Waste Collection Optimization: A case study in Port Coquitlam, British Columbia

Abstract

Waste collection is a re-occuring task that is a version of the capacitated vehicle routing problem (CVRP). CVRP's require a distance matrix which can be expensive to compute. In this paper, we present an efficient method for the distance matrix construction using an "Adjacent Grid Querying Method" in conjunction with Dijkstra's Algorithm that is capable of reducing total queries made by 99.7% and is scalable to large data-sets. We present methods for specialized constraint construction for waste collection vehicles capable of only collecting waste from the right side and for preferential servicing by region. We then perform route optimization using Clarke-Wright's Savings Algorithm, implemented by Google OR Tools.

To measure the success of our algorithm, we perform a case study on Port-Coquitlam, a mid-sized municipality in British Columbia. Port-Coquitlam requires us to address an unusual constraint, the prioritization of waste collection in forested regions to avoid bears that forage through bins awaiting collection. The routes generated by our algorithm are up to 32% faster for most waste collection zones and 16% slower for other zones, compared to the current implementation by the municipality. Our data collection and processing procedures and optimization algorithms are all presented, with source code as well.

1 Introduction

1.1 The Problem Setting

Waste collection is a routine service that either the local governing body or a privately operating company provides [Erd+19]. There is considerable pressure to optimize the waste collection procedure to reduce the economic and environmental burden [Erd+19]. Many municipalities employ similar strategies for waste collection: five distinct waste collection zones operating on a five day collection schedule, with m waste collection vehicles and m routes per zone. Although not entirely identical, the underlying features of the optimization algorithm such as the routing between locations, the clustering of service points and the geographical data required all have many similarities. Due to the shared features between regions, optimization methods need not differ greatly; the techniques used in a particular Waste Collection Optimization (WCO) algorithm are often applicable in other settings as well.

In this paper, we outline the methods used to obtain the dataset and describe in detail the procedure used to generate the inputs to our WCO model. We explain the techniques used to construct and optimize the model and its constraints in the context of Port Coquitlam, a municipality in British Columbia that we have chosen to work with. Port Coquitlam's waste management services operates under similar constraints to the collection procedures previously described:

- A five-zone-five-day collection procedure
- Four to seven waste collection vehicles, varying by demand throughout the year
- Waste collection vehicles outfitted with mechanical arms, enabling collection only from the right side of the street
- Three waste types collected: organic, garbage and recyclables, with organic waste collected every week and garbage and recyclables alternating each week
- Organic waste must be collected from homes near forested areas first, as bears forage through the bins in search of food

All of the above constraints will be accounted for in the solutions generated by our algorithm.

1.2 Hierarchical Clustering

Hierarchical clustering [Hen+15] [KJK14] [War63] is a well accepted and developed method for hierarchical grouping. This clustering technique requires a *complete distance matrix* D, and number of clusters k as input, where $k \leq n$ if D is an $n \times n$ matrix.

Definition 1.1. A *Distance matrix* D is defined as $D \in \mathbb{R}^{n \times n}$, where $D_{i,j}$ is the element in the i^{th} row and the j^{th} column. $D_{i,j}$ then contains the distance from house i to house j. A Distance matrix is also a graph.

Definition 1.2. A graph is called *complete* if each pair of distinct vertices, i, j, are connected by a unique edge.

The *metric* itself, in the traditional mathematical sense, is used to determine which points in the data-set, in our case the houses, are most similar to one another. It begins by grouping together the two points in the dataset that have the smallest distance [War63], i.e., the houses that are closest together. These two points then become a new single point and the method is reiterated until the desired number of clusters, k, remains. In this paper we will describe the use of Hierarchical Clustering to generate waste collection zones for Port Coquitlam.

1.3 Vehicle Routing Problem

The Vehicle Routing Problem (VRP) is a generic name given to a class of problems that involve creating travel routes for either a single vehicle or a fleet of vehicles. The vehicles start at a central depot(s) and are required to visit specific service locations with both capacity and distance restrictions [PK86]. The aim of the problem is to minimize the total distance or deterioration of goods, maximize fulfillment, and so on [And+21]. In the context of this paper, the capacitated nature of the problem is solved by varying the number of waste collection vehicles. Port Coquitlam currently utilizes between four and seven waste collection vehicles depending on the estimated waste amount for a given time of year, most notably, in the summer months there is an increase in the amount of organic waste produced. To account for this, we

have generated solutions for all waste collection zones using four to seven waste collection vehicles.

1.4 The Waste Collection Optimization Problem

The VRP in the context of WCO has been studied for decades [JST08]. There are several heuristic methods that achieve success in finding sets of optimal solutions for both single vehicle and multi vehicle implementations of the VRP.

