CPC Final Report

Client: Denver Public Schools

Key Client Contact: Tyler Maybee

Team Members: Connor McIntee, David Russman, Karthik Vempati, Justin Chen, Nathan LaSalle

Date: 6/08/2022

Executive Summary

There is an ongoing nationwide shortage of school bus drivers, resulting in a transportation problem for our client, Denver Public Schools (DPS). With both a shortage of drivers and rise in salary costs, it is becoming increasingly difficult for DPS to reliably operate their school bus system at an affordable level with their existing routing. As a result, we were approached by our client to look into potential methods of reducing the number of buses and bus drivers. Any changes to the current routing system with the goal of bus reduction must ensure that all students are still provided with adequate and timely transportation services. An effective solution to this problem will allow Denver Public Schools to map potential effective routes, and weigh trade-offs between bus reduction and student travel experience to aid in transportation policy decisions.

Our proposed solution is a mathematical model, implemented in Python, that solves a vehicle routing problem for each school individually. The model takes inputs of maximum student demand numbers for each stop-to-school pair and a time matrix for travel time between stops. This model outputs a set of optimized routes for each school, which are then combined using a greedy heuristic aided by a secondary cleaning function. This second stage allows the user to designate a maximum number of routes that can be created, and then looks to minimize total travel time of each route.

We are providing the client with a Python framework that implements our proposed solution and generates the user's desired number of routes. After generating routes, the framework outputs an HTML file, with all stops and routes plotted on a map of Denver. The map is easily navigated, with all routes capable of being toggled on and off for easier viewing. A compact Excel file is also produced, detailing metrics pertinent to the client, such as route time, order of stops visited, and average capacity. Additionally, we supply detailed instructions on how to successfully run the framework, including how to specify the number of routes desired and load new data. With this framework, Denver Public Schools can make more informed decisions on potential changes to their bus system moving forward and continuously update the system with more recent student data.

Introduction

Our client, Denver Public Schools, is the public school district that serves the city of Denver, Colorado. The district serves over 90,000 students as of October 2021. In order to more easily manage the large area and student population, the district is divided into six planning regions (Figure 1). Our focus lies on the Far Northeast (FNE) region, which operates as a school-choice enrollment zone. Students who live within the zone are able to rank their choice of schools in the FNE region that they wish to attend, rather than being assigned to the nearest school. This enrollment system ensures that any student can attend any school regardless of socio-economic status, so that educational opportunities are equitable.

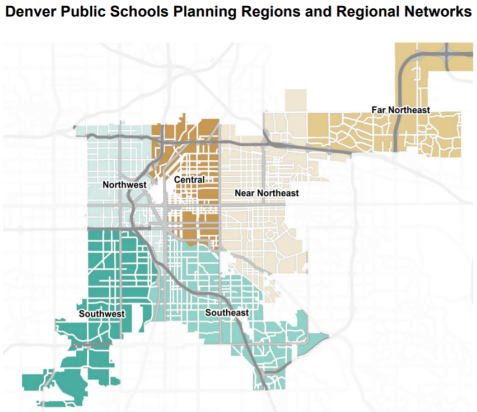

Figure 1: DPS Regions

While the school-choice system allows students access to the best fit schools in the region, it also creates a unique transportation problem, as many students need to travel far to reach their school. As a result, a traditional yellow school bus model would be inefficient and result in long travel times for students. Currently, to ensure all students can reach any potential school, DPS operates the "Success Express Shuttle", which is a shuttle service consisting of 26 routes within the FNE zone with one bus each. These routes guarantee that any given school, with some exceptions, can be reached from any other stop. Students are each assigned to the bus stop nearest to their home, and are required to walk there to board the shuttle route that makes a stop at their school. Within the 26 routes, there are 36 stops, most of which are located at schools to minimize a student's walking distance.

DPS manually created these routes with "Yellow", "Green", and "Express" shuttles that operate in the morning and afternoon. The 12 "Yellow" shuttles were designed to operate in a relatively counterclockwise manner, while the 12 "Green" shuttles were designed in a clockwise direction. The two "Express" shuttles are shorter routes that cut through the center of the system. Two of the current routes, Yellow 2 (in red) and Green 9 (in green) are shown below as an example (Figure 2).

