Analysis of OC Transpo Bus Routes through Network Creation and Simulation

By Owen Allen, Marco Boracci, Lucas Colwell

---

1. Introduction:

Residents of Ottawa are intimately familiar with the OC Transpo, the bus and light rail company which seeks to provide far-reaching and efficient transportation to the greater Ottawa area. The routes of the OC Transpo span the east to the west, from Kanata to Orleans, and the south to the north, from North Gower to Gatineau. Given the large width of Ottawa, the over 1 million residents, and the disconnected 'hubs' of activity (High-tech in Kanata, Federal offices downtown, etc.), citizens have often criticized the OC Transpo for late and tardy service, unreliable and aging equipment, and insufficient resources that do not account appropriately for traffic[1]. These concerns highlight the aim of our paper.

While improving travel is not novel, and many academic papers examine the effects of changes along routes, openings and closures of routes, and the cost of driver choice, we wanted to look specifically at OC Transpo. Having been citizens of Ottawa, we often have had to rely on bus or light rail and have felt the pangs of inefficient travel, whether due to congestion, buses or rail not showing up on time, weather conditions, or equipment decay and malfunction. As such, we wanted to combine our experiences in class and our experiences trying to get to class into one research question: what could be done to improve the OC Transpo, and how would this affect the system as a whole.

Modelling our examination off of concepts such as Nash Equilibrium and traffic network concepts introduced in the lecture on Braess' Paradox, such as social cost (energy cost for drivers) and best response dynamics, we decided to construct a simulation of the OC Transpo network, and model our changes to bus behaviour and route structure based on different desired outputs. Since OC Transpo is an incorporated municipal body[2], the council of members might want to look at specific priorities. We decided that council members would consider two main priorities: prioritizing lower travel time (modelled as the travel cost of getting passengers from point A to point B) or prioritizing profit (which coincides with picking up a more significant amount of passengers and is therefore modelled by the number of passengers picked up along the route.)

By creating our database, we were allowed to run our simulations on the data, tweaking routes, buses and passenger counts to examine different conditions and make an informed

---

[1] Khan, T. (2021, October 12). *Problem-plagued OC Transpo is teaching me to lower my expectations of public transit.*

[2] *Decisions*. IPC.

analysis of their results. Between creating our route network and examining the generated results, we discovered interesting situations and relationships on the node network and how they could pertain to the real world.

2. Creating the Model

Given that there are almost 6,000 bus stops serviced by 170 bus routes, two light rail trains and a paratransit system in Ottawa, we could not fill in all the data for bus routes and stops ourselves. Since the track for the light rail had already been laid, there was no point in trying to analyze a better route, so we did not examine this. Buses and their routes are flexible and can be changed, and this is where we wanted to focus our effort. Despite our enthusiasm for the project, the prospect of riding the bus routes to gather reliable and normalized data for days on end was not appealing. Luckily, OC Transpo and Google Maps could provide the raw data needed, which we could then modify and add to as we saw fit.

```
82551723-JAN22-301Shop-Weekday-01,08:58:00,08:58:00,WR265,6,0,0
82551723-JAN22-301Shop-Weekday-01,08:59:00,08:59:00,WR270,7,0,0
82551723-JAN22-301Shop-Weekday-01,08:59:00,08:59:00,WR285,8,0,0
82551723-JAN22-301Shop-Weekday-01,08:59:00,08:59:00,WR295,9,0,0
```

Fig 1, an example of the data retrieved from OC Transpo, showing, in order: the trip id, the arrival time, the departure time, the stop id, the stop sequence, the pickup type, and the dropoff type (from stop_times.txt)

To begin our modelling, we downloaded bus stop and route information from the OC Transpo API[3], which gave us comma-separated-value files of the routes (in routes.txt and trips.txt of the attached submission), bus stops and identification numbers (in stops.txt), and stop times for the given day (in stop_times.txt). Once we had retrieved this raw data, we noticed some issues regarding the time of arrival and departure (as seen in Fig #1), which will be examined later in this section.

This is where we began to build the network. We created the empty graph G, to be filled with nodes later. Once we had a basic understanding of the file structure, we used trips.txt to identify a route number (ex. 7-332) and retrieved its trip identification number. We then took this trip ID and checked all of the stops of that specific trip in stop_times.txt. Since the bus follows a sequence of stops, and numbers each stop as an increasing sequence, finding the order

[3] https://www.octranspo.com/en/plan-your-trip/travel-tools/developers/

in which stops were visited was trivial, and times could be compared between stops by determining their position in the stop.

