CPC Final Report: Enhancing Operational Efficiency Through a Digital Overhaul of the Route Allocation Process

Client: North America Central School Bus

Aden Benson, Aiden McCoy, Humza Khan, Jonathan Wu

June 10, 2025

Executive Summary

Every week, the charter coordinators at North America Central School Bus, a leading provider of student transportation, physically post a copy of the charter and field trip routes available to the bus drivers. Employees then engage in a "bidding process" in which they rank their top-choice routes. Each location abides by a unique collective bargaining agreement (CBA), which outlines the process charter coordinators must follow and the factors (i.e., seniority and absenteeism) they must consider when allocating bus routes. Issues arise when considering that (1) this process is done entirely by hand using physical spreadsheets (thus leaving room for human error), (2) the number of field trip/charter routes fluctuates each week (thus the allocation process takes longer during busy times in the school year), and (3) there is no formal documentation that depicts why a driver did or did not receive a route, thus requiring coordinators to have to step back through their entire process if a grievance is filed. These factors, in turn, lead to higher costs for the company, as both the driver who received a given route by mistake and the driver who should have received that route are compensated. Furthermore, drivers lose trust in the process and with management as issues arise.

The current allocation process requires an accumulated knowledge base of routes that most managers learn from acting as dispatch. This role gives many charter coordinators real-world experience to organize bus routes on the fly. Some include instructing drivers to split routes where one driver takes the trip to a location and a new driver takes the trip back, or instructing drivers not to return to base and go immediately to a new location. These decisions are actions a model cannot replicate, as they require shifting routes that a model assumes to be constant. As such, we design a process that requires charter coordinators to double-check the model's decisions. We cannot replicate the decision-making capacity of the coordinator, so we need to incorporate the coordinator's knowledge into the process.

We propose a two-part solution (front end and back end) that recreates the current process, increases the digital footprint and traceability, and decreases the margin for error in the process. The front end features an online form where drivers rank their preferred routes, as well as a web application for charter coordinators to assign rotating weekly routes to drivers. The back end utilizes the Gale-Shapley algorithm and a pre-processing step that replicates the current allocation process with added transparency. With digital technologies, we can timestamp and track every action taken in the allocation process, from bids to allocation and posting. Finally, we can decrease the amount of allocation work for managers by removing much of the manual work and focusing on the decision-making process in a digital ecosystem. The model is packaged into an executable file (exe) that opens on the user's default internet browser, recreating a similar experience to many web applications without the need for web hosting or Python packages. Our process overhaul will offer significant benefits by cutting excess work from grievances, increasing transparency, and streamlining existing work in allocation.

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, where drivers then bid for the routes they want. Managers manually assign routes, typically based on seniority (measured by tenure) and additional rules outlined in the location-specific CBA. 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 Implementable Solution

Our solution features a fully digital allocation system that turns raw driver preferences and operational constraints into finished allocations. Managers will use an app where they can set key parameters such as maximum weekly hours and rejected bids to produce union-compliant outputs. Downloadable diagnostic reports provide both the final schedule and a breakdown of each bid decision. Figure 1 illustrates the workflow process from preference collection to final assignments.

