ABSTRACT

Title of Thesis:	ANALYSIS AND OPTIMIZATION OF SERVICING LOGISTICS FOR SELF-DRIVING E-SCOOTERS
	Hao Da Dong, Master of Science, 2021
Thesis Directed By:	Dr. Derek A. Paley, Department of Aerospace Engineering and Institute for Systems Research

In recent years, the shared scooter market has seen tremendous growth along with other micromobility industries as the future means of urban transport. One particularly interesting innovation that companies have begun experimenting with in this field is that of self-driving e-scooters.

This thesis presents a study on the benefits of an autonomous or teleoperated scooter fleet with self-assembly capabilities: the ability to cluster nearby scooters and reduce the number of locations for servicing. To this end, the application is tackled as two separate optimization problems in clustering and routing. The full algorithm pipeline is described and several metrics evaluated against independent variables and algorithm parameters using real-world GBFS scooter data collected over several months.

This thesis shows that self-assembly reduces total service times by as much as 50%, and can serve as a stepping stone for early adoption of the technology while more complex capabilities are being developed.

ANALYSIS AND OPTIMIZATION OF SERVICING LOGISTICS FOR SELF-DRIVING E-SCOOTERS

by

Hao Da Dong

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Master of Science
2021

Foreword

The work presented in this thesis is completed as part of the ReZoom initiative at the University of Maryland – College Park led by Dr. Derek A. Paley. The initiative involves the development of a wide range of autonomous capabilities for e-scooters and research into how these capabilities may be used to benefit stakeholders in the shared scooter industry. As such, the project scope (see Chapter 2), assumptions on the scooter fleet (see Chapter 3), and specific ranges of certain parameters (see Chapter 4) were chosen to complement the other capabilities being developed at the time of writing.

Dedication

This thesis is dedicated to my parents. While holding Masters engineering degrees themselves, they have never pressured me onto any one path and have always encouraged me to find a career that is suitable for me. In completing this thesis, I hope to follow in their footsteps and continue my life-long journey of learning and continuous improvement in my field of study.

Acknowledgements

This thesis was generously supervised by Dr. Derek A. Paley. I cannot express the gratitude I have for his support, flexibility and insurmountable patience during the entirety of this project. From helping me find my direction on the ReZoom team, to giving me technical advice for my work, to providing support in all administrative matters right up to the very end, he has enabled me to complete a research project virtually in the midst of a global pandemic. I give my absolute thanks to Dr. Paley for an amazing year and know that any student who may to taking over my work in the future will be very fortunate to have his incredible support and guidance.

I would also like to thank Dr. Ashish Santosh Kabra, who was kind enough to share with us hundreds of gigabytes of GBFS scooter data collected by his team over the course of several months to assist us in our work.

Table of Contents

Chapter 1: Introduction

1.1 *Motivation*

Micromobility is a mode of transportation that involves small, lightweight vehicles that typically travel below 25 km/h. This includes but is not limited to bikes, e-bikes, scooters, e-scooters and skateboards. In recent years, both private and shared micromobility have tremendously risen in popularity as an option for urban transportation. There has been much discussion recently about the potential for micromobility to play a role in solving several of the transport related issues large cities face worldwide, as well in reducing their carbon footprints by facilitating the move away from private fossil fuel vehicles.

Of the various micromobility vehicle categories, perhaps none has seen more visible growth than the shared e-scooter segment. However, with rapid growth and expansion comes a rapid need for an efficient scooter fleet management solution, which is one of the most important key factors to profitability. This includes the logistics of collecting, charging, servicing, and rebalancing scooters to high demand areas. Currently, scooter operators must allocate tasks for their own employees and/or third-party contractors that move and charge scooters for payment. With the miniaturization of computing resources and sensor suites in recent years due to smartphone industry, several scooter companies are exploring the possibility of adopting a fleet with self-driving capabilities to complement humans in this task. Although self-driving scooters have several potential benefits to operators in this area, a central challenge that is not yet fully solved is the optimization of human and scooter directives to decrease servicing costs and increase ride revenue.

1.2 *Relation to State of the Art*

The idea of adding autonomy to personal mobility scooters has been demonstrated as early as the 2016 MIT Open House, with MIT and the National University of Singapore presenting a joint project which saw the replication of an architecture and sensor suite normally used on cars on a scooter [1]. Surveys at the time showed that the public is generally receptive to the concept of an autonomous personal mobility device. More recently, several companies are making plans to bring this technology to the commercial space. One of the more visible figures in this field is Tortoise, led by Uber's former director of business development, which aims to provide a standard autonomy operating system for micromobility vehicles. Tortoise has already partnered with shared scooter operators Go X and Spin to develop remotely operated scooters as the first phase of this technology rollout [2] [3]. Another prominent name in the space is Segway-Ninebot, which has also developed and showcased its own brand of self-driving e-scooters that can find their way back to charging stations [4]. The company intends to sell its scooters to Uber and Lyft, both of which it claims are advancing towards semi-autonomous micromobility vehicles. Furthermore, several smaller startup companies are also beginning to fill the space with their own versions of self-driving scooters, autonomy module additions, and software.

One key to profitability in the micromobility industry is successful fleet management. Understanding the logistics of collecting/servicing scooters and then re-distributing or re-balancing them to high demand areas is central to reducing costs and increasing ridership (and thus revenue) for shared scooter operators. To this end, there have been several studies conducted in the following areas:

1. The spatial-temporal distribution variations of shared scooters [5] [6]
2. The forecasting of supply/demand and ride trip prediction [7]
3. The relation between ridership and region demographics [8]

These studies all focus on conventional shared scooter operation models, where scooters do not have self-driving capabilities and the data analytics are used by shared operators to determine the best areas to re-balance scooters to each morning, either via their own employees or third-party contractors.

Self-driving scooters increase ridership by automatically re-balancing to high demand areas. One recent study estimates the ideal fleet size under varying assumptions of fleet operations and that up to 10 times higher utilization of scooters can be achieved with self-driving capabilities [9].

1.3 Contributions of Thesis

There has been much work in the data analytics of conventional shared scooter systems, and studies into the benefits of self-driving scooter systems in increasing ridership and revenue are underway, with more expected in the near future. However, there have been no studies yet to our knowledge that look at the benefit of self-driving scooters in terms of the other component of fleet management: the reduction in operating costs from collection and/or servicing. The focus of this thesis aims to bridge that gap with a study into one specific application of self-driving scooters that would aid operator companies in reducing operating costs.

We investigate the application of scooter self-assembly, or clustering. The idea is that the self-assembly of scooters close to each other allows for batch collection and/or servicing, thereby reducing the number of stops and time it takes for service personnel to complete a service run. Our work is based on the model where an operator company employs its own service personnel, which all depart from one central location or depot in the area that the scooter fleet is deployed. The two service types investigated are collection (for charging) and battery swapping. Collection was chosen as it is a task that currently every scooter operator company needs to do, either to recharge scooters or as the first step in re-balancing. Battery swapping was chosen due to the realization that several companies are transitioning to this type of re-charging method, as it is quicker than the traditional method of taking scooters off the road and bringing them back to a depot or charging station.

We evaluate the benefits of self-assembly for different operational approaches by performing analyses on different combinations of scooter servicing parameters. We utilize real-world GBFS scooter data collected over several months for this evaluation, and in doing so we formulate preliminary motivations for the development of self-driving e-scooters and the foundations for impactful requirements.

1.4 Technical Approach

We tackle the scooter self-assembly application using a two-stage approach, each with a separate optimization problem and solution.

The first stage is the clustering stage, in which scooters in need of servicing are identified and located geographically on a map. The geographic coordinates are fed into a custom variation of the agglomerative clustering algorithm [10]. The basic algorithm is applied with the scooters' travel distance matrix and a specified maximum travel distance threshold in order to determine preliminary clusters. Afterwards, the clusters and cluster centers are refined with a check that each scooter respects the maximum travel distance threshold relative to its cluster center. Finally, the refined cluster centers are slightly adjusted to snap to the nearest road, ensuring that the final destinations for the scooters are valid and feasible.

The second stage is the service routing stage, in which near-optimal routes are determined for a given number of service personnel to each scooter cluster center. This is a classic vehicle routing problem (VRP) and conventional heuristic-based solvers are used to provide solutions. The distance matrix used in the solver actually consists of driving times between each cluster center rather than physical distances. Furthermore, the time of each leg is adjusted to account for the service time at each cluster. Two different service time calculations are used, depending on whether the service type is collection or battery swapping.

We quantify the amount of cost reduction in terms of man-hours, number of stops and the maximum time taken to complete a service run. We use real-world data collected from scooter companies over several months to investigate these metrics against various independent variables and parameters, including scooter fleet size, time of year, number of service personnel and maximum scooter travel distance.

1.5 Outline of Thesis

The remainder of this thesis is organized as follows. Chapter 2 provides background information on the optimization algorithms used as the foundations for the two stages of the solution to the self-assembly application. Chapter 3 describes in detail the methods and algorithms pipeline for the solution to the self-assembly application. Chapter 4 describes in detail the data and various metrics used to evaluate the solution and the independent variables and metrics used to compare them. Results are provided to quantify the trends in metrics with respect to independent variables and parameters. Chapter 5 summarizes our findings and provides suggestions for future work.

Chapter 2: Background

This chapter provides background information on the optimization algorithms used as the foundation for the two stages of our solution to self-assembly.

