# Midterm Report

Client: North America Central School Bus

Aden Benson, Aiden McCoy, Humza Khan, Jonathan Wu

June 8, 2025

## 1 Introduction

North America Central School Bus is a leading private provider of transportation services for public schools. The company primarily serves school districts, offering two main types of routes: regularly scheduled ("static") routes, such as consistent student pick-up and drop-off throughout the school year, and field trip/charter routes. Our focus is on the assignment of charter routes, which vary from week to week and are time-consuming for the business. At present, the process is not automated and is handled on paper.

In the status quo, the client's regional managers assign routes based on the region's collective bargaining agreement (CBA) with their drivers. The process starts with a manager posting a sign-up sheet for charter routes. Drivers bid for the routes they want. Managers manually assign routes typically based on seniority (measured by tenure) and additional rules outlined in the CBA. Each CBA has unique rules. If assignment rules are violated, drivers can file grievances. If upheld, both the driver who received the route and the one who should have received it are compensated, leading to double payment and significant time consumption for managers. Beyond the financial cost, this process undermines the company's trust and culture.

### 1.1 Proposed Solution

Figure 1 shows the proposed solution in visual form, with automated elements in blue and manual elements in green. We propose a two-part solution, each with two subparts: a front-end online form and a backend optimization model. The front-end consists of an online form and a web app for the front-end interface. Drivers utilize the form to submit bids, and managers use the web app to make assignments. This process allows the company to automate many time-consuming aspects of route assignments. Additionally, this approach provides flexibility and replicability for both the user interface (UI) and the backend logic.

The backend optimization model is based on the Gale-Shapley matching algorithm that takes in driver preferences and route preferences (represented as the seniority order here) and returns a set of matches with beneficial mathematical properties that ensure the solution is optimal [1]. Our model consists of an iterative process, with each iteration having a preprocessing step and a Gale-Shapley allocation step described in Section 2.2. The preprocessing step will remove bids that drivers cannot complete, for reasons such as time conflicts and hour limits. These steps assign routes to drivers one at a time in compliance with union rules.

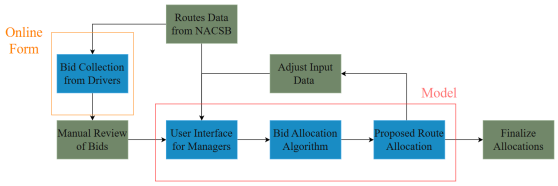


Figure 1: Proposed structure of bid process, with automated processes in blue

### 1.2 Background Research

We considered two main approaches: (1) linear programming, which systematically accounts for drivers' varying levels of interest in routes and management's priorities; (2) the Gale-Shapley matching method, which ensures that drivers receive stable assignments where no two drivers would prefer to swap routes. Because scheduling rules and seniority add complexity, the Gale-Shapley method can be enhanced by additional iterative steps to handle real-world issues like overlapping routes or hourly limits. Combining these approaches creates a practical solution that satisfies both drivers and management, reducing dissatisfac-

tion and improving operational efficiency.

### 1.3 Benefits

Our project should be able to streamline the work that managers do to assign routes to people. This should save company time. Furthermore, we plan to have outputs which allow the client to have increased transparency. This protects our client, and improves the trust and working culture in the locations.

## 2 Analysis

### 2.1 Data and Software Resources

All data used in this project has been provided directly by the client, specifically from their Carol Stream regional depot. The provided data includes a collective bargaining agreement (CBA) that outlines bid assignment rules the client must follow, their current paper bidding process, digital spreadsheets of static routes, and digital spreadsheets of chartered routes. All provided datasets were anonymized before our analysis to comply with confidentiality standards. All software used is either within the client's digital infrastructure or freely available.

#### 2.1.1 Data Overview

The CBA for the Carol Stream region outlines business practices, particularly around Work Assignments and Route Assignments. This outlines the process that the client currently uses when assigning bids and provides us with the standards that our final implementation must follow to avoid costly grievances from employees.