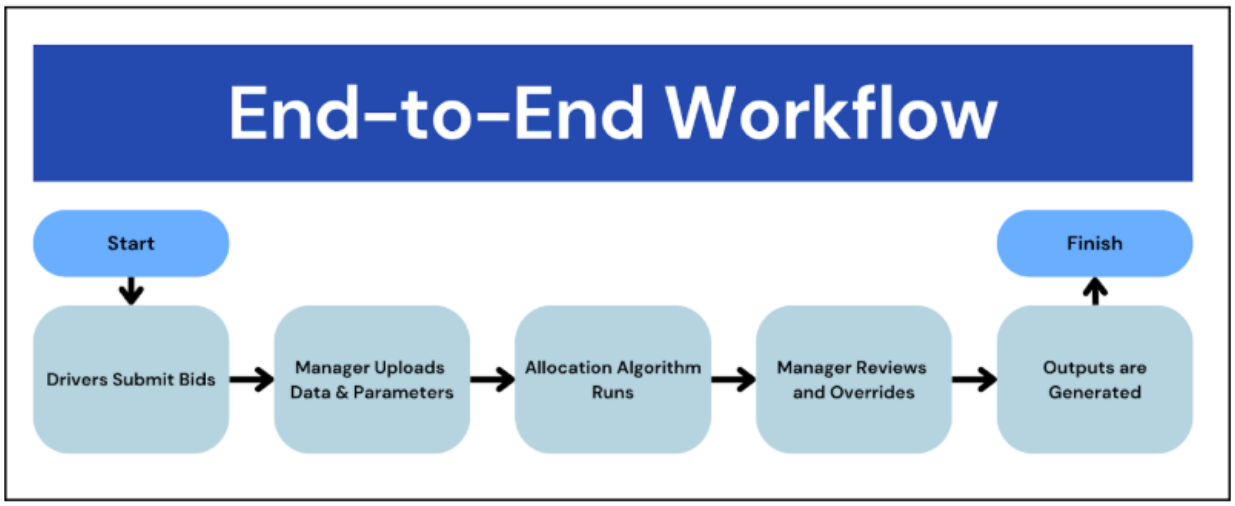

Figure 1: Workflow process from bid submission to output generation

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 [1], and (2) the Gale-Shapley matching method, which ensures that drivers receive stable assignments where no two drivers would prefer to swap routes [2]. 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 [3]. Combining these approaches creates a practical solution that satisfies both drivers and management, reducing dissatisfaction and improving operational efficiency.

1.3 Proposed Benefits

Our project aims to streamline how managers assign routes to bus drivers. Our proposed solution will boost operational efficiency, as assignments are distributed through computation rather than manually. By doing so, our program removes any human bias or error from its final allocations and allows for simplified error-checking through the force rejection mechanism. Additionally, because the inputs for the web application are required to follow templates developed by our group, we are helping our clients think proactively about how they handle company data. We plan to create outputs that increase transparency, further protecting our clients and improving the trust and working culture. Furthermore, by automating this process, we will reduce the time required for route allocation from 16 hours between two people to 2 hours for one person, reducing manual time by 87.5%.

2 Solution

2.1 Solution Overview

Figure 2 shows the proposed solution in visual form, with automated elements in blue and manual elements in green. We propose a two-part solution: a front end and a back end. The front end contains an online form for drivers to submit their bids, as well as an app front end for managers to run the allocation process. 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 has two parts: an optimization model based on the Gale-Shapley matching algorithm that takes in driver preferences and route preferences (represented as the seniority order here), returning a set of matches with beneficial mathematical properties that ensure the solution is optimal [3]. The second part is the preprocessing step, which removes 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. These steps are combined into an iterative process as described in section 2.5.

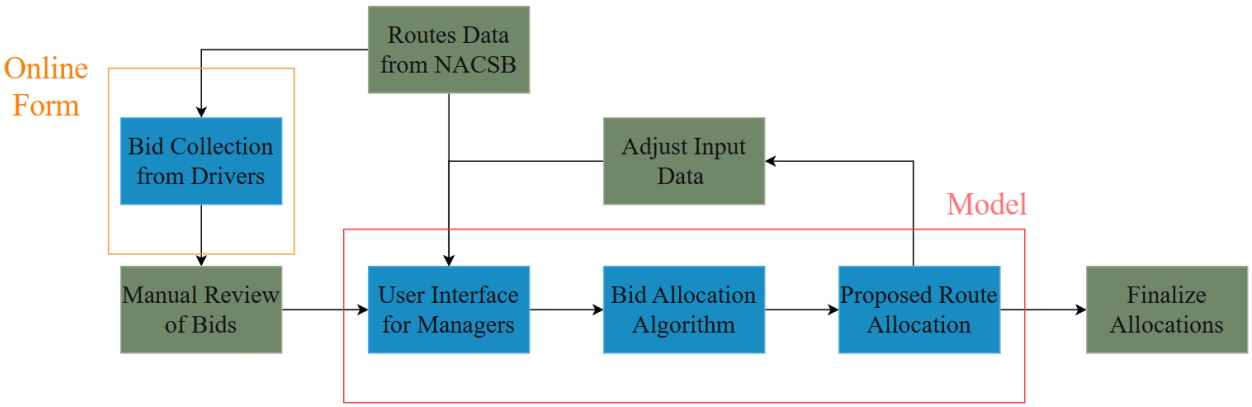

Figure 2: Proposed structure of bid process, with automated processes in blue

2.2 Driver Preference Collection

To capture all drivers' charter priorities in a user-friendly, digital way, we built a Microsoft Form that allows individuals to input their name, driver ID, and route numbers in order of preference, formatting data consistently for the manager. As per the client's request, drivers can bid on anywhere between 0 and 50 charter routes.

Once submitted, responses are automatically stored in an Excel spreadsheet. Each submission triggers a

new row to be appended onto the previous data, consisting of the driver's name, the times they started and submitted the form, driver ID, and ranked bids (in order). At the start of each allocation cycle, the manager will export this Excel file as a CSV file to be placed into the allocation web application. From there, the preprocessing steps outlined in 2.5.1 will check the structure of the CSV and make sure it aligns with the input schema required by the Gale-Shapley algorithm.