2.1 Data Clustering

There are several clustering algorithms available from the fields of statistical data analysis, pattern recognition, and machine learning. The goal of each one is to group data points or objects that are in some sense similar to each other. However, the notion of a cluster cannot be precisely defined, as the metrics for doing so varies between applications. Clustering algorithms can generally be grouped into the following broad categories.

Centroid-Based: Algorithms in this category provide a vector representation of a cluster's center, calculated as the mathematical centroid of the data points within it [11]. The most popular algorithm of this category is the classic k-means clustering algorithm [12], where a given number of known centroids are continuously shifted in the data space to minimize a distance cost function, usually the squared error between the cluster centers and their points.

Connectivity-Based: This type of clustering works under the premise that objects are more related to objects nearby than those that are farther away [13]. This category is most often associated with hierarchical or agglomerative clustering [10], in which the closest objects are grouped together first, before larger clusters are formed and those are in turn grouped even further.

Distribution-Based: This type of clustering envisions the data space as being composed of a series of probability distributions, with each point holding a full or part membership to a distribution or cluster [14]. The Gaussian Mixture Model (GMM) method [15] is a prominent algorithm of this category. One distinction of this category from the previous two is that the clusters here are permitted to intersect and overlap with one another.

Density-Based: In this category, clusters are identified as areas of high density in the data space [16]. Object is sparse areas are sometimes considered as outliers and omitted from the calculation of the cluster centers. The Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm [17] is a prominent example in this category.

The *agglomerative clustering algorithm* is chosen to be the foundation of the first stage of our solution due to its natural conformity to our data of interest. When looking for common locations for scooters to aggregate to on a map, the primary metric is travel distance. Scooters close to each other should be clustered together. Since the scooters are not known to obey any particular spatial probability or density distributions, distribution- or density-based algorithms would be less suitable. The centroid-based k-means algorithm was considered and actually implemented as a prototype for our solution, but agglomerative clustering suits our application better for the following reasons:

1. The number of natural clusters formed from the scooters is not known a priori, and thus the k-means algorithm must be applied iteratively while increasing the number of clusters until the travel distance threshold between scooters and their cluster centers are satisfied. Because the cluster centers are randomized at the start of each iteration, this introduces several opportunities for errors.
2. The k-means algorithm works best for a small number of clusters relative to the number of data points [18]. It was found experimentally that due to the general sparsity of our scooter data, the number of clusters is very large, especially for smaller travel thresholds.
3. By definition, the k-means cluster centers are calculated as the centroids of scooters rather than geographic centers among them. It uses this definition in optimizing its cost function, which is not always realistically correct in our application since we are interested in geographic centers for scooters to travel to. Agglomerative clustering does not have this limitation.

2.2 Combinatorial Optimization for Operations Research

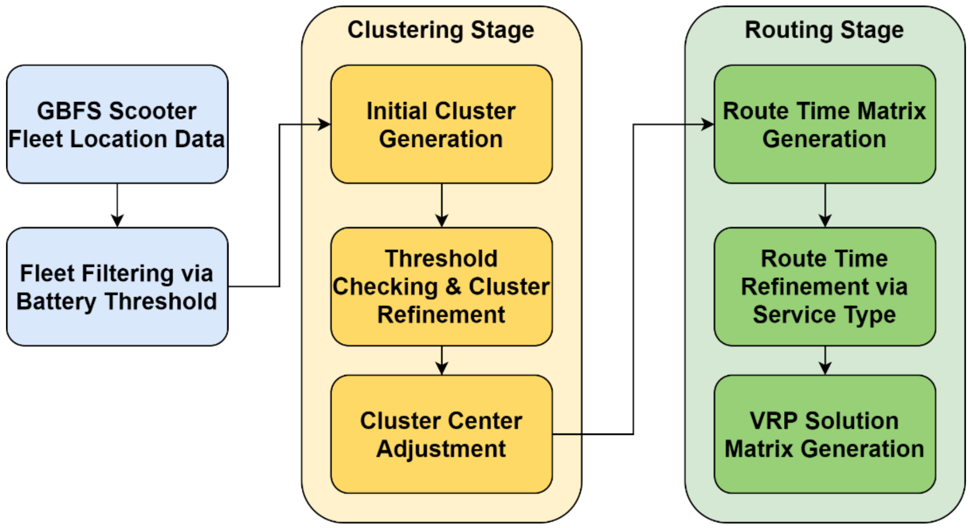
Determining the optimal way to route an agent between several locations is called the Travelling Salesman Problem (TSP) [19]. The task is to find the shortest route for a salesman to visit customers at various locations and return to the starting point. For routing several agents, the more general Vehicle Routing Problem (VRP) [20] can be employed. The task of the VRP is to have one vehicle in a group visit a subset of locations while minimizing the longest route taken by any one vehicle and ensuring all locations are visited exactly once. This problem has a number of variations, including capacity constraints and time windows. Although exact solution techniques such as branch and bound [21] do exist, the computation time for these approaches become unreasonable for larger problems. A more common approach to the VRP is to conduct a limited search of the problem solution space using a metaheuristic. One such approach is simulated annealing [22]. More recently, machine learning techniques such as genetic algorithms have also been leveraged in the formulation of new solvers [23].

Chapter 3: Optimization of Servicing Logistics

This chapter describes in detail the methods and algorithms pipeline for our solution to the self-assembly and collection problem.

3.1 *Overview of Approach*

We tackle the scooter self-assembly application using a two-stage approach, each with a separate optimization problem and solution. The complete algorithm pipeline is illustrated in Figure 1.



GBFS Scooter Fleet Location Data
Fleet Filtering via Battery Threshold
Clustering Stage
Initial Cluster Generation
Threshold Checking & Cluster Refinement
Cluster Center Adjustment
Routing Stage
Route Time Matrix Generation
Route Time Refinement via Service Type
VRP Solution Matrix Generation

Figure 1 - E-scooter Self-Assembly and Collection

Scooter location data is collected in the General Bikeshare Feed Specification (GBFS) [24] format, an open data standard for shared micromobility. After extracting the locations into a suitable data structure, the entire fleet is filtered to include only low battery scooters that fall below a certain threshold. The data used in this study is described in Section 4.1.

The first stage of the application algorithm is the clustering stage, in which scooters in need of servicing are identified and located geographically on a map. The geographic coordinates are fed into a custom variation of the agglomerative clustering algorithm, which returns the cluster center locations as well as the scooters that belong to those clusters.

The second stage of the application algorithm is the service routing stage, in which near-optimal routes are determined for a given number of service personnel to each scooter cluster center. The distance matrix used in the solver for this problem consists of travel times between each stop location (clusters) as well as a pre-assigned depot for the service personnel. Each leg is adjusted to account for the two different types of service time at each cluster.

To analyze the metrics of the self-assembly application, the algorithm pipeline is run twice for a given scooter fleet. During the first run, only the routing stage is executed, treating every scooter as its own cluster center. This generates benchmark results for the theoretical optimal service run using the current industry practice. During the second run, the clustering stage is executed before the routing stage, and thus reduces the number of stops that is fed into the VRP solver. The results from the second run show the theoretical optimal service run generated by the proposed self-assembly application. The two metrics are compared and further analyzed. Figure 2 shows the full logic pipeline for the dual benchmarking runs, along with the intermediary and final products of each process and component of the algorithm.