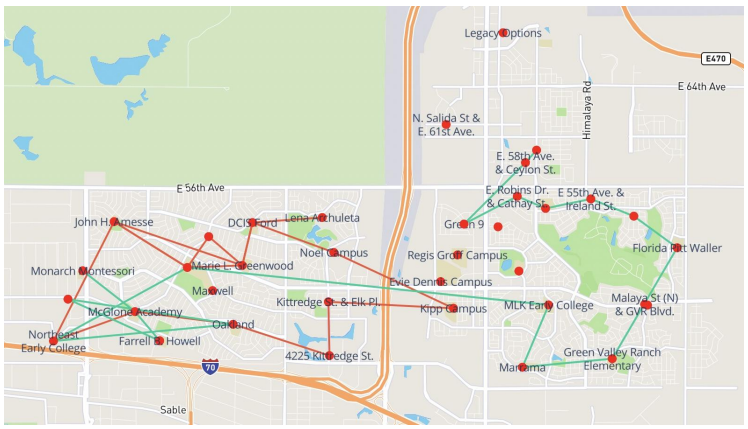

Figure 2: Example Current DPS Shuttle Routes

DPS and many other school districts throughout the United States are experiencing a general decline in the availability of school bus drivers, which has been exacerbated by the COVID-19 pandemic. School districts are averse to incurring the higher costs to employ enough drivers to operate the buses. Due to this shortage, DPS has asked us to reduce the number of buses needed to get students to their respective schools in a reasonable amount of time. We were given the ability to change routes, school bell times, and some flexibility for adding and removing stops. However, for this project, we focus on the creation of new morning routes and leave bell time and stop adjustment to the DPS administration. Once DPS has new routes in a stop-by-stop format, they will be able to make informed decisions regarding future transportation policies.

Reduction of even one bus from the system could prove impactful, reducing transportation costs and easing the urgency of hiring drivers. The money saved could be reallocated into providing resources for students and teachers to improve learning environments. We hope to help relieve the pressure caused by the driver shortage, and provide useful resources for the district in transportation policy decisions.

Solution

Our solution to the DPS policy challenge is a comprehensive Python tool that allows transportation facilitators to generate and analyze different routes to aid in policy decisions. Our framework provides two main recommendations: a new routing system (NRS) and a modified version of the current routing system (MCRS). We want to ensure that our solution is flexible and allows transportation facilitators to edit and analyze their potential and current routing systems. For this reason, the framework is "input-heavy" and allows users to choose the number of routes to reduce, the number of redundancies to scan for within a route, and which scenarios they would like to explore. In addition to generalized routes, our framework has the capability to analyze the impact of new routing options and output summary statistics of the entire system and of individual routes. These statistics include travel time, number of buses, unmet demand, and more. We are providing DPS with a framework that does not require maintenance but will output updated results as they use newer input data.

Recommendation 1: New Routing System (NRS)

The first recommendation we are providing our client is a new, optimized routing system. This system ignores the routes that DPS currently has in place and creates new routes based on maximum student demand. In this recommendation, our main objective is to reduce the number of buses while ensuring student travel time is not too high. Since we were not given a clear maximum travel time, we decided to minimize it instead, subject to using a predetermined number of buses. We are using average travel time as a metric to see how many buses in the system we can reduce while still having an acceptable travel time in the eyes of our client. We used our framework to solve this problem for 26 to 20 buses. By analyzing the output of the framework for each scenario we determine that a decrease of 4 buses, to 22 total buses, appears to be the best reduction. Average time only increases by ~5% (compared to our NRS 26 bus solution), and any more than 4 buses leads to a steep increase in average students per stop, which could potentially cause capacity issues. The summary statistics below show how the NRS compares to the current system.

	New System	Current System
Number of Buses	22	26
Total Travel Time of the System (minutes)	1793.49	1566.27
Avg Travel Time (minutes)	81.52	60.24
Longest Route (minutes)	112.21	86.55
Unmet Demand	4	1675

Table 1: New System versus Current System

As we can see in Table 1, the NRS appears to have a large increase in travel time over the current system, with an average estimated increase of 35% per route. However, the new system was created using maximum demand data. This means that it was engineered to support an unrealistic number of students. In Table 1, the Unmet Demand row shows how many students the system leaves behind if every eligible student takes the bus. Additionally, we can see that the current system ignores at least 1675 students (around 14%) from the maximum demand data. Because of this, we believe that once implemented, the administration would be able to cut unused stops out of routes in the new system and decrease the average travel time to below that of the current system.