1.4.1 Route Optimization in the context of Waste Collection

The main aspect of route optimization problems consists of the design of optimal and cheapest distribution patterns to serve customers that are scattered over a region of space [BE06] [BG11]. Frequent objectives of these problems is to reduce the number of waste collection vehicles used, minimize the travel distances and reduce the maximum length of a route. These objectives all have the underlying intention to decrease logistical costs and environmental emissions [LGA07] [Nuo+06] [Del+19]. Due to these complexities, the WCO problem has posed a considerable number of operational problems for municipalities. In order to develop an effective waste management system a reduction of operational expenses and optimization of vehicle fleet size is absolutely necessary [Sah+05]. More recent studies have noted that minimizing the distance may not be the best parameter to optimize [Tav+09]. This is due to the fact that in more urban areas some of the shorter routes may be inconvenient for a large waste collection vehicle to drive through, leading increased fuel consumption and congestion. Thus, in this paper we will be minimizing the total route time, not the route distance. Route time is the current metric that Port Coquitlam uses as well, it is one of the easiest to understand, implement and generalize, and therein lies the value.

1.4.2 Clarke-Wright's Savings Algorithm, an optimal route finding method

Clarke and Wright presented a greedy heuristic that is widely known as the Clarke-Wright(CW) savings algorithm in 1964 [Cao] [CW64]. The algorithm generalizes the so called *Travelling Salesman Problem*, and finds the optimal set of routes for a fleet of m vehicles to traverse a given set of points [CW64].

Given a Distance matrix as input and m vehicles, the algorithm returns the optimal routes to traverse all the points in the Distance matrix with the m vehicles. The CW algorithm starts with an individual route for each house and merges routes iteratively as long as the combination can reduce cost and meet the constraints until no further merge is possible and the number of routes is less than m, the number of vehicles. This algorithm has proven to be an effective approach for solving real VRPs and in particular, finding good initial solutions for VRPs [Cao]. Since the CW algorithm was first proposed, many sub-sequential papers have presented alternate versions of the original implementation featuring several improvements, as well as other solution methods that utilize CW savings algorithm to generate the first solution before applying other heuristic methods to improve the solution [Cao]. In this paper we will utilize a sophisticated implementation of the CW savings algorithm that was developed by *Google Operation Research Tools* [PF19]. The algorithm uses CW savings to generate an initial solution and then uses several solution improvement techniques to create an optimal heuristic solution.

2 Data Collection

The goal of the data collection procedures is to generate a complete *Travel Time* matrix that is the required input to the optimization model.

Definition 2.1. The *Travel Time* matrix is a Distance matrix, where $T_{i,j}$ is the travel time from house i to house j

It is also convenient for the purpose of route optimization, that the graph representation of the Travel Time matrix be complete. In this section we will describe the procedure that generated our Travel Time matrix. The matrix itself is approximately $10,000 \times 10,000$, and is stored both locally and in a cloud server.

2.1 PoCo Maps Geographic Information System

Port Coquitlam has built a web based Geographic Information System that allows us to collect the address and municipal zoning information regarding every house and business located within its jurisdiction. We were able to extract the information in a collection of "comma separated value" files (.csv).

Collecting the addresses of each service point is a necessary step to create the desired Travel Time matrix for our optimization model, as we will need them to query the geographical coordinates.

2.2 Google Geocoding API

Once the address of each house is collected, the Google Geocoding API [Goo21] is used to approximate the longitudinal and latitudinal coordinates of each home. The Geocoding API is accessed through an HTTPS request using an API authentication token obtained through the Google Cloud Services Console. A Python script is used to create the HTTPS request and collect the data. Once the geographical coordinates are obtained the houses can be arranged geo-spatially by referencing their coordinates.

2.3 Adjacent Grid Querying Method

2.3.1 Motivation

The next step in the procedure to generate the Travel Time matrix would be to query the travel times between homes. However, to query the travel times between every pair of houses ($n = 10,000$) to generate a complete Travel time matrix would require $100,000,000$ API queries. This would be a waste of resources. Thus, reducing the number of queries stemmed the creation of the "Adjacent Grid Querying Method". This method is based on the following notion : *At any point in time, the next closest house to any given house will be a house that is directly adjacent to it*

2.3.2 Explanation

From the Geocoding API we have obtained the coordinate of each home, thus each home can be arranged in space according to its geographical position. From here we can create an $n \times n$ grid that covers the entire space.

Definition 2.2. The adjacent house grid G is defined such that for every house h_i, $i \in [0, 10000], h_i \in G_{j,k}$ for $j, k \in [0, 60]$. Thus G is a 60×60 grid, uniformly spaced over the spatial region of the City.