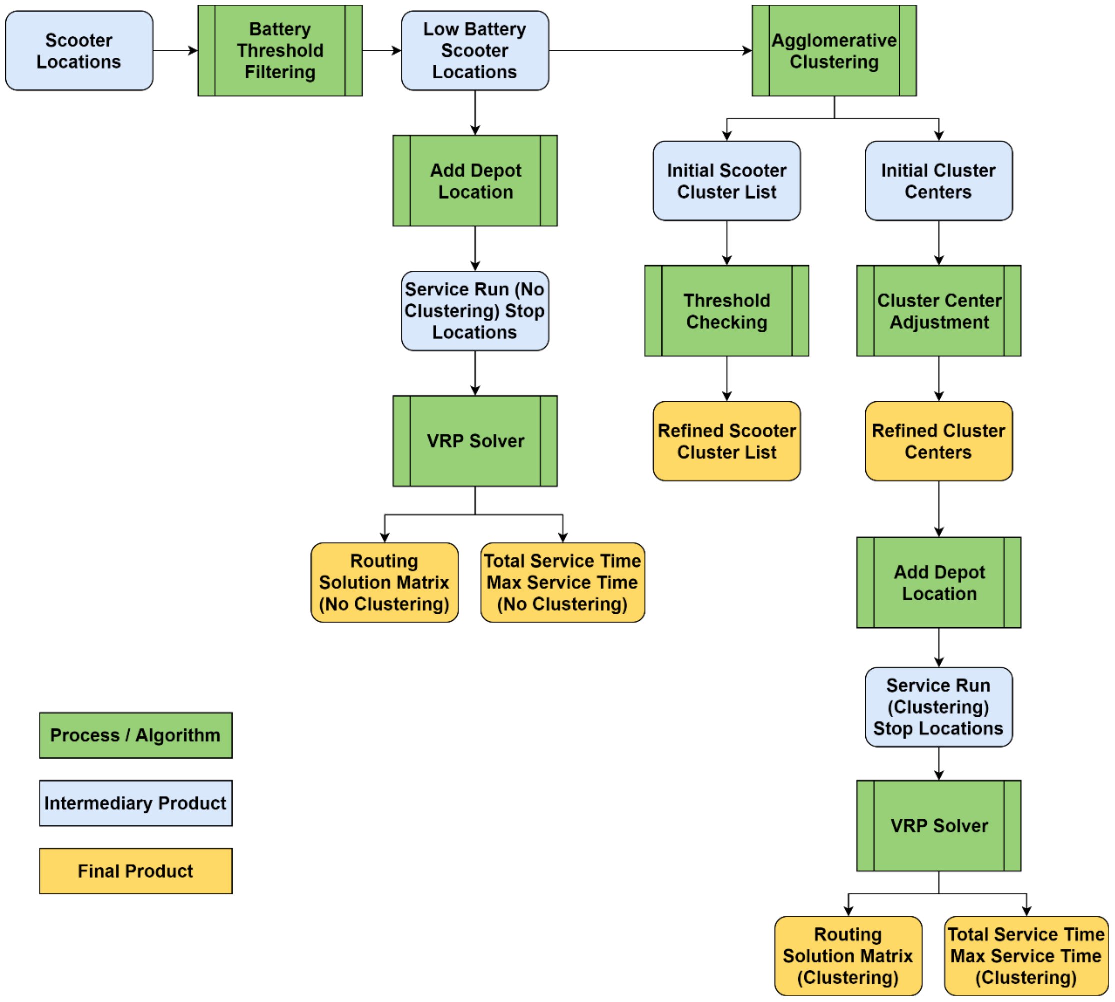

Scooter Locations
Battery Threshold Filtering
Low Battery Scooter Locations
Agglomerative Clustering
Add Depot Location
Initial Scooter Cluster List
Initial Cluster Centers
Service Run (No Clustering) Stop Locations
Threshold Checking
Cluster Center Adjustment
VRP Solver
Refined Scooter Cluster List
Refined Cluster Centers
Routing Solution Matrix (No Clustering)
Total Service Time Max Service Time (No Clustering)
Add Depot Location
Process / Algorithm
Intermediary Product
Final Product
Service Run (Clustering) Stop Locations
VRP Solver
Routing Solution Matrix (Clustering)
Total Service Time Max Service Time (Clustering)

Figure 2 – Algorithm Benchmarking Pipeline with Final Products

3.2 *Clustering by Micro-scale Re-positioning*

This section describes in detail the first stage of the self-assembly algorithm. We work with the assumption that a scooter operator company already has access to an e-scooter fleet with the following autonomous or teleoperated capabilities:

1. Scooters that are able to self-orient into an upright position suitable for mobility;
2. Scooters that are able to navigate a few hundred meters given a set of destination GPS coordinates, while driving safely on public roads, avoiding obstacles and obeying traffic laws;
3. Scooters that are able to identify valid parking spaces at a destination and self-park in line with other scooters at the location, if any.

These capabilities are being developed as part of the ReZoom initiative at the time of writing this thesis.

With the previous autonomous capabilities available to a scooter fleet under consideration, the problem is reduced to spatial data clustering in two dimensions. As stated in Section 2.1, agglomerative clustering was selected to be the foundation of our approach, with slight modifications to improve the real-world feasibility of the solutions generated. The specific steps in the clustering stage pipeline are detailed below.

3.2.1 Initial Cluster Generation

The agglomerative clustering package from the `scikit-learn` machine learning framework [25] is used for generating the initial clusters. The following key parameters are set specific to our application during usage:

1. **`linkage`** This parameter is set to complete, which tells the solver to use the maximum of the distances between all scooters of two clusters when deciding whether or not to merge them. This choice better ensures that scooters will satisfy the travel distance threshold to their cluster centers.
2. **`distance_threshold`** This parameter tells the solver to continue merging clusters until the given threshold can no longer be satisfied. This choice is in contrast to setting a specific number of clusters for the solver to generate.
3. **`affinity`** This parameter tells the solver what metric to use in computing the linkage distances. It is set to precomputed for our application, which states that a custom linkage matrix will be used to determine distances. We utilize the Bing Maps Distance Matrix API [26] from Microsoft in order to generate the required matrix that comprise of the route distances between scooter locations.

After the clusters are generated from the solver, the centers are calculated as the spatial midpoint of the scooter locations in each cluster.

3.2.2 Cluster Refinement

Each cluster is put through a refinement process to ensure every scooter satisfies the given travel distance threshold. The process involves using the Bing Maps Routes API [27] to calculate the distance between each scooter and its cluster center and comparing it with the distance threshold. The Distance Matrix API cannot be used here since it cannot handle the edge case of a 1x1 matrix request. If a scooter violates the threshold, it is separated from the rest of the cluster and forms its own cluster. The center of the original cluster is then recalculated. This process repeats until all clusters have been processed.

This refinement procedure does not produce the absolute optimal result. However, it is expected and also observed through preliminary testing that the vast majority of clusters formed during the initial step are unchanged after refinement, and does not significantly affect the overall results presented in Chapter 4.

3.2.3 Cluster Center Refinement

After the clusters are refined, their recalculated centers are also refined. Because the centers are calculated purely as the spatial midpoint, it is possible that some are infeasible destinations for scooters to travel to on a map, such as in the middle of a building. To correct for this, we utilize the Google Maps Platform Roads API [28], which provides the functionality to snap any GPS coordinate to its nearest road. All cluster centers are adjusted to their nearest roads, and this is the final step in the clustering stage.

Figure 3 illustrates the results of applying the modified agglomerative clustering algorithm on the low-battery scooters of a sample fleet in Washington DC on February 2, 2020. The fleet consists of scooters below 40% battery, with a travel distance threshold of 300m. Refer to Section 4.1 for a detailed explanation of these parameters. Scooters are represented as dots and cluster centers are represented as X's. The colors identify unique clusters. The red circles outline two instances where the cluster refinement step described in section 3.2.2 is executed. Notice how spatially, these areas contain scooters which appear as if they should from a cluster, but have been separated in the refinement step. Also notice that for some single scooter clusters, the center locations are slight shifted from the scooter location as a result of the cluster center refinement step.

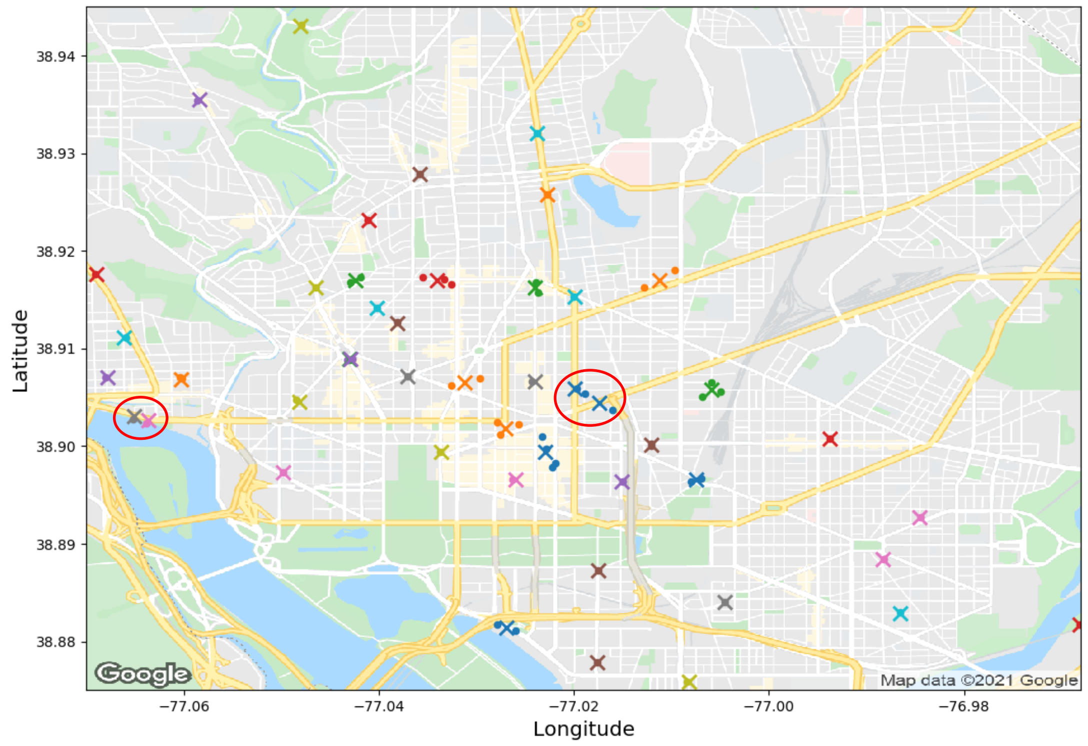


Figure 3 - Sample Cluster Diagram of an E-Scooter Fleet
Battery Threshold: 40%
Service Type: Collection
Travel Distance: 300m

3.3 *Optimization of Service Routes for Multiple Personnel*

This section describes in detail the second stage of the self-assembly algorithm. For the purpose of this study, we make the following assumptions about a sample scooter operator company:

1. The company services its fleet via its own employees rather than third party contractors;
2. All service personnel depart and return to one depot location within a given area of operation;
3. All personnel drive motor vehicles to scooter locations to perform servicing;
4. There are only two types of servicing: scooter collection and battery swapping;
5. In the case of the service being collection, all vehicles shall have sufficient capacity to store all collected scooters.

Under these assumptions, the formulation is a simple Vehicle Routing Problem (VRP) with no time windows or capacity constraints. The Google OR-Tools framework [29] and its VRP solver package is used as the foundation for this stage of the algorithm. The specific steps in the routing stage pipeline are detailed below.