For each stop we found along the route, we would check if the stop (as a vertex) was present in G. If it was not, we would add it to G. If it was present in G, we would parse the stop to retrieve the vertex representing the stop, then add a new edge to the stop (from the previous stop). This new edge was weighted by taking the difference in departure time against the previous stop on the route. Since routes followed a sequential (and labelled) order, computing the departure time between two stops and adding an edge between stops on the same route was simple to compute and prepare.

```
82551723-JAN22-301Shop-Weekday-01,08:59:00,08:59:00,WR270,7,0,0
82551723-JAN22-301Shop-Weekday-01,08:59:00,08:59:00,WR285,8,0,0
```

Fig 2, an example of two stops with 8:59:00 as their departure and arrival times

However, we noticed an issue with the stop and departure times after weighing these edges. Since the OC Transpo API rounded the departure and arrival times to the nearest minute, if stops were only seconds apart (or had been passed without pickup), the departure time would be the same. That is to say, if a bus arrived at stop A at 10:30:01 (hh:mm:ss), departed at 10:30:11, arrived at stop B at 10:30:19, and departed at 10:30:30, all times would show only as 10:30:00, for both stop A and B. This is a significant issue, but since we were modelling this data ourselves, we conferred and decided an appropriate adjustment, based on our personal experiences with OC Transpo, was for each bus to take 30 seconds between stops (if the time between stops was equal, as seen in the above case with A and B.). This resolved the issue and used our personal experience with the data.

However, after running these algorithms for every bus, we noticed our network was too extensive. As mentioned in the introduction, modelling over a hundred routes with nearly 6,000 stops was laggy, time-consuming, and, most importantly, challenging to interpret and analyze. So, we decided only to examine four buses in a core area of Ottawa. We decided on the downtown core due to the frequency of travel and the high population density, and due to our personal experiences, as we have lived in the downtown area and have experienced more travel in this sub-network. In addition, there are also many more side routes and alternative paths in the downtown core, so it was a perfect choice for us to model buses taking different paths and examine their effects.

We looked at buses that served routes centred on the downtown core of Ottawa and found four buses that fit the bill: the 14,7,6 and 10. We decided to use these four bus routes, which serviced 234 stops, going both directions (e.x: we modelled the 6 Rockliffe and the 6 Greenboro), for a total of 8 routes along with our network. The buses acted as agents on our network, choosing which node to visit, being given information on the graph as a whole, and acting on that information. This will be further explained in the next section, where we will explain our algorithms for simulating agents (buses) on the network.

After parsing the data, we decided to add new paths to the network based on our own experiences with the routes and our intuition for what may constitute an alternative path. In our examination, we noticed that specific routes would only go one way on a two-way street. This presented an opportunity for change. For example, on Sunnyside Avenue, buses on route 7 only went one way and would make buses turn down a side street (Grosvenor), then turn left on another side street (Grove) before heading back onto Sunnyside.

