Math 480 Project Draft
Captain Scheduling for Ride the Ducks

Zach Bickel and Michael Yoon

March 2, 2015

Abstract

This paper describes the model and solution we derived from a scheduling problem for a small business, Ride the Ducks of Seattle. The objective was to create scheduling software that not only created a schedule, but also satisfied the wants and needs of the company as well as the workers. The problem is modeled as a Mixed Integer Problem and fitted with a branch and bound algorithm. After simplifying the problem at hand to fix the model we wanted to use, we were able to derive decision variables and a cost function to minimize. We used a two-step process. First, we began with a schedule creator with no branch and bound algorithm implemented simply to satisfy the constraints of the workers and the company. We then implemented the branch and bound algorithm to obtain a feasible solution with a better score than the original solution. Our finished product will be used by Ride the Ducks with the option of changing constraints to fit their week-to-week schedule making.

Goal

Scheduling problems are very common for small businesses where many managers are still doing it by hand. This can lead to hours of tinkering with the schedule to get the most out of the workers. Ride the Ducks of Seattle is a major tourist attraction that takes their customers on a tour of Seattle on land and water. A captain directs each tour; the captains are the ones who operate the vehicles, known as ducks, and gives the tours. The ducks are the pride and joy of the company. These vehicles are capable of going on land to water and vise versa. Each tour is 2 hours long, which includes about 90 minutes of touring with a 30-minute break for the captains. The amount of tours run depends seasonally with the warmer seasons needing more tours than the cooler seasons. Scheduling the captains and their ducks in a way that satisfies working conditions by the company as well as the individual captains can be tricky. Management at Ride the Ducks of Seattle currently spends upwards of ten hours per week scheduling captains for tours. Ten hours creating a schedule is much too long, we believe we can not only create a schedule for Ride the Ducks much faster, but also create a schedule that allows the company to run tours efficiently while satisfying the captains and the companys needs. Our goal is to build scheduling software that will drastically cut time down and automatically build a schedule. This will be flexible enough so that a user will

be able to update the constraints as they come up. Ride the Ducks of Seattle is a very private about the details for how their company is run. They keep much of their data to themselves to protect the company from those that want them gone. Due to their privacy, the notion of making the scheduling software easy to edit and make changes to is crucial. The company can then make their own schedules with the data that they might not have given us. With this schedule creator, we hope that Ride the Ducks will spend less time working on scheduling the captains and spend their newly acquired time efficiently.

Simplifications

Ride the Ducks is open 7 days a week from 9 AM to 9 PM and offers 90-minute tours where the driver ideally gets 30 minutes of rest before the next tour. There is one captain and one duck per tour. Each captain is not supposed to work more than four days a week and there are special constraints for individual captains, such as Captain A cannot work on Tuesday. Our objective is to minimize the total number of 4-day stretches all captains have to work. Minimizing these 4-day stretches is a way to keep the workers rested and keep hours even among the captains that need them. We started by figuring out what a Ride the Ducks schedule looks like. The schedule is broken down into which captains are working what days and the times that day that captain will be running a tour. The schedule also gives one or two captains that are on-call so that if a captain has a conflict day of or more tours than expected or scheduled the captains on-call can come in and help. From the schedule given by the company we were able to break down the schedule into decisions. The decisions include, how many captains are available, how many captains we want in a day, how many time slots for tours, and a captains own restrictions on which days he or she can work. These decisions we believe are the most important to creating a schedule. With these decisions we made assumptions to certain aspects of the variables involved. For the tours, we capped the total length at 120 minutes which incudes the tour and rest time for the captains. The time slots for those tours are set to a default of one tour every 10 minutes. The number of captains in our problem is 40, with 22 captains scheduled a day, 2 of which are on-call. We simplified each block of tours to be 6 blocks of tours per day. A block of tours is the 2-hour period in which a captain is out on a tour. So, if the company is open for 12 hours a day, there are 6 blocks of tours.

Literature Overview

Before we tried to model the problem at hand, we read research papers on modeling problems similar to our own. One paper we came across discussed America West and the Arizona State University using a mathematical model to decide on the best airplane boarding strategy (Hogg, 2002). This paper gave us insight on handling mixed integer problems and how they are modeled. We were able to better derive the decision variables in our own problem after learning about how the ASU and America West team simplified their problem. The ASU and America West team simplified the problems of boarding an airplane by classifying the problems into two groups (Hogg). This helped us to understand how to simplify our own

