MATH 187: Deterministic Operations Research
Final Project: Single School Bus Routing

Shu Bin[1] and Carrie Yang[2]

[1]Claremont McKenna College - sbin19@cmc.edu
[2]Harvey Mudd College - xiyang@hmc.edu

7 May 2019

Abstract

In this project, we attempt to create an algorithm that determines the routes a single school's school bus fleet takes in order to pick up all the students while minimizing the distance traveled by the bus fleet. We use the data from the Boston Public School Transportation Challenge, and normalize it to fit the data onto a perfect grid. We first create a linear program and implement it in AMPL. Because the linear program's big-O is 2^n, its runtime proves too large for AMPL to handle. To circumvent this, we construct a hybrid program that combines the linear program with a heuristic program that uses a greedy algorithm to pick the bus stops on a route. This significantly decreases the model's runtime and we're able to implement it in Python. We then discuss the effect that changes in grid size, students' maximum walking distance, and bus capacity has on the best objective function values of our model. In the end, although the hybrid algorithm does not guarantee an optimal solution, it is the more viable solution among the two due to its small runtime.

Executive Summary

In this project, we create an algorithm that determines the routes a single school's school bus fleet takes in order to pick up all the students while minimizing the distance traveled by the bus fleet. In order to simplify the data to a more manageable scale, we assume that the roads in the school bus fleet's operating region is a perfect grid, and processes the data in this way. We first make a linear program based in AMPL. However, this program requires too much time to run, and is almost unsolvable when the set of data becomes too large. To solve this problem, we create a hybrid linear-heuristic program based in Python. This program successfully solves even problems with very large data sets and returns solutions that are satisfactory. It should be noted that the hybrid program does not guarantee that its solution is optimal. We introduce one and recommend several other techniques to alleviate this problem so that the hybrid program's solution approaches optimality, but this problem is ultimately unavoidable in order to make the model feasible in real world scenarios. We seek to contribute to the creation of a Claremont College school bus system through this project.

1 Introduction

This project aims to tackle a school bus routing problem for a single school. Given the location of a local school, the locations of students' homes, bus capacity, number of buses in the fleet, and students' walking limitations, we attempt to construct an optimal route for the school buses to take such that they deliver all students to the school while minimizing the distance traveled by the bus fleet. We use a combination of linear programming and heuristic programming methods to construct our algorithm. Albeit relatively small in scale, our work can be used in the creation of a Claremont Colleges school bus system, where school buses take students to popular destinations such as Target, Trader Joe's, and Walmart and back. This service can help low-income students with their financial situation by eliminating some transportation costs, save students' time for more late-night studies, and generally improve students' quality of life. Our algorithm can help the Claremont Colleges school bus services find routes that maximizes the buses' efficiency. It can even serve as a foundation for more complex problems such as the optimization of public transportation systems of entire cities and regions.

2 Assumptions

In the linear program, we attempt to minimize the collective distance traveled by the school bus fleet. In order to simplify the real world problem such that it can be expressed through a mathematical model, we make the following assumptions:

1. The road system in the school buses' operating region is a perfect grid, where the distances between any two adjacent intersections (nodes) are the same.
2. All school buses have the same capacity.
3. School buses can only pick up students at nodes.
4. Only one school bus operates on each route.
5. Each student can take a direct path from their home to a bus stop. Hence the distance between a student's home and a node is their Euclidean distance.
6. Each bus can only travel on roads (represented as arcs) to go from one node to another. Hence the distance between two nodes is their Manhattan distance.
7. Each non-school node only can be visited at most once by a bus.
8. Each student is picked up at a node that's within reasonable walking distance away from their home.

9. The number of students assigned to a route cannot exceed the maximum capacity of a bus.

10. Each student is picked up exactly once by a bus.

3 Data and Data Processing

Our project uses the simulated data set from the Boston Public School Transportation Challenge.[1] In this data set, there are 22420 students distributed among 133 different schools. For the purpose of this project, we randomly select a school and its set of students to calculate the location parameters for our model. We use the "latitude" and "longitude" data associated with each student and the "school latitude" and "school longitude" data associated with each school to determine their relative location on the map.

