Fixing Webtree with Mixed Integer Programming

Matthew Days and **Sarah Hancock**
{madays,sahancock}@davidson.edu
Davidson College
Davidson, NC 28035
U.S.A.

Abstract

At Davidson College, course selection is done through a process called Webtree, in which students submit "trees" that are ordered from root to leaf by preference and are given pseudo-random lottery numbers to indicate in which order relative to other students their preference tree will be traversed. It is performed in multiple rounds and gives preference to students in order of seniority. In this paper, we frame the problem of course selection in as an optimization to be solved via mixed integer programming (MIP). We specifically look to maximize the number of top-choices that students receive based on a scoring system assigned to student preference trees. Our program outperforms the vanilla Webtree implementation in every test, which indicates that, with some refinement, MIP could be a viable alternative the current course selection system at Davidson College.

1 Introduction

For course selection, Davidson College uses a program called WebTree. This program currently relies on a semi-random lottery system to place students into their classes. Students who happen to get a better lottery number will have a better chance of getting into their preferred courses. The current algorithm traverses the student's tree, in order of the student's preferences until it comes across a class with open space, then places the student in that class.

We aim to solve this problem with mixed integer programming (MIP) using Google's OR-Tools (Google 2019b). The solver aims to maximize a score representing the number of students being placed in their most preferred courses. In this paper we review the current WebTree algorithm, present our MIP solution, and compare the performance of these differing scheduling algorithms.

2 Background

WebTree

While the WebTree selection process can be a source for trepidation among the Davidson student body, it actually provides a fairly novel solution to the complicated problem of course selection. In essence, the WebTree selection runs on a simple if-then logic centered around primary and secondary choices for classes. Davidson College frames the logic in the following manner: "You can think of WebTree as, in effect, lining up the whole student body and asking each student a pair of questions. Each question depends on the answer to the one before it. The first question is "what class do you most want?" The second question is "what do you want if that one is unavailable?" Questions like that continue until the student gets one course. The program then goes to the next student in line. Once each student has one class the process repeats until everyone's WebTree choices have been exhausted." (Davidson 2013)

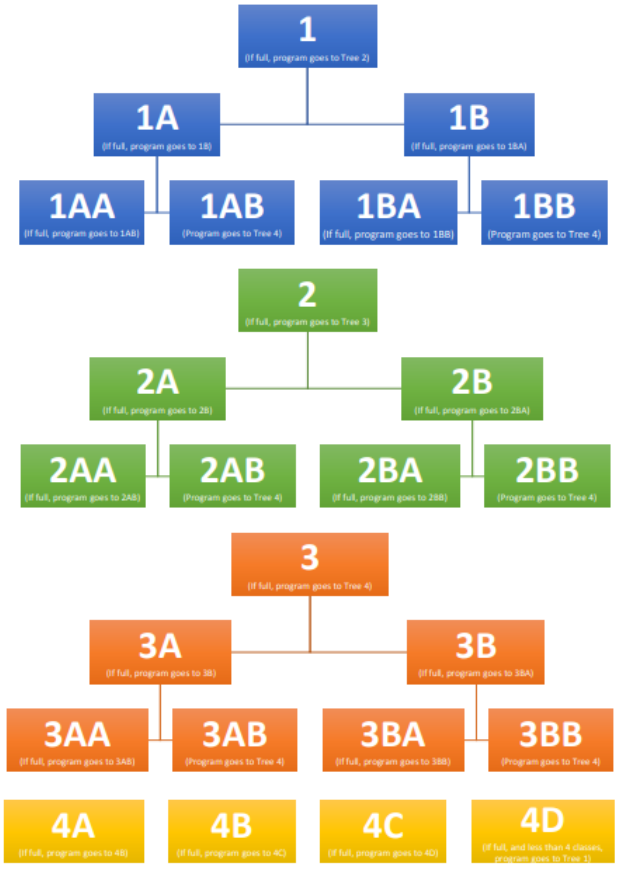

Figure 1: WebTree Logic. Image courtesy of Davidson

Mixed-Integer Programming

For this paper, mixed integer programming (also referred to as mixed integer linear programming) techniques were used in order to model the assignment problem and optimize the number of students who received classes they preferred. Generally speaking, MIP is a variant of integer programming in which some of the variables are restricted to integers while others take on non-discrete values. Additionally, the task of solving a MIP problem is NP-complete (Del Pia, Dey, and Molinaro 2017).

Assignment as a MIP Problem MIP can be applied to a number of problems ranging from scheduling to production planning. In this paper, the authors used the "Assignment as a MIP Problem" example given in the Google OR-Tools documentation as inspiration for modelling the Davidson WebTree problem (Google 2019a). This example attempts to assign teams of workers to various tasks that must be accomplished, with the constraint that each task is assigned exactly to one worker and each team takes on two tasks. This problem uses a matrix of workers and tasks along with a corresponding cost matrix to minimize the cost of assigning workers to tasks given the provided constraints, a structure that lends itself well to the scheduling problem of WebTree.

3 Experiments

In this experiment we used Google's OR-Tools mixed integer programming (MIP) to assign students to their classes. We sought to maximize the number of students that would get into their preferred classes based on where they ranked classes on their tree.