NRS Limitations

The implementation of a completely new system has its drawbacks. Ideally, we would use the average ridership of each shuttle route and average daily student demand per school to get an accurate estimate of ridership numbers. However, due to inconsistent reporting and human error, many sections of this data are incomplete. Instead, following a recommendation from our client, maximum demand numbers are used. Hence, large demand at certain stops require bus capacity to be ignored, which could result in students not being able to secure a seat on a shuttle. Since it is unclear to what extent the demand data overestimates the number of students, DPS would likely have to modify the routes slightly to make sure every student is accounted for.

A second shortcoming of our recommendation is the absence of school bell time as a factor in creating new routes. With the current route outputs, student arrival time is not considered and late arrivals may escalate as a consequence. However, DPS offers schools the opportunity to decide their own bell time based on the scheduling of bus routes, which may enable more efficient routing.

Lastly, when considering this solution, it is important to note that large scale policy changes are difficult to implement. All decisions need to be discussed and voted on by the DPS Board of Education, and conflicts arise that stagnate progress. This motivated the suggestion of a smaller alternative recommendation, with a relatively easy implementation.

Recommendation 2: Modified Current Routing System (MCRS)

The second recommendation we are providing to our client is a modified version of the current routing system. Unlike the NRS, this recommendation does not utilize maximum demand data to create new routes, but instead takes in the current routing system and looks for optimal route combinations. The MCRS joins routes based on stop sequence similarity and proximity, then locally improves along each route by removing stop redundancies.

We used our framework to solve this problem for 26 to 20 buses. By analyzing the output of the framework for each scenario we determine that a decrease of 2 buses, to 24 total buses, appears to be the best reduction. Specifically, combining shuttles Yellow 1 with EXP 101 and Yellow 9 with Yellow 12. Unmet demand only increases by ~5% while average time decreases by ~17% (compared to the current bus routing system). The summary statistics below show how the MCRS compares to the current system.

	New System	Current System
Number of Buses	24	26
Total Travel Time of the System (minutes)	1197.55	1566.27
Avg Travel Time (minutes)	49.9	60.24
Longest Route (minutes)	73.03	86.55
Unmet Demand	1757	1675

Table 2: New System versus Current System

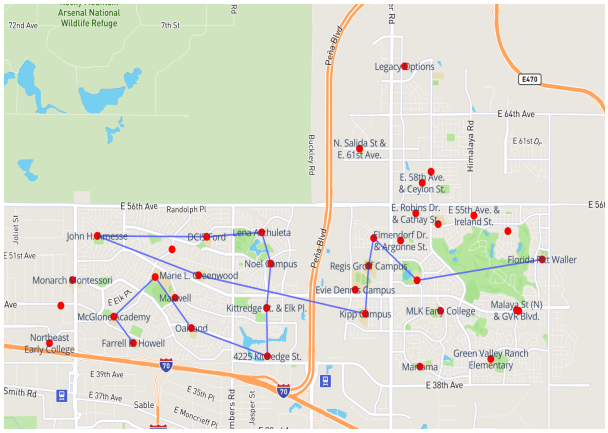

Figure 3: Yellow 1 joined with EXP 101

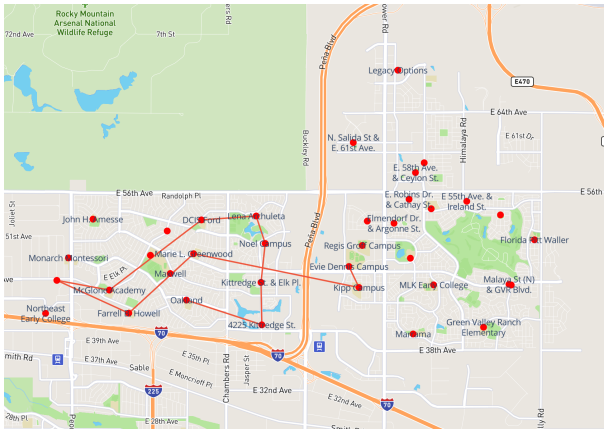

Figure 4: Yellow 9 joined with Yellow 12

MCRS Limitations

Similarly to the NRS, the MCRS disregards school bell times and requires bus capacity to be ignored for the combined routes, which could hypothetically result in students not being able to secure a seat on a shuttle. Since this capability of the framework takes in a current routing system, student demand data isn't a direct input. Therefore, the MCRS should be used as a guide when considering small scale changes based on route similarity and proximity. The MCRS may need to be followed up with independent modifications to ensure every student is accounted for.