2.3 Manager Allocation Interface

To allow charter coordinators to assign charter routes using the allocation algorithm, our team developed an accessible, easy-to-use Shiny web application that integrates the backend optimization model into its outputs. When a user enters the required datasets (including the driver seniority numbers, the standard routes they operate, their route preferences based on results from the Microsoft Form, and the charter routes available for a given week) and the maximum number of hours drivers at a given location can work before entering overtime, clicking "run driver assignments" generates two tables—one that depicts the driver IDs with the route(s) they received and another that displays the charter IDs that remain unassigned after running the Gale-Shapley algorithm.

Additionally, the web app features an optional "force-rejection assignments" input, which allows charter coordinators to automatically remove routes from a given driver's submitted preferences before running the Gale-Shapley algorithm. Incorporating this feature, along with an integer "time padding" input that represents the minutes cut off from route end time, into our model ensures that the route assignments are free of time conflicts and fit into the standard schedules of bus drivers. Figure 3 shows the interface that managers use to input data into. Information boxes provide additional context of what files managers should input.

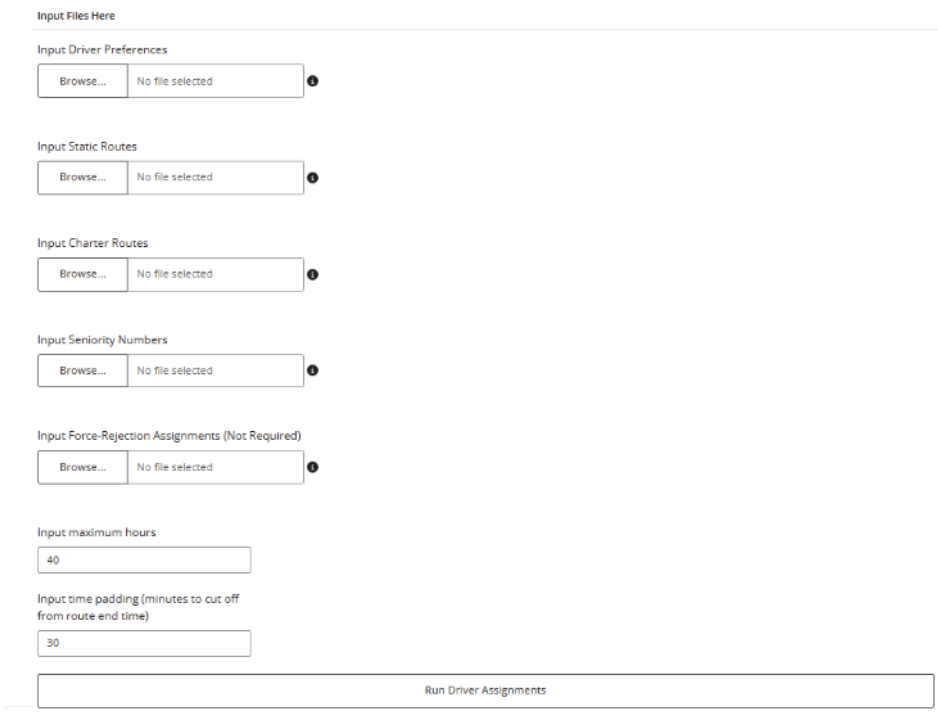

Figure 3: Manager interface to input all required information

Users can download three Excel files—the diagnostic sheet provides documentation depicting why a driver did or did not receive each route they bid on, the charter assignments sheet depicts the employee-to-charter route assignments, and the unassigned charter details sheet presents the unassigned charters. Figure 4 shows

this interface. These files can be downloaded from the interface and named with the date to provide easier way to search for said files. This naming convention should make it easier to find data from months prior in case of a grievance.

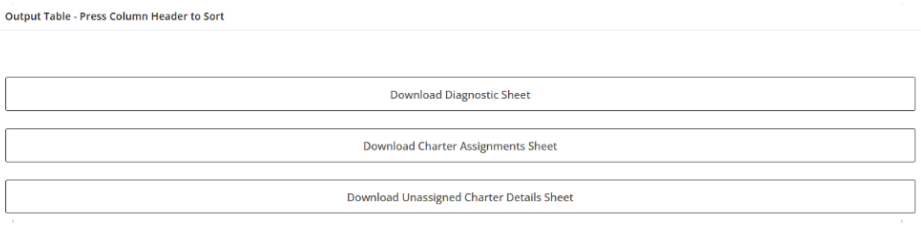

Figure 4: Manager interface to input all required information

2.4 Data Sources & Software Summary

To design our allocation process, we integrated the client's static route schedule, weekly charter routes, driver bid preferences, and seniority rankings. The static routes, charter routes, and seniority rankings are digital datasets provided by the client, while the driver preferences are created through the Microsoft Forms solution outlined in Section 2.2. Figure 5 shows the relationships we gather from our inputs. These relationships allow the model to match data across the files.