Currently, the client manually performs the bid assignment process on paper, where drivers fill out preference forms and area managers assign routes based on this stack of paperwork. These documents gave our team examples of the inefficiencies that the client currently exhibits in the assignment process. We will use these examples to design the digital solution that imitates the process drivers currently use, but making assignments much easier for contract managers. This includes a seniority list of drivers. The higher an individual ranks on the seniority list, the greater priority their preferences receive.

The static route data provided by the client contains a digital spreadsheet with each driver's assigned regular routes. The spreadsheet contains driver identification numbers, specific departure and return yards, and defined task start and end times. Importantly, this spreadsheet adds buffer periods for pre-trip and post-trip activities like inspections and walking time, giving realistic constraint parameters for our model.

The charter route data was also provided to us in a digital spreadsheet, containing pickup and return times, route destinations, and the number of buses required for each charter. However, this dataset lacks depot entry and exit times, which are included in the static route data. Because we don't have any measure of driving time from the Carol Stream depot to the charter destination before pick-up and after drop-off, we are currently working with the client to calculate estimated parameters for these values.

For model development (described in Section 2.2), all data will be converted into .csv files and structured like Figure 2, which shows an entity-relationship diagram. Each table represents one of the .csv files and the associated data. We assume static routes will match an output similar to one from ByteCurve. A column that we are missing is between the driver and the assigned static route. Another intricacy of data input is if the route is designated as a "drop pick" where the driver does not stay, thus letting the route be split amongst two drivers.

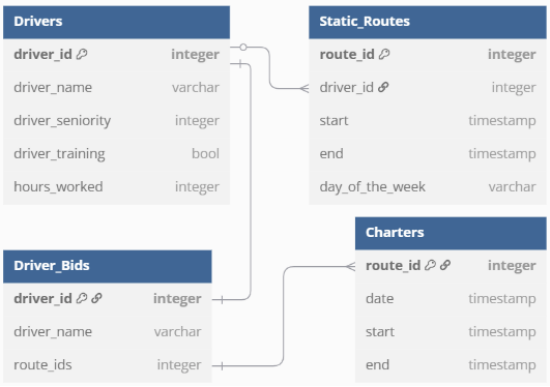


Figure 2: Structure of the various objects throughout the project

#### 2.1.2 Software Resources

For the front end of our solution, we are developing a user-friendly web interface utilizing Microsoft's Web App Service. Microsoft was chosen due to its compatibility with the client's existing infrastructure. This web application will emulate the current paper bidding form in a digital format, allowing drivers to easily type in their preferences and review them before submission. It also provides area managers an organized view of their submissions, as the form enters data directly into a Microsoft Excel spreadsheet.

On the backend, we are using Python integrated with Shiny's dashboard generator to allow managers to

assign routes. We envision that the final version of this application can take in driver route preferences, driver static routes, and driver seniority numbers as .csv files, perform the Gale-Shapley allocation outlined in Section 2.2.1, and output all assigned routes. The outputs will consist of a simple allocation list that the manager can provide to their employees, along with diagnostic results detailing why each employee was assigned their charter. This increases transparency in the company and ensures no disputes between managers and drivers regarding union regulations. We also will add the ability to edit assignments after the allocation process. This addition will prevent unusable solutions in the case of bad data entry, such as incorrect static routes being entered that inaccurately interfere with a chartered route.

In the future, we plan to integrate model outputs with the Bytecurve interface, a school bus operating system that the client uses for dispatch. This integration will enable our solution to automatically communicate with the client's existing operational data.

## 2.2 Modeling and Analysis

Any proposed model must provide an optimal allocation of routes while following the various union and mechanical constraints that apply to the assignment process. In order to achieve this goal, we have proceeded in an iterative manner with four candidate models, each building off of the previous one(s). Figure 3 illustrates this process.

