only one job to a person. Any person can perform any job, but each pairing of a person and a job has a certain cost. The goal of the problem is to find assignments for everyone so that the total cost of the assignments is minimized.

In [1], a solution (the Hungarian method) to the assignment problem is discussed. One must simply place the costs into a matrix and perform a series of operations by row or column to arrive at the solution.

In the case of the member-candidate assignments, the constraints can be viewed as the costs of the assignment problem. However, there are a few nuances in this situation that make it difficult to apply the assignment problem directly.

First, there is not a one-to-one mapping of members to candidates. Each candidate is assigned to only one member, but each member must have multiple candidates. Thus, this situation looks more like a generalized assignment problem.

Second, there are numerous constraints. It would be simple to turn one constraint into numerical values and use those directly as the costs, but how does one account for multiple constraints? One solution may be to calculate a linear combination of the constraints to obtain a 1-dimensional value that can be used as a cost, but the correct weights that reflect the importance of each constraint would first need to be found. Another solution may be to evaulate cost as a vector of these constraints and to minimize some function of the vectors.

Third, time availability adds a difficult wrinkle to the problem. Pairings of candidates and members must be scheduled in a way that all pairs can have a lesson during the week. This requirement has the potential to disrupt the initial member-candidate assignments since it is the strictest constraint.

It is entirely possible that this specific situation will not be able to fit a known model for combinatorial optimization. In that case, it might be realistic to use a probabilistic algorithm to estimate at least part of the solution. In the end, it will be likely that a combination of different techniques will be necessary.

Plan

The first step is to explore possible techniques and methods that would help solve this problem. This may involve an adaptation of the Hungarian algorithm and/or other techniques in the field of optimization. Once an appropriate algorithm is found, it will be implemented in Python. I have chosen Python for the ease of following the ideas behind its code; it also has convenient libraries for manipulating data structures. Once the correctness of this program is verified, I will create a web interface so that users can input their information. Since this part is not the core of the project, I will use Google Forms to perform this function and to place information onto a spreadsheet. The spreadsheet can be exported into a csv file for my program to parse.

Deliverables

- A web interface to accept members' and candidates' schedules
- A Python program to perform the matching

- A sample set of user inputs
- An output from the sample inputs
- A final project report

References

[1] J. Munkres, "Algorithms for the Assignment and Transportation Problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32 - 38, Mar. 1975.