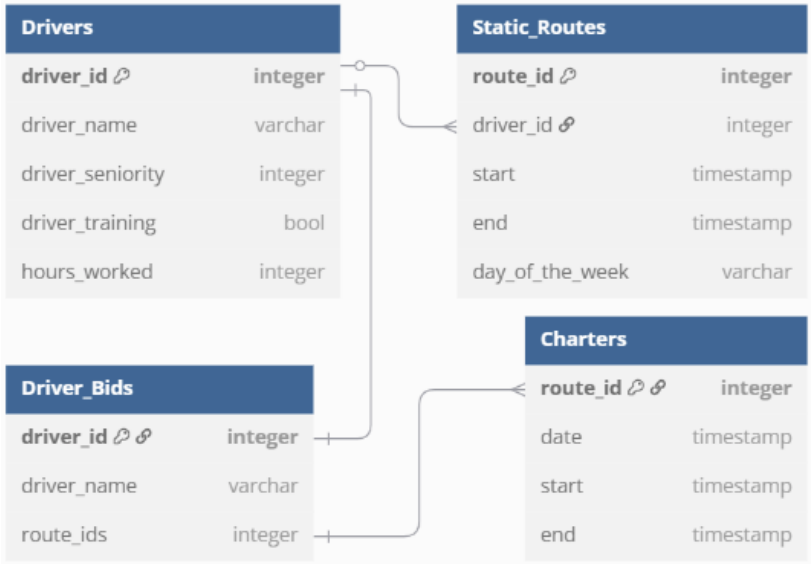

Figure 5: Structure of the various objects throughout the project

Our product offers the company flexibility, as the solution can be self-maintained within the client's existing Microsoft 365 environment and relies on open-source components. On the front end, drivers submit ranked preferences through Microsoft Forms, which automatically writes responses into an Excel spreadsheet. The allocation model that managers use is implemented in Python, using Shiny's Python library to create a dashboard for charter coordinators. To simplify deployment, the solution is packed as a single executable file (exe) that opens on the user's browser, delivering a web app experience without additional licensing fees. While initial startup times may vary depending on the local machine, the executable gives the company full control over updates and zero hosting costs.

2.5 Gale-Shapley Model

2.5.1 Gale-Shapley Algorithm

The core of our model is the Gale-Shapley algorithm, first proposed in Gale and Shapley 1962 [2] and Shapley and Shubik 1971 [4]. 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. For explanations, see Appendix C.

A limitation of the Gale-Shapley model is that it assumes all matches are possible. We know, however, that some drivers will bid on routes they are not able to take. To combat these mistakes, we use a preprocessing step outlined in Appendix C. Our model uses the following process for allocation:

1. Preprocess bids by checking for time conflicts, hour limits, and training
2. Run one iteration of Gale-Shapley
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
4. Repeat Steps 2 and 3 until all routes are assigned or no drivers have remaining bids

2.5.2 Post-Processing

However, the buffer times provided are likely wrong because they are applied across the board rather than customized to each route. Thus, after we provide an allocation, the user may need to fix the output if the time buffer assumption is incorrect. We allow the user to input overrides. After the output, the client will be able to see the displayed output and override the model's decision, which we dub as "force rejection." The override procedure is as follows:

1. The user inputs a new csv with the driver ID and route ID pairings to override
2. The assignment runs again, but the bid is forced to be excluded during the first step of preprocessing described in C.2
3. The user gets the output back with this new allocation and is prompted again, repeating steps 1 and 2 until the faulty assignments are removed
4. The user saves the output

This process allows multiple overrides that also are saved across iterations. This process helps the client ensure that faulty bids are not assigned to drivers. According to managers who currently oversee the process, there are only a handful of these instances, but removing faulty bids improves the company bottom line as these bids that cannot be taken are now reallocated to drivers who can take them. This process prevents emergency bid processes that allow drivers to go over the maximum workable hours as the location needs to fulfill the route and will allow anyone to take the route.