Then for every house h_i we will query the distances from house h_i to every house located in its own grid and the houses in the *adjacent* grids. Here, an

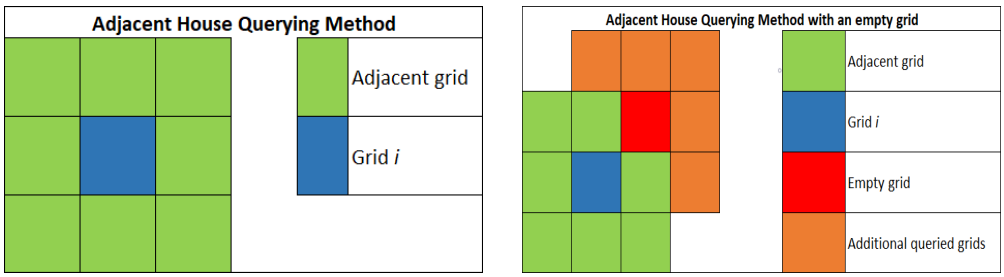

Figure 1: How the Adjacent Grid Querying Method calculates adjacent grids in the regular case (left) and in the case with an empty grid (right)

adjacent grid is any grid that either above, below, on either side or diagonal to the grid in question (**figure** 1). If no houses are located in any adjacent grids then we will search the next adjacent grids for houses. The number of grids, $n = 60$, was determined experimentally by increasing the number of grids until the total number of queries required to create the Travel Time matrix was reduced to a "reasonable amount". In this case the reasonable amount was $300,000$ queries, a reduction of 99.7%. Computationally, $300,000$ is a reasonable number of queries that our hardware can perform. Of course, this method generates a sparse matrix that is *not complete* (but still connected). Since we require the final Travel Time matrix to be complete, this poses a problem, however this can be alleviated as later shown.

2.4 Google Maps Directions API

The Google Maps Directions API was used to query the travel times and directions between pairs of houses that were computed using the Adjacent Grid Querying Method. Similar to the Geocoding API this is done using an HTTPS request using a secure API authentication token. Once the distances and directions are queried we have a sparse matrix of Travel Times, the next step in this procedure is to generate a complete matrix.

2.5 Complete graph generation: Dijkstra's Algorithm

To generate the complete matrix we have used Dijkstra's Algorithm [Dij59] [CFM18], a highly referenced and well known algorithm for finding the minimum spanning tree of a graph. Numerous forms of Dijkstra's original algorithm exist [Zha+16], however the original version is sufficient for our

purposes. The algorithm can be summarized as follows.

Definition 2.3. A graph is called *connected* if there exists a path between any two points on the graph

Definition 2.4. Let $t(n_i, n_j)$ be the travel time from node i to node j

We start with a connected graph, the current sparse Travel Time matrix. We arbitrarily select our starting node n_1. We examine the travel time from n_1 to all adjacent nodes - that is, the nodes directly connected by an edge - and select the node that has the minimum time to n_1, let it be n_2. We then record that we have visited n_1 and we record the time from n_1 to n_2 as $t(n_1, n_2)$, we then traverse to the n_2 node. We examine the distance from the n_2 node to each adjacent node that has not been visited yet and select the node with the minimum time, let it be n_3, we then record the time from the starting node n_1 to n_3 as $t(n_1, n_3) = t(n_1, n_2) + t(n_2, n_3)$. We continue this process until we have visited every node in the graph, the result is a *minimum spanning tree* for the graph which contains the shortest path from the starting node to each other node in the graph.

Definition 2.5. A *spanning tree* of a connected graph is a tree that minimally includes all the vertices of a graph.

Definition 2.6. The *minimum spanning tree* is a spanning tree with an assigned weight less than or equal to the weight of every possible spanning tree of a connected graph.

Definition 2.7. The weight of a spanning tree is the sum of all the weights assigned to each edge of the spanning tree

Once we have computed Dijkstra's Algorithm we obtain the shortest path from the starting node to every other node on the graph. We then use this path to calculate the minimum distance from the starting node to every other node. Then for a given node i, we have computed all the $T_{i,j}$ entries for the i^{th} row of our Travel Time matrix. We reiterate this procedure and select a different starting node until we have computed $n = 10,000$ minimum spanning trees and thus have filled all n rows of our Travel Time matrix.

The time complexity of Dijkstra's Algorithm is $O(|E| + |V| \log(|V|))$, where E is the number of edges and V is the number of vertices. Since

our initial Travel Time matrix is sparse the number of edges is significantly less than the number of vertices, thus the time complexity in application is $O(|V|\log(|V|))$ for a single iteration and $O(|V|^2\log(|V|))$ to compute the entire complete Travel Time matrix. This makes Dijkstra's Algorithm quite feasible to use on even a modest computer, taking just 2.5 hours to compute the complete Travel Time matrix for Port Coquitlam. We computed these results using a Python implementation that is available through a 3rd party package [Vir+20] the results of this packaged implementation were cross referenced with our own implementation of the algorithm to validate our results.