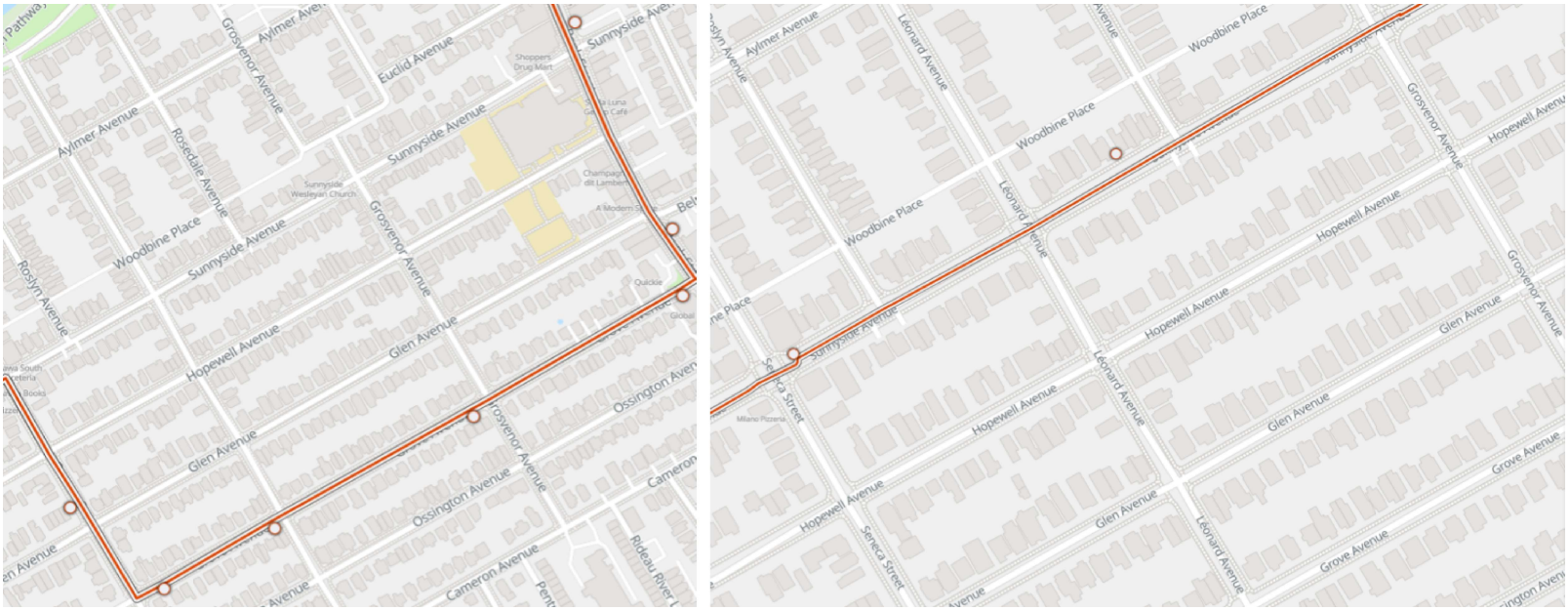


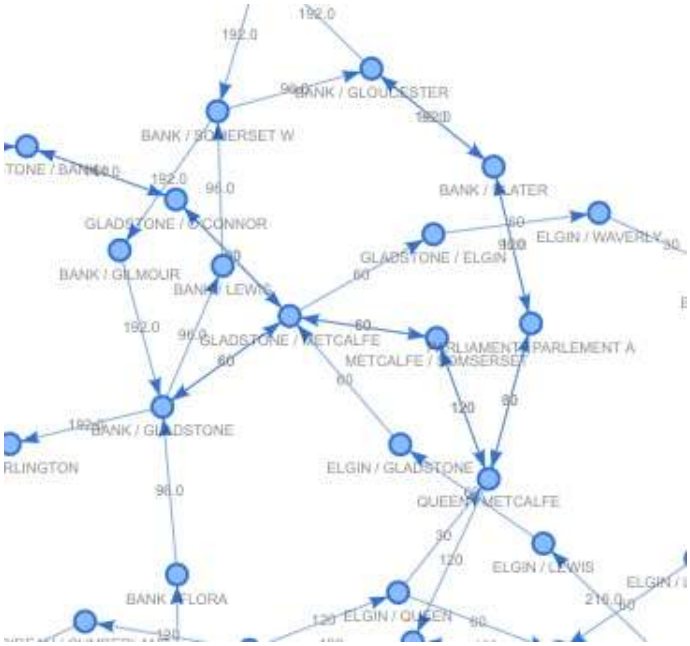
Fig 3, showing the original bus route for the 7 St. Laurent (right), vs the 7 Carleton (left)[4]

However, as shown in Fig #3, the 7 Carleton does not take this detour and instead goes down Sunnyside. We decided to visit the area in person, observe the differences between Sunnyside and Grove, and observed that Grove was much narrower (as it was one-way), meaning parked cars created bottlenecks and made passage difficult. The route was not plowed

[4] https://moovitapp.com/ottawa_on-422/lines/7/234393/686807/en?customerId=4908&ref=2&poiType=line

as often as Sunnyside (during the winter), and Grove had no major intersections on either end, which led to increased congestion and bottlenecks. Since Sunnyside was a two-way street, and the 7 Carleton already went down it, we decided to open up the path for the 7 St. Laurent, opening up directed edges along this path for the bus to choose. We did this with various other vertices and edges in the network. However, since they all followed this similar process of investigation and change, it is sufficient only to show one example so as not to clutter this paper.

In addition to opening up pre-existing vertices for travel, we also wanted to add edges and vertices where we believed they could offer alternate routes between vertices on the network where buses only currently had a single choice of destination, as if buses only had one choice, we could not effectively model their decisions, as they would have none to make. In order to effectively add routes, we looked at real-world maps and added stops and routes near the original route, but they provided some benefits or drawbacks.



One such example is the addition of a route to travel the network from the Bank/Gladstone vertex to the Queen/Metcalfe vertex. We also added a new vertex, representing Metcalfe/Somerset, along the new route. Our reasoning behind this was that Bank Street is a highly congested area, where buses are highly likely to decide to take shortcuts. Hence, we needed to ensure they had options to keep our model practical and valuable. Figure 4 (generated as output from our code) shows the new.

Fig 4, showing an excerpt of the network graph generated by our code, with the new path having been added

Adding these two types of changes (one-way to two-way and addition of edges and vertices) also highlighted the difference between the two priorities an agent might have; maximizing profit or minimizing travel time. In the Sunnyside example, travelling down Sunnyside may be faster, but there may be more passengers on the Grosvenor-Grove route, so this is a consideration the agents on the network will have to make.