3.3.1 Route Time Matrix Generation

Similar to in the clustering stage, the Bing Maps Distance Matrix API is used to generate a distance matrix for all stop locations (cluster centers) calculated from the previous stage. The values of the matrix are travel times rather than distances since we are interested in optimizing service time rather than checking against a certain distance threshold. Furthermore, the depot location is prepended to the list of stops to account for the first and last legs of each trip.

3.3.2 Route Time Refinement via Service Type

Before the route times are passed to the VRP solver, they are adjusted to account for the service time at each stop location, with the exception of the depot. The models used for the two service types, collection and battery swapping, are shown below.

Collection:

$$ST_{Col,i} = 5 \qquad i = 1,2 \dots N_C$$

Battery Swapping:

$$ST_{Bat,i} = 4 + 1 * N_{S,i} \qquad i = 1,2 \dots N_C$$

The variables are defined as follows:
ST_{Col} is the collection service time per stop (minutes)
ST_{Bat} is the battery swapping service time per stop (minutes)
N_S is the number of scooters at a particular stop location
N_C is the number of clusters (stop locations) for the service run

During collection, we assume that it roughly takes a flat 5 minutes for a service personnel to park the vehicle, exit, load all scooters into the vehicle and then depart. For battery swapping, we assume that it roughly takes a flat 4 minutes to park the vehicle, exit and depart. Furthermore, it roughly takes an addition 1 minute per scooter for the personnel to open the battery compartment, swap cells and then close the compartment. These parameters are adjustable. The number of scooters at each cluster is obtained from the list of clusters generated in the algorithm's first stage. In the absence of clustering, the battery swapping model reduces to the collection model.

3.3.3 VRP Solution Matrix Generation

The refined route times along with the number of service personnel are passed to the Google OR-Tools VRP solver. The solver uses a heuristic-based approach in generating near-optimal routes. Its first solution generation method and local search strategy (metaheuristic) are set to automatic, which allows it to select the best parameters based on its internal analysis of the dataset. The solver returns a list of ordered stop locations for each vehicle, which corresponds to the near-optimal routes found for each. A solution matrix of these routes is saved, along with the total and maximum travel times for the service run.

The following figures show the routes generated by the VRP solver for service personnel, both for the non-clustering scenario (Figure 4) and the clustering scenario (Figure 5). The sample scooter fleet used is the same as that in Figure 3. The number of service personnel is set to 3 and the service type is set to collection. See Section 4.1 for a detailed explanation of these parameters. In each plot, the depot location is marked with a red star and each color represents a route taken by the personnel. The stop locations are shown as dots on the routes and labeled with numbering system that follows the scheme:

$$(Personnel\ Number).(Stop\ Number\ in\ Route)$$

Notice how the number of stops in Figure 5 is reduced from that in Figure 4.

The paths between each stop location are generated with the Google Maps Platform Directions API [30], which has the functionality to return the wapoints of a route's polyline on a map between two GPS coordinates.

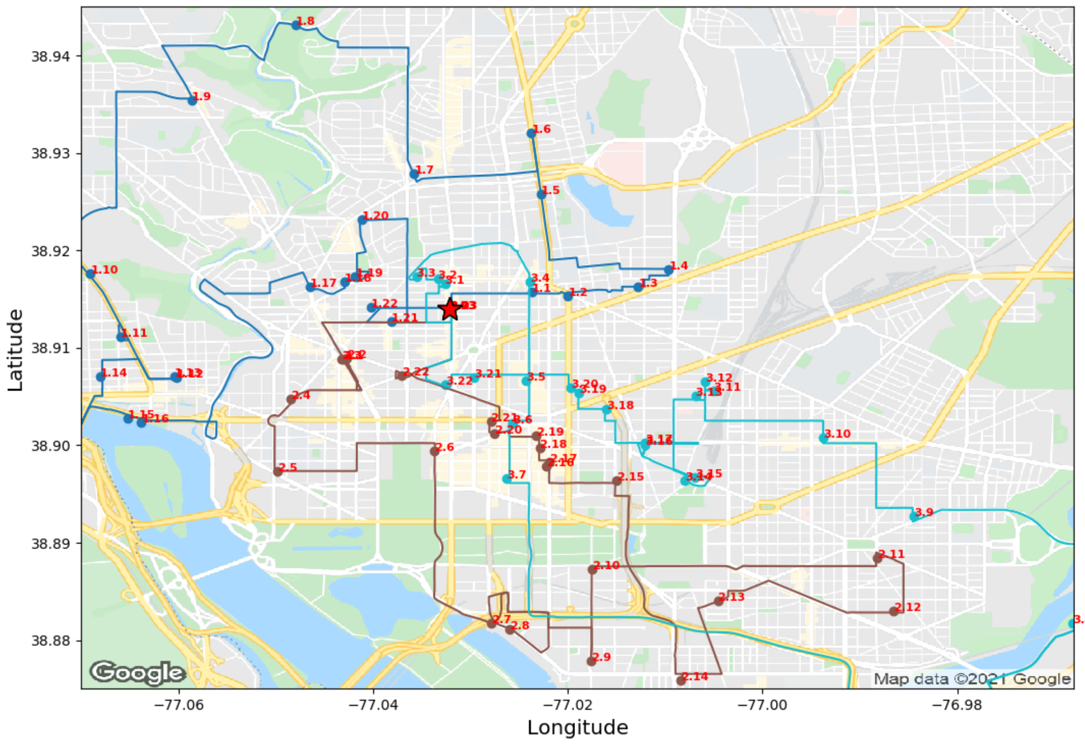


Figure 4 - Sample Routing Diagram of an E-Scooter Fleet without Clustering
Service Type: Collection
Number of Personnel: 3

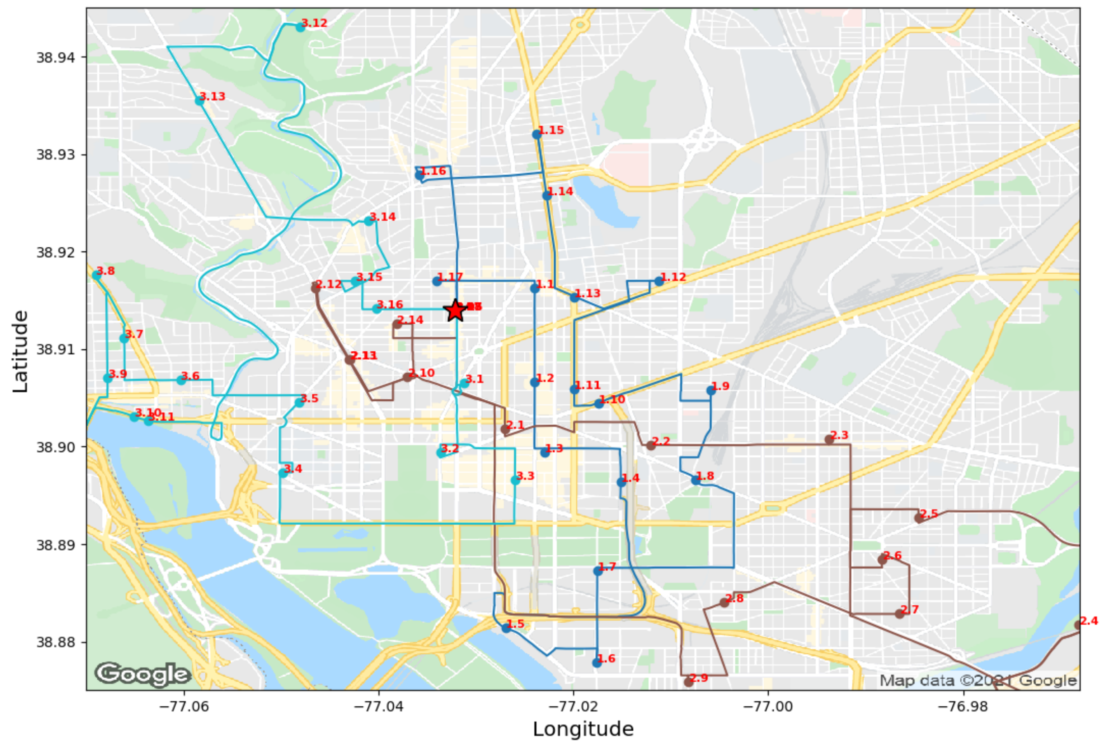


Figure 5 - Sample Routing Diagram of an E-Scooter Fleet with Clustering
Battery Threshold: 40%
Travel Distance: 300m
Number of Personnel: 3
Service Type: Collection

Chapter 4: Numerical Analysis of Servicing Logistics

This chapter describes the data and various metrics used to evaluate how the solution depends on certain independent variables.

4.1 Data and Performance Metrics

4.1.1 Data Description

As stated in section 3.1, the data used in this study is originally collected in the GBFS format. The GBFS data available to us consists of scooters from various operator companies in the Washington DC area from late 2019 to late Summer of 2020. The data was eventually filtered to include only scooters from the operator Bird, since it is the only company to include battery levels in its GBFS data and the company with the least amount of data gaps within the 8-month range.

After the Bird GBFS data was separated from the rest of the dataset, it was further filtered to include only scooters with the first timestamp after 3am of all the available days. This effectively provides a snapshot of scooter locations in the fleet at around 3am each day, which is the expected time that service personnel will become active. For each day at the 3am timestamp, the scooters' GPS coordinates, battery levels and datetime are extracted and saved as CSV files, which are the inputs to the self-assembly algorithm.