Impacts

There are multiple impacts from our framework and recommendations. The first is the financial benefit from a reduction in buses. Bus driver salaries in Denver are roughly $40,000 per year and there are additional costs associated with housing a large number of buses.

The second is the social impact of financial benefits such as funding that can be reallocated towards after school programs and teacher's salaries. Additionally, by making the transportation system more affordable, we hope that equal education and opportunities will continue to be a core part of the Denver school community.

The third is that the framework gives the transportation facilitator autonomy to analyze multiple solutions to their current and future problems. While we only provide two recommendations, the various functions and outputs of the framework allow the facilitator to generate, evaluate, and visualize new and current routing systems.

Data and Software

We were given several data files by the client. First, we have the Shuttle Ridership data file that shows actual ridership counts collected in the 2019-2020 school year. The table has raw data showing the shuttle routes (both am and pm), the locations of the stops, and how many students got on or off the bus for specific days. The ridership data relies on bus driver tallying on four random days throughout the school year. Due to inconsistent reporting and human error, many sections of this data are incomplete. The second data file, Shuttle Schedule with Targets, shows the current target time that each bus gets to each bus stop. The final data file that we have is Student Count by Bus Stop. This file shows how many students going to a certain school are eligible to be picked up at each stop. In our model this is treated as the maximum demand data. The file also contains matrices of the distance between each stop/school and the time it takes to drive (without traffic) between each stop/school. Since the main use of time in our analysis is to compare different transportation systems to each other and it is not used as a standalone metric to evaluate the system, we determined that the absence of traffic travel times does not influence the efficacy of our analysis.

For route generation, the first stage of our optimization, we were given access to a Python framework developed by IEMS graduate student Min Fei for optimizing a randomly generated transportation system using Google OR-Tools. In her code, she solves a Vehicle Routing Problem (VRP) for each randomly generated school individually and creates separate routes for each school. We adapted her framework to instead perform this task for the Denver Public Schools, using the demand data in the Student Count by Bus Stop file.

For route joining, the second stage of our optimization, we created our own Python framework to combine the routes from the first stage. Doing this allows us to create and test multiple different functions and algorithms to match, combine, reorder, and visualize routes.

For our analysis, we created an optimization class in our framework containing all the functions and variables needed for the second stage optimization. We also created a route class that stores the stops and other information about each route. We wrote an algorithm that uses a custom metric to compare similarities between each possible combination of two routes. Then, another algorithm uses a greedy heuristic to combine the best routes that were identified based on the aforementioned metric. The implemented optimization function will continue to combine routes until the target number of routes has been reached. Additionally, our framework has two algorithms that re-sort the order of a given route to make the total time of the route shorter. This is used to eliminate some drawbacks of utilizing a greedy heuristic. Finally, with the help of Min Fei, we developed a function that produces a plot of the given routes, overlaid on a satellite map of the Denver Public Schools FNE area. This function gives our client the ability to compare routes and transportation systems holistically. A brief description of useful functions within our framework is seen in Appendix B.

Modeling and Analysis

Modeling

After reviewing previous work done by Bertsimas et al. (2020), we decided that a two stage optimization was the best way to optimize this transportation system.