Having now generated vertices (stops), edges (weighted for travel time between vertices) and agents (buses), we needed to add passengers. Passengers, we decided, would be modelled as having a randomly assigned initial starting vertex location somewhere on the graph network and a randomly assigned destination vertex somewhere else on the graph. When a bus would visit a node, a passenger would 'board' and inform the bus driver of their desired location, which would allow the bus driver to make decisions based on the information given, and the information of the network. However, this introduced the issue of passengers potentially wanting to go somewhere that a single bus from their current location cannot take them, but that will be discussed later in the analysis section of this paper.

We decided on three amounts of passengers on the network, 1,000, 10,000 and 100,000. We chose these amounts as we wanted to examine how different passenger counts would interact with a broadly consistent travel cost, as the number of passengers currently riding a bus does not affect the individual bus' speed (1 passenger on a bus makes the bus move at the same speed as 30 passengers do). We also set a limit of passengers on the bus as fifty to allow for congestion and give the buses more say in choosing passengers based on their priorities, allowing for a more significant difference in results. In addition, we simulate that picking up passengers takes 10 seconds and that dropping off passengers also takes 10 seconds. We ran two simulations (based on differing priorities) of the four agents (buses) acting on the graph network, each with the three different amounts of passengers. We produced graphs of their output to model the different amounts of passengers.

3. Running Simulations on the Model

Now that we had a working model, with vertices and edges, and passengers and buses as weights of those edges and agents that would make decisions on the graph network, we needed to write the algorithms to allow the agents to make said decisions. First, it would be beneficial to discuss, in exact terms, the two priorities, travel cost and profit, and what they mean for the agents making decisions on the network.

If an agent (bus) has been told to prioritize travel costs on the network, the bus will prioritize reaching passenger destination nodes as fast as possible (i.e. travelling along the most minor weighted edges whenever possible). However, with this method, we cannot just have every passenger that boards the bus being the next closest destination, or the bus would be too 'short-sighted' and would constantly be changing its path, resulting in terribly inefficient routes.

So, given this, we decided that the bus would poll the bus passengers, find the passenger with the furthest destination node on the route, and find the shortest path to that specific node. If there were too many passengers trying to board, priority would be given to passengers travelling to the furthest nodes on the network. Thanks to this method, the above issue of 'short-sighted' buses is fixed, as buses are still finding shortest paths, but to a mostly unchanging endpoint because the bus is always taking the shortest path to the next node on the shortest path to the end node.

For example, a run of the 7 St. Laurent with travel cost priority would look like the following: At the first node on the route, the bus will stop, poll all passengers, and see if any of the passengers are going to the 7 St. Laurent's final destination. If there are no passengers who are, it will recursively ask about passengers going to the next furthest node until it finds a passenger. Once the bus departs, it will take the shortest possible path to the next node along its path (to the furthest passenger's destination node). Once it arrives at the new node, it will poll all passengers to see if anyone is *further* than the current furthest passenger's destination node (it still must be along the bus' route). If so, it will change the furthest passenger's destination node to the new passenger's node and repeat the process. We will talk more about our findings given this priority in the result section of this paper.

For situations where the priority of the bus is profit, we model each passenger that boards as being worth five dollars of profit. The bus' priority is to pick up as many passengers as possible while still trying to save on travel time (though not as ruthlessly as the previous priority). The bus models this priority by arriving at a node and, if there are too many passengers trying to board, picking up passengers who are going to the following closest nodes on the graph (i.e. will only be on the bus for a short amount of time), allowing for more passengers to cycle through the bus. This again assumes that the bus driver will be polling the passengers to get an exact destination, which, in a real-world scenario, could be done via an app (inputting a destination) or simply telling the bus driver the desired stop. Once a bus departs, the shortest path to the next closest node is taken, and this process repeats.

Now that priorities were defined and algorithms had been written representing agent (bus) behaviour with a given priority, we had to find the best method of calculating the shortest distance. Dijkstra's algorithm[5] was the most apparent solution for the shortest distance of positively weighted edges and a graph network. We implemented this as the buses' algorithm to find distances for the shortest paths between two given nodes (start and endpoints). We added an

[5] Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford (2001). "Section 24.3: Dijkstra's algorithm".

extra variable to the algorithm to shorten run-time, which stored the previous node, which allowed us to retrieve a list of the shortest path nodes by iterating from the target node and calling the stored previous node value until the source node was reached. This meant that the return from the algorithm would be a list of shortest paths of all nodes between the starting node and the end node, which the buses can use to determine the shortest paths along their travel route. This was the last piece needed for the algorithm to work, so now we can move on to examining the output of our simulation and what it means for our central research question.