Within the algorithm, the depot location is set to be the office of Capitol Scooter Rental. Since Bird does not operate any depots in Washington DC, an arbitrary local scooter operator's location is chosen as a substitute. Furthermore, the low battery threshold is set to 40%, although this choice is adjustable.

4.1.2 Parameters, Independent Variables and Performance Metrics

We define three types of variables for the analysis of the self-assembly algorithm. Parameters are tunable variables that affect the algorithm's performance. Metrics are values that measure the performance of the algorithm (*y*-values in plots). Independent variables are variables that metrics are plotted against (*x*-values in plots).

The following tables list and describe specific variables of each type that we use in our numerical analysis.

Name	Description	Value Range
Scooter Travel Distance Threshold	Maximum distance a scooter can autonomously travel to a cluster center	100m, 300m, 500m
Service Type	Type of service performed by service personnel at each cluster center	collection, battery swapping
Number of Service Personnel	Number of personnel sent out for a service run	1, 3, 5

Table 1 - Algorithm Parameters

Name	Description
Time	The date that a service run occurs on
Number of Low-Battery Scooters	The number of low-battery scooters for a service run

Table 2 - Independent Variables for Performance Evaluation

Name	Description
Fleet Service Total Time (No Clustering)	The total man-hours used in a service run **without** clustering
Fleet Service Total Time (Clustering)	The total man-hours used in a service run **with** clustering
Fleet Service Time (No Clustering)	The maximum time it takes for a service run to be competed **without** clustering
Fleet Service Time (Clustering)	The maximum time it takes for a service run to be competed **with** clustering
Number of Service Stops (No Clustering)	Number of locations for personnel to visit for a service run **without** clustering
Number of Service Stops (Clustering)	Number of locations for personnel to visit for a service run **with** clustering

Table 3 - Metrics for Performance Evaluation

Under metrics, fleet service total time refers to the sum of times across all routes taken by service personnel. This is equivalent to the total man-hours required to complete a service run, and therefore has a direct correlation with the cost required for a company to perform such a run. By contrast, the fleet service time refers only to the longest route taken by service personnel. This is the metric that the VRP solver minimizes and assists companies in forecasting how many personnel are needed to complete a service run given a time constraint. If there is only one service personnel, the fleet service time is equal to the fleet service total time.

For the analysis shown in subsequent sections, we use parameter configurations that resemble those of a sensitivity analysis. In other words, all parameters are held at a reference value while a single parameter is perturbed across its full range of values. This results in the parameter configuration table below. The reference configuration set is highlighted in green.

Travel Distance Threshold (m)	Number of Personnel	Service Type
100	1	Collection
300	1	Collection
500	1	Collection
300	3	Collection
300	5	Collection
300	1	Battery Swapping

Table 4 - Parameter Configuration Sets (Reference Set Shown in Green)

We take the reference configuration to be {Travel Distance Threshold, Number of Personnel, Service Type} = {300, 1, Collection}. We then cycle each parameter through its range while avoiding repetitions. For each parameter configuration, the metrics are compared for both the clustering and non-clustering cases.

4.2 *Performance vs. Time*

We first evaluate the self-assembly algorithm's performance with respect to time. Figure 6 shows a plot of the fleet service total time comparisons for the parameter configuration {500, 1, Collection}.

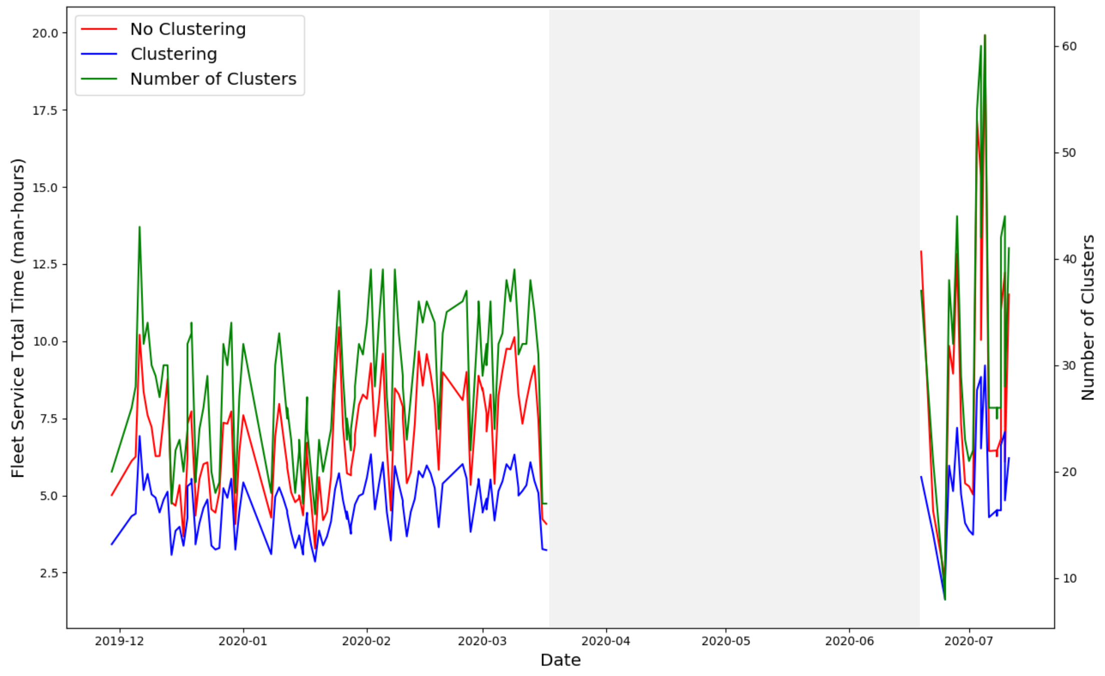


Figure 6 – Fleet Service Total Time vs Time (December 2019 to July 2020)
Battery Threshold: 40%
Travel Distance: 500m
Number of Personnel: 1
Service Type: Collection

The plot displays the fleet service total times for the clustering and non-clustering scenarios on the left *y*-axis, and the number of clusters formed on the right *y*-axis. All three metrics in this case is a measure of scooter usage, since usage is directly correlated with the number of low-battery scooters, which in turn affects the service total times, both with and without clustering.

There is a large data gap between mid-March and mid-June 2020, which is most likely due to Bird temporarily stopping their services due to the COVID-19 pandemic. Apart from this gap, the overall usage trends appear to be quite stable from late 2019 to mid-March 2020. There is a slight increase in usage in February, which may be attributed to the public and especially students returning to a daily commute after Winter Break. Furthermore, there appears to be another slight increase in usage after the service suspension. Although the timeline does not extend very far to provide more insights, it is possible that this increase is due to people transitioning away from public transit and toward shared mobility for daily commuting.

Apart from these usage trends, however, there is not much else that can be inferred from examining the metrics with respect to time. The next section provides a alternative way to examine the data that better highlights the relationship between the metrics and parameters.

4.3 Performance vs. Number of Scooters

The most useful insights from the metrics are visible when they are plotted with respect to the number of low-battery scooters in a fleet. Consider the sample fleet service total time plot of parameter configuration {500, 1, Collection} in Figure 7.

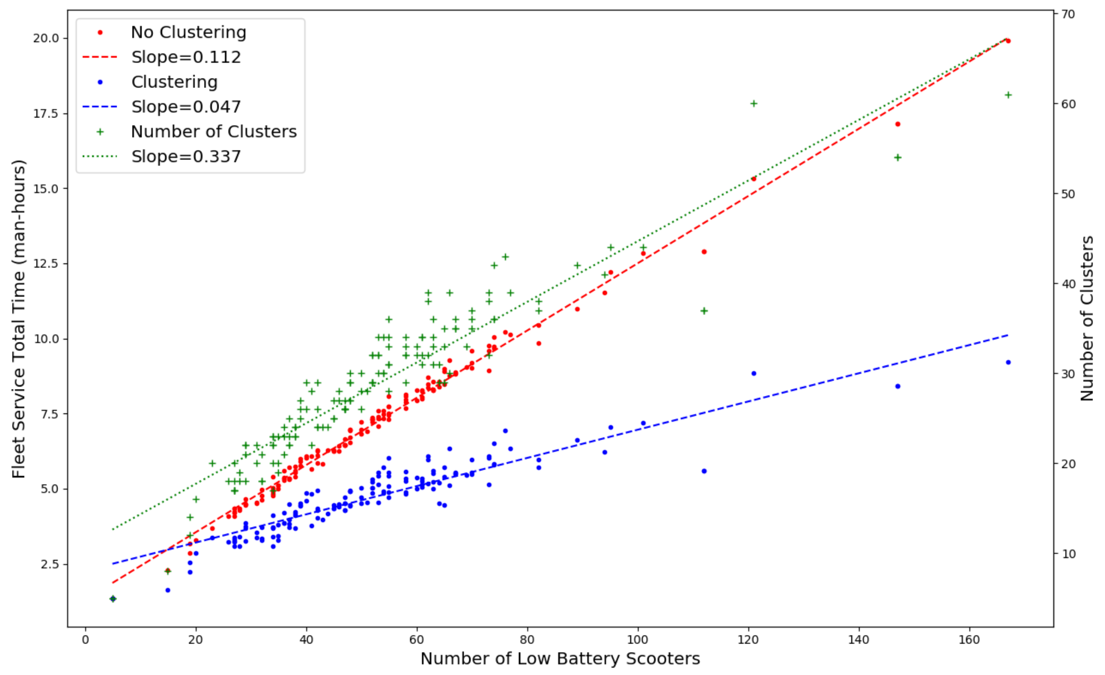


Figure 7 - Fleet Service Total Time vs Number of Scooters
Battery Threshold: 40%
Travel Distance: 500m
Number of Personnel: 1
Service Type: Collection

The plot displays the service total times for the clustering and non-clustering scenarios on the left *y*-axis, and the number of clusters formed on the right *y*-axis. However, unlike in the previous section, the metrics here are arranged as a scatter plot. There is a positive correlation between the number of scooters, the clusters formed, and the total times for both clustering and non-clustering. This correlation is modeled with a linear regression.