First Stage

In the first stage of our optimization, we solve a VRP for each individual school using the mathematical model seen in Appendix A, which is similar to work done by Li et al. (2007). In this model we assume that our fleet of buses are homogenous, all with the same capacity and no wheel-chair accommodations. We also assume that students are picked up from the same stop every time. Our objective function (1) is to minimize the total distance traveled by all vehicles. Constraint (2) ensures that

the total number of students each bus picks up cannot exceed the bus capacity we set. Constraint (3) makes sure that each vehicle starts at the depot, which we designate as stop 0. Constraint (4) states that if a bus has a path that enters a stop, there must also be a path that leaves that stop. Constraint (5) ensures that the sum of all students getting on each bus from a single stop is at least equal to the total number of students that use the stop. Constraint (6) makes sure that students can only be picked up by a vehicle that visits their designated stop. Constraint (7) makes sure that students are only picked up once. This is especially important because our model allows vehicles to visit the stop again to pick up additional students. Constraint (8) exists to keep the number of students getting on each bus a positive number. Constraints (9) and (10) make sure that our decision variables are binary.

To solve our model, we create an optimal route for each school, ignoring all other schools. To generate an optimal route for each school, we use the cheapest insertion heuristic, followed by a local guided search to improve on initial routes.

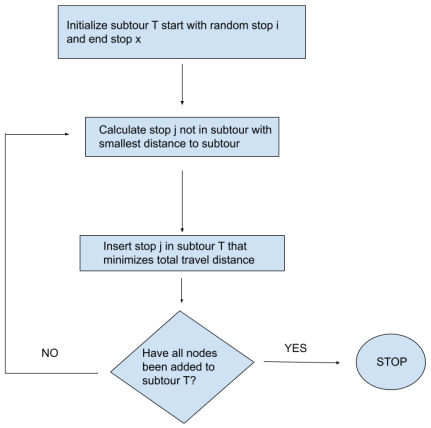

Figure 5: Diagram of Cheapest Insertion

The way cheapest insertion works is that it initializes an empty subtour. It starts by picking an arbitrary stop with the end stop being our school of choice and inserts them into our subtour. Then, it picks a new stop that minimizes the difference between our most recently inserted stop and the remaining stops not in the subtour. Once a new stop is picked to be inserted into the subtour, the newest stop is then inserted where it minimizes the total travel time in the subtour. This loop continues until all stops are added into our subtour and thus we reach our optimal route. We perform this optimization on all schools individually to create a route for each school.

However, a problem with using just the cheapest insertion by itself is that it may be up to twice as large as the real optimal route. As such, we perform a local guided search on the routes we created with cheapest insertion.

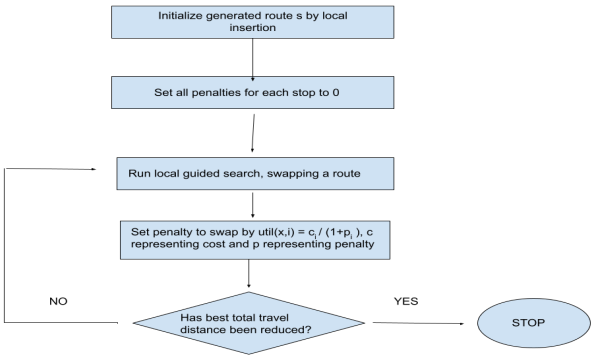

Figure 6: Guided Search Heuristic

According to Alsheddy et al. (2016), the local guided search is a metaheuristic used to improve on our routes generated by the cheapest insertion. The local guided search swaps the ordering of a route to see if there is an improvement in the total time traveled until it reaches a local minimum. Whenever changes are made to the route, penalties are applied to any change to reduce the computational cost and ignore inferior options.

Second Stage

For the second stage of our optimization model, after reviewing Park and Kim (2010), we decided to use a greedy algorithm to combine routes based on a similarity score. The algorithm computes a score for each combination of two routes and chooses the combination of routes that produces the highest score to combine. Our similarity metric is as follows:

$$\frac{(\%\ of\ stops\ in\ each\ route\ that\ is\ in\ the\ other\ route + 0.1)}{length\ of\ the\ combined\ route}$$

We added a factor of 0.1 to allow for the possibility of relatively small routes with no stops in common to be combined.

The greedy algorithm then takes two routes to combine and outputs a single route. It does this by adding the first stop of the first given route as the first stop in the combined route and removes that stop from the first route. Then, it looks at the first stops in each of the input routes and adds the stop that is closest to the last stop in the combined route. When it adds a stop to the combined route it removes it from the input route. It does this until there are no more stops in the input routes. Combining routes this way preserves the ordering of each individual route in the combined route.