4. Results and Analysis

We ran two simulations for each of the eight buses, with three different passenger counts, resulting in a considerable amount of network graphs (for all different runs), as we also included the results of the original route (which contained no modifications). We also generated line graphs to demonstrate the difference between runs as results instead of paths (showing how much profit was made on the modified versus original and what was the travel cost of the original route versus the modified route). As a result, showing every graph generated is impossible in this paper, so instead, we will focus on graphs for just one bus and see what trends can be examined and analyzed from the results shown for the given bus and whether or not this was reflected in other buses, and whether or not it had an effect on other buses (as they all share the same network). Of course, all graphs are generated by the attached code, so all graphs can be viewed, if desired, under the graphs folder of the relevant attachment. The bus selected for this purpose is the 7 St. Laurent, as other paths were too short or did not vary enough for our purposes. The bus route starts at Carleton University and ends at St. Laurent/Lemieux no matter what modifications or differing decisions the bus decides to take along the route, the beginning and end destination node must be the same. Therefore, the best way to analyze the results is to look at the sections *between* the beginning and endpoints that have changed and examine why they have done so and what it means for the network as a whole. It is important to note that the graph may look slightly different in node placement when generated. However, the overall general structure is preserved, even with slight visual differences.

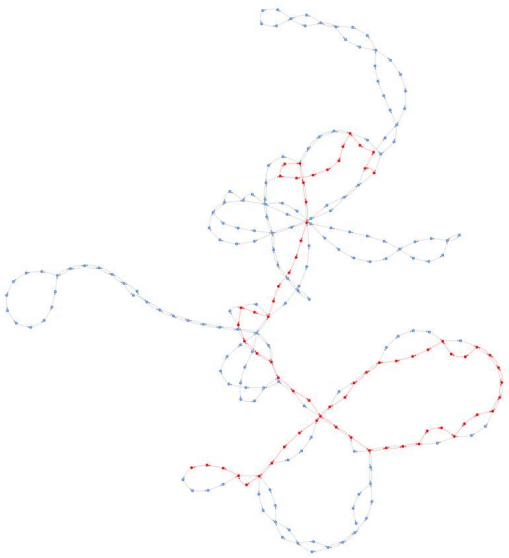


Fig 5, showing the original route of the 7 St.Laurent along a network graph generated by our code

Seeing as the above graph is for the original, non-modified route (the bus was unable to make any decisions), let us examine the output of a run where the bus *is* allowed to make decisions, given a travel-cost priority, and 10,000 passengers on the network. The 7 St. Laurent now looks a little different. It has taken a vastly different path, skipping over many stops, such as the sizeable loop-like route in the bottom right of the network graph (representing Ottawa's Beechwood-St. Laurent area). It has made other decisions, such as which node to go to immediately after beginning the route (originally was University/Gymnasium, but is now Raven/University). Of course, by following the algorithm as described in section 3, we can understand how these changes came to be.

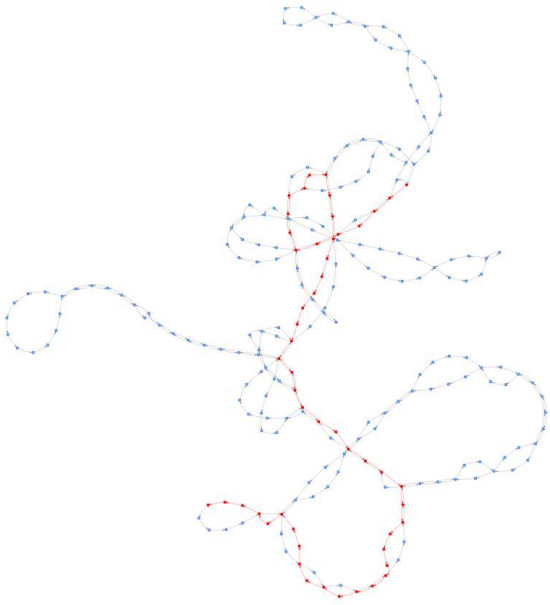


Fig 6, showing the the modified route of the 7 St.Laurent, with 10,000 passengers and a travel cost priority along the network

In this modified path, a shorter route was taken between destinations that passengers who had been picked up wanted to reach, often skipping over significant stops (such as the Beechwood-St. Laurent stops). While, as one would expect, the travel cost was significantly lower for passengers who managed to board, a large number of passengers who would have been included on the original route were skipped over and did not have the chance to board, resulting in a much lower profit, and many fewer passengers picked up. So, as we expected, focusing on maximizing travel efficiency (and therefore minimizing travel cost) was adequate, but at the cost of the bus becoming a less proper form of travel to the many passengers left behind. Another takeaway from this simulation (and knowing our algorithm) is that the bus in this graph has reached Nash Equilibrium given its best response dynamic of travel-cost priority, as no other path on the network would allow it to reach its destination faster with the passengers, as it could not ignore its passengers. It is forced to at least drop off every passenger it picked up to a destination along the way.