constraints as well as which decision variable we wanted to minimize. We came across Patrick Perkins paper on creating weekly timetables for the Math Study Center at the University of Washington (Perkins, 2004). Perkins paper tackled their scheduling problem in a two step process. The first step of their modeling process utilized a relaxed version of constraints to just obtain a valid schedule. The second step involved stricter constraints that influenced the objective function. This second step would go on to produce better schedules. Perkins states that these better schedules are better optimized and thus produce more efficient schedules, 7Another paper we read was by Nicholas Beaumont on scheduling staff that drove to and serviced customers (Beaumont, 1997). In Beaumonts case study, the staff he was trying to schedule had many similarities to the staff we were trying to schedule. This included constraints on the staff based on times they could work and the demand of their work (Beaumont). Using binary values and other variables assigned to certain staff members to restrict their start times was something that we wanted to implement in our model. These constraints may change by week for our own partner, thus knowing the formulation of these constraints was necessary to write up code that would create these constraints for our partner to utilize once we are done with the project. Similar to this, the demand of the staff in Beaumonts study and our own problem of varying numbers of tours based on the time of year. These papers offered us guidance on our own problem and model. Simplify the process; solve the problem two ways, one more relaxed and the other fit to tighter constraints, and how to model certain restrictions that may change based on the situation. All of the ideas listed we utilized to give our partners the best possible solution.

Decision Variables and Parameters

First, let's define some parameters:

$$\begin{aligned} n &\equiv \text{Number of captains,} \\ m &\equiv \text{Number of tour slots per 2-hour blocks in one day,} \\ b &\equiv \text{Number of 2-hour blocks in one day,} \\ k &\equiv \text{Number of time slots per day.} \end{aligned}$$

If there are n captains and k time slots in a day, then there are $7kn$ decision variables. Note that $k = mb$. Our decision variables will represent when the captains will start their shifts. We can understand them as a matrix X, where $x_{ijd} = 1$ if captain $i \in \{1, 2, \ldots, n\}$ begins their shift at time $j \in \{1, 2, \ldots k\}$ on day $d \in \{1, 2, \ldots, 7\}$ and 0 otherwise.

Objective Function

Our objective function will represent the number of 4-day stretches captains have to work and our goal will be to minimize this in order to keep captain fatigue low. For readability purposes, let's define a separate function $w(i, d)$ to represent captain i working on day d:

$$w(i, d) = \sum_{j=1}^{k} x_{ijd}.$$

So, $w(i,d) = 1$ if captain i works on day d and 0 otherwise. Then $w(i,1)w(i,2)w(i,3)w(i,4) = 1$ implies that captain works on days 1, 2, 3 and 4. Our objective function f is thus:

$$f(i) = [w(i,1)\cdots w(i,4)]+[w(i,2)\cdots w(i,5)]+\cdots+[w(i,4)\cdots w(i,7)]+[w(i,5)w(i,6)w(i,7)w(i,1)].$$

Note that the last term in the function wraps around to make sure that we are not loading the schedule too heavily towards the weekend shifts.

Constraints

(1) Number of daily captains constraint: We want to have c_d captains at work on day d.

$$\sum_{i=1}^{n}\sum_{j=1}^{k} x_{ijd} = c_d.$$

(2) Multiple start times constraint: The captain can only have one start time for each day. For each day d, and for each captain i,only

$$\sum_{j=1}^{k} x_{ijd} \leq 1.$$

(3) Number of days worked per week constraint: Each captain can work a maximum of 4 days per week. Some have to work less, so we'll define the maximum number of days captain i can work as y_i. For each captain i,

$$\sum_{d=1}^{7}\sum_{j=1}^{k} x_{ijd} \leq y_i.$$

(4) Tours per time slot constraint: We would like to have a certain number of captains available at each time slot in order to run the required number of tours. If we want t_{jd} tours at time slot j on day d we have to first obtain the number of captains that are available for a tour (i.e. haven't run 5 or more tours in a day already). Define captain i starting a shift or eligible for the tour j on day d by z_{ijd}.

$$\sum_{i=1}^{n} z_{ijd} = t_{jd}.$$

(5) Captain unavailability constraint: If captain i is unavailable on day d,

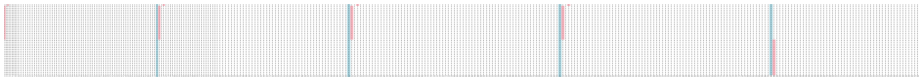
$$\sum_{d=1}^{7}\sum_{j=1}^{k} x_{ijd} = 0.$$

(6) Captain required tour constraint: Some captains are requested to run specific tours, so if captain i has to work day d on time slot j,

$$x_{ijd} = 1.$$

Results

As of this writing, the code has been fairly buggy and we haven't been able to obtain a feasible solution. Sample output from our code is below and it is formatted to show the starting schedule visually.



What this shows is that the captains are front-loaded on each day and none of the tours per time slot constrains are satisfied. Further improvements of our algorithm are needed to obtain a solution.