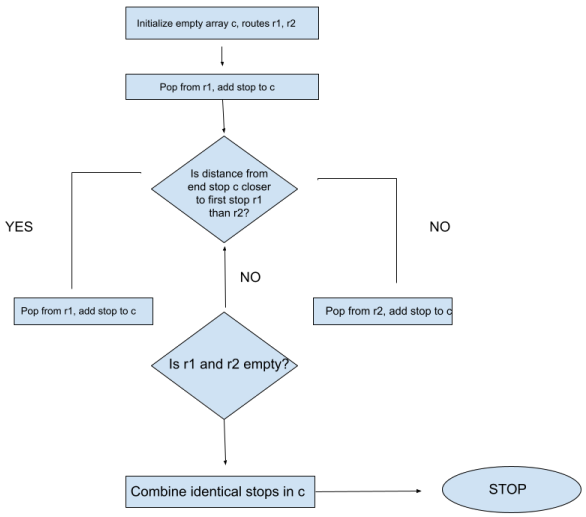

Figure 7: Flow Chart for the Greedy Algorithm

Once we have combined enough routes together so that the number of routes equals the target, we re-sort the routes to decrease the time for each. The re-sorting algorithm takes in a route and for each stop in said route, it looks at every other stop and sees whether swapping the order of the two stops decreases the total time of the route. Running this resorting algorithm multiple times generally leads to a decrease in the route's time. We have found that running this algorithm preserves each stop to school requirement. In other words, optimizing each route in this way allows all the students to get to their respective schools.

Analysis

To come up with our recommendations (NRS and MCRS) we tested out a variety of combinations of cleaning and optimization to develop new routes. For the NRS, we determined that the combination of cleaning the first stage routes once, optimizing to a certain number of buses, and then cleaning 10 times worked the best. For the MCRS we followed this same process but didn't initially clean routes in order to preserve the stop sequence of the current system. This procedure decreased the total time of the respective system significantly, while generally preserving each stop to school requirement. We then decided that since our goal was to decrease the number of buses used, we would try this procedure for 26 to 20 buses. We found that using less than 20 buses results in an infeasible solution due to a very high level of unmet demand.

NRS:

For our NRS recommendation, in our evaluation of each system that our framework outputs, we are using a newly created optimized system with 26 buses (the same number as the current system) as a baseline because we are unable to calculate load comparison metrics for the current system. The two main metrics we used to compare these systems was the increase in average route travel time and the increase in a load metric, "Students per Stop", averaged over all routes. This load metric was calculated by estimating the total number of students who use a route from the maximum demand data and then dividing by the number of the stops on that route.