While the no-clustering service total time can be modeled linearly even as the number of scooters grow very large, the same cannot be said for the clustering service total time or the number of clusters formed. The reason is that given a distance threshold and a constrained space, there is a limit to the number of clusters that can be formed in that space. As more scooters are added to the space, they would automatically belong to one of the existing clusters instead of forming a new one. Thus, we expect that as the number of scooters grows very large, the number of clusters formed would be more accurately modeled with a plateauing exponential and asymptotically approach a limiting value. The same logic applies to the clustering total service time, as it is directly related to the number of clusters. We can already see this relationship in Figure 7. Although the data points for large numbers of scooters are sparse, it is clear that the overall trend is not linear. That said, since the cluster number limit is expected to very large, and because the majority of real-world data is shown to fall within the plot segment with smaller scooter numbers, a linear model serves as a good first-order estimate to provide the results needed to quantify the benefits of self-assembly, while also making calculations very convenient.

Since all metrics are modeled linearly, their slopes can be used to provide more insight into the data and results. For instance, an estimate of the percent reduction in total service time by using self-assembly can be calculated as follows:

$$PR_{FSTT} = \frac{y_{NC} - y_C}{y_{NC}} = \frac{m_{NC}x - m_C x}{m_{NC}x} = \frac{m_{NC} - m_C}{m_{NC}}$$

The variables are defined as follows:
PR_{FSTT} is the percent reduction in fleet service total time
x is the number of scooters for a run
y_{NC} is the fleet service total time with **no clustering**
y_C is the fleet service total time with **clustering**
m_{NC} is the slope of the fleet service total time line with **no clustering**
m_C is the slope of the fleet service total time line with **clustering**

Although the fitted lines do not have exact zero-intercepts, we expect that this assumption does not have a large impact on benefit estimate.

Furthermore, the percent reduction in the number of stops for service personnel can be calculated from the slope of the cluster number metric:

$$PR_{SS} = 1 - m_{CF}$$

The variables are defined as follows:
PR_{SS} is the percent reduction in the number of service stops
m_{CF} is the slope of the number of clusters (stops) line

If no scooters are self-assembled, then the number of clusters would be equal to the number of scooters, since each cluster would only have one scooter. In this scenario, the slope is 1. As more scooters self-assemble, the number of clusters and service stops decreases.

The percent reductions in service total time as well as number of service stops provide a quantitative way to measure the benefits of the self-assembly application. The sensitivity of these quantities with respect to each algorithm parameter are examined in the following sections.

4.3.1 Travel Distance Threshold Variation

We begin by examining the sensitivity of our metrics as the scooter travel distance threshold is varied. Figure 8 shows the fleet service total times for each of the values in the parameter's range, while Figure 9 shows the number of clusters (service stops).

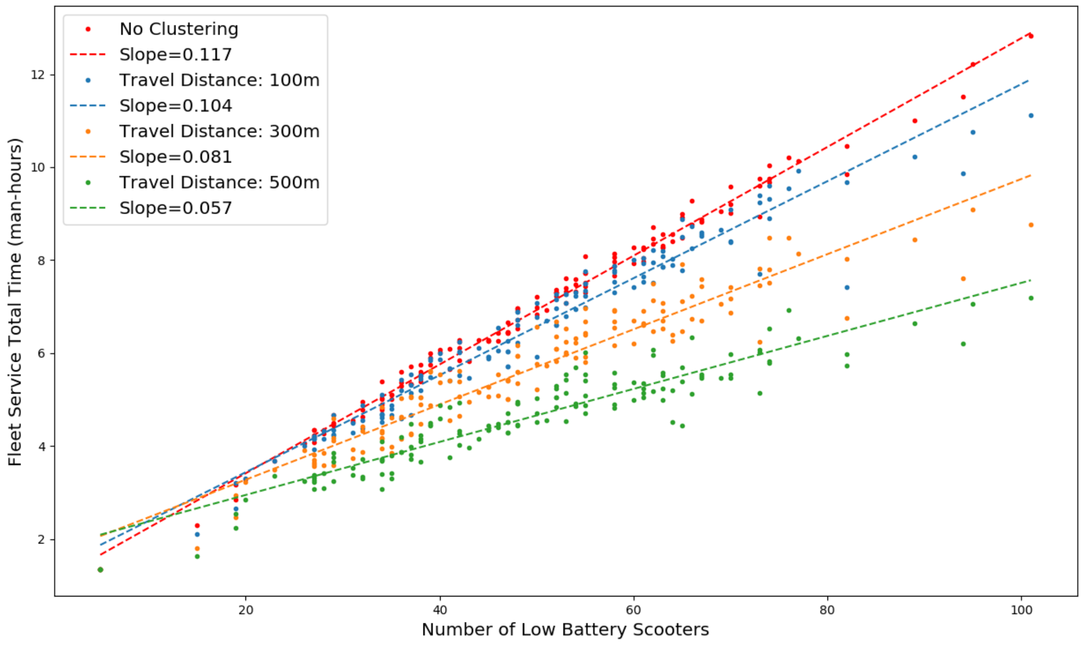


Figure 8 – Fleet Service Total Time vs Number of Scooters
Battery Threshold: 40%
Number of Personnel: 1
Service Type: Collection

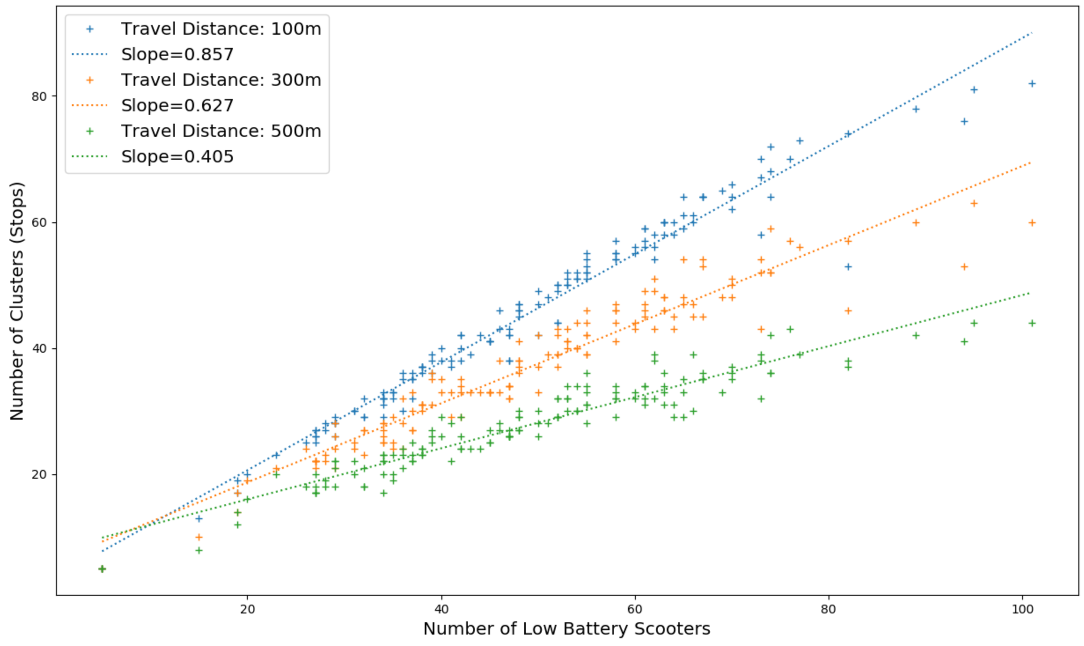


Figure 9 – Number of Service Stops vs Number of Scooters
Battery Threshold: 40%
Number of Personnel: 1
Service Type: Collection

Using the equations in Section 4.3, the following table summarizes the benefit in terms of the percent reduction in service total time and number of stops.

Scooter Travel Distance Threshold (m)	% Reduction in Fleet Service Total Time	% Reduction in Number of Service Stops
100	11.1%	14.3%
300	30.8%	37.3%
500	51.3%	59.5%

Table 5 - Performance Benefits with Respect to Scooter Travel Distance

4.3.2 Number of Service Personnel Variation

Next, we examine the sensitivity of our metrics as the number of service personnel is varied. Figure 10 shows the fleet service total service times for each of the values in the parameter's range.