To fit the school and its students' locations into a perfect grid, we first normalize our data. Denote a school or student as i and their longitude and latitude as X_i and Y_i, respectively. Then, the data is transformed so that no negative numbers or unnecessarily large numbers exist. For instance, in a given data set, if all students' and the school's X-coordinates are greater than 42, there is no reason to simulate the blocks between X-coordinate 0 and 42 as they would not be used by the buses. To achieve this, we use the following formula:

$$X_j^{Normalized} = X_j - min(X_i) \quad \forall\, j$$
$$Y_j^{Normalized} = Y_j - min(Y_i) \quad \forall\, j$$

This will ensure that no negative value is present and that the student or school with the smallest X-coordinate and Y-coordinate values will become (0,0) on the coordinate system. Then, all of the coordinate values are multiplied by 1000 and turned into integers. The students and school's locations are plotted on a coordinate system that spans from 0 to $max(X_i)$ on the horizontal axis, and from 0 to $max(Y_i)$ on the vertical axis. Next, we take an arbitrary value n that divides $max(Y_i)$. The value n dictates the size of each identical square on the final grid that represents the buses' operating region. Each square's height and width as well as the distance between two adjacent arcs are equal to $\frac{max(Y_i)}{n}$.

After fitting the location data onto a grid, the data is further processed to compute if a student can reach a stop from their home as well as the distance between two stops. These values are saved as matrices to serve as parameters for our model.

The above steps are all done through Python. Figure 7 and 8 in the Appendices give an example of our data manipulation process.

[1] https://www.bostonpublicschools.org/transportationchallenge

4 Model Formulation

4.1 Algebraic Formulation

- **Sets**

  Student for the set of students

  Stop for the set of nodes or potential stops

  Bus for the set of buses or routes

- **Decision Variables**

  $$X_{k,i,j} = \begin{cases} 1 & \text{if bus } k \text{ traverses the edge between node } i \text{ and node } j \\ 0 & \text{otherwise} \end{cases}$$

  $$Y_{k,i} = \begin{cases} 1 & \text{if bus } k \text{ visits node } i \\ 0 & \text{otherwise} \end{cases}$$

  $$Z_{s,k,i} = \begin{cases} 1 & \text{if student } s \text{ is picked up by bus } k \text{ at node } i \\ 0 & \text{otherwise} \end{cases}$$

- **Parameters**

  $Capacity$: the maximum number of students that can be assigned to a bus/route

  $mdist_{i,j}$: the Manhattan distance between node i and node j

  $reach_{i,s}$: binary variable that's 1 if node i is within walking distance from student s, 0 otherwise

- **Objective Function**

  Minimize total distance:

$$Min \sum_{\forall i \in Stop} \sum_{\forall j \in Stop} \sum_{\forall k \in Bus} mdist_{i,j} X_{k,i,j}$$

- **Constraints**

A bus can only traverse on an edge that leads to node i if it makes a visit to node i:

$$\sum_{\forall j \in Stop} X_{k,i,j} = Y_{k,i} \qquad \forall\, (i,k) \in\ Stop \times Bus$$

The number of times a bus traverses between node j and i is the same as the number of times it traverses between node i and j:

$$\sum_{\forall j \in Stop} X_{k,i,j} = \sum_{\forall j \in Stop} X_{k,j,i} \qquad \forall\, (i,k) \in\ Stop \times Bus$$

Each non-school node is visited at most once by a bus:

$$\sum_{\forall k \in Bus} Y_{k,i} \leq 1 \qquad \forall\, i \in\ Stop \setminus \{0\}$$

Each student is picked up at a stop that they can reach:

$$\sum_{\forall k \in Bus} Z_{s,k,i} \leq reach_{i,s} \qquad \forall\, (s,i) \in\ Student \times Stop$$