Other runs (for buses not shown, 6,8,10) on the network ran similarly, getting to the final node much quicker, but at the cost of passengers being left behind. Since all buses run simultaneously for our simulation, the actions one bus takes affect the rest. The locations where the 7 St. Laurent deviated experienced higher than usual congestion, meaning paths usually taken by the 7 St. Laurent became cheaper. Hence, buses that shared paths in common with the

7 St. Laurent found their regular routes cheaper (as the 7 St. Laurent was no longer taking these paths, clearing congestion), and therefore had no reason to deviate. This resulted in many original runs showing the same on the network graph, as the original path was still in-line with the new priority strategy.

The priority of profit strategy also produced interesting results. Here, the newly modified path (for 10000 passengers) is shown in red, and the areas where this original path would deviate from the new path are shown in black. Again, many sections are largely ignored in favour of a more 'appealing' section of nodes, according to the best response dynamic. However, unlike with the travel cost priority, we can see that more passengers are picked up, meaning that although sections were ignored (like they were in the travel cost priority), the overall usefulness of the bus (in terms of passengers picked up) was improved. This was an interesting result and showed that if buses can service areas of the high-density population (i.e. more passengers) exclusively, then the bus becomes more effective, at least for those living in the densely populated areas. Of course, the purpose of a bus is to provide transport to all, but these are interesting indicators of where and how the OC Transpo may choose to modify its networks. Again, the other bus routes (although not shown in this section due to the large number of them) behaved similarly, picking up more passengers, although leaving out many sections of paths and stops. Again, Nash Equilibrium was reached for all buses, as the number of passengers picked up was maximized for this best response dynamic.

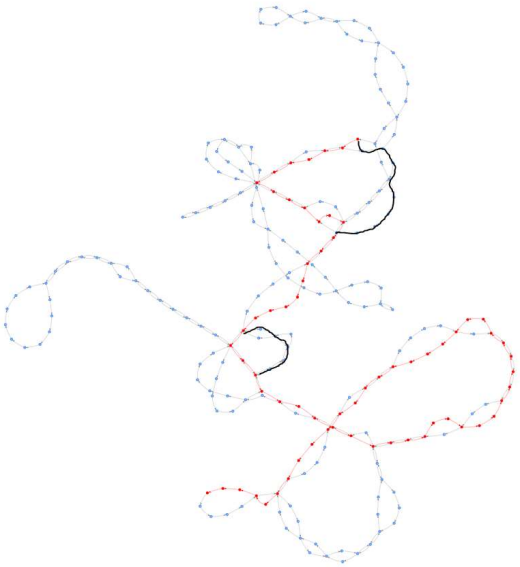


Fig 7, showing the the modified route of the 7 St.Laurent, with 10,000 passengers and a profit priority along the network

The network graphs do not tell the entire story as useful as they are. Since we are using numerical values to measure the success of these strategies (the number of passengers picked up for the profit priority or the amount of time spent on the route for the travel priority), we need a way to visualize these amounts and how they differ between passenger amounts and priorities, which is why we generated line graphs for each of the runs. They can be found under the 'line' folder of the submission, but we will again examine the results of the 7 St. Laurent here. Each

bus route has six graphs, showing either a passenger count, a profit count, or the total travel cost for each strategy. However, since the passenger count is closely related to the profit count, only showing one of these two types of graphs is sufficient.