While there was discussion with the company to include an option to force time-conflicted bids to be considered, this option, called force approval, introduces unnecessary complexity and risk considering the penalty for failure. If a manager makes a mistake and does not approve a route that a driver could take, the penalty is having to pay both the driver who took the bid and the driver who should have received the bid. This is 2x the normal rate that the business pays drivers. In the case where a manager makes a mistake to reject a route a driver is unable to take, the penalty is that a driver who may be working overtime will take the route.

The driver who takes the route will be paid 1.5x the normal rate. Thus, when we consider the bottom-line impacts, it is better to fail to reject than to fail to accept.

2.6 Outputs & Diagnostics

Our model outputs 3 CSV files that the user can download seen in Figure 4. These files provide three key insights. First, we output the allocations made, which managers typically post to a general board to let drivers know which routes they have been assigned. Second, we output a diagnostic sheet that provides generalized information regarding the bid status of every bid made by a driver. By including every bid, the sheet allows managers to easily find why a bid was not awarded to a particular driver. This file provides traceability for our model's decision-making process. Finally, we include an output that lists all unassigned charters. This file provides a quick and easy way to track charters that need to be bid out, as any charter that requires additional drivers is output with the number of drivers still required.

2.7 Software Sustainability

As our typical user will not have a use case for understanding Python, directories, or command-line scripting to launch our front-end code, outside of this code, we packaged our code into an executable file (.exe). This file removes the need to understand how to operate backend code and acts similarly to a regular app. This experience is comparably more intuitive for the user and requires no initial setup.

While we believe the code can be rolled out to new locations with minimal changes, these changes would require the code to be repackaged into a new executable. Our code relies on Python's open-source libraries, and if there are changes to the library, repackaging code may not be possible without more significant edits, requiring someone with knowledge of Python. As most open-source libraries typically avoid deprecating code, this issue should not arise frequently. If it does, more in-depth changes may be needed.

The more likely issue is during the rollout of new software. The exe currently runs on Windows 10 and 11, but we cannot guarantee that the executable file will run on later operating systems, and we can guarantee that the executable will not run on MacOS in the foreseeable future. Solutions to this issue can be found in the future development section.

3 Limitations and Future Development

A limitation of our model (and any model) is its dependency on data inputs. There are somewhat strict requirements on the format of input data, and if data isn't updated for events like snow days or spring break, allocations may be skewed by invalid assumptions, leading to incorrect allocations. We've tested various configurations of invalid data and attempted to provide documented errors when they arise. For example, we need static routes to have the driver ID. A potential solution is to integrate this code with Bytecurve to pull drivers' active routes and allocate routes internally rather than having to run allocations. This feature would only be possible if ByteCurve releases an application programming interface (API) or if the company were to build a robotic process automation (RPA) system to collect data. However, one bright spot is that smaller errors, like minor changes in schedules, would be easily handled by the force-rejection mechanism.

Another layer of complexity comes from CBA rules, as each location has different requirements and procedures for the bid process. Our model is designed to work for a select few locations and will require varying modifications to accommodate other locations. One example of this is absenteeism. At some locations, drivers who miss routes are not allowed to bid for charters and field trips. The amount of time depends on how many times the driver has missed routes. However, this data is not provided to us, nor are we able to handle the absenteeism issue, as this requires knowing the history of attendance. We expect that the charter coordinator will manually remove the drivers referenced in this section of the CBA and run the driver assignments process without considering their route preferences (thus technically accounting for absenteeism). Nevertheless, there is room for an automatic implementation of this feature.

Although our model is specifically designed to replicate the process at certain locations, we do not have time to gather data and test this assertion. The extension to fit more locations that have similar processes is possible, but we lack the requisite time to test and validate with the client. Thus, we strongly encourage continued testing and verification to ensure the model's accuracy.

Another area for improvement relates to travel times. In that case, we recommend replacing our padding approach with dynamic time estimates drawn from a routing API such as Google Maps [5]. This feature removes the assumption in our model of constant travel time between all routes. Although this feature would yield extra costs for API usage, it would allow our allocation engine to resolve any schedule overlaps automatically without requiring force rejection or force acceptance.

Finally, managers currently need to run an application (.exe file) on their computer to allocate routes. As technology continues to improve, our .exe may fail on different computers. If the company wanted to extend this product further, the company could consider web hosting the shiny frontend. This option may improve the app's lifecycle, as web hosting can receive live updates while applications are static.

4 Conclusion

In this project, our team has created a fully digital allocation app for North American Central School Bus that mirrors and improves upon the existing paper-based process. By allowing managers to work entirely with digital native data, we have built an executable that eliminates much of the manual (and on-paper) work that managers and charter coordinators have historically endured. The new process results in measurable time savings, freeing managers from paperwork so they can focus on more valuable projects. This process also reduces the frequency and duration of grievance-related disputes, as managers won't make process-related errors, leading to fewer driver complaints due to human errors. Finally, the organization will benefit from faster turnaround times on weekly charter assignments.