The number of students on a route cannot exceed bus capacity:

$$\sum_{\forall s \in Student} \sum_{\forall i \in Stop} Z_{s,k,i} \leq capacity \qquad \forall\, k\ \in\ Bus$$

If bus k does not visit node i, student s cannot be picked up by bus k:

$Z_{s,k,i} \leq Y_{k,i} \qquad \forall\, (s,i,k) \in\ Student \times Stop \times Bus$

Each student is picked up only once:

$$\sum_{\forall k \in Bus} \sum_{\forall i \in Stop} Z_{s,k,i} = 1 \qquad \forall\, s\ \in\ Student$$

The number of edges connecting the nodes in the route of bus k must be less or equal to the cardinality of Q minus 1, where Q is an element of the powerset of the set of potential stops:

$$\sum_{\forall i \in Q} \sum_{\forall j \in Q} X_{i,j,k} \leq |Q| - 1 \qquad \forall\, Q \subseteq P(Stop \setminus \{0\})$$

5 AMPL Implementation

5.1 Solving the Model

The first implementation of our model is done in AMPL. The *.mod* file is constructed exactly as it is shown in our algebraic formulation. We find the *.dat* file to be challenging, because we need to calculate the distance between every two nodes (adjacent or not) and whether node i is reachable by student j for all nodes and students. These values are calculated in Python before rearranged into the format required by the AMPL *.dat* file. Note that if we want to change the values of parameters such as grid size and maximum walking distance, we need to run the Python script and construct a new *.dat* file. We find the process of constructing the *.dat* file very time-consuming. It also makes the postoptimality analysis difficult to carry out, since we need to get the model's results from AMPL manually in order to plot them in Python. Moreover, due to this project's time constraints, we cannot tune the parameters as frequently as we would like.

The AMPL implementation uses a small example of 18 students and 5 buses on a 3×3 grid. The model generates 3 routes and the optimal objective function value is 15. Below is the assignment of students to routes:

- Bus 1 stops at node 5 to pick up student 12 and 13, then stops at node 8 to pick up student 14, 15, 16, and 18.
- Bus 2 stops at node 3 to pick up student 4, 5, 6, 7, 8 and 9.
- Bus 3 stops at node 1 to pick up student 1, 2, 3 and 17, then stops at node 2 to pick up student 10 and 11.

5.2 Limitations

Even though our model has feasible outputs for data with 18 students, as mentioned before in Section 5.1, we cannot get the algorithm to work for larger datasets since its runtime becomes too large for our computers and even NEOS to handle. This is due to the fact that one of the algorithm's subproblems is a travelling salesman problem, and that the last constraint involving a power set has 2^n variables. As such, this linear program is unfit to be used with larger real world datasets.

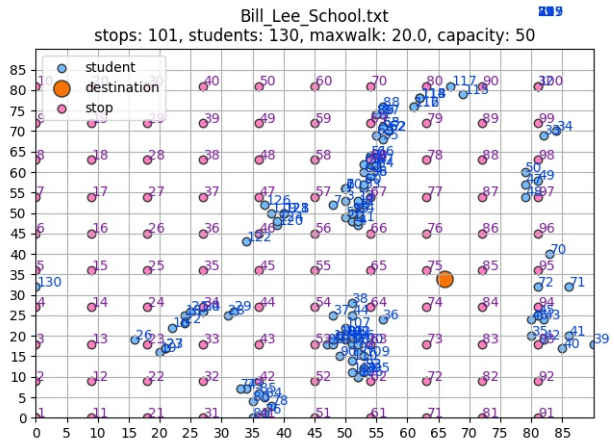
6 Python Implementation

To lower our program's runtime such that it becomes feasible with larger datasets, we make a second implementation in Python. For this enhanced algorithm, we take inspiration from a greedy heuristic algorithm of a capacitated vehicle routing problem [1] to make a hybrid linear-heuristic program. We construct a helper set of stops that are within walking distance to some students. The algorithm first start from a random choice of a stop. To choose the next stop, our

algorithm picks the stop that is the closest to the current stop and reachable by some students. We pick the stop only if the bus has enough room left to hold the students that are only assigned to this stop. If a bus does not have capacity to hold all students that are assigned to a stop, it will prioritize the students who can only reach this stop.