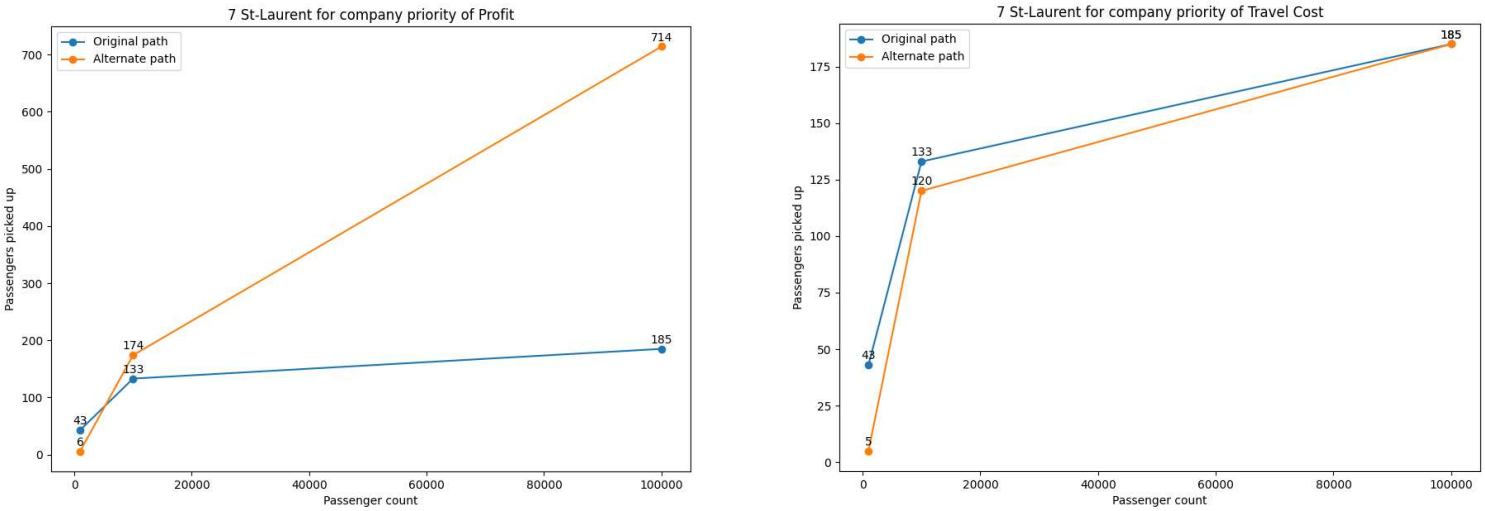


Fig 8, showing two line graphs comparing the passengers picked up for different best response dynamics strategies.

The above two line graphs show the results between all runs for the 7 St. Laurent and how they differ between strategy and the total number of passengers on the network (y-axis) when it comes to picking up passengers (x-axis). Broadly, the results are what one would expect, the Profit priority picks up a more significant number of passengers, growing larger with larger numbers of passengers to pick up. In contrast, the travel cost priority, which does not prioritize passengers, picks up far fewer than if it had stuck to the original path. However, one fascinating result is that if the passenger count is equal to 1000, the profit strategy fails. It picks up far fewer passengers than if it had stuck to its original route. Why is this? With the profit strategy, an agent wants to pick up more passengers but also wants to 'save on gas,' so a shorter route is desired. Despite wanting the most passengers, the agent still takes the shortest path to its destination. When there are low passenger counts, the agent may finish the route after taking the shortest path, so it does not visit the original path (as it has found a faster route). However, this faster route (for those who are boarded) likely visits nodes with very few passengers, as all 1000 passengers are randomly assigned to all nodes. We can see that this odd case is corrected as the total passenger count improves, with the profit strategy picking up multitudes more passengers than with the original route.

Another interesting result is that the number of passengers picked up for the travel cost priority route and original route converge at 100,000 passengers. This is due to a situation that can develop with our model. On the original path, we stop at every required node. However, any passenger picked up at the first node who wants to travel to the end node will be on the bus for the entirety of the run, taking up space from passengers who may want to get on and travel a shorter amount of distance before disembarking. This case of passengers' overstaying' their welcome causes the convergence at higher numbers.

The final graph to be examined is the line graph showing the difference in travel cost (x-axis) between the two strategies for different total passenger counts (y-axis).

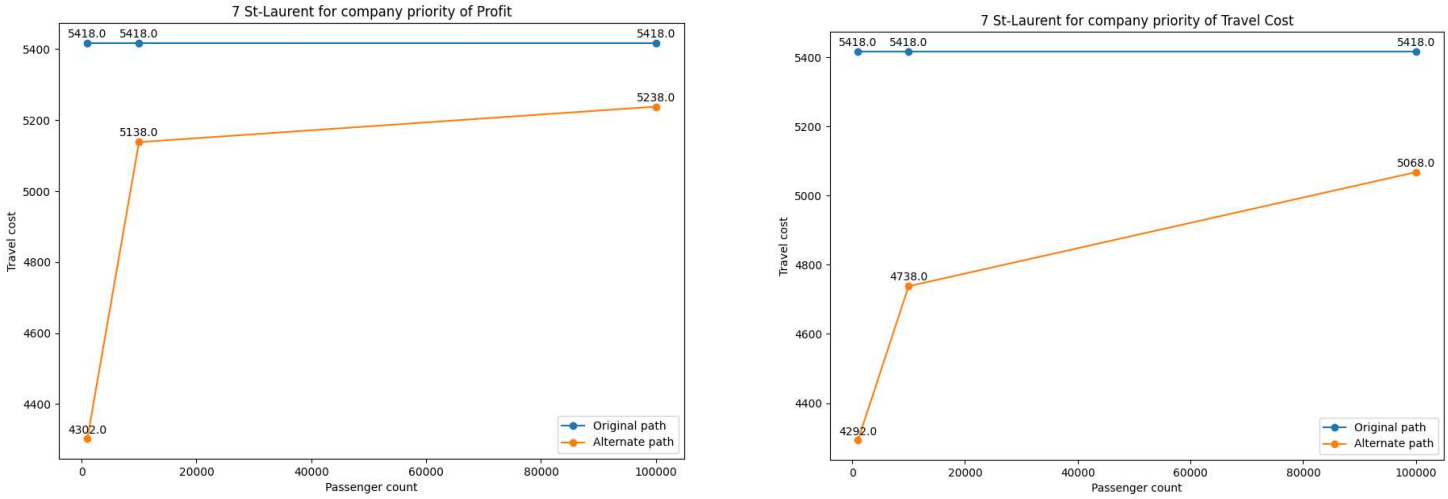


Fig 9, showing two line graphs comparing the total travel cost for different best response dynamics strategies.