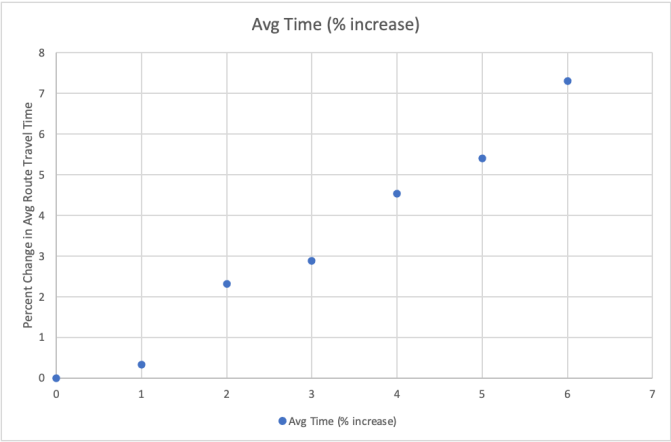

Figure 8: Percentage Increase in Average Route Time from the baseline

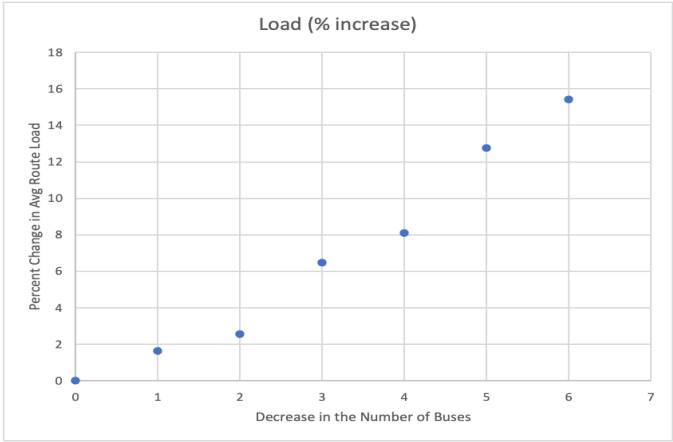

Figure 9: Percentage Increase in Students per Stop from the baseline

As we can see in Figures 8 and 9, a decrease of more than 4 buses leads to a steep increase in the Students per Stop metric. Since the lack of bus capacities are the main source of concern for the validity of the model we chose a decrease of 4 buses to avoid more than a 10% increase in Students per Stop over the baseline.

MCRS:

For our MCRS recommendation, in our evaluation of each system that our framework outputs, we are using the current system as a baseline to measure the impact of route combination and cleaning. The three main metrics we used to compare these systems were the increase in average route time, the increase in unmet demand, and the unmet demand of the MCRS that was met in the current system.

System	Current	Current Cleaned	25R	24R	23R	22R	21R	20R
Number of Routes	26	26	25	24	23	22	21	20
Total Time (minutes)	1566.27	1282.67	1229.01	1197.55	1181.55	1140.2	1113.21	1067.57
% Change		*-18%*	*-22%*	*-24%*	*-25%*	*-27%*	*-29%*	*-32%*
Avg Travel Time (minutes)	60.24	49.33	49.16	49.9	51.37	51.83	53.01	53.38
% Change		*-18%*	*-18%*	*-17%*	*-15%*	*-14%*	*-12%*	*-11%*
Longest Route (minutes)	86.55	73.03	73.03	73.03	73.03	73.03	73.03	73.03
Unmet Demand	1675	1773	1768	1757	1757	1425	1482	1500
% Change		*6%*	*6%*	*5%*	*5%*	*-15%*	*-12%*	*-10%*
Unmet Demand Against Current	0	147	142	134	134	189	251	269

Table 3: System Output Summary

As we can see in Table 3, unmet demand against current is lowest in systems 24R and 23R, at 134 students. This metric is extremely important in analyzing the impact of the MCRS recommendation because it quantifies modifications to a routing system in place. In addition, a decrease in average travel time compared to the baseline is also favorable for the routing system, inferring smaller, more efficient routes. Taking into account both these factors we found system 24R most favorable.

Sensitivity

To assess the versatility and future implications of our model, we analyzed its sensitivity to various changes in the input student demand data. As a proxy for fluctuations in maximum demand data, we changed our VRP bus capacity constraint while leaving the input demand data unedited. The goal of this modification was to increase or decrease the relative load of one student without manually editing the data. We tested output changes in the first and second stage of our model. We found on average, a decrease of ~12% in VRP bus capacity led to an increase of ~9% in the number of routes required to transport all students. We further explored what the impact of changing VRP capacity would be in the second stage of our model. After optimizing multiple VRP solutions with varying bus capacities, we found no significant changes in total travel time but found, on average, a decrease of 10 spaces in VRP bus capacity led to an increase of ~4 students in unmet demand.

Because changing VRP bus capacity serves as a proxy for changing the input student demand data, we can say that unmet demand is sensitive to changes in student demand while travel time remains relatively consistent. With this information we suggest that our client should collect more thorough data for student demand and ridership such that our framework will output a solution that meets all of the demand requirements. Additionally, since we know that the given demand data is overestimated, we can say that this likely increased the unmet demand of our new systems, but the travel times were likely accurate.

Bibliography

Alsheddy, Abdullah & Voudouris, Christos & Tsang, Edward & Alhindi, Ahmad., 2016, Guided Local Search. 10.1007/978-3-319-07153-4_2-1.

Bertsimas, A. Delarue, W. Eger, J. Hanlon and S. Martin, 2020, "Bus routing optimization helps Boston Public Schools design better policies," INFORMS Journal on Applied Analytics, Vol. 50, No. 1, https://doi.org/10.1287/inte.2019.1015.

Google, Google, "Vehicle Routing Problem; OR-Tools; Google Developers." https://developers.google.com/optimization/routing/vrp.