Since the introduction of the greedy heuristic algorithm brings a factor of randomness to our algorithm, we run the routing algorithm 100 times and pick the solution with the highest objective function value as our best solution. Note that this does not guarantee the optimality of the solution. However, we believe this is a reasonable trade-off in order to lower the model's runtime and make it workable with larger datasets.

Figure 1: Students, school, and potential bus stop locations for Bill Lee School

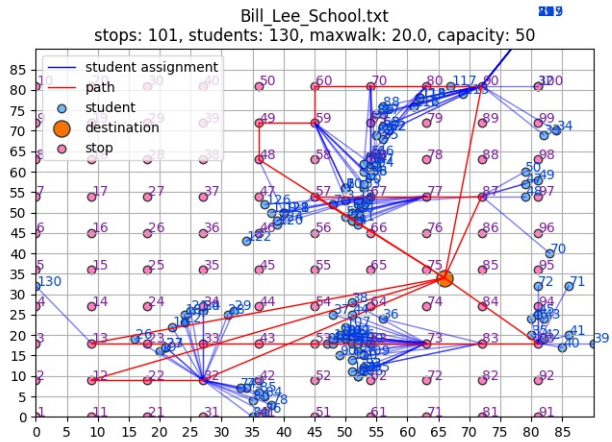


To give an example of our algorithm in action, we use it to find the best routes for Bill Lee School. Bill Lee School has 130 students spread on a 10×10 grid. Students' maximum walking distance is set to 20 and bus capacity is set to 50. Our algorithm returns the following routes:

- Bus 1 stops at node 100, 90, 80, 70, 60, 59, 49, and 48.
- Bus 2 stops at node 72, 62, 52, 42, 32, 22, and 12.
- Bus 3 stops at node 86, 76, 66, 56, 46, 36, 26,and 16.
- Bus 4 stops at node 91 and 92.

Figure 1 illustrates the distribution of Bill Lee School students' locations on the buses' operating region. Note that the figure says there are 101 nodes instead of 100. This is because the school (shown as "destination" on the figure) is also counted as a node. Figure 2 illustrates the bus routes calculated by our algorithm.

Figure 2: This figure shows the routes constructed by the hybrid algorithm. Blue lines assign a student to a stop. Red lines denote bus routes



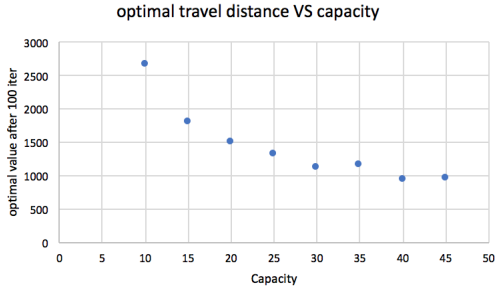
7 Postoptimality Analysis

In this section, we conduct a postoptimality analysis on the Python implementation. As explained in Section 5.2, due to the limitations of the AMPL formulation, it requires a lot of manual input every time we want to tune the parameters, making the AMPL postoptimality analysis very time consuming. We're also unable to tune the parameters however we want in the AMPL formulation due to runtime errors. Thus, we choose to focus on the Python model for our postoptimality analysis.

7.1 Changes in Bus Capacity

We keep students' maximum walking distance at 20 and grid size at 10×10 while changing bus capacity.