3 Waste Collection Zone Optimization: Hierarchical Clustering

One of the mathematical undertakings for this paper involves creating new waste collection zones for the City using hierarchical clustering. The City currently operates on a common five day collection schedule, to optimize the zoning while adhering to this scheduling we used hierarchical clustering with the travel time between houses as the similarity metric. We utilized an implementation by Scikit-Learn called Agglomerative Clustering [Ped+11] with a cluster number $k = 5$ and linkage type 'average'. The linkage type determines the method used to measure the distance from one cluster to another, since there are multiple homes in a single cluster there are many ways to measure this distance, including the maximum distance (supremum) and minimum distance (infimum), using the average distance proved the most effective.

Despite multiple model tuning efforts, the clustering attempt was relatively unsuccessful because the algorithm generated two large clusters and three small clusters. The formation of these clusters are a result of the geography of Port Coquitlam where there is a divide created by railroad tracks across the city that separates it into two zones, along with several small pockets of land that naturally provide singular clusters. These clusters can be see in **figure** 2. To combat this we developed a *balancing algorithm*, let $C := floor(\frac{10{,}000}{5})$, where we are dividing the total number of houses by the number of clusters. Then C is the approximate desired number of houses

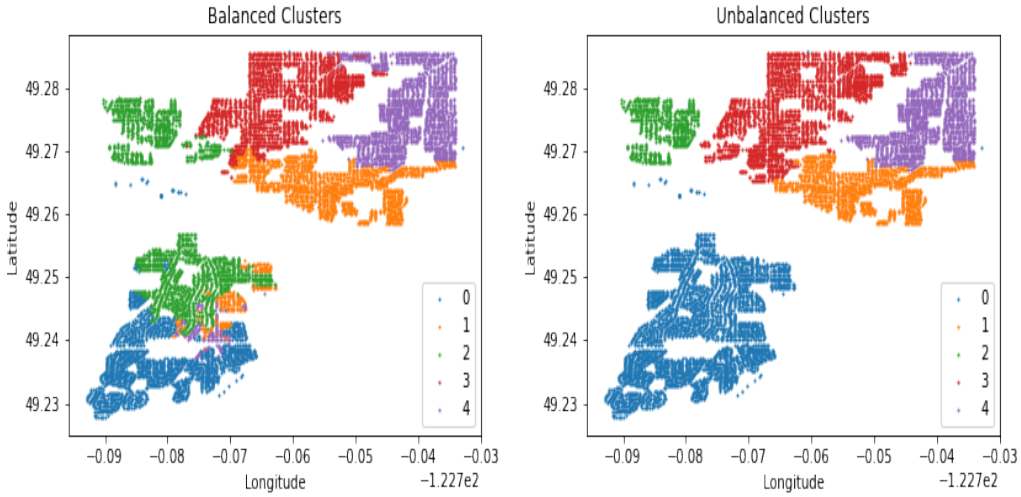

Figure 2: Balanced (left) and unbalanced (right) clusters using hierarchical clustering

per cluster. After generating the initial Hierarchical clustering solution we proceeded to remove move the element most similar to the smallest cluster that currently resided in the largest cluster to the smallest cluster. This was repeated until the number of houses in every cluster was approximately C. This resulted in much more balanced clusters (**figure** 2), however, these clusters were still not ideal because the clusters were not well-defined. For example, given two houses on the same street, it was possible that the algorithm resulted in each house being a part of two different clusters. From a simplicity and public relations perspective this cannot be allowed. Thus, since our efforts to re-optimize the waste collection zones did not result in any improvements, this indicates the original clusters are well designed, and for the remainder of the paper we shall utilize the predefined zones that the City currently uses (**figure** 3).

Figure 3: Current waste zones in The City

4 Waste Collection Route Optimization

The Waste Collection Route Optimization model determines the best possible routes that pass through every single house in a given waste collection zone. The inputs to the model are

- m waste collection trucks
- The addresses of the houses to service
- The Travel Time matrix
- The maximum time before houses near forested areas must be serviced

The model assumes that all trucks are identical, which is valid in the case of Port Coquitlam. The output of the model is the optimized route that is presented as a sequence of addresses for a given waste collection zone.

4.1 Vehicle Routing Problem: Constraints

Our model has three main constraints:

- The maximum length of a route

- The maximum time before houses near forested areas must be serviced
- Waste collection can only occur on the right side of the waste collection truck

The collection of organic, waste and recyclable waste all follow identical routes, thus reducing the number of problems required to be solved. To account for the varying number of trucks used by the City (four to seven), we have generated solutions for all numbers of trucks for all five waste collection zones. For the optimization portion of the algorithm, the model assumes that waste pickup is instantaneous. After the solutions are computed an additional sum of 15 seconds per house will be added to each route to approximate the waste pickup time for each route.