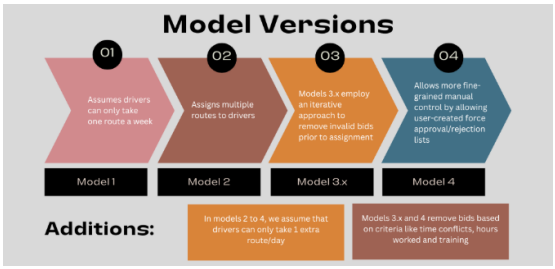


Figure 3: Models 1-4, with included features

The current model as of writing is Model 3.x. Sections 2.2.1 and 2.2.2 discuss the various features included in Model 3.x, as well as the justifications for why each feature is necessary and what problems they address. Sections 2.2.3 and 2.2.4 discuss the proposed features and procedures for Model 4.

### 2.2.1 Allocation

The core of all four models is the Gale-Shapley algorithm. This algorithm is designed to solve a matching problem, where one group has preferences over another group, and vice versa. In this case, we have drivers with preferences (bids) over routes and charter routes that have preferences (seniority order of the driver) over drivers. Additionally, the Gale-Shapley algorithm is especially valuable for two main reasons: (1) it returns a solution that is mathematically guaranteed to be optimal; (2) it replicates a major step used in the current allocation process, allowing us to capture the benefits of a linear programming approach while respecting the CBA.

The specific beneficial mathematical properties are proved in [1] and [2]. One key result is Pareto Efficiency: there is no way to assign the routes to make someone better without making someone worse off. This property implies that no one would want to swap routes, meaning that we have an allocation everyone is at least reasonably happy with. Another implication is that this model is "strategy-proof": drivers can't collaborate to game the allocation - a key concern for management.

The second reason is that the algorithm is compliant with the union rules for allocation of bids. The union mandates a round-robin process, with the most senior driver going first. The bids of each driver are iterated through to see if the charter route has not been taken, and if open, assigned to the driver. The process repeats in seniority order for all drivers. Lastly, every driver must either be assigned a charter route or exhaust all their bids before moving to the next driver. Gale-Shapley operates similarly, as each driver will propose (bid) on each charter route, and if the route is open, the driver will receive it. Gale-Shapley then proceeds down the seniority list, assigning a route or exhausting bids for each driver.

One concern with the Gale-Shapley algorithm is the deferred acceptance process, as it may switch a driver after they have been assigned, causing a violation of union rules. We can circumvent this process because all routes have the same preference: the seniority list. This means there will never be a switch. A brief proof is as follows:

Suppose we have driver $i$ and driver $j$, where driver $i$ has a higher seniority ranking than driver $j$. Additionally, suppose we have a tentative acceptance, resulting in a switch; that is, driver $j$ is assigned route $a$, which is then switched to driver $i$. For this to happen, driver $i$ must either pass over route $a$ and be assigned a lower preference route, or be assigned a higher preference route. However, the former scenario cannot happen, since if driver $i$ could have done route $a$, they would have been assigned it before driver $j$, since we descend in seniority order. The latter scenario also cannot happen, since it would mean that

driver $i$ already has a route assigned on this iteration and is not eligible for an additional route, so they cannot have also been assigned route $a$. Thus, we have a contradiction, and the tentative acceptance scenario is not possible.
For the initial Model 1, we define the following ranking matrix:

| | $R_1$ | $R_2$ | $R_3$ | $R_4$ | $R_5$ | ... |
|---|---|---|---|---|---|---|
| $S_1$ | 3 | 2 | 1 | – | 4 | ... |
| $S_2$ | 2 | 3 | 1 | – | 3 | ... |
| $S_3$ | 2 | – | 1 | – | – | ... |
| $S_4$ | – | 1 | – | 2 | 3 | ... |
| ... | ... | ... | ... ... | ... | ... | ... |

