Improving Transit for Baltimore City School Students

JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

About Us

We are the **System Engineering Senior Capstone Design Team**. Systems engineering is the intersection of programming, engineering, statistics, and optimization. The systems engineering process is to break down a intricate problem into a set of goals to achieve, decisions we can make, and constraints on those decisions. This year, we worked with the **Maryland Transit Administration (MTA), Baltimore Banner, and JHU Faculty.**

The Problem

Baltimore City Public Schools offer an open choice policy to middle and high school students: they may enroll in any school across the city, regardless of their distance from that school.

However, City Schools **does not offer a yellow bus system to students beyond elementary school.** Older students seeking public transit must use the MTA's bus routes, or other forms of public transit.

MTA Routes have the following problems:

- **Buses are unreliable:** they are rarely on time, and sometimes never appear.
- **Buses may not stop for students:** drivers may not stop for high schoolers because it is at-capacity or they refuse to transport high school students.
- **Transfers are inconsistent:** MTA does not optimize time spent waiting for the next bus to arrive.

Background Research

Improving bus routes is an example of the **Vehicle Routing Problem (VRP): a class of problems where a set of customer demands (students) must be served (taken to school) by a fleet of vehicles (MTA / yellow busses).**

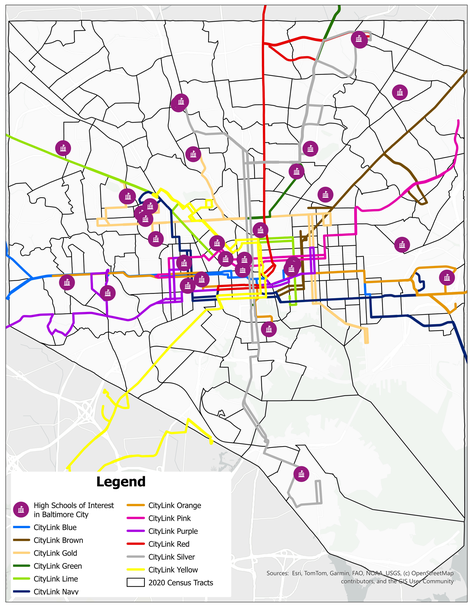
The VRP is a computationally complex problem, meaning it quickly grows in the run-time as the problem gets larger. Hence, **heuristics (ways to approximate solutions) are used** to solve the problem as they reduce run-time. These approximations are still highly effective.

The heuristic we chose was the Genetic Algorithm:

1. Treat each set of **bus routes as a "chromosome."**
2. **Evaluate** each "chromosome" based on evaluation criteria.
3. **Combine strong routes** to create new candidates, "crossover."
4. Apply **small, random changes** to new candidates, "mutation."
5. **Keep the best-performing candidates** and any mutations that improve performance.
6. **Repeat** the process, allowing better solutions to emerge over time (natural selection effect), ending when reaches some stopping-criterion.

Results

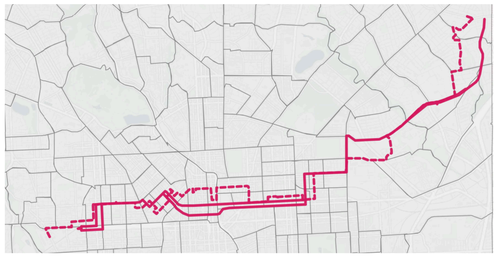
Our model was effectively able to modify the MTA's bus routes without significant changes, as seen below. We broke up Baltimore into its **census tracts**, and approximated each MTA bus route by the census tracts it passes through. Then, we run the **Genetic Algorithm on these broken up route**s while limiting the number of changes, leading to a similar system which is better for students. **Our model is highly adaptable:** we can easily adjust the amount we allow bus routes to change, the metrics we optimize for, and the amount of time spent searching for an optimal solution. Further, since student demand is an input, the **model can easily be adjusted to seasonal changes** by updating where students live. This also means the model can be optimized for *all* riders, not just students, if the amount of people commuting between each census tract is known.



Current Bus Routes, Census Tracts, and High Schools in Baltimore City, Maryland

Current and Optimized Centroid to Centroid Bus Routes Overlayed

Current and Optimized Pink Route Centroid to Centroid

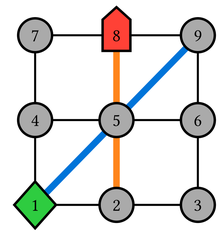


Current and Optimized Pink Route Fitted to Streets

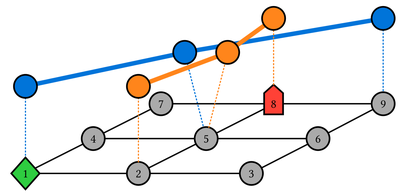
There are **many avenues for expanding our model.** We currently allow the same number of changes to each bus route, but we could customize these values: high ridership routes can have minimal changes, whereas low ridership routes can be modified more significantly. We stop at adapting the current set of routes, but the model could be expanded to create and evaluate new routes. These could be routes ran by the MTA or yellow busses operated by Baltimore City Schools. Finally, if this model were to be adopted by the MTA, **we could use a finer level of detail** rather than just using centroids of census tracts.

Implementation

We simulate commutes using a **Shortest Path Algorithm.** In this toy example, we model a student getting to school with blue and orange bus routes (right). To account for transfer times, we create **imaginary "transfer" nodes** for each bus, where it takes the transfer time (15 minutes) to reach them, and they can return to walking by getting off the bus (moving down to grey nodes). **This guarantees optimality without needing to modify our algorithm.**

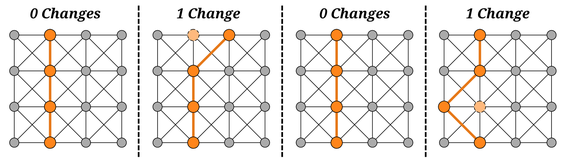


Toy Bus Route Example



Network Projection to account for transfer times

We want to minimize impacts on non-student commuters. We achieve this by restraining the amount of changes any bus route can have. Below, we limit the orange route to only experiencing one change at a time. This way, even after hundreds of mutations, the overall system will not be significantly changed.



Evaluation

Evaluation Criteria	Outcome
Longest Ride Time	17% Reduced → Improves Equity for Worst-Off Students
Number of Transfers	41% Decrease → More Reliable Routes
Flexibility/Redundancy	Moderate → Small Route Adjustments Allow Adaptability but Limited Backup Paths
Required Resources	Small Increase →Requires Route Adjustments but Not Entirely New System
Projected Cost	Low → Builds on Existing MTA Infrastructure