Our solution adds value beyond pure cost savings. Every bid and allocation is now logged digitally and timestamped, which builds transparency and trust between drivers and management. Despite the complexity and extent of these features, the interface is intuitive enough so managers never need to download, interact with, or even know about Python. Finally, the manager's ability to quickly edit their inputs and re-run the model allows for quick troubleshooting in case of adjustments, where previously mistakes could force the entire paper allocation process to be repeated.

Looking forward, the framework we've put together allows the client to perform analysis on their outputs. The client can add analytics, which allows them to notice patterns in charter demand or scheduling conflicts. Real-time routing, such as APIs, can also be integrated in the future for dynamic travel time estimates. These steps will further reduce human error and enhance the operational efficiency of the business.

We would like to thank North America Central School Bus for their partnership in building this application. Specifically, we extend our sincere thanks to Cafria Hart, Jason Walker, Vince Ramirez, Stephanie Ingram, Scott Allen, the Carol Stream region, and many others for their valuable feedback at every stage. Your willingness to share your expertise in the industry has been crucial to building a solution that not only aligns with today's needs but also lays the foundation for future growth.

References

[1] X. Bu, Z. Li, S. Liu, J. Song, and B. Tao, "Fair division with allocator's preference," *arXiv preprint arXiv:2310.03475*, Jan. 2023.

[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.

[3] L. S. Shapley and H. Scarf, "On cores and indivisibility," *Journal of Mathematical Economics*, vol. 1, no. 1, pp. 23–37, Mar. 1974.

[4] 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

[5] Google, "Platform pricing & api costs - google maps platform," https://mapsplatform.google.com/pricing/?utm_source=google&utm_medium=cpc&utm_campaign=gmp25_us_search_api&gad_campaignid=22283564552, 2025, accessed: June 2, 2025.

[6] K. Iwama, S. Miyazaki, and H. Yanagisawa, "Improved approximation bounds for the student-project allocation problem with preferences over projects," *Journal of Discrete Algorithms*, vol. 13, pp. 59–66, May 2012.

[7] P. Ortoleva, E. Safonov, and L. Yariv, "Who cares more? allocation with diverse preference intensities," *SSRN Electronic Journal*, 2021.

[8] D. Manlove, D. Milne, and S. Olaosebikan, "Student-project allocation with preferences over projects: Algorithmic and experimental results," *Discrete Applied Mathematics*, Aug. 2020.

[9] M. Boyle, "Welfare economics," Online; accessed Jun. 2024, 2024. [Online]. Available: https://www.investopedia.com/terms/w/welfare_economics.asp

Appendix

A Background Research

The goal of this project is to provide a more efficient and automated way to allocate drivers to various charter and field-trip bus routes, thus saving time for the charter coordinator, the drivers, and NACSB management. The allocation must be cognizant of fixed routes that drivers hold regularly, equipment needs (such as accessibility aids and bus size), and hourly limits. We examine two methodologies to approach this challenge: linear programming and Gale-Shapley.

A.1 Linear Programming Allocation

Manlove, Milne, and Olaosebikan test an existing allocation algorithm, finding that it can equitably provide stable matches between two sets of subjects. Their study deals with the Student-Project Allocation problem, which examines a scenario where university lecturers propose projects for students, who must provide preferences for the projects they wish to work on. Manlove et al. test Iwama et al.'s solution [6], which assumes that students and lecturers have preferences over projects and establishes an upper bound of 3/2 on the number of students each project can accommodate. They demonstrate that the 3/2-approximation algorithm identifies an optimal number of stable matchings, accounting for lecturer and student preferences. This focus on dual-sided preferences is directly relevant to our project, as it resembles the framework of drivers and routes bidding for one another. Similarly, Bu et al. [1] discuss dual preferences to a further extent by emphasizing the allocator's preferences. In our project, this can be modeled as union rules, seniority, and equipment requirements, which usually do not affect the driver's preferences but are vital to management. The algorithm outlined first sorts items according to the allocator's utility function, then performs a round-robin selection where agents choose their preferred item according to their utility function. This framework can be adapted to provide a systematic way to focus on driver preferences with overarching managerial constraints.