4.1.1 Route Balancing

By constraining the maximum length of the route we are able to ensure the routes are balanced. For a given instance of m waste collection vehicles, the total length of each route in seconds must be less than some constant r. This forces the routes to be somewhat evenly distributed. The value of this constant is determined experimentally by first generating a solution with a single truck to determine the maximum possible route time for a single route, r_{max}, then $r = \frac{r_{max}}{m} + \gamma(r_{max})$, where $\gamma(r_{max})$ is some added padding to ensure a feasible solution can still be generated mathematically. The exact value of $\gamma(r_{max})$ is determined on a case to case basis using a binary search, **table** 1 lists the values by zone and number of vehicles.

4.1.2 Bear Avoidance

Routes must collect organic waste from bear dense zones first to minimize the risk of bears foraging through waste bins and posing a threat to bystanders and vehicle operators. For every house near a forested area we have constrained the estimated waste pickup time at that house to be less than or equal to the smallest constant value that allows for the model to find a feasible solution. This forces the model to collect waste from those locations prior to collecting waste from other locations. This constant is also determined experimentally using a binary search (**table** 1).

Coeff Type	# Vehicles	Zone 1	Zone 2	Zone 3	Zone 4	Zone 5
Route Balancing	4	5200	5400	5000	4375	4400
	5	4400	4500	4000	3600	3400
	6	3800	4000	3500	3125	2800
	7	3200	3500	3200	2750	3600
Bear Avoidance	4	3600	3600	NFA	NFA	NFA
	5	3600	3600	NFA	NFA	NFA
	6	3600	3600	NFA	NFA	NFA
	7	3600	3600	NFA	NFA	NFA

Table 1: Table of experimentally determined model parameters, NFA denotes No Forested Areas

4.1.3 Right side waste collection

The waste collection vehicles can only collect from the right side of the street Without this constraint, the model routes waste collection in a zig-zag fashion, collecting waste from houses across the street from one another, as seen in **figure** 4. This is impractical, due to the mechanical arm that lifts the waste bin only being on the right side of the truck. To prevent the algorithm from creating routes in this way we have inflated the travel times between houses of even/odd numbers on the same street by 90 seconds, the approximate time required for a truck to turn around. The house numbering system in Port Coquitlam places even houses on one side of the street and odd houses on the other. By inflating these travel times between even and odd houses we can prevent the algorithm from moving to a house across the street whenever it is looking for the next best house.

4.2 Model Realization

Google OR Tools provides a comprehensive toolkit for a variety of OR problem types, in particular for the VRP. Contained within the Google OR Tools package is an implementation of the Clarke-Wright's saving algorithm. The route finding method uses Clarke-Wright to generate the first feasible heuristic solution, the model then performs sophisticated solution improvements that create the final solution. Clarke-Wright was chosen as the initial solution method because of the speed of the computation, other methods tended not to converge to a feasible solution and were unable to find the desired

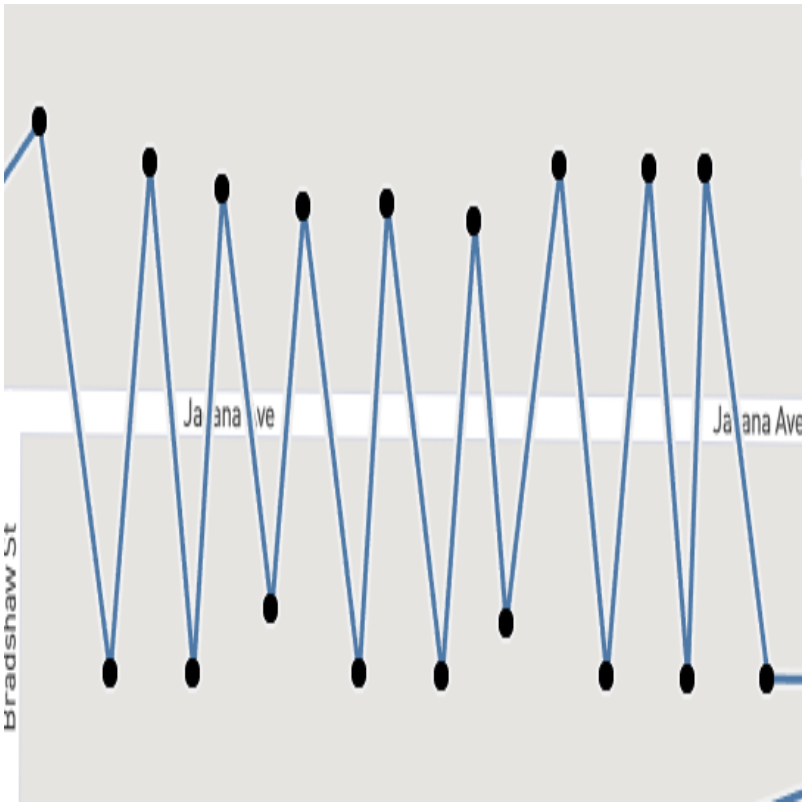

Figure 4: Optimized Routes without time inflation between odd and even houses

solution quickly.