Figure 3: A plot of optimal path distance versus bus capacity

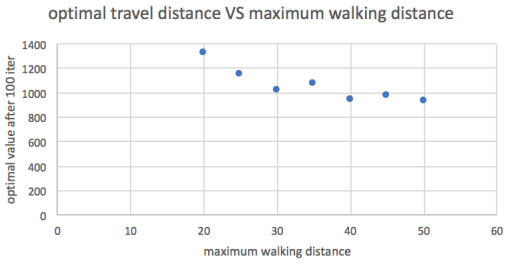


As shown in the figure 3, as we increase bus capacity, the number of routes calculated by the model decreases because fewer buses are needed if each bus can hold more student. As we increase the bus capacity, the optimal objective function value decreases. This is due to the fact that having a larger capacity allows each bus to visit more stops before having to travel back to the school, thus increasing the fleet's overall efficiency and decreasing overall travel distance.

7.2 Changes in Students' Maximum Walking Distance

We keep bus capacity at 25 and grid size at 10×10 while changing students' maximum walking distance.

Figure 4: A plot of optimal path distance versus maximum walking distance

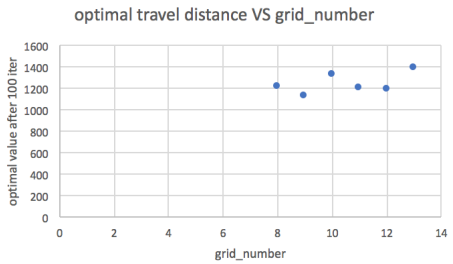


As we increase the maximum walking distance, the total distance traveled doesn't have a clear increasing or decreasing trend. Although it may seem like there is a weakly decreasing trend from the figure, we think that it is caused by the randomness of the greedy algorithm and the data we used.

7.3 Changes in Grid Size

We keep bus capacity at 25 and students' maximum walking distance at 20 while changing grid size.

Figure 5: A plot of optimal path distance versus grid number. For example, if grid number is 5, there will be 5×5 stops in a grid.



Changing the grid number means changing the number of potential stops. As we increase the number of potential stops, the total bus travel distance exhibits an increasing trend. Note that if the maximum walking distance is set below 20, not all students can be assigned to a route by the algorithm.

8 Model Validation and Future Improvement

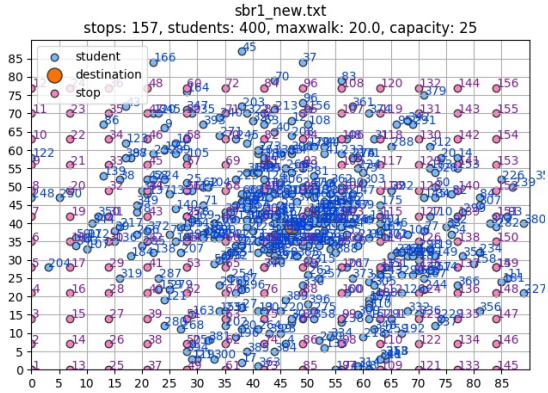
This paper includes two example solutions generated by our hybrid linear-heuristic program on two different sets of data. The first one, Bill Lee School, is shown in Section 6. We notice the solution requires the bus to stop at node 48, 49, and 59 in that order. By visual inspection, we can see that stop 49 is unnecessary and can be taken out of the route. In the second example (Figure 6), there exists crossing in path that could be resolved by using a 2-opt algorithm. Both these two examples show that our hybrid program, unlike the linear program, does not return the optimal solution. This is due to the random factor introduced by the greedy algorithm. As we have discussed before in Section 6, this is a sacrifice we have to make in order to construct a model that's workable with larger datasets.

In order to partially overcome this limitation, our algorithm runs multiple iterations and returns the best result among these iterations. In the Bill Lee School example with 130 students, we run 100 iterations, which causes the objective function value to drop from 1305 to 504, and it takes less than 5 seconds to run the script. In the second example (Figure 6) with 400 students, it takes less than 120 seconds to finish 100 iterations. Therefore, using many iterations is a great way to improve our results while keeping the fast computation time low.

