# Assigning and Scheduling Teachers and Students

Keshane Gan
Advisor: James Aspnes
CPSC 490
Department of Computer Science
Yale University

May 5, 2016

## 1 Introduction and Motivation

Every school year, the Yale University Guild of Carillonneurs recruit new members to join their group to play a musical instrument called the carillon. During this process, each member of the Guild teaches several students individually how to play the carillon (through a process they call Heel) before those students audition for membership. These lessons are scheduled once a week over nine weeks. Since the founding of the Guild, the responsibility of scheduling these lessons lay in the hands of one Guild member.

The popularity of the Guild has made this responsibility more time consuming and stressful. Several hours are usually spent with one or two additional helpers trying to schedule teachers and students. Several hard [H] and soft [S] constraints must be satisfied:

- [H] Both the teacher and the student have to be available for the lesson time.
- [H] Because there is only one practice instrument available, only one lesson may be taking place at a time.
- [S] Each lesson is 30 minutes long and are assigned to half-hour slots between 8:00 AM and 12:00 AM. For various reasons, some of these slots may have to be made unavailable for lessons.
- [H] Students must be taught by teachers at least one college class year higher. For example, freshman may be taught by a sophomore, junior, or senior, but a sophomore can only be taught by a junior or senior. For the cases when a graduate or professional school student is taking lessons, a Guild member also in the graduate or professional schools should teach the student. The Guild may not have such a member, so a college senior may also be the teacher.

- [H] Students must not be taught by a Guild member whom he or she personally knows.
- [S] Teachers should be given students with a variety of musical experiences.
- [S] All teachers should have roughly the same number of students.

A couple of other important considerations is that the number of members in the Guild has historically been between 15 and 20 and the number of students in Heel has been between 50 and 100. Automating the scheduling process would allow the responsible Guild members to focus their energy and efforts on other important aspects of coordinating Heel. I aim to provide them that automation. Note: for brevity and clarity, members of the Guild who act as the teachers will be referred to as *Guildies* and the students who are taking lessons to learn carillon will be called *Heelers*.

# 2 Background

The nature of this problem resembles several others that are widely known in the optimization field, namely the assignment problem and the interval scheduling problem.

## 2.1 Assignemnt Problem

In the assignment problem, $p$ people must be assigned to $j$ jobs (for now assume $p = j$). There is a specific cost for each person performing a specific job. The goal is to assign people to these jobs so that the total cost of those jobs is minimized.

In [1], Munkres presents an algorithm to find this minimum cost. In his discussion, he represents all the costs in a matrix. This matrix in turn can be represented as a complete bipartite graph. Each element in the matrix can be discussed as the weight of an edge between a vertex in the set of rows $P$ and a vertex in the set of columns $J$. The goal is to turn this bipartite graph into a perfect matching between the two sets of vertices. Using a matrix makes this algorithm easier for computation. For a matrix, the goal is thus to essentially find an independent set of 0s (created by simple subtraction operations) in the matrix. This can be completed in $O(n^3)$ time, which is much faster than the naïve $O(n!)$ solution (where $n$ is either the people or the jobs).

In the current problem, figuring out the cost of a Guildie having a particular Heeler is not straightforward. Some of our constraints cannot be represented solely in a one-dimensional cost. In addition, there are many more Heelers than Guildies. The solution presented by Munkres cannot handle this many-to-one mapping.

### 2.2 Interval Scheduling

Interval scheduling seeks to assign the most tasks in a given time frame, with the parameters that the tasks can be arbitrary lengths and the constraint that no more than one task may run at one time.

A greedy algorithm provides the solution to this problem. On each iteration, the task that will finish earliest should be assigned. All other tasks whose intervals intersect the assigned task must be discarded.

As opposed to the assignment problem, the interval scheduling problem is easier in the current case. Assuming we already have Guildie-Heeler pairs, each of those pairs will likely have multiple times that it is available together. Thus, a pair need not be removed from the set of pairs if it does not end up being scheduled in an interation since there will be more chances for it. In addition, all lessons last the same time, so finding the one with the earliest finishing time is unnecessary. A straightforward comparison of the start time is sufficient.

While the aforementioned problems form the core of the program, their implementations are not. There are many subtle and not-so-subtle wrinkles that have been (and still need to be) solved.

## 3 Implementation

In this section, I will give a broad overview of my implementation and the reasons behind my decisions. The comments in the source code are mostly helpful in explaining the details.

### 3.1 Generating Data

To gather data, I used Google Forms. The service provides a convenient way of viewing the data and exporting it into a tab-separated file. The link to the form is here.

## Acknowledgements

I would like to thank Darien Lee and Alex Carillo for their support and company on my all-nighters. I would also like to thank Megan Brink for her encouraging words. Special thanks goes to Christopher Shriver for the idea and motivation for creating this. Thank you Professor Aspnes for starting me on the right track on this project.

## 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.