4.3 Computational Tools

The entirety of our source code is available on [link redacted for anonymous submission], the data collected from Port Coquitlam has been removed to protect the City's privacy. The programs are all written in Python. The computing hardware used is an AMD Ryzen 3700X CPU running on a 64 bit version of Windows 10.

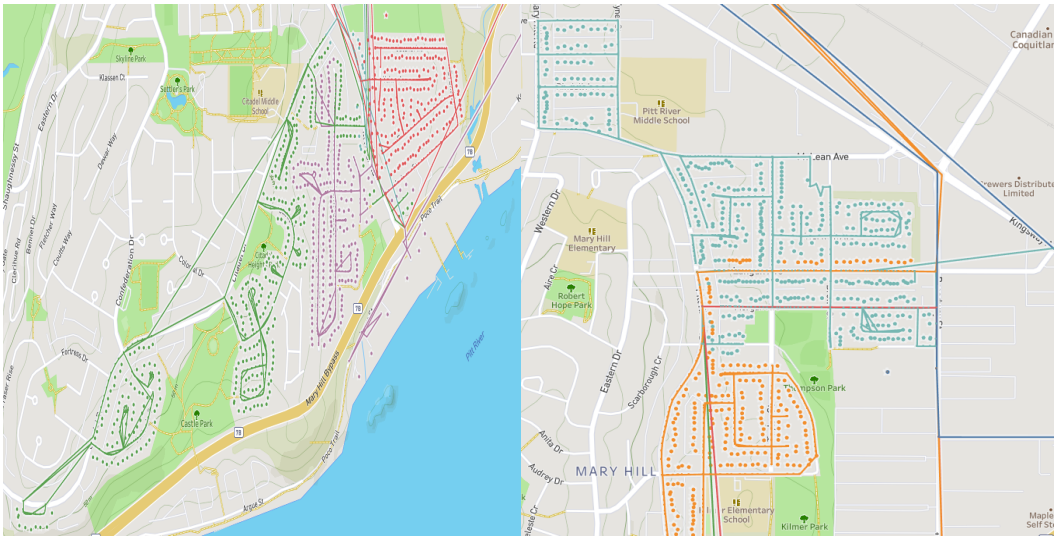

Figure 5: Examples of 2 routes for zones 5 (left) and 2 (right)

5 Results

The computed solutions (**figure** 5) show the routes for m vehicles for a given waste collection zone. The results are relatively successful, the optimized routes traverse all the service points successfully and do so in a somewhat efficient fashion. Compared to the current implementation used by Port Coquitlam, the solutions generated by our algorithm are 32% faster for zones 3,4 and 5 and up to 16% slower for zones 1 and 2. Although the solutions successfully prioritize collection from forested areas in Zone 1 and 2, these are also the two zones that the algorithm was unable to optimize faster than the current implementation. The forested area priority collection constraint could be the cause of this. After the route optimization, the vehicle routes are approximately equal in terms of travel time. However, once we account for the 15 seconds of waste pickup time per house the route time increases significantly for routes with more homes. This results in routes that are somewhat imbalanced. The computed routes can be seen in **figure** 5. A view of the full optimized routes for zone 1 are shown in **figure** 6

An important factor in the usability of this method is its computational time, a solution for a single zone using m trucks can be generated in three

minutes on consumer grade hardware, something similar to that which a waste collection company can acquire with a reasonable budget. To generate solutions for all zones using every variation of truck numbers the current computational time is 60 minutes using a single process, with a parallelization of 8 cores these numbers can be reduced to merely 24 minutes. These computational times are quite appealing, and increase the usability of this solution method.

Vehicles	Zone 1		Zone 2		Zone 3		Zone 4		Zone 5	
	C	A	C	A	C	A	C	A	C	A
4	N/A	20	18	21	28	19	N/A	19	16	16
5	20	21	20	23	20	20	25	21	18	18
6	24	23	24	24	24	22	24	22	24	19
7	24	25	24	26	24	24	24	24	24	21

Table 2: Average Current (C) route times and Algorithm (A) generated route times in hours