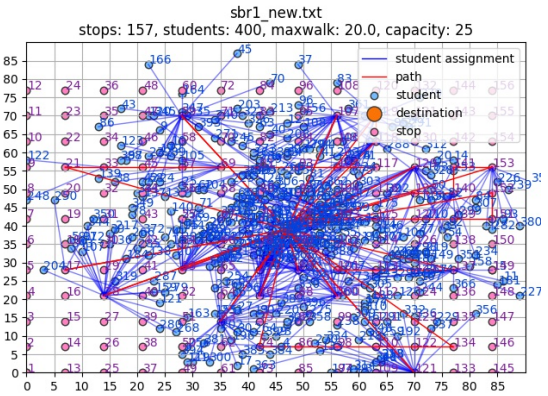
The other limitation is that the hybrid algorithm doesn't always find a solution. For instance, if the maximum walking distance value is too small, we can run into a case where there aren't enough stops to accommodate for all students. In this case, we need to manually increase the maximum walking distance until a solution can be found. In the future, we can resolve this problem by writing a script that automatically tunes the parameters.

9 Conclusion

In this project, we construct a linear program and a hybrid linear-heuristic program in an attempt to find the route that minimizes the distance traveled by a school bus fleet while making sure that the buses pick up all the students. Although the linear program is guaranteed to return an optimal solution, its large runtime makes it essentially useless when it comes to solving problems with very large datasets, thus we turn to the hybrid program. While the hybrid program does not guarantee an optimal solution, it returns a viable solution in very little time. Looking to the future, the hybrid program can be further improved by merging it with an opt-2 algorithm and another algorithm that calculates the optimal parameters for a given set of data.



(a) Student and Stop Location



(b) Final Routes

Figure 6: A routing example with 400 students, 157 stops, maximum walking distance at 20, and bus capacity at 25. This result is generated with the hybrid program.

References

[1] Kirci, P. (2016). An optimization algorithm for a capacitated vehicle routing problem with time windows. Sdhan, 41(5), 519-529.

Appendices

Figure 7: Illustration of part of the Boston High School Data in *.xlsx* format. We extract latitude and longitude of student and school from it.

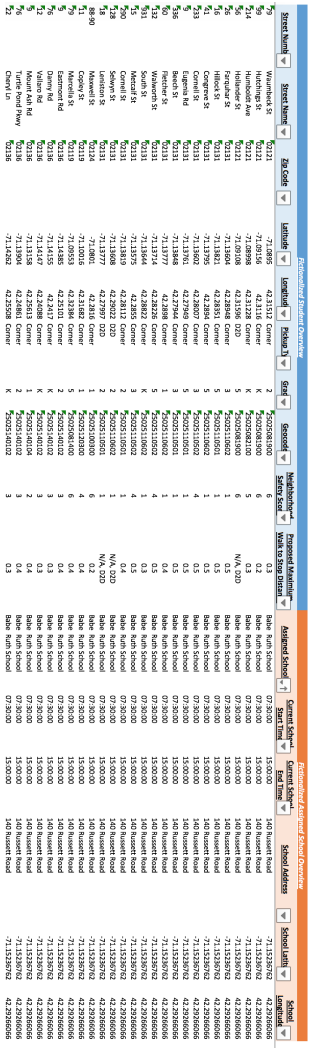


Figure 8: Illustration of part of the output text file with Python scripts data_process.py In this figure, from left to right, the rows represent the index of a potential stop, its x location, and its y location.

```
101 stops, 130 students, 20.00 maximum walk, 50 capacity
0 66 34
1 0 0
2 0 9
3 0 18
4 0 27
5 0 36
6 0 45
7 0 54
8 0 63
9 0 72
10 0 81
11 9 0
12 9 9
13 9 18
14 9 27
15 9 36
16 9 45
17 9 54
18 9 63
19 9 72
20 9 81
21 18 0
22 18 9
23 18 18
24 18 27
25 18 36
26 18 45
27 18 54
28 18 63
```