Li, F., Golden, B., Wasil, E., 2007. The open vehicle routing problem: algorithms, large-scale test problems, and computational results. Computers and Operations Research 34 (10), 2918–2930.

Park and B.-I. Kim, 2010, "The school bus routing problem: A review," European Journal of Operational Research, Vol. 202, No. 2, pp. 311-319.

Appendices

Appendix A: First Stage Mathematical Model

Parameters:

V: Set of all stops
C: Capacity of bus
K: Number of school buses
d_{ij}: time a bus spends traveling from stop i to j
g_i: number of students at stop i

Decision Variables:

x_{ijk}: 1 if bus k goes from stop i to stop j, 0 otherwise
z_{ik}: 1 if students are picked up by vehicle k at stop i
y_{ik}: 1 if vehicle k visits stop i
s_{ik}: number of students getting on vehicle k from stop i

Objective Function:

$$min \sum_{i=0}^{n} \sum_{j=0}^{n} \sum_{k=1}^{K} d_{ij} x_{ijk} \quad (1)$$

Subject to:

$$\sum_{i=1}^{n} s_{ik} y_{ik} \leq C \qquad k \forall \{1, 2, \ldots, K\} \quad (2)$$

$$\sum_{k=1}^{K} y_{0k} \leq K \quad (3)$$

$$\sum_{j \in V} x_{ijk} = \sum_{j \in V} x_{jik} = y_{ik} \qquad \forall i \in V,\ k \forall \{1, 2, \ldots, K\} \quad (4)$$

$$\sum_{k=1}^{K} s_{ik} \leq g_i \qquad \forall i \in V \quad (5)$$

$$z_{ik} \leq y_{ik} \qquad \forall i, k \quad (6)$$

$$\sum_{k=1}^{K} \frac{s_{ik}}{g_i} z_{ik} = 1 \qquad \forall i \in V \quad (7)$$

$$s_{ik} \geq 0 \qquad \forall i \in V \quad (8)$$

$$y_{ik} \in \{0, 1\} \quad (9)$$

$$z_{ik} \in \{0, 1\} \quad (10)$$

Appendix B: Overview of the Framework

Data Reader (reader.py)

This script reads in demand data, time matrices and current system scheduling and stores each respective data source

First Stage (first_stage.py)

This script runs the first stage of the optimization process and outputs the generated routes to a json file

Second Stage (second_stage.py)

Implements an Optimizer Class with input variable *current_system* which is a boolean that, if it equals True, makes the Class using the current routing system, and if it equals False, makes the Class using the output data from the first stage.

Optimizer Class Functions and Attributes

optimize(n):

This function combines routes until the number of routes equals n.

clean_all(n, aggressive):

If aggressive is set to True: The function rearranges each route n times by iteratively choosing the next closest stop in the route to the current stop. This will likely drastically change the ordering of each route.

If aggressive is set to False: The function rearranges each route n times by looking at each stop in the route and seeing if switching that stop with any other stop on the route would lead to a reduction in total time of the route. While this function is not guaranteed to preserve each stop to school requirement in the route, in general it seems to do so.

unsatisfied_demand():

This function iterates through each stop to school requirement in the demand data file and checks to see if, given infinite bus capacity, the students at each stop could get to their respective schools on any of the system's routes. It returns the number of students that were unable to get to their schools.

unsatisfied_demand_ac():

This function is the same as unsatisfied_demand() except that each time the function determines that there is unsatisfied demand at a given stop to school pair, it checks to see if the current system meets this demand and only includes those students as unsatisfied if the current system does satisfy the demand.

summary():

This function outputs summary statistics for the system including

- Number of Routes
- Total Travel Time of the system
- Average Travel Time for the system
- Longest RouteTravel Time
- Unsatisfied Demand

- Unsatisfied Demand AC

graph(routes):

This function plots the given routes over a map of DPS

remove_stop(stop):

This function iterates through all of the routes in the system and removes the stop from all of the routes

output_system():

Takes the current route and summary information for the system and outputs the data to an excel file

load_summary():

Returns summary statistics pertaining to the load of each route in the system, including

- Load for each route
- Load per stop for each route
- Load per school for each route

Here, load is the estimated number of students who would use the route, calculated from the maximum demand data.