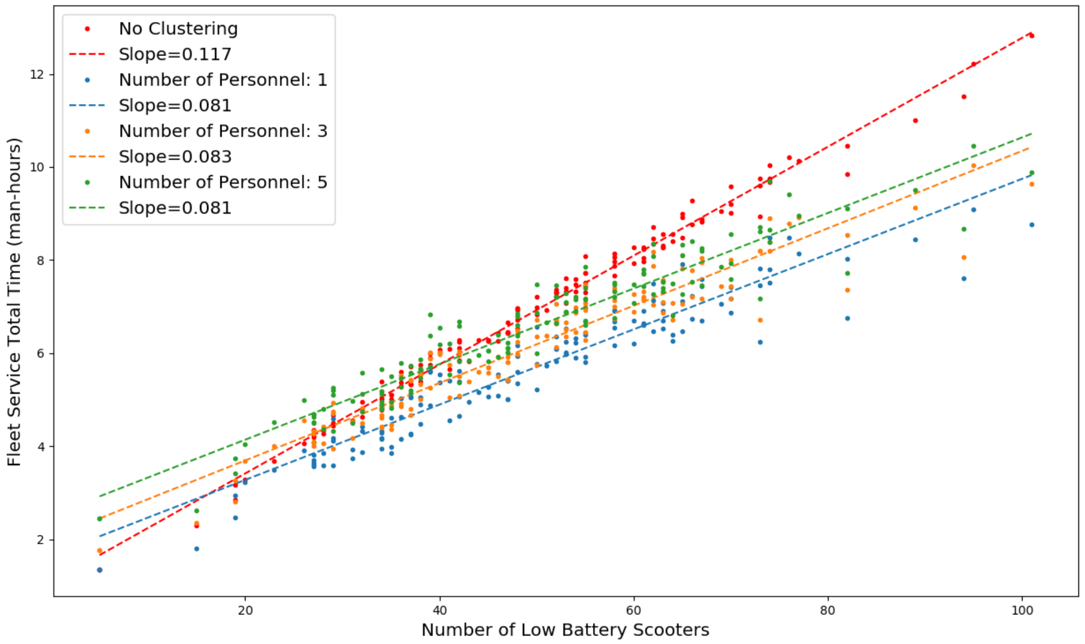


Figure 10 – Fleet Service Total Time vs Number of Scooters
Battery Threshold: 40%
Travel Distance: 300m
Service Type: Collection

Similar to the previous section, the following table summarizes the benefit in terms of the percent reduction in service total time. Percent reduction in service stops is not calculated, as this metric is not affected by changing the number of service personnel.

Number of Service Personnel	**% Reduction in Fleet Service Total Time**
1	30.8%
3	29.7%
5	31.4%

Table 6 - Reduction in Fleet Service Total Time with Respect to Number of Service Personnel

We see here that changing the number of personnel does not seem to affect the total service time. This makes sense, since roughly the same total distance needs to be travelled to each cluster regardless of how many personnel are sent. The only differences that occur are minor variations in the route solution matrix generated by the VRP solver.

The only metric we do expect to see great benefit from an increase in personnel, however, is the fleet service time. The plot for this metric is shown in Figure 11.

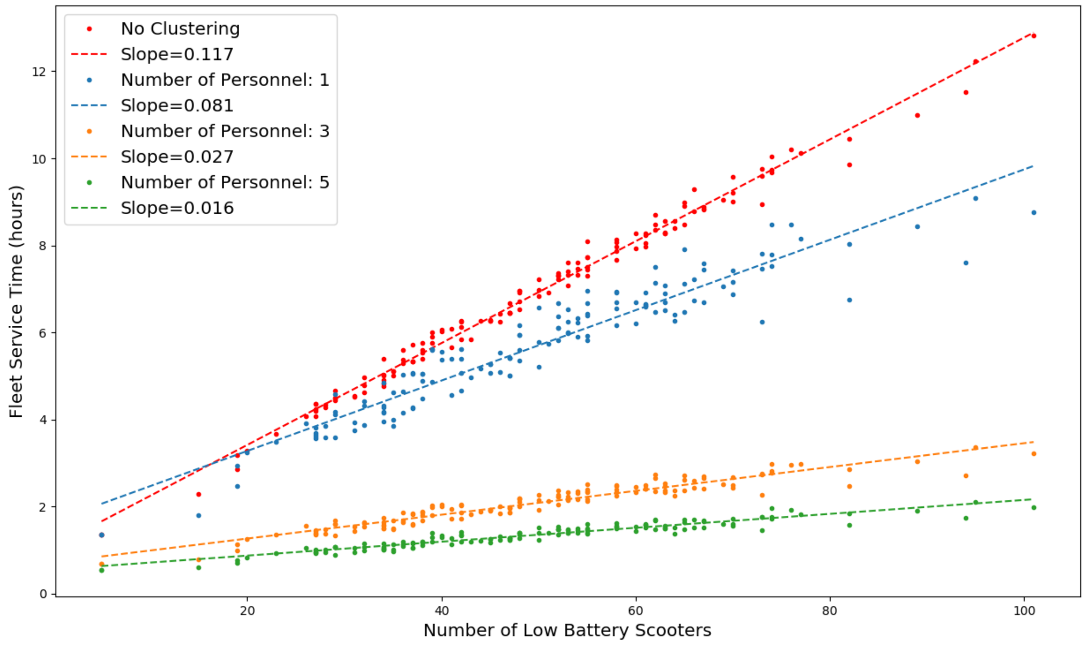


Figure 11 - Fleet Service Time vs Number of Scooters
Battery Threshold: 40%
Travel Distance: 300m
Service Type: Collection

Instead of calculating the percent reduction between the clustering and non-clustering, here we calculate the percent reductions with respect to the nominal scenario of only one service personnel. The same equation from section 4.3 can be used, but with different inputs.

Number of Service Personnel	% Reduction in Fleet Service Time with 1 Personnel
1	0%
3	66.7%
5	80.2%

Table 7 - Reduction in Fleet Service Time with Respect to Number of Service Personnel

As expected, we see a dramatic improvement in reducing the fleet service time as the number of personnel is increased.

4.3.3 Service Type Variation

Finally, we examine the sensitivity of our metrics as the service type is varied. Figure 12 shows the fleet service total times for each of the values in the parameter's range.

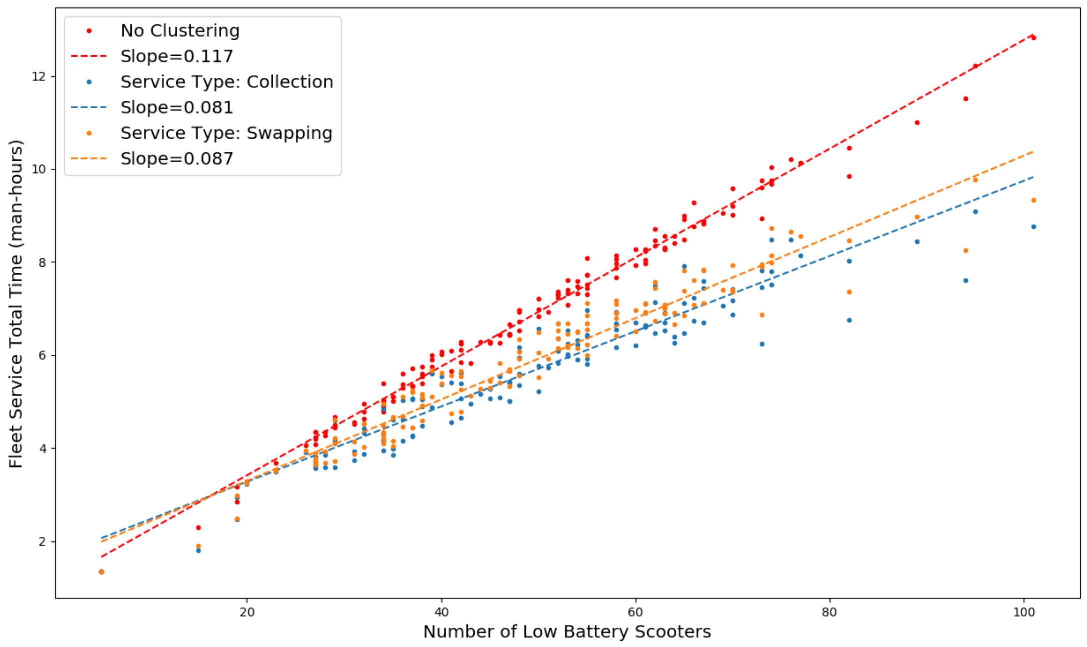


Figure 12 - Fleet Service Total Time vs Number of Scooters
Battery Threshold: 40%
Travel Distance: 300m
Number of Personnel: 1

The percent reduction in service total time can be calculated as before.

Service Type	**% Reduction in Fleet Service Total Time**
Collection	30.8%
Battery Swapping	25.6%

Table 8 - Performance Benefits with Respect to Service Type

We see that if the service type is battery swapping, the self-assembly algorithm offers less benefit than if the type is collection. As the number of scooters per cluster increase, the battery swapping times at those clusters become greater than the collection times. As in Section 4.3.2, the percent reduction in service stops is not calculated as this metric is not affected by the service type.

Chapter 5: Conclusion

5.1 *Summary of Contributions*

This thesis presents a study on self-assembly as an application of self-driving e-scooter fleets to reduce costs for operator companies. To this end, the application is tackled as two separate optimization problems in clustering and routing. A full algorithm pipeline is developed to solve both of these problems using modified versions of agglomerative clustering and a generic VRP solver.

We quantify the benefit to shared scooter operators as percent reductions in the number of service stops, fleet service total time (man-hours) and fleet service time if self-assembly is utilized. We studied the sensitivity of these metrics and their relation to the parameters of our algorithm, namely scooter travel distance, number of service personnel and service type.