Student Preference Matrix

We create a preference matrix where each row represents a student and each column represents a course. Each element of the matrix is filled with an integer $0 - 7$, relating to the number of diversions away a class is from the student's ideal schedule. An element earns a score of 0 if the course did not appear on the student's tree, otherwise the element earns a score between 1 and 7 depending on where in the tree the student placed the course. Each course in a student's ideal schedule: 1, 1A, 1AA, 4A would earn the highest possible individual score (7) in order to facilitate the maximization of our objective function. Each diversion, or "branch" right on the tree, away from the ideal schedule decrements the given score. For Example. 1A earns a score of 7, while 1B and 2 both earn a score of 6, and 1BB and 2B would receive a 5, and so on.

Assignment Matrix

The assignment matrix mirrors the shape and size of the student preference matrix, with the same ordering of students and courses. The solver fills each element of this matrix with a BoolVar, designating whether a student is placed into that class or not: 1 if a student is placed into that class, and 0 if not.

Constraints and Objectives

WebTree gives each student up to 4 classes, a constraint we mimic in our MIP solver. The sum of each row of the assignment matrix must be less than or equal to 4, ensuring that no student gets an excessive number of courses. Each class has a capacity on the number of students it can seat, thus another constraint involves enforcing the class caps. The sum of each column of the assignment matrix has to be less than or equal to the corresponding course's class capacity. Our objective is to maximize a score based on students preferences for the classes they receive.

Scoring

For each student, we isolate both of their rows, or vectors, from the assignment matrix and student preference matrix. The dot product of these two vectors gives the student's personal score, which we then proceed to weight with the student's scalar class multiplier. These multipliers, chosen in order to reflect WebTree's seniority preference, are as follows: Seniors: 2.00, Juniors: 1.75, Sophomores: 1.50, First-Years: 1.25, and Other (individuals who did not fit within these classifications, i.e. auditors): 1.0. Thus, our final score is the sum of all student's individual scores. We then use this score to compare our performance to the original WebTree algorithm.

4 Results

Results without Seniority Weights		
	WebTree	MIP
Fall 2013	22973.00	34854.00
Spring 2014	22677.00	34133.00
Fall 2014	22826.00	34754.00
Spring 2015	23781.00	36111.00

Figure 2: Performance of WebTree vs MIP based on student preference of assigned courses, without seniority preference.

Results with Seniority Weights		
	WebTree	MIP
Fall 2013	36777.00	55859.25
Spring 2014	36239.50	54389.75
Fall 2014	36683.50	55283.25
Spring 2015	38084.00	57493.00

Figure 3: Performance of WebTree vs MIP based on student preference of assigned courses, with seniority preference

The results of our various experiments are displayed above. As is evident, our MIP approach to student schedules outperforms WebTree by a similar window in each round of experiments (about 12000 unweighted for seniority, and 20000 weighted for seniority, according to our scoring method). In scoring student class preferences based on deviations from the desired WebTree path, our model prioritizes choosing classes along the desired path on a lower

tree regardless of conflicts. Such a disregard for conflicting classes comes in part from the fact that students made their selections based on WebTree's logic rather than that of a MIP problem. However, prioritizing pure score maximization over all else means that students could end up in a different section of the same class if that nets them a higher score according to our scoring system. This highlights another mitigating circumstance that could explain the performance gap between MIP and WebTree, specifically the scoring method itself. The multipliers were chosen to capture the seniority preference of WebTree; however, their affect is exaggerated by their various magnitudes. For instance, with any given class, the MIP approach is much more willing to sacrifice the preferences of a first-year to give a senior a desired class, not taking into account any restrictions (say, it could only be open to first-years).

5 Conclusions

In this paper we presented an alternative algorithm for WebTree scheduling. This algorithm approached schedule assignment as an MIP problem. According to our objective function, which maximized a score based on students course preferences and the classes they received, our algorithm consistently scored higher than WebTree in assigning students to or near to their ideal schedule. As such, MIP could pose a viable alternative to WebTree given further refinement. Aside from modifying the weights that model seniority preference, one avenue we would explore involves giving priority to declared majors when selecting in-major classes.

6 Contributions

Both authors collaborated heavily in modelling the problem and planning the approach to its solution. S.H. designed and implemented the scoring function, while M.D. wrote the code to fit the MIP model and setup the experiments. Both authors ran the experiments, and wrote and proofread the document collaboratively.

7 Acknowledgements

The authors of this paper would like to thank Dr. Ramanujan for providing insight and guidance with this problem, and Ryan Strauss for nudging us in the right direction with our example model.

References

Davidson. 2013. Webtree logic. https://www.davidson.edu/offices/registrar/course-registration-and-webtree. Retrieved on Mar. 14, 2019.

Del Pia, A.; Dey, S. S.; and Molinaro, M. 2017. Mixed-integer quadratic programming is in np. *Mathematical Programming* 162(1-2):225–240.

Google. 2019a. Assignment as a mip problem. https://developers.google.com/optimization/assignment/assignment_mip. Retrieved on Mar. 14, 2019.

Google. 2019b. Google or-tools. https://developers.google.com/optimization/. Retrieved on Mar. 14, 2019.