These graphs showcase a very intriguing and helpful result. Regardless of the strategy chosen, total travel time is improved along with the network. Of course, as the passenger count of the network grows, so does the travel cost, but it never exceeds the flat travel time of the original routes. The travel cost priority is, as we would expect, faster, but not by as much as expected when compared to the profit priority, which also improved on travel time, while picking up a much larger number of passengers. After a bit of confusion, we realized that because buses on a profit priority do not stop at every node (and seek out specific higher density nodes), they reduce their travel cost. These results, showing that travel costs can be improved by choosing either strategy, show that the buses' original route is not the best (when considering these two priorities) and can be improved.

5. Conclusion

Given all of our results, we can now revisit our research question and see what questions have been answered. Our research question was primarily examining how the modification of the road and bus network by applying social network theory impacts the overall OC Transpo bus system? Given different company strategies, buses will find different Nash Equilibriums (best profit, lowest travel), giving different meanings to what Nash Equilibrium means on our network, and that as these buses reach Nash Equilibrium, it affects the congestion of the road network resulting in different outcomes for all subsequent buses, as we saw from the network graphs. In addition, our results from the line graphs show us that there is a better path possible than the original. Although we can choose to maximize profit or minimize travel costs, the overall travel cost will still be much lower than the original path. Solid cases could be made to adopt either of these two strategies, which choose either to prioritize customer experience or prioritize profit. There is a clear advantage to choosing the profit model from a pure number basis. The travel cost is lower, while the profit/passenger count skyrockets to huge numbers, not even approached by the original route or travel cost strategy. However, if the company was trying to set records or improve travel time, there is solid reasoning for choosing the travel cost. Choosing which strategy comes down to what priorities the company holds.

Overall, we are pleased with the discoveries made in this paper, but there is always more to learn. It would be helpful to model buses choosing different strategies on the same run in the future. In our current program, all buses are given the same priority as other buses. Mixing and matching priorities between buses and routes could generate more valuable data and show a different kind of Equilibrium, where agents on the networks have differing ideas of their own best response. This would better bridge the gap between two strategies and perhaps inform that a mixture of the two strategies would be helpful and more beneficial than a rigid adoption of one single strategy. In the future, modelling cars and changing congestion based on time of day would also generate more helpful information. Perhaps adopting different strategies during different times of the day would result in a more optimal network. These would allow the network to become more adaptive to dynamic changes, and upgrading our strategies to respond to and accommodate these possibilities would significantly impact passenger experience and perhaps the public attitude towards the OC Transpo as a whole. Much like the people and the city, the roads and networks of Ottawa are ever-changing, and new strategies are needed so that the city does not fall behind.

CITATIONS/SOURCES:

Khan, T. (2021, October 12). *Problem-plagued OC Transpo is teaching me to lower my expectations of public transit*. Capital Current. Retrieved April 18, 2022, from https://capitalcurrent.ca/problem-plagued-oc-transpo-is-teaching-me-to-lower-my-expectations-of-public-transit/

The Environmental Advantages of Commuting by OC Transpo Bus : a Report to OC Transpo. Ottawa: Environment House, 1990. Print.

Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford (2001). "Section 24.3: Dijkstra's algorithm". Introduction to Algorithms (Second ed.). MIT Press and McGraw–Hill. pp. 595–601. ISBN 0-262-03293-7.

A Tradition of Service -- and the Continuing Quest for Excellence. Ottawa: OC Transpo, 1991. Print.

Serving the Customer : the Quest for Superior Public Transit. Ottawa: OC Transpo, 1990. Print.

Decisions. IPC. (n.d.). Retrieved April 18, 2022, from https://decisions.ipc.on.ca/ipc-cipvp/orders/en/item/128197/index.do#:~:text=of%20West%20Carleton.-,OC%20Transpo%20is%20an%20incorporated%20body%20consisting%20of%20nine%20members,services%20in%20the%20regional%20area.

https://moovitapp.com/ottawa_on-422/lines/7/234393/686807/en?customerId=4908&ref=2&poiType=line

https://www.octranspo.com/en/plan-your-trip/travel-tools/developers/