We find that the scooters' travel distance threshold plays the biggest factor in reducing the number of stops and service total time of a fleet, with reductions of up to 50% possible if the scooters are able to autonomously travel 500m, and over 10% reduction if the travel distance is 100m. We also found that self-assembly provides slightly greater benefits in terms of total service time for collection type services as compared to battery swapping. Finally, our results show that the number of service personnel plays no role in reducing the service total time, but greatly helps in reducing the fleet service time.

The two main insights that result from this study are as follows:

1) As self-diving e-scooter fleets are developed, efforts should be focused on improving the range for which the scooters can travel autonomously or teleoperated, since this factor is shown to have the greatest impact on reducing servicing times.
2) Self-driving e-scooter fleets do not necessarily have to be a replacement for humans in the servicing process. When self-assembly is used together with the conventional servicing procedure, increasing the number of personnel can help counter low scooter travel distance thresholds, and the service time for a fleet can still be dramatically reduced.

5.2 *Ongoing and Future Work*

Although the work done in this project presented some interesting results regarding the benefit of self-assembly to shared scooter operator companies, we have only scratched the surface of the possibilities and scenarios of self-driving e-scooters.

The algorithm presented in this thesis for the self-assembly application has several areas for improvement. The clustering stage currently consists of finding the solution for an optimization problem and then performing a series of refinement steps to tweak the solution, thereby making it sub-optimal. It would be ideal if the clustering problem can be reformulated to include these refinements as constraints, thus providing one coherent solution for the full problem instead of doing it as separate pieces. The Google Maps Roads API was used under the assumption that roads will always provide valid destinations for scooters to cluster to, but this is certainly not the case, especially if the road is a highway. A more rigorous method for determining the validity and feasibility of a destination is needed.

As stated in Section 4.3, the metrics in our study were modeled linearly, which is valid for a slightly reduced dataset and made benefit calculations convenient. It would be interesting, however, to apply the exponential model to certain metrics as previously stated and see how the results would change. We would no longer be able to assign a percent reduction value for an entire parameter configuration, but this would be a more realistic model for large scooter fleets or scooters that have a large travel distance threshold.

Finally, since this thesis only studies one particular application of an autonomous scooter fleet, it would be beneficial to use it as a starting point in the study of other applications. For example, scooters may be able to autonomously re-position to directly reduce the travel time of service personnel, rather than doing so indirectly by reducing the number of stops. Studies can also be conducted on how utilization of the fleet to reduce costs can be integrated with utilization of the fleet to increase ride revenue by re-positioning to high demand areas. There may be a way to combine these two applications into one complete optimization problem to solve.

Appendices

6.1 *Appendix A: Tools and Software*

Software Language and Packages
Python 3.7.6
NumPy 1.18.1
Matplotlib 3.1.1
Pandas 1.0.4
Scikit-learn 0.23.2
Google OR-Tools 8.0.8283

Microsoft Bing Maps APIs
Distance Matrix API
Routes API

Google Maps Platform APIs
Static Maps API
Roads API
Directions API

Bibliography

[1] H. Andersen, Y. H. Eng, W. K. Leong, C. Zhang, H. X. Kong, S. Pendleton, M. H. Ang and D. Rus, "Autonomous Personal Mobility Scooter for Multi-Class Mobility-on-Demand Service," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, Rio de Janeiro, Brazil, 2016.

[2] M. Toll, "'Self-Driving' Shared Electric Scooters Are Here – Are They Awesome or Terrifying?," electrek, 20 May 2020. [Online]. Available: https://electrek.co/2020/05/20/g0-x-self-driving-shared-electric-scooters-are-here/.

[3] H. Field, "Spin Teams Up With Tortoise on Teleoperated E-Scooters," Emerging Tech Brew, 27 January 2021. [Online]. Available: https://www.morningbrew.com/emerging-tech/stories/2021/01/27/spin-teams-tortoise-teleoperated-escooters?__cf_chl_jschl_tk__=8efaa1787f6c8a3221d04b1e291fa98d84985b97-1616685279-0-AaEw52gRsYUOok7sIibLzB-dvuO39p-IQQ3Vi3YVFzB5U9li3aCfSgRRvTt4sUDpJUh-FqQuwS1mY.

[4] K. Pyzyk, "Segway-Ninebot's Newest E-Scooter Can Drive Itself," SmartCities Dive, 20 August 2019. [Online]. Available: https://www.smartcitiesdive.com/news/segway-ninebots-newest-e-scooter-can-drive-itself/561226/.

[5] R. Zhu, X. Zhang, D. Kondor, P. Santi and C. Ratti, "Understanding Spatio-Temporal Heterogeneity of Bike-Sharing and Scooter-Sharing Mobility," *Computers, Environment and Urban Systems,* vol. 81, no. 101483, 2020.

[6] J. Jiao and S. Bai, "Understanding the Shared E-scooter Travels in Austin, TX," *ISPRS International Journal of Geo-Information,* vol. 135, 2020.

[7] M. Li, S. Somenahalli and S. Berry, "Policy Implementation of Multi-Modal (Shared) Mobility: Review of a Supply-Demand Value Proposition Canvas," *Transport Reviews,* pp. 670-684, 2020.

[8] R. Mitra and P. M. Hess, "Who are the potential users of shared e-scooters? An examination of socio-demographic, attitudinal and environmental factors," *Travel Behaviour and Society,* vol. 23, pp. 100-107, 2021.

[9] D. Kondor, X. Zhang, M. Meghjani, P. Santi, J. Zhao and C. Ratti, "Estimating the Potential for Shared Autonomous Scooters," *IEEE Transactions on Intelligent Transportation Systems,* 2020.

[10] C. D. Manning, P. Raghavan and H. Schütze, "Hierarchical Agglomerative Clustering," in *Introduction to Information Retrieval*, Cambridge University Press, 2008, pp. 378-382.

[11] S. K. Uppada, "Centroid Based Clustering Algorithms - A Clarion Study," *International Journal of Computer Science and Information Technologies,* vol. 5, no. 6, pp. 7309-7313, 2014.

[12] C. D. Manning, P. Raghavan and H. Schütze, "K-means," in *Introduction to Information Retrieval*, Cambridge University Press, 2008, pp. 360-365.

[13] C. D. Manning, P. Raghavan and H. Schütze, "Hierarchical Clustering," in *Introduction to Information Retrieval*, Cambridge University Press, 2008, pp. 377-400.

[14] "Clustering Algorithms," Google, 2021. [Online]. Available: https://developers.google.com/machine-learning/clustering/clustering-algorithms.

[15] D. Reynolds, "Gaussian Mixture Models," *Encyclopedia of biometrics*, vol. 741, pp. 659-663, 2009.

[16] M.Parimala, D. Lopez and N. Senthilkumar, "A Survey on Density Based Clustering Algorithms for Mining Large Spatial Databases," *International Journal of Advanced Science and Technology*, vol. 31, no. 1, pp. 59-66, 2011.

[17] M. Ester, H.-P. Kriegel, J. Sander and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," in *KDD-96 Proceedings*, 1996.

[18] P. Fränti and S. Sieranoja, "How Much Can K-Means be Improved by Using Better Initialization and Repeats?," *Pattern Recognition*, vol. 93, pp. 95-112, 2019.

[19] J. K. Lenstra and A. R. Kan, "Some Simple Applications of the Travelling Salesman Problem," *Journal of the Operational Research Society*, vol. 26, no. 4, pp. 717-733, 1975.

[20] P. Toth and a. D. Vigo, The Vehicle Routing Problem, Society for Industrial and Applied Mathematics, 2002.

[21] G. Laporte and Y. Nobert, "A Branch and Bound Algorithm for the Capacitated Vehicle Routing Problem," *Operations-Research-Spektrum*, vol. 5, no. 2, pp. 77-85, 1983.

[22] A. S. Alfa, S. S. Heragu and M. Chen, "A 3-Opt Based Simulated Annealing Algorithm for Vehicle Routing Problems," *Computers & Industrial Engineering*, vol. 21, no. 1-4, pp. 635-639, 1991.

[23] M. A. Mohammed, M. K. A. Ghani, R. I. Hamed, S. A. Mostafa, M. S. Ahmad and D. A. Ibrahim, "Solving Vehicle Routing Problem by Using Improved Genetic Algorithm for Optimal Solution," *Journal of Computational Science*, vol. 21, pp. 255-262, 2017.

[24] "GBFS and Open Data," North American Bikeshare Association, 2021. [Online]. Available: https://nabsa.net/resources/gbfs/.

[25] "scikit-learn Machine Learning in Python," scikit-learn, 2021. [Online]. Available: https://scikit-learn.org/stable/.

[26] "Calculate a Distance Matrix," Microsoft, 2021. [Online]. Available: https://docs.microsoft.com/en-us/bingmaps/rest-services/routes/calculate-a-distance-matrix.

[27] "Calculate a Route," Microsoft, 2021. [Online]. Available: https://docs.microsoft.com/en-us/bingmaps/rest-services/routes/calculate-a-route.

[28] "Roads API Overview," Google , 2021. [Online]. Available: https://developers.google.com/maps/documentation/roads/overview.

[29] "Vehicle Routing Problem," Google, 2021. [Online]. Available: https://developers.google.com/optimization/routing/vrp.

[30] "The Directions API Overview," Google, 2021. [Online]. Available: https://developers.google.com/maps/documentation/directions/overview.