Table 1: Ranking matrix with assigned routes

where $S_1, S_2, \ldots$ is the driver corresponding to that location in the seniority order. $R_1, R_2, \ldots$ is each route in some arbitrary order. We use specific numbers here for illustration purposes, but this process is the same for arbitrary preferences, routes and drivers. Each row of the ranking matrix thus corresponds to the bids or preferences of each driver. As stated earlier, we proceed down each row, assigning routes based on preference order. The outcome of this ranking matrix would be:

| | $R_1$ | $R_2$ | $R_3$ | $R_4$ | $R_5$ | ... |
|---|---|---|---|---|---|---|
| $S_1$ | 3 | 2 | $\boxed{1}$ | – | 4 | ... |
| $S_2$ | $\boxed{2}$ | 3 | ~~1~~ | – | 3 | ... |
| $S_3$ | ~~2~~ | – | ~~1~~ | – | – | ... |
| $S_4$ | – | $\boxed{1}$ | – | 2 | 3 | ... |
| ... | ... | ... | ... ... | ... | ... | ... |

Table 2: Ranking matrix with assigned routes

As shown in Table 2, the most senior driver will receive their first preference. We then proceed to the second-most senior driver. Their first preference has already been assigned, and so they receive their second preference. The third-most senior driver has no valid preferences, as they did not bid on any of the available routes remaining, and are assigned no route. The fourth-most driver receives their first preference. However, this process is a one-shot procedure, as it only assigns one charter route to each driver and then terminates. We can improve this model by relaxing the one-charter-route-per-week assumption and moving to a one-charter-route-per-day assumption. To incorporate this improvement while being compliant with union rules, we employ an iterative approach dubbed "Iterative Gale-Shapley". The process is as follows:

Step 1: Run one iteration of Gale-Shapley

Step 2: Remove all routes that were assigned to a driver, and all drivers whose bids have been exhausted

Step 3: Repeat Steps 1 and 2 until all routes have been assigned or all bids have been exhausted for all drivers

This process maintains the round-robin procedure and ensures that everyone gets a "first helping" before seconds, as stated in union rules. To see the Model 2 formulation, suppose we have the same ranking matrix as Table 1. We then run one iteration of Gale-Shapley and receive the ranking matrix in Table 2. We then apply Step 3 here to get the following ranking matrix:

| | ~~$R_1$~~ | ~~$R_2$~~ | ~~$R_3$~~ | $R_4$ | $R_5$ | ... |
|---|---|---|---|---|---|---|
| $S_1$ | 3 | 2 | $\boxed{1}$ | – | 4 | ... |
| $S_2$ | $\boxed{2}$ | 3 | ~~1~~ | – | 3 | ... |
| ~~$S_3$~~ | ~~2~~ | – | ~~1~~ | – | – | ... |
| $S_4$ | – | $\boxed{1}$ | – | 2 | 3 | ... |
| ... | ... | ... | ... ... | ... | ... | ... |

Table 3: Ranking matrix with drivers and routes slated for removal

We remove driver 3, as their bids have been exhausted, and routes 1, 2, and 3 as they have been already assigned. We then repeat steps 1 and 2 to get the following ranking matrix:

| | $R_4$ | $R_5$ | ... |
|---|---|---|---|
| $S_1$ | – | $\boxed{4}$ | ... |
| $S_2$ | – | ~~3~~ | ... |
| $S_4$ | $\boxed{2}$ | ~~3~~ | ... |
| ... | ... | ... | |

Table 4: Final ranking matrix after two iterations

Driver 1 receives route 5 in addition to their previously assigned route, driver 2 exhausts their bids and driver 4 receives route 4. The process then terminates since all bids have been exhausted. Note that if we had left-over routes, we would still terminate the process since no drivers would be willing to complete those routes.