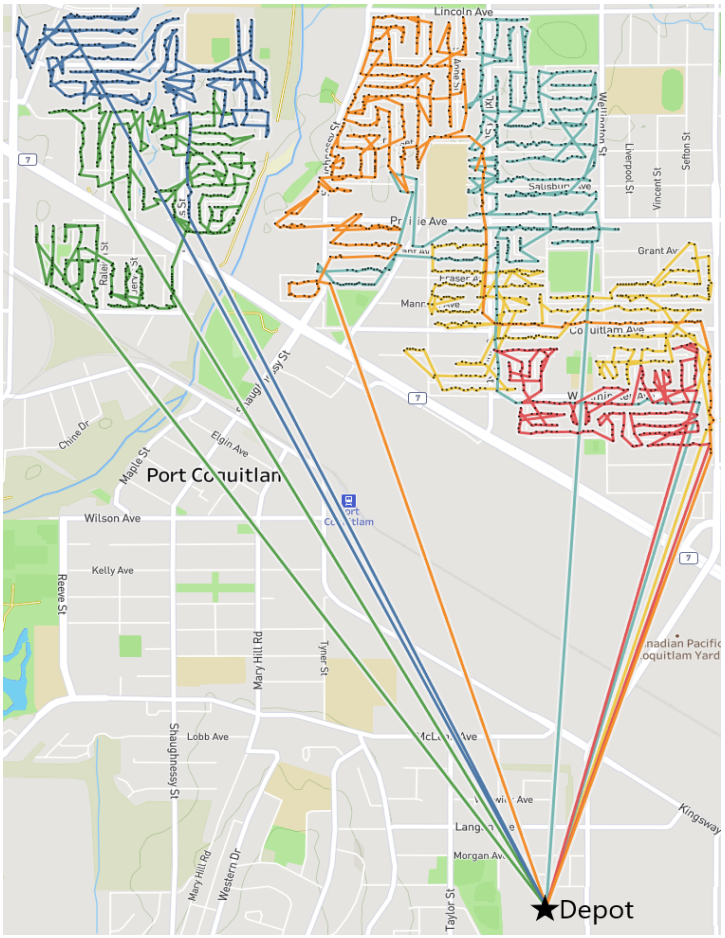

Figure 6: A full view of the optimized routes for zone 1

6 Recommendations

6.1 Cul-de-sac and round-abouts

Inflating travel times between odd and even homes to constrain waste collection to only the right side of the road resulted in the model failing to efficiently traverse cul-de-sacs, round-abouts (traffic circles), and dead-ends. Since houses within a single cul-de-sac will contain both odd and even homes this resulted in the algorithm creating routes that traversed half the cul-de-sac and would leave the other half untouched, only to return later to finish. This causes inefficiencies in the route optimization process. To overcome this

it is recommended to reformulate the constraints governing right side waste collection.

6.2 Route Balancing

Due to the waste bin pickup time being added after route optimization, the computed routes were somewhat unbalanced in certain cases with a higher number of vehicles. To correct for this it is recommended to incorporate the bin pickup time directly into the optimization portion of the algorithm.

6.3 Real Time, Traffic-aware algorithm

There are several aspects of this solution that would be interesting to look into further. For example, waste collection is a dynamic problem, it is impacted by the real time traffic statistics on a day to day basis. To implement a real time, traffic-aware algorithm one would need to reduce the computational time, generate a consistent method to estimate model parameters, and have the required hardware available in the waste collection trucks. The advantages of this method are clear, at any given time based on the current traffic and weather conditions the optimal waste collection route can vary, by implementing a real time system the routes will always remain optimal.

7 Concluding Remarks

We have presented a thorough and detailed summary of our methods for data collection, data preparation, and model design, as well as outlined the model constraints and how to generate them in a reasonable manner. Our optimization algorithm is up to 32% faster for zones 3,4 and 5 and up to 16% slower for zones 1 and 2. Further, our model presents several methods: for reducing the total number of queries needed to generate a distance matrix, to construct a "right side only" constraint and for regionally prioritized waste collection.

8 Acknowledgements

We would like to thank our point of contact at Port Coquitlam Waste Management, Tom, for providing us access to Port Coquitlam's database and

answering our many questions regarding the details of waste collection in the municipality. We also would like to thank our supervisor, Tamon, for their efforts and guidance throughout the research, development and writing process.

References

[And+21] Alexandra Anderluh et al. "Multi-objective optimization of a two-echelon vehicle routing problem with vehicle synchronization and 'grey zone' customers arising in urban logistics". In: *European Journal of Operational Research* 289.3 (2021), pp. 940–958. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2019.07.049. URL: https://www.sciencedirect.com/science/article/pii/S0377221719306289.

[BE06] M.F. Badran and S.M. El-Haggar. "Optimization of municipal solid waste management in Port Said – Egypt". In: *Waste Management* 26.5 (2006), pp. 534–545. ISSN: 0956-053X. DOI: https://doi.org/10.1016/j.wasman.2005.05.005. URL: https://www.sciencedirect.com/science/article/pii/S0956053X05001534.

[BG11] T.G. Bosona and G. Gebresenbet. "Cluster building and logistics network integration of local food supply chain". In: *Biosystems Engineering* 108.4 (2011), pp. 293–302. ISSN: 1537-5110. DOI: https://doi.org/10.1016/j.biosystemseng.2011.01.001. URL: https://www.sciencedirect.com/science/article/pii/S1537511011000134.