Building off this framework, diverse preference intensities in an assignment allocation problem can be described using a more detailed objective function. Ortoleva, Safonov, and Yariv [7] propose an objective function that can be used to handle route assignments because we know drivers' preference intensities. This linear program takes a set of routes defined as $q_1, q_2, \ldots, q_n$) and finds the maximum of the objective function

$$W(q_1, q_2, \ldots, q_n) = \sum_{i=1}^{N} \alpha_i \mu_i \sum_{j=1}^{D} (u_i(d) * q_i(d))$$

where D is the set of drivers, $u_i(d)$ is the utility or ranking of routes by the driver, and μ_i is the probability that a driver would rank the route. α_i is an arbitrary set of weights that can be used to boost certain routes. This can emphasize the value of a route q_i if it is an important client.

The three articles described above are all concerned with fairness and validity in the allocation process. Their central idea is that preferences from both the allocator and the driver are crucial to getting outcomes that minimize dissatisfaction. The sources differ in their complexity and overall relevance to our problem. Manlove et al. [8] provide a general stable matching framework, while Bu et al. [1] add allocator influence. Ortoleva et al. [7] suggest that some preferences carry more weight than others and allow us to define "fairness" in a more generalized manner. By providing a mathematical formulation to maximize the benefit of Pareto efficiency, we can maximize the benefit for all actors without making any actor worse off. This assignment process reduces the number of routes that reach 'last call' by matching less popular routes with willing drivers, even if those routes rank low on their preference list. This process provides opportunities for lower-seniority drivers to have an assignment in the initial assignment process.

A.2 Gale-Shapley Matching

The process itself must follow the rules of the collective bargaining agreement (CBA). That is, the allocation must follow seniority and a round-robin fashion, complicating many of the traditional linear programming

methods above. Although it complicates standard job scheduling, this constraint aligns well with the classic Gale-Shapley [2] algorithm. One version of the Gale-Shapley algorithm is the stable marriage problem, where there are a set of men and women, each with preferences over all members of the opposite gender. The men then propose to women, using a "deferred-acceptance" procedure that is guaranteed to lead to a stable solution. A stable solution in the marriage example is one where cheating would not happen, as there are no two married people who would rather be together than with their current partners [2]. In our case, this solution would mean there are no two drivers who would like to swap their routes. The Gale-Shapley algorithm can be applied to our problem, with routes taking the place of women and drivers taking the place of the men who propose to or "bid" on routes. Unlike the previous models discussed, we allow for the routes to have preferences in the form of the seniority ranking.

The benefit of this model comes from a subsequent result discussed in Shapley and Shubik 1971 [4], where the solutions from this design framework are in the "core." Outcomes in the core are mathematically equivalent to those found in market equilibrium and are "strategy-proof." Market equilibrium replicates many of the benefits of the model proposed in Ortoleva et al. [7], as Pareto efficiency is a property of market equilibrium [9]. Additionally, the strategy-proof nature of the solutions means that if drivers collaborate, they are unable to achieve a superior solution. This property prevents groups of drivers from improving their positions by modifying their reported matching preferences.

The Gale-Shapley algorithm does not solve the entire challenge, as there are additional constraints on route choice like hourly limits and overlaps. This problem becomes more complicated when considering the dynamic nature of these issues—each additional route assigned adds to the hours worked by a driver and cannot overlap predetermined routes. We can resolve these constraints by drawing from another allocation method—Top Trading Cycles (TTC) by Shapley and Scarf 1971 [3]. While TTC only deals with consumers who have preferences over goods, the iterative nature of the algorithm provides a framework to handle the external constraints identified. One step of TTC involves removing all consumers who have been allocated a good and all goods that have been allocated. We can modify this step to remove allocated routes and invalid bids based on an arbitrary set of criteria. In sum, with Gale-Shapley and an iterative process, we can combine the mathematical properties of Gale-Shapley while maintaining compliance with the stated constraints.

In summary, our research shows that integrating multiple allocation strategies can effectively address the challenges provided in this project. By combining stable matching frameworks [2] with additional allocator preferences [8] and weighted preference intensities for drivers [7], we can create an approach that balances driver choice with managerial needs. Additionally, heuristic methods of the Gale-Shapley algorithm [2] and TTC [3] provide structures that extend beyond traditional linear programming.

B Data and Software

B.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.

B.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 use these examples to design the digital solution that imitates the process drivers currently use, but make assignments much easier for contract managers. This process 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 created a parameter that allows the client to estimate travel time between routes.

For model development (described in Section C), all data is converted into .csv files and structured like Figure 6, which shows an entity-relationship diagram. Each table represents one of the .csv files and the associated data. We assume static routes match an output similar to one from ByteCurve. A column that we were missing was 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.