### 2.2.2 Pre-Processing

One major issue with both Models 1 and 2 is that both rely on sanitized inputs; that is, every route the driver bids on is valid. While the one-charter-route-per-day assumption prevents time conflicts across

newly assigned routes, and Gale-Shapley automatically handles the removal of already assigned routes, there is no check for time conflicts with the standard routes that the drivers run. Additionally, there are various other reasons that a bid could be invalid. Here we focus on a few, specifically, time conflict with a standard route, inability to complete the route without using overtime, and lack of training. To account for these circumstances, we introduce a preprocessing step. This step checks the bids of each driver for the above conditions and removes them if they are found to be invalid. This step leads us to Model 3, which now uses the following procedure:

Step 1: Preprocess bids by checking for time conflicts, hour limits, and training

Step 2: Run one iteration of Gale-Shapley

Step 3: For each assigned route, remove same-day bids from the assigned driver, remove that route from other drivers' bids, and eliminate drivers with no remaining bids

Step 4: Repeat Steps 2 and 3 until all routes are assigned or no drivers have remaining bids

This procedure ensures that invalid routes are caught and prevented from being assigned.

However, one concern still remains: what exactly is the definition of "start" and "end" time? One might consider the time the driver picks up the passengers as the start and drops them off as the end. But when scheduling charter routes, we must consider the travel time to and from the standard routes in addition to direct time overlaps. The travel time is a highly variable quantity and is hard to estimate without complicated calls to real-time data or extensive historical data. However, the local managers and dispatchers know these times. Thus, we allow users to input padding or buffer time before and after the standard routes. This, plus the preprocessing step, leads us to the current formulation of Model 3.

Suppose we have the same ranking matrix as Table 1 and additionally suppose that route 1 conflicts with driver 1's existing schedule once a 30-minute buffer is applied, that driver 2 does not have enough hours to complete route 5 and that driver 4 is not sufficiently trained to complete route 5. Finally, assume that routes 2, 3, and 4 are on the same day, and all other routes are on different days. We run the preprocessing step and get the following ranking matrix:

| | $R_1$ | $R_2$ | $R_3$ | $R_4$ | $R_5$ | ... |
|---|---|---|---|---|---|---|
| $S_1$ | ~~3~~ | 2 | 1 | – | 4 | ... |
| $S_2$ | 2 | 3 | 1 | – | ~~3~~ | ... |
| $S_3$ | 2 | – | 1 | – | – | ... |
| $S_4$ | – | 1 | – | 2 | ~~3~~ | ... |
| ... | ... | ... | ... ... | ... | ... | ... |

Table 5: Ranking matrix after preprocessing

If we run one iteration of Gale-Shapley and remove invalid bids, we get:

| | ~~$R_1$~~ | ~~$R_2$~~ | ~~$R_3$~~ | $R_4$ | $R_5$ | ... |
|---|---|---|---|---|---|---|
| $S_1$ | ~~3~~ | ~~2~~ | $\boxed{1}$ | – | 4 | ... |
| $S_2$ | $\boxed{2}$ | 3 | ~~1~~ | – | ~~3~~ | ... |
| $S_3$ | ~~2~~ | – | ~~1~~ | – | – | ... |
| $S_4$ | – | $\boxed{1}$ | – | ~~2~~ | ~~3~~ | ... |
| ... | ... | ... | ... ... | ... | ... | ... |

Table 6: Ranking matrix after preprocessing, one round of Gale-Shapley and with slated removals

In addition to the normal removals from Table 3, we remove the routes that are on the same day as another assigned route. The iterative process continues as specified in Model 2. One thing to note is that the buffer added here affects the removal of bids. For example, it is possible that driver 1 is able to complete route 1, but not with a 30-minute buffer added.

### 2.2.3 Output Alteration

However, the buffers provided could likely be wrong, or be outdated to account for changing local conditions. Thus, after we provide an allocation, the user may need to fix the output if the time buffer assumption is incorrect. For example, the assumption that the travel time from one route to the next is constant for all drivers and their bids may be a faulty assumption. As this is likely to occur, we will need to allow the user to input overrides. We propose the following procedure based on our client's input. After the output the client will be able to see the displayed output. The override procedure would be as follows:

Step 1: The user is prompted with two options: one to override an existing assignment, and another to override its removal due to a time conflict.