[Cao] Buyang Cao. "Solving Vehicle Routing Problems Using an Enhanced Clarke-Wright Algorithm: A Case Study". eng. In: *Computational Logistics*. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 190–205. ISBN: 9783642335860.

[CFM18] Krzysztof Chris Ciesielski, Alexandre Xavier Falcão, and Paulo A. V Miranda. "Path-Value Functions for Which Dijkstra's Algorithm Returns Optimal Mapping". eng. In: *Journal of mathematical imaging and vision* 60.7 (2018), pp. 1025–1036. ISSN: 0924-9907.

[CW64] G Clarke and J. W Wright. "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points". eng. In: *Operations research* 12.4 (1964), pp. 568–581. ISSN: 0030-364X.

[Del+19] Laura Delgado-Antequera et al. "A bi-objective solution approach to a real-world waste collection problem". In: *Journal of the Operational Research Society* 71.2 (2019), pp. 183–194. DOI: 10.1080/01605682.2018.1545520.

[Dij59] E.W Dijkstra. "A note on two problems in connexion with graphs". eng. In: *Numerische Mathematik* 1.1 (1959), pp. 269–271. ISSN: 0029-599X.

[Erd+19] Oğuzhan Erdinç et al. "Route optimization of an electric garbage truck fleet for sustainable environmental and energy management". In: *Journal of Cleaner Production* 234 (2019), pp. 1275–1286. ISSN: 0959-6526. DOI: https://doi.org/10.1016/j.jclepro.2019.06.295. URL: https://www.sciencedirect.com/science/article/pii/S0959652619322607.

[Goo21] Google. 2021. URL: https://developers.google.com/maps/documentation/geocoding/overview.

[Hen+15] David Henry et al. "Clustering Methods with Qualitative Data: a Mixed-Methods Approach for Prevention Research with Small Samples". eng. In: *Prevention science* 16.7 (2015), pp. 1007–1016. ISSN: 1389-4986.

[JST08] Nicolas Jozefowiez, Frédéric Semet, and El-Ghazali Talbi. "Multi-objective vehicle routing problems". In: *European Journal of Operational Research* 189.2 (2008), pp. 293–309. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2007.05.055. URL: https://www.sciencedirect.com/science/article/pii/S0377221707005498.

[KJK14] Peter Kent, Rikke K Jensen, and Alice Kongsted. "A comparison of three clustering methods for finding subgroups in MRI, SMS or clinical data: SPSS TwoStep Cluster analysis, Latent Gold and SNOB". eng. In: *BMC medical research methodology* 14.1 (2014), pp. 113–113. ISSN: 1471-2288.

[LGA07] D. Ljungberg, G. Gebresenbet, and S. Aradom. "Logistics Chain of Animal Transport and Abattoir Operations". In: *Biosystems Engineering* 96.2 (2007), pp. 267–277. ISSN: 1537-5110. DOI: https://doi.org/10.1016/j.biosystemseng.2006.11.003. URL: https://www.sciencedirect.com/science/article/pii/S1537511006003746.

[Nuo+06] Teemu Nuortio et al. "Improved route planning and scheduling of waste collection and transport". In: *Expert Systems with Applications* 30.2 (2006), pp. 223–232. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2005.07.009. URL: https://www.sciencedirect.com/science/article/pii/S0957417405001375.

[Ped+11] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[PF19] Laurent Perron and Vincent Furnon. *OR-Tools*. Version 7.2. Google, 2019. URL: https://developers.google.com/optimization/.

[PK86] Yang Byung Park and C.Patrick Koelling. "A solution of vehicle routing problems in a multiple objective environment". In: *Engineering Costs and Production Economics* 10.1 (1986), pp. 121–132. ISSN: 0167-188X. DOI: https://doi.org/10.1016/0167-188X(86)90033-9. URL: https://www.sciencedirect.com/science/article/pii/0167188X86900339.

[Sah+05] Surya Sahoo et al. "Routing optimization for waste management". In: *Interfaces* 35.1 (2005), pp. 24–36.

[Tav+09] G. Tavares et al. "Optimisation of MSW collection routes for minimum fuel consumption using 3D GIS modelling". In: *Waste Management* 29.3 (2009), pp. 1176–1185. ISSN: 0956-053X. DOI: https://doi.org/10.1016/j.wasman.2008.07.013. URL: https://www.sciencedirect.com/science/article/pii/S0956053X08002717.

[Vir+20] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[War63] Joe H Ward. “Hierarchical Grouping to Optimize an Objective Function”. eng. In: *Journal of the American Statistical Association* 58.301 (1963), pp. 236–244. ISSN: 0162-1459.

[Zha+16] Jin-dong Zhang et al. “Vehicle routing in urban areas based on the Oil Consumption Weight -Dijkstra algorithm”. eng. In: *IET intelligent transport systems* 10.7 (2016), pp. 495–502. ISSN: 1751-956X.