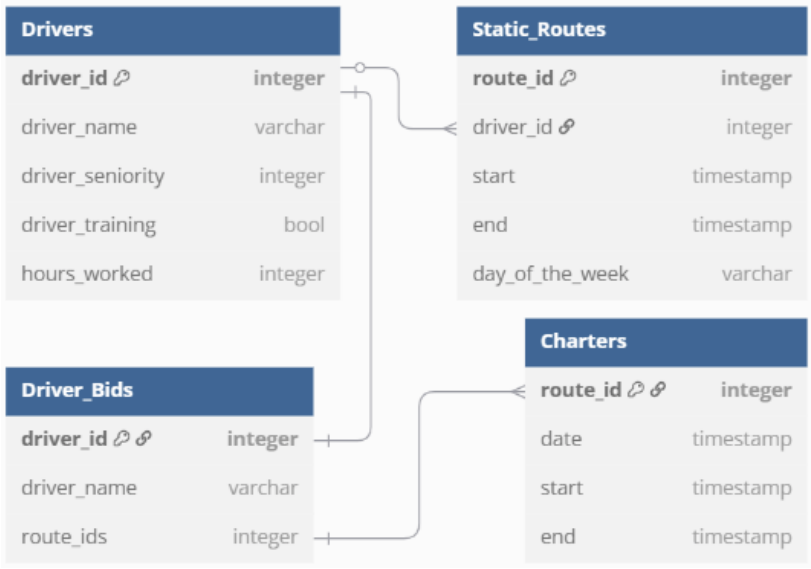

Figure 6: Structure of the various objects throughout the project

B.1.2 Software Resources

For the front end of our solution, we developed 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 emulates 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 used Python integrated with Shiny's dashboard generator to allow managers to assign routes. 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 C.1, and output all assigned routes. The outputs 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 diagnostic sheet increases transparency in the company and ensures no disputes between managers and drivers regarding union regulations. We add the ability to edit assignments after the allocation process. This addition prevents unusable solutions in the case of bad data entry, such as incorrect static routes being entered that inaccurately interfere with a chartered route.

C Models 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 7 illustrates this process.

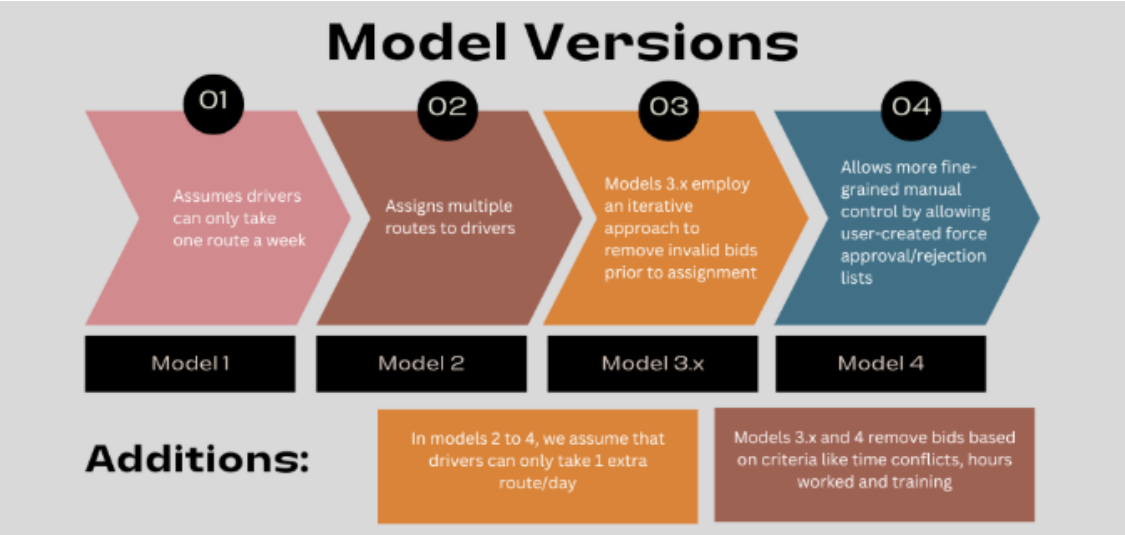

Figure 7: Models 1-4, with included features

Sections C.1 and C.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 C.3 and C.4 discuss the proposed features and procedures for Model 4.

C.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 [4] 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.

C.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.

C.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 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 C.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.

C.4 Saving data + Model Interpretation

A major focus of our client's needs is transparency of the results. We have an output the client can download which will be an Excel file with the bid assignments for each charter route. This "diagnostic sheet" contains the status of each bid for the drivers, indicating whether the route was assigned, or rejected and why. We worked 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.