Step 2: The user inputs a driver id and route id pair, for the route they wish to force approve/reject.

Step 3: The assignment runs again, but the bid is either forced to be included during the first

step of preprocessing described in 2.2.2 or removed during the same step.

Step 4: The user gets the output back with this new allocation and is prompted again, repeating steps 1 to 3 until the faulty assignments are removed.

Step 5: The user saves the output.

Notably, we force each iteration to change only one assignment. This is because we anticipate the first faulty assignment will lead to many other assignments to change especially early on in the assignment process.

#### 2.2.4 Saving data + Model Interpretation

A major focus of our client's needs is transparency of the results. We plan to have an output the client can download which will be an Excel file with the bid assignments for each charter route. This "diagnostic sheet" will contain the status of each bid for the drivers, indicating whether the route was assigned, or rejected and why. We will work with the client on the structure and content of this sheet, particularly whether to split or combine the results across drivers and the granularity of the bid statuses.

## 3 Future Steps

Moving forward, our team plans to build out Model 4, the Microsoft web app, and the charter coordinator user interface for route allocations.

One of our main priorities with the model is to test its ability to detect time conflicts for drivers between their existing or previously assigned charter routes and any new, overlapping routes they bid on. As noted by the charter coordinator at the Keller, Texas depot, there are instances where bus drivers may bid on routes that technically overlap with their current ones but still fit within the window between the pick-up and drop-off times of their original charter/static route. In such scenarios, the driver would complete the initial student drop-off, handle the additional charter route if the timing works, and then return to transport the first group of students back to their original location. To make this work, we determined that a 30-minute gap between the drop-off of the initial route and the pick-up of the next is enough to allow a driver to fulfill the additional route. To account for the variable nature of drive times, we would also like to add an option within the web app that allows the charter coordinator to approve or remove a bid before its assignment.

Additionally, we plan on developing the pre-processing step found in Model 3.x and Model 4, which implements bid removal functions such that the algorithm automatically removes driver bids from routes for a specific reason (i.e., lack of training, time conflicts, seniority). The pre-processing step would be run before each round of Gale-Shapley to remove any routes newly assigned to a driver during the previous round.

According to the collective bargaining agreement at the Carol Stream location, drivers who miss routes are not allowed to bid for the following week of charters and field trips. However, since the data doesn't clearly show when a driver missed a route and thus was prevented from bidding, we plan to discuss this with the charter's coordinator. Establishing that removing drivers from the assignment process requires the manual effort of the charter coordinator is an imperative clarification to make with our client.

Another top priority as we move forward is to integrate the Python model with our front-end application. NACSB employees will use this to retrieve the driver preference responses from the Microsoft Form, upload its corresponding CSV file, along with seniority data and route information, run the Gale-Shapley algorithm, and output the assignments. To handle all these tasks on a single device, our team is looking into packaging the Python file/necessary packages and the web application into an EXE file, which will be compatible with any Windows computer.

To conclude, our team's primary goal is to create a fully functioning application for the Carol Stream location that is user-friendly and aligns with their current initial assignment process. We aim to create an application that minimizes any interaction with the Python code unless there is a desire to "upgrade" its functionality or change any pre-set parameters (i.e., the 30-minute overlap window, the maximum number of hours that a worker is allowed to work to prevent overtime charges, etc.). To reach this goal, we plan on investing our time in the integration of the front-end and back-end elements, as our application must be (1) accurate in creating stable matchings and (2) easy to use for any non-technologically trained users.

## References


[1] L. S. Shapley and M. Shubik, "The assignment game i: The core," RAND Corporation, Research Report R-874, Oct. 1971. [Online]. Available: https://www.rand.org/pubs/reports/R874.html

[2] D. Gale and L. S. Shapley, "College admissions and the stability of marriage," *The American Mathematical Monthly*, vol. 69, no. 1, p. 9, Jan. 1962.